12-22-2000

A simulation-based heuristic for fleet assignment

Sonia Rosario Anorga
Florida International University

DOI: 10.25148/etd.FI14032323

FLORIDA INTERNATIONAL UNIVERSITY

Miami, Florida

A SIMULATION-BASED HEURISTIC FOR FLEET ASSIGNMENT

A thesis submitted in partial fulfillment of the

requirements for the degree of

MASTER OF SCIENCE

in

INDUSTRIAL ENGINEERING

by

Sonia Rosario Anorga

2001

To: Interim Dean Richard K. Irey
College of Engineering

This thesis, written by Sonia Rosario Anorga, and entitled A Simulation-Based Heuristic for Fleet Assignment, having been approved in respect to style and intellectual content, is referred to you for judgment.

We have read this thesis and recommend that it be approved.

Ronald Giachetti

Shih-Ming Lee

Martha A. Centeno, Major Professor

Date of Defense: December 22, 2000

The thesis of Sonia Rosario Anorga is approved.

Interim Dean Richard K. Irey
College of Engineering

Interim Dean Samuel S. Shapiro
Division of Graduate Studies

Florida International University, 2001

DEDICATION

I dedicate this thesis to Jorge, my husband and to Bertha and Ramon, my parents. Without their love, the completion of this work would not have been possible.

ACKNOWLEDGMENTS

I wish to thank the members of my committee for their support, patience and good humor. They gentle but firm direction has been most appreciated. Dr. Ronald Giachetti was particular helpful in guiding me toward the use of the appropriate tools. Dr. Shih-Ming Lee's views regarding the scope of the study assisted me greatly. I would like to thank my major professor, Dr. Martha A. Centeno, for her support and direction.

ABSTRACT OF THE THESIS

A SIMULATION-BASED HEURISTIC FOR FLEET ASSIGNMENT

by

Sonia Rosario Anorga

Florida International University, 2001

Miami, Florida

Professor Martha A. Centeno, Major Professor

Integer programming, simulation, and rules of thumb have been integrated to develop a simulation-based heuristic for short-term assignment of fleet in the car rental industry. It generates a plan for car movements, and a set of booking limits to produce high revenue for a given planning horizon.

Three different scenarios were used to validate the heuristic. The heuristic's mean revenue was significant higher than the historical ones, in all three scenarios. Time to run the heuristic for each experiment was within the time limits of three hours set for the decision making process even though it is not fully automated. These findings demonstrated that the heuristic provides better plans (plans that yield higher profit) for the dynamic allocation of fleet than the historical decision processes.

Another contribution of this effort is the integration of IP and rules of thumb to search for better performance under stochastic conditions.

TABLE OF CONTENTS

CHAPTER PAGE

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

FIGURE PAGE

CHAPTER 1:

INTRODUCTION

This thesis developed a heuristic for the assignment of fleet in the car rental industry. The fleet is considered a perishable good in the car rental industry. Hence, its allocation to the right place, at the right time is essential. It is considered a perishable good because its potential revenue is a function of the days each car is rented. If in a given day a car is not rented, there is lost revenue for that day that will never be regained since the day is gone. Similarly, if a customer requests a rental and there is no car to satisfy the request, the company incurs opportunity losses, good will losses, and customer dissatisfaction. Researchers have tried to find solutions to this problem using traditional operations research approaches. However, the dynamics of car rental industries demand models that respond dynamically as well. Hence, there is the need for multidisciplinary efforts that combine the use of methodologies and techniques from Operations Research (OR), Statistics, Information Technology (IT), and the rationale of the General Systems theory (GS).

The main contribution of the thesis is the development of a heuristic that allocates and moves fleet across locations and establishes booking limits with the objective of maximizing revenue while minimizing cost.

The remainder of this chapter presents the problem statement, the goal, and specific objectives. Chapter two reviews the literature in the areas of dynamic assignment problems and fleet management in the car rental industry. Chapter three discusses the methodology used for this effort, the characteristics of car rental systems, and a conceptual framework for

the heuristic. Chapter four describes the implementation of the prototype. Chapter five describes the various experiments done to verify and validate the model. Chapter six summarizes the results of the research and recommends future courses of investigation.

1.1 Problem Statement

High costs, poor quality, dissatisfied customers, market losses and low revenue are associated with inadequate allocation of fleet in the car rental industry. Every car that is not rented is a loss for the company. Not renting a car may be caused by shortages in the fleet, by having damaged units, or by having cars in the wrong place at the wrong time.

The nature of the car rental industry is such that if the car is idle, it does not generate any revenue, but it generates expenses. When a car is seating in the lot, there is an inventory cost associated with it. Furthermore, the depreciation and financial cost don't stop, and there is the possibility of an additional maintenance expense due to the need of a rewash. In addition, when a car is not where it should be, it creates dissatisfied customers whom in turn may go someplace else to rent a car.

From an industry that grew based on low cost cars during the period from 1970 to 1990, it is now an industry with an expensive perishable good. This makes the optimal consumption of fleet vital to generate revenues and survive. However, it is not easy to manage a fleet because most of the time companies have several departments dealing with the same fleet but with different interests. These departments include Fleet Planning, Marketing/Sales, Revenue Management, and Operations. The overall car assignment problem has three levels: Strategic, Tactical and Operational. The Strategic level deals with

the allocation of cars during the next 12 to 18 months, the Tactical deals with the allocation in the next 5 months, and the Operational deals with the week to week assignment. Each corporate department works at different levels, looks at the fleet in unique ways and hence, individual objectives may differ. For example, at the corporate level, it may be decided that a car should be returned to the manufacturer; yet, the local rental station may refuse on the basis of demand. Similarly, a local rental station may set productivity and customer satisfaction goals that are not necessarily 100% compatible with the long-term corporate goals.

The Fleet Planning department is concerned with the number of cars that need to be bought or returned to manufacturers in the short, medium and long terms. Its goal is procuring the correct amount of vehicles in each rental station, at any period in time, at a minimum cost. On the other hand, the Marketing and Sales department concentrates in the "business volume" trying to increase market share; it is in constant search of new distribution channels, new products, and new promotions, which would lead to increase demand in each and every location. The Revenue Management department is concerned with maximizing revenues by looking at the current demand and supply and establishing measures and directives to extract the most revenue. It also dictates the pricing and fleet availability policies as well as overbooking levels. Finally, the Operations department and individual rental stations are concerned with customer service, high utilization of fleet, and the proper maintenance of it. They manage the day to day operation of the fleet; their decisions have a direct impact in customer service.

As it can be seen, there are multiple interests within the corporate structure. The short and long term goals of these various interest groups may conflict. The ultimate goal of

proper allocation of fleet to maximize overall revenue can not be achieved without taking in consideration all the relevant elements and corporate objectives. In most cases, the car rental industry manages its fleet focused in one corporate objective at a time, and it does not cross reference the individual results. Common assignment models used in industry, concentrate in one location (or location pools) to set the available fleet for each day and for each car class. These models do not include the fact that cars can be moved between locations (Geraghty and Johnson, 1997). Recent optimization models developed for two major car rental companies assume that there are a fixed number of cars at each location during the planning horizon. They also consider a base fleet of car in every location, every day, in spite of the fact that cars rented in one location could be returned to another one, that new cars can be bought, and old cars could be sent back to the manufacturers.

Fleet allocation is a dynamic stochastic problem; it is very similar to the allocation of rooms in the hotel industry (Bitran and Mondschein, 1995) and to the allocation of airplanes in the airline industry (Rexing et al., 2000). In these two latter industries, the use of heuristics has been successful when combined with mathematical analytical models. Modeling this situation is not an easy task. In fact, it is a combinatorial problem of all possible alternatives and the high level of integration of all the elements. Although the number of previous efforts explicitly for the car rental industry is limited, methods and models from the airline and hotel industry can be used to derive solutions for the car rental industry. However, these models need to be tried out and possibly modified; hence, the goal of this effort is to research for better tools and models for fleet allocation in the car rental industry.

1.2 Goal and Specific Objectives

The main goal of this effort was to develop a heuristic for fleet allocation at the operational level. The end product is a heuristic that integrates a simulation program with linear programming and with rules of thumb. The heuristic acquires fleet status, dynamics, demand projections, and finds a "best solution" for the fleet allocation problem.

The specific objectives were to:

- Understand the Fleet System.
- Conceptualize a framework for the heuristic.
- Data Collection and Data Analysis.
- Heuristic development.
- Experimentation to test the heuristic.

Chapters three, four, and five provide details on how each of these objectives was satisfied; Chapter six summarizes the effort, analyzing its contribution and future extensions.

CHAPTER 2:

LITERATURE REVIEW

This chapter reviews previous research and applications in the areas pertinent to the goal. The review has been organized by industries. Assignment problems can be found in many industries such as manufacturing, hotels, car rental and airlines. However, this review is focussed in the Hotel, Airline, and Car rental industries primarily.

2.1 Dynamic Assignment Problems in the Hotel Industry

One of the earliest works for the Hotel Industry is the one developed by Rothstein (1974). He proposed the solution of the Hotel Overbooking problem by the use of a Markovian sequential decision model. The overbooking problem is the difficulty in setting limits for the booking of reservations for a given arrival date with the objective of achieving maximum profit and minimum loss of good will. What happens is that not all reservations are realized, some will be cancelled or will not show-up; because of that, hotels need to accept more reservations than they can honor, so as to minimize the opportunity for empty rooms. If more reservations than the hotel can accommodate are requested, the hotel will lose good will and probably some customers for life.

Rothstein (1974) showed an approach to solve this problem with a model based on his previous work in the airline industry. His model considers the overbooking problem as a finite stationary markov process in which there are economic rewards. The process changes

states base on transition probabilities and at each point in the sequence decisions can be made towards obtaining the total maximum expected reward. The transition probabilities are based on the cancellation, no-show, and demand probabilities. An important insight gained from this work is the way he model the probabilities, the optimization problem, and how return and revenue are measured.

Another early effort is the one done by Ladany (1976), who developed a succinct sequential process to select booking limits for single and double bedrooms in a hotel. He experimented with a small case and suggested the grouping of demand data and the reduction of decision periods when solving real world problems. Williams (1977), describes a method for the setting of optimal reservations policies. The strategy is to determine expected values for future events and to evaluate the optimal number of reservations that will minimize losses. By losses he considers the opportunity cost of a room that it is not rented and the cost of not honoring a reservation because of overbooking. He analyzes and projects three types of events: stayovers, reservations, and walk-ins.

The problem in the Hotel industry is how to optimally rent rooms. It is a dynamic assignment problem because the status of the system varies continuously as customers arrive or depart. However, it is a fixed resource problem because the number of rooms does not vary. The complexity of the problem is similar to that of the car rental industry because its customers have different renting patterns and come from different market segments. Bitran and Mondschein (1995) showed how heuristics could achieve good solutions for discount allocation. Specifically, they present three models: a model without reservations, a multiple-product case, and a model with reservations. The model without reservation is the simplest of the three; it does not deal with requests for rooms prior to the arrival date and it only

models one type of room. The decision is to accept or reject a customer when it arrives based on the maximum expected profit at any period in time. The price associated with a customer and the rejection cost are considered in reaching a decision. The second model considers different types of rooms and substitutions. Similar to a car rental, the initial type of room may be substituted for one of higher value. The third model considers reservations made in advanced and deals with possible no-shows and walk-ins.

2.2 Dynamic Assignment Problems in the Car Rental Industry

Edelstein and Melnyk (1977) present a way to solve the short-term dynamic fleet assignment problem in the Car Rental industry. Short-term planning is affected by the demand, the initial supply, and the future decisions. The company they worked for administers its fleet based on individual or pool locations; in both cases, an adequate way to redistribute fleet was necessary for the company's survival. They describe an interactive model-oriented management tool called the Pool Control System (PCS) developed for Hertz Rent A Car. This model turned out to be a practical and a clear descriptive tool to assign fleet. The PCS system is an application that receives input from field managers about real data from the prior day and a forecast for the week ahead; it also receives information from the distribution managers about scheduled and projected car distribution. After all information is gathered, the model produces a report with the status of the system for the next seven days. This report serves as an analytical tool to foresee shortages or excess of fleet. The distribution managers can readjust the car distribution and run the report over and over again until they are satisfied with the outcome. PCS is a semi-manual *what-if* tool for

them. Once the model contains the final distribution policies, three reports are produced and sent to the field.

Carroll and Grimes (1995) show how Hertz has been improving its fleet planning, rate optimization and demand control by the use of non-linear programs, marginal values, and simulation. Hertz has developed and integrated Yield Management System (YMS) since 1989. YMS supports the decision process in: (1) Planning fleet levels in the long run, (2) Deploying fleet in the short-term, (3) Managing revenue or yield and (4) Offering products based on the marginal cost of offering them. The Yield Management System gets its input from the operational systems and feeds them with the directives that have been accepted by the analysts. For long Term planning of fleet, YMS uses the marginal value of a car to determine if it should be added or deleted from the fleet. However, for short-term planning of fleet, Hertz still uses the Pool Control system named the DPDA model (Daily Planning and Distribution Aid). In managing revenue, YMS uses optimization models to suggest rates that will give the maximum revenue to the company. Finally, in the area of demand control, it suggests when, where and how many cars to offer, based on the forecasted demand and fleet availability.

Geraghty and Johnson (1997) explain how capacity is managed at National Car Rental by the processes of fleet planning, planned upgrades, and overbooking. These processes are based in forecasts, heuristics and linear programs. They stated that there has been a dichotomy between the inventory and pricing functions in the car rental industry. Thus, they proposed that the main function of aligning all interests should be centralized. Within a year, the new directives originated profits for the company.

2.3 Dynamic Assignment Problems in the Airline Industry

Fleet Allocation for the Airline industry has been extensively studied. For example, American Airlines, United Airlines, Swissair and Delta have accomplished major milestones in the area of Yield and Revenue Management. These companies have approached the overall problem by decomposing it into sub problems. Different OR models run in parallel, and the outcomes of one model feed other models in search of a good or optimal global solution for the complete system.

Airlines take into consideration behavior, different locations, market segments, types of aircraft, rates, federal restrictions, capacity limits, crew assignment and corporate objectives. Their models can be used as the basis for the development of algorithms to holistically manage fleet in the car rental industry.

American Airlines has done a lot in the last 10 years in the area of decision support systems (DSS) with OR models. Cook (1998) explains the evolution of SABRE's IT department's effort to develop Decision Support Systems (DSS). He elaborates on the basic decision variables of its applications: Where to fly?, How often to serve a market and over what hub?, What times of day to fly?, What type of aircraft to assign to each route?, etc.

At DELTA Airlines, Subramanian et al. (1994) describes a time-space network model for aircraft assignment and its solution by the use of the OB1 interior point code. OB1 converts the program into a smaller mixed integer problem that is solved with the OSL mixed-integer programming code. A typical model contains 40,000 constraints and 60,000 variables; OB1 reduces the problem using the "lonely-plus/lonely minus" method. It is stated that the interior-point method dominates the simplex method for this class of problems.

Another example of the use of linear and non-linear programming is found in the area of crew scheduling for the airlines, Bixby et al. (1992) describe the experience of solving a linear problem of 12,753,313 variables with an hybrid method that combines the interior point and simplex methods. They compared the time required to solve the same problem with only simplex or interior point (CPLEX and OB1) and concluded that the hybrid approach was better.

Rothstein (1974) showed how to solve the overbooking problem through the implementation of a markovian model. Subramanian et al. (1999) also uses this approach for the problem of a single leg airline. They propose heuristics to determine the optimal booking policies when there are multiple fare classes, time-dependent arrival probabilities, no-show and cancellations with probabilities and refunds that are class and time dependent and with overbooking permitted. They use queuing-control techniques and test the approach with real data from Delta.

McGill and Van Ryzin (1999) analyzed the use of Dynamic Programming (DP) in general for revenue management related problems and point out the infallibility of it for real-world problems because of their size. The actual approaches have been to identify and exploit the structural properties of optimal or near optimal solutions and the implementation of systems based on monotonic threshold curves or control limits that contain the solutions (optimal or close to optimal). They state that DP can not be used for a real-time revenue management in the airline industry. However, they agree that it can be used in an exception basis for specific problems that are critical.

In Europe, the University of Geneva has been doing research using Branch and Bound algorithms implemented with a library of parallel search algorithms created at ETH

Zurich. They also have researched the use of genetic algorithms for this problem. The research has been applied to Swissair.

Artificial intelligence and Expert Systems have been tried as well for the allocation problem. There are applications that deal with rule oriented algorithms to maximize revenue in the airlines; they embed the expertise in the problem domain inside the models. As mathematical or analytical models can be infeasible to implement when the variables grow exponentially, Artificial Intelligence could be the answer in those cases. Ritcher (1989) says in his analysis of Operations Research in the Airlines, that Neural Networks can give faster time responses.

2.4 Dynamic Assignment Problems in Other Industries

OR Models for Dynamic assignment problems can be found in Manufacturing supporting the Inventory Control and Logistic functions. Bitran and Dasu (1992) present two approaches for the problem of setting ordering policies in the semiconductor industry. In this environment there are stochastic yields and substitutable demands just like in Car Rental. In their first approach, they solve finite horizon stochastic programs on a rolling horizon basis. In their second approach, they developed a heuristic that is based on the structure of the optimal policy for two period problems. They conclude that the heuristic is efficient and further research on it is worth.

OR Models for Dynamic assignment problems are also found within financial Decision Support Systems as Yield or Revenue Management. Since revenue is a function of the timely allocation of the assets in these industries, revenue management has the

responsibility to make the most important decisions: How much to sell?, At what price to sell?, When?, and Where?. As Weatherford and Bodily (1992) stated, Revenue Management has to maximize profit/contribution, capacity, utilization and revenue while minimizing loss of customer good will.

Eom and Lee (1990) identified Decision Support Systems for distribution planning, train dispatching and fleet configuration. These DSS contain OR methods like Network Optimization, shortest route algorithm, maximum low algorithm, PERT/CPM and GERT. Ciancimino et al. (1999) presented a study about the Yield Management Railway passenger transportation problem in Europe. They state that it can be solved by the use of three different methods: a deterministic linear program, a probabilistic non-linear program and an algorithm that exploits the structure of the network optimization problem and yields comparable results with more efficiency.

2.5 Summary

From the literature reviewed, it can be seen that the dynamic assignment problem is a complex one. The problem is commonly found in industries with perishable assets like airlines, hotel, and car rental. Therefore, there has been a lot of time and money devoted to the development of applications and tools to help with its solution.

CHAPTER 3:

DESIGNING THE HEURISTIC

This Chapter describes the methodology used to design the heuristic. The methodology consists of:

- Understanding the Fleet System.
- Conceptualization of a framework for the heuristic.
- Heuristic Development
- Data collection and analysis
- Experimentation.

Heuristic development, data collection, and data analysis are discussed thoroughly in Chapter 4.

3.1 Methodology

The Heuristic has been designed in five steps. Four of them resemble the general phases in the development of Information Systems (Figure 1).

The first step was the analytical phase in which the main objective was the understanding of the Fleet System, its components, goals and limitations. The real system was studied and its elements and dynamics were learned; it applied the principles of the General Systems theory (GST) to find the objective of the system, and the relevant components that affect it.

The second step was the Conceptualization phase, which allow the design of the framework for the heuristic. The heuristic's goal, scope, elements, relationships, inputs and outputs were also determined.

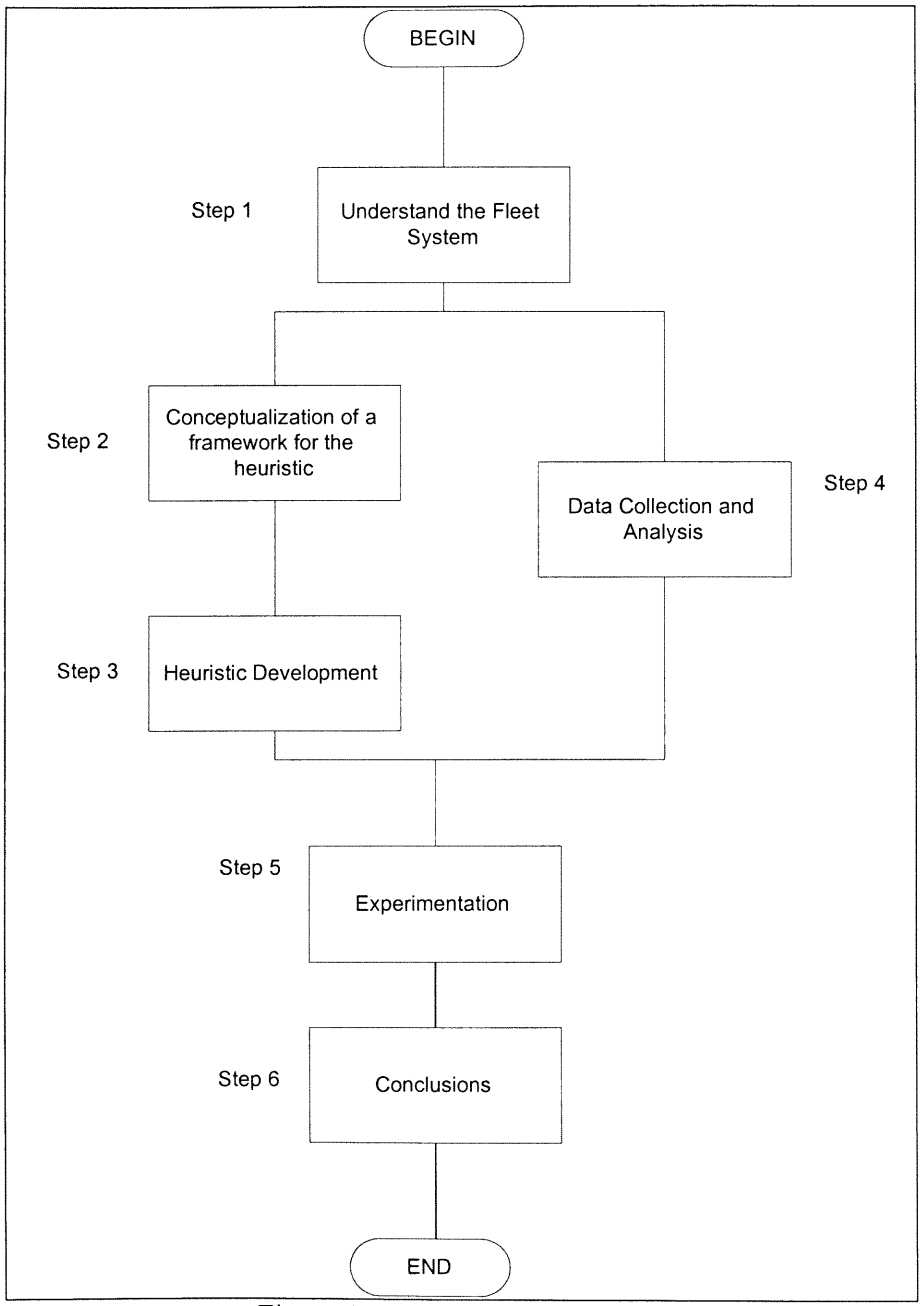

Figure 1: Heuristic Design Steps.

The third step was the actual development of the heuristic. In this step, the problem size to implement is established; data needs, physical representation of the framework, and the tools to be used were established. Lingo, Arena, Visual Basic for Application (VBA), and Microsoft Excel were selected as the tools for implementation because they offer programming flexibility as well as integration capabilities.

Implementing the framework requires the utilization of commercial-of-the-shelf (COTS) software and the development of customized user interfaces. There are several alternatives for each one of the components. For the optimization model, it is possible to use CPLEX. It has good speed and it is prepared for big size problems. It has products with simple command structure, easy problem entry formats, and on-line help screens. It is available in different platforms and can be called from different programming languages making possible the full automation of the heuristic. We also have Lingo, which is a product from Lindo systems. It has the capability of solving linear, integer, and non-linear programs, and its price is convenient. The Hyper version of Lingo 5.0 was used for the optimization model for its capabilities for data manipulation through sets, loops and functions. It minimizes the editing effort while providing with a structured and clear way to represent the problem.

For the simulation model one can choose any general-purpose simulation software with external programming capabilities. There are several well-known, simulation packages, like Arena, marketed by Systems Modeling Corporation, AweSim, marketed by Pritsker corporation, and Symix (1999), GPSS/H [Henriksen and Crain (1994) and Schriber (1991)], Micro Saint, Micro (1998), MODSIM III (Banks et al. 1996) and Promodel. Arena 4.0 was used for the simulation model for its flexibility to model different types of systems. It is not

focus on one specific type of model, and it interfaces with other Windows products for added flexibility.

For the rules, all that is needed is a programming language such as Visual Basic, Fortran, Java, or C. The criterion to select the programming language is that it should have the compatibility to interface with the optimization and simulation software. Visual Basic is the most recommended one because it would enable fully automation of the heuristic. However, for prototyping, Excel can be used, with its functions, macros and data manipulation capabilities, Excel is certainly a useful tool.

The fourth step was data collection and data analysis. In this step, historical data for reservation and rental transactions, fleet, prices, arrival pattern, breakdown percentages and over stays was gathered and projections made. Historical transactions, prices, arrival patterns and percentage of over stay were obtained by collecting information about rentals, no-shows, cancellations and turndowns. Fleet figures were obtained by using the daily fleet running average in industry. Breakdown percentages were estimated by gathering data from manufacturers. The review of the historical data identified three scenarios for the Fleet system, in which two of them are caused by extreme demand conditions and the other one happens during average demand levels. The data obtained was masked and grouped to represent the selected problem size and net revenue was computed for the different scenarios. Historical data and a triangular distribution were used to project demand and this one was also grouped accordingly to the problem size.

The fifth and last step was the validation of the heuristic and is called the experimentation step. This step established experiments, measures of performance and performed the hypothesis testing. Three experiments were set; each of them represents a

different status of the fleet system, namely two extreme scenarios and an average one. The level of usability for the heuristic resides in its capability to appropriately react at the different situations that may occur in real life. By testing it in an average situation as well as in extreme ones, its validity and scope can be assessed.

Additional experiments could have been set; however, the scarcity of input data prevented it. Further experimentation is highly desirable. As part of the experimentation, the heuristic was executed for each of the scenarios, and the net revenue yielded was compared against the historical revenue. A hypothesis test (t-test) for comparing two population averages was used to assess the performance of the heuristic. More details are given in Chapter 5.

3.2 Understanding the Fleet System

The General Systems theory was used to study the fleet system and to identify its relevant components. The system of interest exists at the operational level and its major goal is to generate the highest possible profit from the existing fleet capacity and demand. Its relevant components are the car rental locations (stations/branches), the fleet, demand, corporate policies and goals (Figure 2).

The rental locations are places in which cars are rented, returned, cleaned, maintained and where repairs can be managed. They are distributed geographically and can share fleet. They can make their own decisions, but corporate policies take precedence.

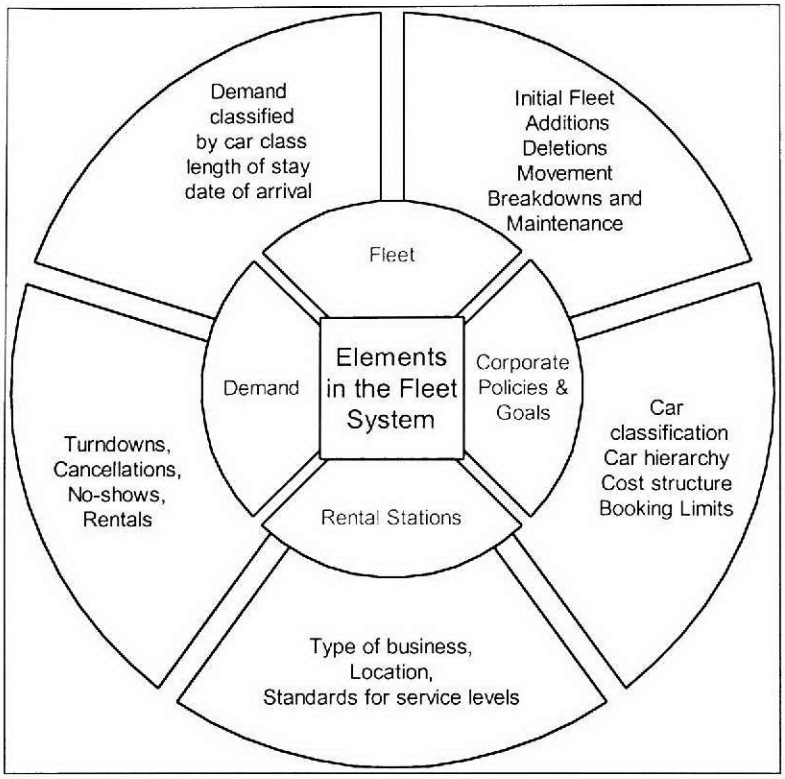

Figure 2: Elements in the Fleet System.

The fleet is composed of cars that are differentiated by classes. The car classes are arranged hierarchically by the company; a given car class can be substituted by any other that is higher than itself in the hierarchy (Table 1). Cars can be moved from one location to another by truck, train, or individually. Every movement originates cost.

Class	Description	Hierarchy
E	Economic	1
C	Compact	2
I	Intermediate	3
F	Full Size	4
S	Specialty	5

Table 1: Car Class Hierarchy.

Five things can happen to a car after it is rented:

1. The rental can last the contracted number of days or a different number of days (more or less).
2. It may come back in good condition and be ready for the next rental.
3. It may need maintenance or repair.
4. It may be moved to another location or depart the company.
5. It may not return at all because it was severely damaged or stolen.

These five possible situations generate costs and expenses to the company; some of them can be avoided, while others can not. The transition from these states is considered deterministic and probabilistic respectively. The maintenance time for example may be considered a known average, whereas the repair time is considered uncertain because it depends on the specific car problem. Demand for cars is seasonal and stochastic. It varies by location, market segment, day of the week (DOW), day of the year (DOY), car class and by the length of rent or "length of stay (LOS)" (Table 2). Most of the time is economically better to serve demand for longer days of stay. It is important to point out that LOS is stochastic because a customer may keep the car less or more time than originally planned, creating what is called over stay or under stay.

LOS	Description
N	{1,2,3,...7} N days of rent
8+	8 or more days

Table 2: Length of Stay.

Total or Unconstrained demand is a term that refers to the demand produced by all possible customers regardless of the existence or not of cars for them. Total demand can be further classified into two groups depending if it is realized or not. The demand that is realized (net demand) is the effective number of customers that would actually rent a car, while the demand that is not realized is the number of customers that will eventually cancel or will not show up. The demand can also be described by type of customers, market segments or by distribution channels (Table 3).

Segment	Description
1	Consumer
2	Travel Agent
3	Corporate
4	International
5	Tour Operators

Table 3: Market Segments.

The fleet is managed under the framework set by corporate or local policies. These policies include prices, car classification, cost, movement policies and additions or deletions to the fleet. To achieve the goal of maximum net revenue, decisions need to be made over time to take full advantage of the demand and supply available to the system, subject to the different constraints that affect it.

The planning period at the operational level is from one to four weeks. Decisions are made at different stages to make sure supply meets demand effectively. The Cost of movement is compared against possible revenue and booking limits are evaluated to appropriately redirect the use of fleet.

The movement of cars between rental stations is approved when the net revenue it generates is positive. The total cost of movement during a given planning horizon is the transfer cost plus the lost revenue of all possible rentals in the original rental station. This total cost is contrasted with the revenue they can generate in the target rental station.

Booking limits are set for some types of demand whenever their expected return is not appropriate. In this way, fleet capacity is saved for more profitable demand. Booking limits are a complement of fleet allocation and they both need to be synchronized. The probabilistic behavior of the demand sometimes creates the need for overbooking.

3.3 Conceptualization of a framework for the Heuristic

The framework for the heuristic is composed of two models of the fleet system and a set of rules. The first model is an optimization one, while the second is a simulation model (Figure 3). The decisions of what to move and what to rent are the heuristic's major goals.

The heuristic models the rental stations, fleet capacity, demand, rental patterns and corporate policies. Corporate policies are the different decisions and structures that support and rule the company and its services. The most important ones are car classification, car hierarchy, cost structure, car movement and booking limits. Some of these elements are considered exogenous factors; hence, they are inputs to the heuristic; some others, like car movements and booking limits, are considered endogenous elements because they are generated and evaluated within the heuristic (Table 4). The heuristic looks for better alternatives for car movement through its optimization model and its rules. The best alternative is the one that generates the maximum net revenue.

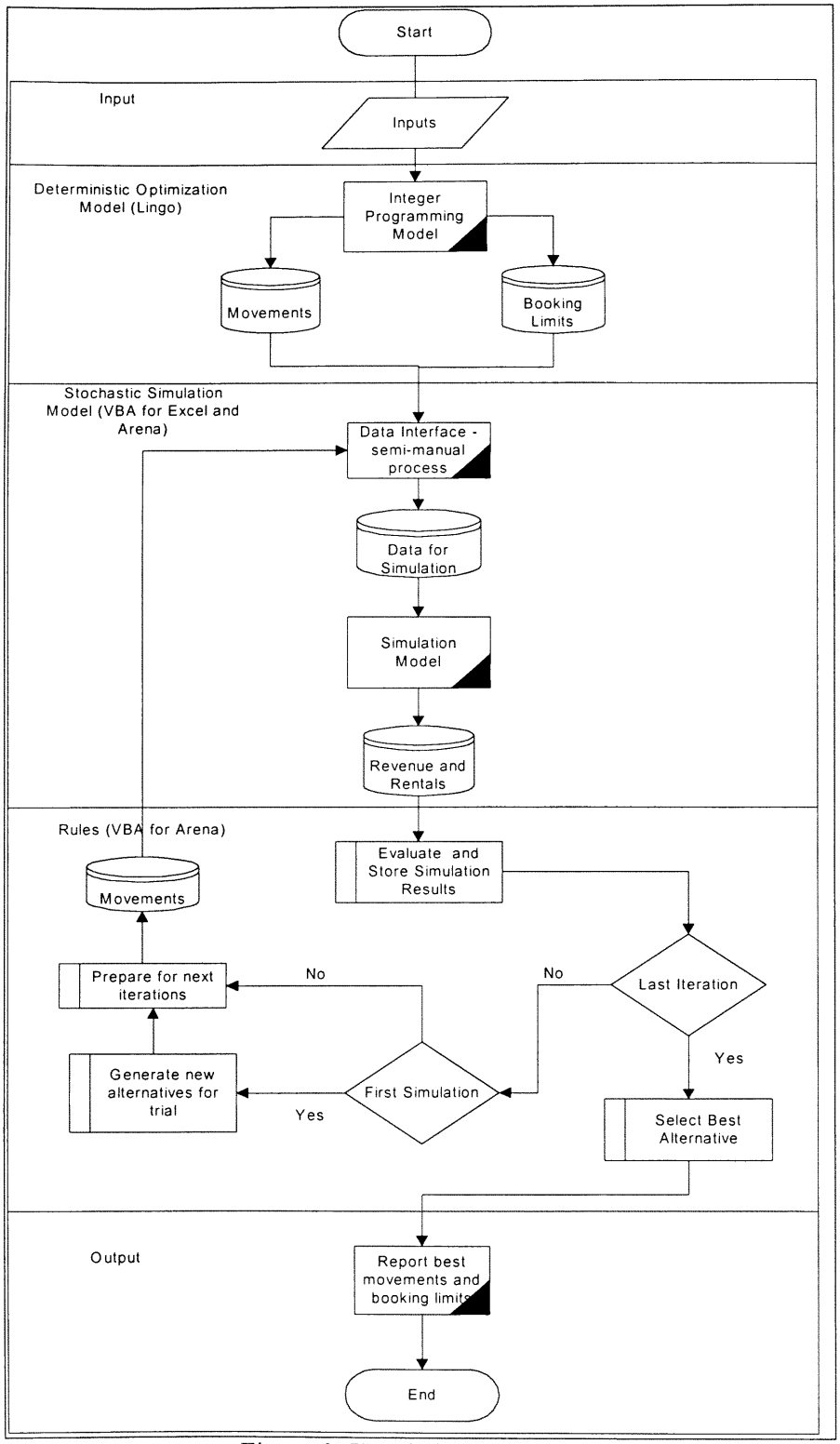

Figure 3: Heuristic Framework.

Fleet elements modeled in the heuristic
Exogenous Elements:
Initial fleet, fleet deletions, and fleet additions. Unconstrained demand classified by car class, length of stay, and day of week Car classes and hierarchy Probability of car breakdown Turndowns, cancellations and no-shows.
Endogenous Elements:
Rentals Arrival Patterns Constrained Demand Fleet Movement Probability functions for overstay or under stay. Corporate policies and goals of what to move and what to rent.

Table 4: Fleet Elements Modeled in the Heuristic.

The optimization model generates an initial set of decisions when given a deterministic input. It optimally solves the questions of *what to move* and *what to rent* by generating a set of rentals and fleet movements between stations that will generate the maximum profit. The mathematical model maximizes a function that represents the revenue from rentals minus the cost of movement and has restrictions on demand and supply.

The simulation model enables experimentation with rental and movement alternatives under stochastic conditions, using a set of rules. The rules evaluate the revenue obtained from the simulation and suggest new decisions for trial whenever they are possible. These rules seek to improve the search for the best assignment of cars across rental stations. The simulation model experiments with these new decisions until no more alternatives are suggested by the rules or until the decision-maker stops the process. The rules will not suggest more alternatives if its determined that they have found the best solution.

CHAPTER 4:

IMPLEMENTATION OF A PROTOTYPE

This chapter describes the prototyping of the heuristic. Detailed discussion of the three major components of it is given.

4.1 Data Collection and Data Analysis

Average rental, fleet capacity and operational data was acquired from a major U.S. Rental Car Company. The data was modified to protect the interests of the rental Car Company and to guard against release of sensitive information.

Data was gathered regarding rental station operation, fleet management, demand behavior and corporate policies (business logic). The different functions and characteristics of rental stations were collected and analyzed. The three stations selected for the prototype were chosen for their proximity and for their different type of business. One is predominantly commercial (Fort Lauderdale), the second one is for mostly leisure customers (Key West), and the third one is a combination of the other two (Miami). Data was also collected to represent the time a customer spends at the rental station.

Data on fleet management, such as capacity, cost structure, car classification and upgrades was obtained from historical information. The Demand behavior was gathered by location, car class, length of stay, date of arrival and arrival pattern. Corporate policies and the business logic in general were obtained from observation and study of the rental business.

It was determined that Rental Agents are expected to perform according to industry standard. This standard calls for 8 minutes for the average service time and 15 minutes for the average waiting time. The upper limit for the wait time is set to be 60 minutes. Most of the customers are really upset and leave the rental station after an hour of waiting.

Fleet management and policies were analyzed and represented using sample data. Initial fleet at each station was determined using historical information. Two car classes were established by grouping economic, compact (2-door and 4-door) and medium (sedan) into one car class, and full size, luxury, specialty, SUV, and vans into another class. Initial Fleet at each rental station as well as cost of movement was established using historical averages (Table 5 and Table 6).

Locations	Car Class 1	Car Class 2	Addition Car Class 1	Addition Car Class 2
Miami	150	497	500	500
Fort Lauderdale	393	922	200	200
Key West	29	29		

Table 5: Initial Fleet and Planned Additions.

	Fort Lauderdale	Key West	Miami
Miami	$20	$30	
Fort Lauderdale		$30	$20
Key West	$30		$30

Table 6: Cost of Movement.

Demand, arrival patterns, percentage of over stays, and under stay rentals were estimated from historical data as well. It was noticed that the arrival pattern changed as the day passed; hence, it was necessary to divide each day into 12 2-hour periods of time.

Demand is classified as Constrained or Unconstrained depending if it is limited but fleet capacity or not. Furthermore, it is sub-classified as Total or Realized demand, if it represents all possible customers that will make reservations or if it only contains the ones that will actually show-up the day of the rental and excludes the ones that may cancel or no-show. The four types are then called: Total Unconstrained demand (TU), Realized Unconstrained demand (RU), Total Constrained demand (TC), and Realized Constrained demand (RC).

Total Unconstrained demand represents all possible customers even if no fleet is available for them or even if some of them cancel or do not show up. Total Unconstrained demand for each location l and time period t is obtained by adding up historical data for rentals, no-shows, cancellations and turndowns (Formula 1).

$$TU_{lt} = R_{lt} + N_{lt} + C_{lt} + T_{lt} \qquad (1)$$

Where,

R_{lt} = Number of actual rentals at location l at time t.

N_{lt} = Number of no shows at location l at time t.

C_{lt} = Number of cancellation at location l at time t.

T_{lt} = Number of turndowns at location l at time t.

Realized Unconstrained demand is the unconstrained demand that could have actually showed up. It is obtained by subtracting historical and projected no-shows and cancellations from the Total Unconstrained demand. In other words, it is the actual rentals

plus the percentage of turndowns that would have shown had they be given a reservation. Historical no-shows and cancellations are the ones that actually happened from the reservations booked. Projected no-shows and cancellations are the ones that may have happened if all the turndowns would have been accepted by the rental stations and booked a reservation. The Realized Unconstrained demand is given by

$$RU_{lt} = R_{lt} + T_{lt}(1 - \eta_{lt} - \varsigma_{lt}) \tag{2}$$

Where,

$$\eta_{lt} = \frac{N_{lt}}{(R_{lt} + N_{lt} + C_{lt})} \tag{3}$$

$$\varsigma_{lt} = \frac{C_{lt}}{(R_{lt} + N_{lt} + C_{lt})} \tag{4}$$

and

η_{lt} = Percentage of no shows among turndowns at location l at time t.

ς_{lt} = Percentage of cancellations among turndowns at location l at time t.

Total Constrained demand is the demand that is limited by fleet capacity. It contains all possible customers even if some cancel or no-show. For location l and time period t, it is obtained by the sum of rentals, no-shows and cancellations. It did not consider the turndowns because those are the customers that were not accepted for lack of fleet capacity. It is given by

$$TC_{lt} = R_{lt} + N_{lt} + C_{lt} \tag{5}$$

Finally, the Realized Constrained demand is the constrained demand that shows up for location l and time period t it is equal to the number of rentals.

$$RC_{lt} = R_{lt} \tag{6}$$

Historical data from three locations was gathered for three different weeks. The first week represents a very low demand period, the second week represents an average period, and the third week represents a high peak in demand. Historical data for the Realized Unconstrained demand was further grouped by car class, length of stay and arrival patterns using historical percentages. Summaries of the data gathered for one location are shown from Table 7 to Table 18 in units of customers. Information was collected from three different weeks with lower, average and high demand respectively. The summaries for the other two locations are given in Appendix A.

In addition to these data, information regarding an acceptable decision making time frame was established. Generally the short-term assignment of fleet does not exceed three hours. If it takes longer, it is considered extremely inefficient and useless. The corporate fleet analysts and the rental station managers have to constantly plan ahead for the short, medium, and long-term horizons. Their decisions vary in range from one day to one year. They plan and review past decisions and all these need to be done in a speedy fashion. The rental station manager can not afford more than half of his/her daily hours to convey the short time planning. He or she has to deal with the supervision of the daily activities at the rental station: the production and rental side, the personnel administration, etc. The corporate fleet analysts, on the other hand, are also concerned about additions and deletions of fleet and can not devote more than half of their days to short term planning.

Time period	Realized Constrained Demand R_{lt}	No Shows N_{lt}	Cancellations C_{lt}	Total Constrained Demand (TC_{lt})	Turn-downs T_{lt}	Total Unconstrained Demand TU_{lt}
Sun	136	42	25	203	22	225
Mon	146	47	29	222	33	255
Tue	139	44	19	202	32	234
Wed	87	19	13	119	19	138
Thu	96	18	16	130	36	166
Fri	129	35	18	182	70	252
Sat	200	58	23	281	70	351

Table 7: Demand for Miami by day of the week – First week.

Time period	0-2 am	2-4 am	4-6 am	6-8 am	8-10 am	10-12pm	12-2 pm	2-4 pm	4-6 pm	6-8 pm	8-10 pm	10-12 pm
Sun	0.04	0.04	0.01	0.04	0.04	0.09	0.09	0.12	0.32	0.1	0.06	0.05
Mon	0.03	0.01	0.03	0.05	0.06	0.08	0.08	0.13	0.17	0.19	0.13	0.04
Tue	0.03	0.01	0.03	0.05	0.06	0.08	0.08	0.13	0.17	0.19	0.13	0.04
Wed	0.01	0	0.05	0	0.03	0.05	0.05	0.19	0.42	0.11	0.04	0.05
Thu	0.02	0	0.01	0.03	0.06	0.12	0.06	0.21	0.24	0.13	0.1	0.02
Fri	0.01	0.01	0	0.04	0.03	0.03	0.13	0.15	0.26	0.21	0.08	0.05
Sat	0.01	0.01	0.03	0.04	0.05	0.04	0.07	0.12	0.26	0.18	0.11	0.08

Table 8: Demand for Miami – Arrival Pattern – First week.

Time period	No Show % η_{lt}	Cancellation % ς_{lt}	Realized Unconstrained Demand RU_{lt}	1 day LOS %	3 days LOS %
Sun	0.21	0.12	151	0.02	0.98
Mon	0.21	0.13	168	0.01	0.99
Tue	0.22	0.09	161	0.01	0.99
Wed	0.16	0.11	101	0.01	0.99
Thu	0.14	0.12	123	0.04	0.96
Fri	0.19	0.10	179	0.01	0.99
Sat	0.21	0.08	250	0.01	0.99

Table 9: Demand for Miami – Percentages - First week.

Time period	RU demand for Car class 1 & 1 day Los	RU demand for Car class 1 & 3 days Los	RU demand for Car class 2 & 1 day Los	RU demand for Car class 2 & 3 day Los
Sun	2	74	2	74
Mon	1	76	1	90
Tue	1	84	1	75
Wed	0	44	1	56
Thu	2	48	3	70
Fri	1	83	1	94
Sat	1	81	2	165

Table 10: Demand for Miami by Car class and Length of Stay – First week.

Time period	Realized Constrained Demand (Rentals) R_{lt}	No Shows N_{lt}	Cancellations C_{lt}	Total Constrained Demand (Reservations) (TC_{lt})	Turndowns T_{lt}	Total Unconstrained Demand TU_{lt}
Sun	512	160	100	772	85	857
Mon	496	185	96	777	74	851
Tue	382	167	75	624	55	679
Wed	212	95	42	349	53	402
Thu	295	78	46	419	97	516
Fri	333	142	64	539	133	672
Sat	525	270	126	921	167	1088

Table 11: Demand for Miami by day of the week – Second week.

Time period	0-2 am	2-4 am	4-6 am	6-8 am	8-10 am	10-12pm	12-2 pm	2-4 pm	4-6 pm	6-8 pm	8-10 pm	10-12 pm
Sun	0.02	0.01	0	0.06	0.08	0.1	0.14	0.15	0.19	0.1	0.09	0.06
Mon	0.05	0	0.02	0.04	0.04	0.13	0.12	0.13	0.16	0.08	0.09	0.14
Tue	0.02	0.01	0.01	0.03	0.05	0.14	0.15	0.14	0.14	0.15	0.09	0.07
Wed	0.02	0	0.02	0.05	0.07	0.08	0.1	0.25	0.15	0.11	0.06	0.09
Thu	0.02	0	0.03	0.03	0.09	0.05	0.14	0.15	0.2	0.14	0.07	0.08
Fri	0.02	0.01	0.02	0.04	0.12	0.06	0.09	0.11	0.21	0.15	0.11	0.06
Sat	0.03	0.01	0.03	0.03	0.06	0.1	0.09	0.12	0.19	0.17	0.11	0.06

Table 12: Demand for Miami – Arrival Pattern - Second Week.

Time period	No Show % η_{lt}	Cancellation % ς_{lt}	Realized Unconstrained Demand RU_{lt}	One day length of stay %	Three days length of stay %
Sun	0.21	0.13	568.10	0.34	0.66
Mon	0.24	0.12	543.36	0.38	0.62
Tue	0.27	0.12	415.55	0.48	0.52
Wed	0.27	0.12	244.33	0.02	0.98
Thu	0.19	0.11	362.90	0.01	0.99
Fri	0.26	0.12	415.46	0.02	0.98
Sat	0.29	0.14	620.19	0.20	0.80

Table 13: Demand for Miami – Percentages – Second Week.

Time period	RU demand for Car class 1 & 1 day Los	RU demand for Car class 1 & 3 days Los	RU demand for Car class 2 & 1 day Los	RU demand for Car class 2 & 3 day Los
Sun	83	160	110	213
Mon	100	162	108	175
Tue	101	110	97	106
Wed	2	111	3	129
Thu	1	136	2	222
Fri	4	179	5	229
Sat	56	223	68	273

Table 14: Demand for Miami by Car class and Length of Stay – Second Week.

Time period	Realized Constrained Demand (Rentals) R_{lt}	No Shows N_{lt}	Cancellations C_{lt}	Total Constrained Demand (Reservations) (TC_{lt})	Turndowns T_{lt}	Total Unconstrained Demand TU_{lt}
Sun	853	316	158	1327	65	1392
Mon	756	308	175	1239	61	1300
Tue	661	244	126	1031	79	1110
Wed	704	233	186	1123	117	1240
Thu	812	345	175	1332	277	1609
Fri	882	455	227	1564	516	2080
Sat	852	366	204	1422	286	1708

Table 15: Demand for Miami by day of the week – Third Week.

Time period	0-2 am	2-4 am	4-6 am	6-8 am	8-10 am	10-12pm	12-2 pm	2-4 pm	4-6 pm	6-8 pm	8-10 pm	10-12 pm
Sun	0.01	0.01	0.02	0.06	0.06	0.11	0.15	0.11	0.19	0.11	0.1	0.07
Mon	0.04	0.02	0.02	0.02	0.06	0.14	0.14	0.13	0.17	0.12	0.05	0.09
Tue	0.04	0.02	0.02	0.02	0.06	0.14	0.14	0.13	0.17	0.12	0.05	0.09
Wed	0.04	0	0.03	0.05	0.05	0.13	0.13	0.12	0.17	0.12	0.07	0.09
Thu	0.02	0.01	0.03	0.04	0.05	0.13	0.16	0.12	0.13	0.14	0.08	0.09
Fri	0.05	0.02	0.02	0.04	0.06	0.13	0.13	0.13	0.16	0.09	0.08	0.09
Sat	0.04	0.01	0.03	0.04	0.05	0.12	0.13	0.12	0.19	0.14	0.07	0.06

Table 16: Demand for Miami – Arrival Patterns – Third week.

Time period	No Show % η_{lt}	Cancellation % ς_{lt}	Realized Unconstrained Demand RU_{lt}	One day length of stay %	Three days length of stay %
Sun	0.24	0.12	894.60	0.62	0.38
Mon	0.25	0.14	793.21	0.69	0.31
Tue	0.24	0.12	711.56	0.70	0.30
Wed	0.21	0.17	776.54	0.50	0.50
Thu	0.26	0.13	980.97	0.63	0.37
Fri	0.29	0.15	1170.96	0.62	0.38
Sat	0.26	0.14	1023.60	0.54	0.46

Table 17: Demand for Miami – Percentages – Third week.

Time period	RU demand for Car class 1 & 1 day Los	RU demand for Car class 1 & 3 days Los	RU demand for Car class 2 & 1 day Los	RU demand for Car class 2 & 3 day Los
Sun	260	159	293	179
Mon	274	123	274	123
Tue	258	111	239	102
Wed	185	185	200	200
Thu	278	163	340	200
Fri	340	208	383	234
Sat	271	231	282	241

Table 18: Demand for Miami by Car class and Length of Stay – Third week.

4.2 Heuristic Implementation - Optimization

The optimization model of the heuristic was conceived as a deterministic, linear model. The decision of going deterministic was based on the fact that an average demand

behavior can be estimated for the system and used to obtain a starting point in the evaluation of assignment and booking limit alternatives. On the other hand, the decision to go linear was based on the fact that rental revenue is a linear function of the price and the number of rentals and because the movement cost can be estimated as a linear function. Rental revenue is the product of the price of the rental, the length of stay and the number of customers that rent a car. Cost of movement varies marginally, but it can be estimated with averages.

Let:

l = Source location. $\{1..nl\}$ nl = Total number of locations.

r = Target location. $\{1..nl\}$ c = Reserved car class. $\{1..nc\}$

g = Rented car class. $\{1..ng\}$ nc = Total number of car classes.

a = Auxiliary car class index. ns = Total number of LOS.

s = Length of stays, LOS. $\{1..ns\}$ nt = Total number of time periods.

t = Time period for rentals $\{1..nt\}$

p = Time period for the movement of cars. $\{1..np\}$

np = Total number of time periods for car movements.

e = Auxiliary time-period index.

P_{lcst} = Rental price for location l, car class c, LOS s, and time t.

C_{lr} = Cost to move a car from location l to location r.

RU_{lcst} = Realized Unconstrained demand by location l, car class c, LOS s and time t.

ORC_{lcgst} = Optimal Realized Constrained demand by location l, car reserved c, car rented g, LOS s and time t.

ORC_{lcst} = Total ORC by car reserved c. $ORC_{lcst} = \sum_{g=c}^{nc} ORC_{lcgst}$

X_{lrgp} = Number of cars moved from location l to location r, by car class g, and time p.

$F_{\lg t}$ = Total Fleet by location l, car class g, and time t. It is the fleet at beginning of time t, before rentals and returns are processed.

$B_{\lg st}$ = Projected Returns from old rentals by location l, car class g, LOS s and time t.

$AD_{\lg t}$ = Fleet net additions and deletions for location l, car class g and time t.

Leading to

$$\text{Max } Z = \sum_{t=l}^{nl}\sum_{c=1}^{nc}\sum_{s=1}^{ns}\sum_{t=1}^{nt} P_{lcst} s \sum_{g=c}^{nc} ORC_{lcgst} - \sum_{l=1}^{nl}\sum_{\substack{r=1 \\ r\neq l}}^{nr}\sum_{g=1}^{ng}\sum_{p=1}^{np} C_{lr} X_{lrgp}$$

Subject to:

$$ORC_{lcst} \le RU_{lcst} \quad \forall l,c,s,t$$

$$\sum_{s=1}^{ns} ORC_{lcst} \le \sum_{g=c}^{nc} F_{\lg t} \quad \forall l,c,s,t$$

Where,

$$F_{\lg t} = F_{\lg 1} + \sum_{\substack{r=1 \\ r\neq l}}^{nr}\sum_{\substack{p=1 \\ p\le t}}^{np} X_{r\lg p} - \sum_{\substack{r=1 \\ r\neq l}}^{nr}\sum_{\substack{p=1 \\ p\le t}}^{t} X_{lrgp} - \sum_{c=1}^{g}\sum_{s=1}^{ns}\sum_{e=1}^{t} ORC_{lcgse} + \sum_{c=1}^{g}\sum_{\substack{s=1 \\ s\le t-e}}^{ns}\sum_{e=1}^{t} ORC_{lcgsc}$$
$$+ \sum_{s=1}^{ns}\sum_{c=1}^{t} B_{\lg se} + \sum_{e=1}^{t} AD_{\lg e}$$

$\forall ORC_{lcgst}$ and $\forall X_{lrgp}$ are integers

$\forall ORC_{lcgst} \geq 0$ and $\forall X_{lrgp} \geq 0$

The objective is to maximize the rental revenue of the movement cost. The rental revenue is the product of the rental price (P), length of stay (S) and optimal realized constrained demand. The movement cost is the product of the unit cost of movement (C) and the number of cars moved for each possible combination (X).

Additional revenue and costs exist in this process; however, this study concentrates only in the dynamic of booking controls and car transfers, and how these two elements interact to generate more profit. Among the additional costs, there are cost of idle resources, when demand is not met or overstated, and the opportunity cost from lost customers. Among the additional revenue, there is the one from additional products sold like baby seats, sky racks, etc.

There are 4 constraints in the Linear Programming formulation. The first one is a demand constraint, the second is a supply constrain, the fourth states that the optimization variables *ORC* and *X* are integers and the fifth constrain states they are non-negative.

The Demand constraint states that Optimal Realized Constrained demand (ORC) is less or equal to the Total Unconstrained demand (TU). The number of renters in each combination of location, car classes, LOS and time period can not exceed the total demand for that combination.

The Supply constraint states that ORC at each time period can not exceed the available fleet (F) at that period. The available fleet at each time period *t,* in a given location

l, is a function of the initial fleet at time period 1 and all the different decreases and increments in fleet during all the rest of the time periods previous to t. Decreases in fleet are caused by fleet movements from location l to any other location, by rentals from previous time periods and, by planned fleet deletions (down fleet). Increases in fleet in a given location at a given time period are caused, by movements of fleet to that location, by returns from rentals made in the previous time periods or any other rentals made outside the planning horizon and from planned fleet additions (in fleet). For the prototype, the formulation has been simplified dropping the additions and deletions to the fleet and the returns from rentals outside the planning horizon.

In this LP formulation, the number of variables and constraints varies by the problem size. As more locations, cars, and days are considered the model grows geometrically. Table 19 and Table 20 show the total number of maximization variables for different sizes of systems, and from Table 21 to Table 23 the total number of constraints for them are shown.

Let

$$Demand\ variables = nl \times ns \times \left[\frac{nc \times (nc+1)}{2}\right] \times nt \tag{7}$$

$$Movement\ variables = np \times nl \times (nl-1) \times nc \tag{8}$$

$$Total\ variables = Demand\ variables + Movement\ variables \tag{9}$$

$$\text{Demand constraints} = nl \times nc \times ns \times nt \tag{10}$$

$$\text{Supply constraints} = nl \times nc \times nt \tag{11}$$

$$\text{Total non-negativity constraints} = \text{Total variables} \tag{12}$$

System Size	Location *(l)*	Car Class *(c)*	Car combinations	LOS *(s)*	Time *(t)*	Demand Variables
1	2	2	3	2	4	48
2	3	2	3	2	7	126
3	3	3	6	8	14	2016
4	5	5	15	8	14	8400

Table 19: Demand variables in Objective function.

System Size	Location *(l)*	Car class *(c)*	Period *(p)*	Movement variables *(X)*	Total variables
1	2	2	1	4	52
2	3	2	1	12	138
3	3	3	2	36	2052
4	5	5	2	200	8600

Table 20: Movement and Total variables in Objective function.

System Size	Location *(l)*	Car class *(c)*	Length of stay *(s)*	Time *(t)*	Demand constraints
1	2	2	2	4	32
2	3	2	2	7	84
3	3	3	8	14	1008
4	5	5	8	14	2800

Table 21: Demand Constraints.

System Size	Location *(l)*	Car class *(c)*	Time *(t)*	Supply Constraints
1	2	2	4	16
2	3	2	7	42
3	3	3	14	126
4	5	5	14	350

Table 22: Supply Constraints.

System Size	Non negativity constraints	TOTAL Constraints
1	52	100
2	138	264
3	2052	3186
4	8600	11750

Table 23: Non-negativity and Total Constraints.

The Optimization Program was implemented in Lingo. A Lingo model has two sections: the model formulation and the data section. The model formulation is independent of the problem-size, whereas the data section depends on it.

In the Lingo model, eleven entities, called sets, are defined to represent the linear programming formulation. Four of them are primitives and the other seven are derived from them. The primitive sets are the locations, car classes, length of stay (LOS), time and the initial fleet at each location. The derived sets represent the fleet status across the different time periods, the different upgrades, the movement elements, the Realized unconstrained demand and the optimal constrained demand (renters). Table 24 shows the parallel between LP elements and Lingo constructs.

As formulated in LP, the objective function has two major factors: the revenue from rentals and the cost of movements. The rental revenue is represented by the RENTER attribute in the RENTAL set, multiplied by the rental days in DAYS and the price in PRICE. The movement cost is represented by the MOVES attribute in set MOVEMENT multiplied by the cost in COST. Rentals and Moves are defined in Lingo as integers, using the @GIN function. Renter (i,j,k,l) represents the number of renters for each combination of locations, car class, LOS, and time period; i.e. is the Optimal Realized Constrained demand (ORC). Table 25 shows the Lingo formulation.

The Constraints are modeled directly from the mathematical formulation. The demand constraint states that ORC can be less or equal the total number of RU. Customers (i,j,k,l) represents RU for the different combinations of location, car class, LOS and, time period; Renter (i,m,k,l) is the ORC for a particular demand type satisfied with a particular upgrade combination. The constraint has certain condition for the upgrade. ORIGCAR (m) means that the renter is considered to satisfy a demand for a particular car class if it check out the same car class or any higher in an upgrade.

LP Elements		Lingo Constructs	
Components	Variables and indexes	Primitive sets & attributes	Derived sets & attributes
Location	l	LOCATION	
Car Class	c or g	CAR_CLASS : hierarchy	
LOS	s	LOS: days	
Time period Total planning horizon	t nt	TIME NP	
Initial Fleet Daily Fleet Upgrades	F at time 1 F at time t c, g	CARS:inifleet	 DAILYFLEET: dayfleet CAR_ASSIGNED: origcar, upgcar
Movement Cost Number of Movements	C X and l, r,c,p		TRANSFER: cost MOVEMENT: moves
RU demand Rental price	RU P		DEMAND: customers DEMAND: price
Optimal RC demand	ORC		RENTAL: renter

Table 24: Parallel between LP and Lingo constructs.

Model Component	Lingo Code
Objective function	[OBJECTIVE] MAX = @SUM (RENTAL(I,J,K,L): RENTER (I,J,K,L) * DAYS(K) * PRICE(I,ORIGCAR(J),K,L)) - @SUM (MOVEMENT(M, N, O): MOVES (M,N,O) * COST(M,N));
Demand Constraint	@FOR(DEMAND (I,J,K,L) : [SATISFIED_DEMAND] CUSTOMERS (I,J,K,L)>= @SUM(RENTAL (I,M,K,L) | ORIGCAR(M) #EQ#J:RENTER(I,M,K,L)));
Supply Constraint	@FOR (DAILYFLEET (I,J,T) : [USED_FLEET] DAYFLEET (I,J,T) >= @SUM(RENTAL (I,M,K,T) | UPGCAR(M) #EQ# J: RENTER (I,M,K,T))); @FOR (DAILYFLEET (I,J,T) : DAYFLEET (I,J,T) = INIFLEET (I,J) + @SUM(MOVEMENT(M,I,J) : MOVES(M,I,J)) - @SUM(MOVEMENT(I,M,J) : MOVES(I,M,J)) - @SUM(RENTAL(I,M,K,P) | UPGCAR(M) #EQ# J #AND# P #LT# T: RENTER (I,M,K,P)) + @SUM(RENTAL(I,M,K,P) | UPGCAR(M) #EQ# J #AND# P #LT# T #AND# DAYS(K) #LE# (T-P) : RENTER(I,M,K,P)));
Integer Constraint	@FOR(RENTAL (I,J,K,L): @GIN(RENTER(I,J,K,L))); @FOR(MOVEMENT (I,J,K): @GIN(MOVES(I,J,K)));
Non-negativity constraint	Default in Lingo

Table 25: Lingo Optimization Model.

The Supply constraints state that the total number of rentals in each day can not be greater than the cars available on that day. The cars available on that day are the cars that idle or are returning from previous rentals. The supply constraints consider the complete fleet flow across the time horizon from the very first day. Dayfleet(i,j,t) represents the available fleet for a particular location, car class and time period. Renter (i,m,k,t) represents rentals in a given location, upgrade, LOS and time period. The condition #eq# j means that the rental took car class j; it may be that it was reserved for car j or it was upgraded to it. The complete code of the Lingo model and example of a data file to feed the model is given in Appendix B.

The optimization model runs generating feasible solutions for the fixed set of input parameters. It runs in approximately 10 seconds, using a Pentium II Processor with 450 megahertz. The maximum number of iterations is around 236. It provides with a set of optimal values for renters and for moves. These values are entered to the next step in the heuristic. The renters are the booking limits or realized constrained demand and the moves are the scheduled car transfers in the simulation.

4.3 Heuristic Implementation - Simulation

The simulation model is the one that handles the stochastic behavior of the Fleet system by modeling the probabilistic nature of arrival patterns and fleet breakdown. It receives the initial set of decisions generated by the Optimization component and experiments with it. The simulation model has eight processes: customer behavior, fleet flow, customer and car matching, check-out, check-in processes, fleet movement, fleet additions, and fleet deletions (Figure 4).

There are two main entities in the model: the customers and the cars. Customers are generated as directed by the booking limits set by the optimization model, whereas cars are established by historical data at the beginning of the simulation.

Customer behavior has been modeled using historical data, and it varies in the different experiments accordingly to each of the scenarios they represent. Customers that renege the system are also modeled with a tolerance limit of an hour because it is the average across industry.

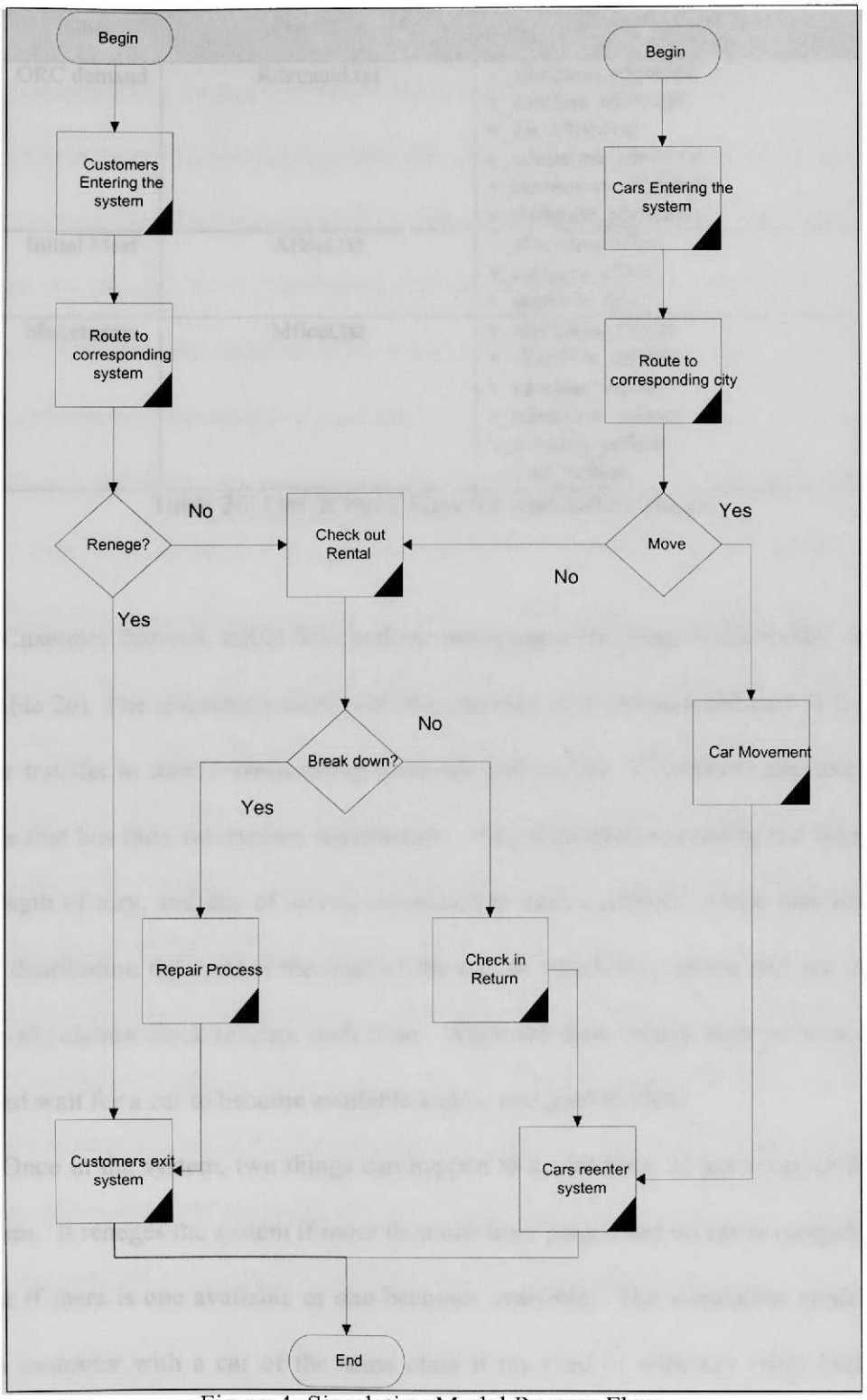

Figure 4: Simulation Model Process Flow.

Element	File name	Structure
ORC demand	Rdemand.txt	v_alocation_rdemand v_carclass_rdemand v_los_rdemand v_adatetime_rdemand v_customers_rdemand v_dailyrate_rdemand
Initial Fleet	Afleet.txt	v_alocation_afleet v_carclass_afleet v_quantity_aflee
Movements	Mfleet.txt	v_alocation_mfleet v_rlocation_mfleet v_carclass_mfleet v_adatetime_mfleet v_quantity_mfleet v_cost_mfleet

Table 26: List of Input Files for Simulation Model.

Customer demand, initial fleet and car movements are given to the model via ASCII files (Table 26). The simulation starts with the entrance of customers and cars to the system and their transfer to their corresponding locations and queues. Customers are read from an input file that has their reservation information. This information contains the location, car class, length of stay, and day of arrival associated to each customer. Once they are read, a discrete distribution determines the time of the day in which they arrive and are held back until the simulation clock reaches such time. When the time comes, they go to a detached queue and wait for a car to become available and be assigned to them.

Once in the system, two things can happen to a customer: 1) get a car, or 2) renege the system. It reneges the system if more than one hour passes and no car is assigned. It can get a car if there is one available or one becomes available. The simulation model tries to match a customer with a car of the same class it reserved or with any other higher class according with the upgrade hierarchy.

The cars are also read from an input file, but they are all created at time 0 and routed to their corresponding location. Once in their location's lot they are either rented, remain idle or they are moved to another location. They are moved if there is a movement for them in the movement file. The movement file is also read at the beginning of the simulation, and it contains the car class to be transferred, location source, target, quantity and time for the movement. Cars are also added or deleted from the system at certain periods of time. The addition and deletion is modeled but not used.

Once a customer and a car are matched, a rental takes place. The rental process or checkout event is modeled with a standard service time for rental agents. A rental can last the reserved length of stay, more, or less time. The simulation model uses a discrete distribution to determine the actual length of stay. A rental can finish successfully or abruptly if there is a car breakdown. Fleet breakdown is modeled using an average percent and assuming car breakdowns occur in the middle of the rental for simplicity. When the rental finishes, the customer exits the system and the car reenters it. In the case of a breakdown, the car will be repaired before it reenters the system.

The simulation model runs for 15000 minutes to gather the revenue from all rentals initiated during the planning horizon of the first 7 days (10080 minutes). It replicates 10 times, and it provides totals for net revenue. It interacts with a set of rules that suggest movement alternatives and evaluates each of them to find the best one. The complete listing of the model is given in Appendix C, and its input files in Appendix D.

4.4 Heuristic Implementation – Rules

The rules are a set of conditions that determine movement alternatives that may lead to better plans than those given by the optimization model. The rules prioritize the locations and car classes that need to be increased in fleet. They help evaluate the amount of business lost for the absence of the different car classes and try to allocate more fleet in locations with the highest amount of lost business.

They use several measures of performance and system parameters to determine the ranking of locations - car class combinations. These measures include daily revenue lost, daily number of customers lost, number of days with losses, and number of cars available. Daily revenue lost is the monetary value of lost reservations, and it comes from the Realized Constrained demand that could not get a car. The daily number of customers lost represents the loss of good will from customers that renege the system. The number of days with revenue or customers lost captures the impact of the absence of fleet across days. The number of available cars in each location-car class combination is the total initial fleet minus the movements suggested by the optimal model.

The rules follow a four-step process. The first step is the selection of locations that have the highest amount of lost business for each car class (In-need locations). The second is the selection of locations that have the least amount of lost business, and therefore, are good candidates to provide fleet to the In-need locations (Might-give locations). The third step is the establishment of allocation alternatives based on the amount of available fleet in Might-give locations and the amount of daily customers lost in In-need locations. The fourth step is the evaluation of each alternative (Table 27).

The possible alternatives are within the range of zero to an amount equal to the minimum between the available fleet in Might-give locations and the daily customers lost in In-need locations. Each alternative evaluates a different level of movement within this range for each car class. The prototype runs for three points in each range, generating a maximum of eight alternatives for the selected system size of three locations and two car classes. To use more than three a binary search is recommended.

In addition to the nomenclature given in Section 4.2, let:

Q_{lc} = Initial fleet for car class c at location l.

M_{lrc} = Number of cars of class c that were moved from location l to location r.

D_{lcst} = Demand in location l for car class c for length of stay s at time t.

P_{lcst} = Daily rate for car class c for length of stay s, at time t in location l.

R_{lcst} = Number of rentals in location l for car class c for length of stay s in time t.

A_{lc} = Total number of cars, class c or higher, available for movement in location l.

DVL_{lc} = Daily revenue lost for the absence of car class c in location l.

DCL_{lc} = Daily customer lost for the absence of car class c in location l.

DFL_{lc} = Daily financial lost for the absence of car class c in location l.

NDL_{lc} = Number of days with lost revenue in location l for car class c.

H_c = Location selected as in need of fleet for car class c. Called "In-need locations".

L_c = Ordered list of locations "with available fleet" for car class c (Descending order). They are called "Might-give locations".

1- Select In-need locations.

1.1 Determine available cars of each class at every location. The number of available cars at location l for class c, is the total initial fleet for class c minus all movements performed to any other location. For all l and c,

$$\text{If } Q_{lc} - \sum_{r=1}^{nl} M_{lrc} > 0 \text{ Then } A_{lc} = Q_{lc} - \sum_{r=1}^{nl} M_{lrc} \text{ Else } A_{lc} = 0$$

1.2 Determine DVL, DCL and NDL

$$DVL_{lc} = \frac{\sum_{s=1}^{ns}\sum_{l=1}^{nl}\sum_{g=c}^{1} s * P_{lcst} * (D_{\lg st} - R_{\lg st})}{nt} \quad \text{and} \quad DCL_{lc} = \frac{\sum_{s=1}^{ns}\sum_{t=1}^{nt}\sum_{g=c}^{1} (D_{lcst} - R_{lcst})}{nt}$$

NDL_{lc} = Sum of days in which $DVL_{lc} > 0$ and/or $DCL_{lc} > 0$.

1.3 Determine one location in need of fleet for each car class H_c. Select the location with $Max(DVL_{lc})$. Solve ties by selecting the one with $Max(DCL_{lc})$; if necessary select the one that also has $Max(NDL_{lc})$. Finally, if there are still ties, chose one location randomly.

2- Determine Might-give locations.

2.1 Determine a ranking of locations with available fleet for each car class L_c. Exclude locations that are H_c, and locations with $A_{lc} = 0$.

2.2 Rank locations in ascending order of DVL_{lc}. If none is selected, no better alternative can be generated to improve the performance of a particular location and car class H_c. If there are ties, break them by evaluating DCL_{lc}. If necessary evaluate the total days with lost revenue. Finally, if there are still ties, chose one randomly.

3- Generate different car movement alternatives for each car class.

3.1 Generate different car movement alternatives for each car class. For each car class c starting from the lowest to the highest, determine movement alternatives using all might-give locations L_c. The range of search is either A_{lc} in L_c, or DCL_{lc} in H_c whichever is less.

$$Min(Q_{lc} - \sum_{\substack{r=1 \\ r \neq l}}^{nl} M_{lrc}, DCL_{l^1 c})$$ Where l is might-give location in L_c and l^1 is in-need location H_c.

4- Evaluate each alternative with the simulation model and determine the best one using the net revenue each one produces as the measure of performance.

Table 27: Heuristic Rules Process.

4.5 Heuristic Implementation – Automation

The intense computational operations in the heuristic require the full automation of it, if it is expected that the heuristic will become a useful tool for the decisions makers.

However, the main objective of this effort was to identify what would it take to have a heuristic of this nature. Namely, the effort sought to identify critical components of it, and to demonstrate that such heuristic would lead to a feasible and useful tool. This goal has been met by partially automating the proposed heuristic using Lingo, Arena, Visual Basic for Applications (VBA) for Arena, and VBA for Excel. There are five main components to the heuristic:

1) Optimization,
2) The interface between the optimization and simulation,
3) The Simulation,
4) The Rules, and
5) The interface between the user and the heuristic.

The optimization code is implemented in Lingo 5.0. It receives an input file and generates another one that contains a set of movement alternatives and the realized constrained demand. The interface between optimization and simulation is implemented with VBA for Excel. It processes the Lingo 5.0 output file, reformats it, and distributes its information in two separate files: one for the movements and the other one for the realized constrained demand. The simulation model is implemented in Arena 3.0. It uses several input files and simulates the system for a little over a week reporting the net total revenue at the end. The Rules are implemented in VBA for Arena, and they are executed in the Run End Replication, Run End Simulation and Run End events.

The interface between the user and the heuristic is in VBA for Arena. There are two Forms that function as interface: The Main form (Figure 5), and the Results form (Figure 6).

The Main form is for input, and it prompts the user for the number of replications, its "Run" button starts the simulation and rules. The Results form is displayed when the heuristic finds the best alternative; it shows the net revenue from each alternative and the best of all. Experimentation with the prototype showed that the maximum number of iterations is 8; however, the prototype will still work properly with less iterations.

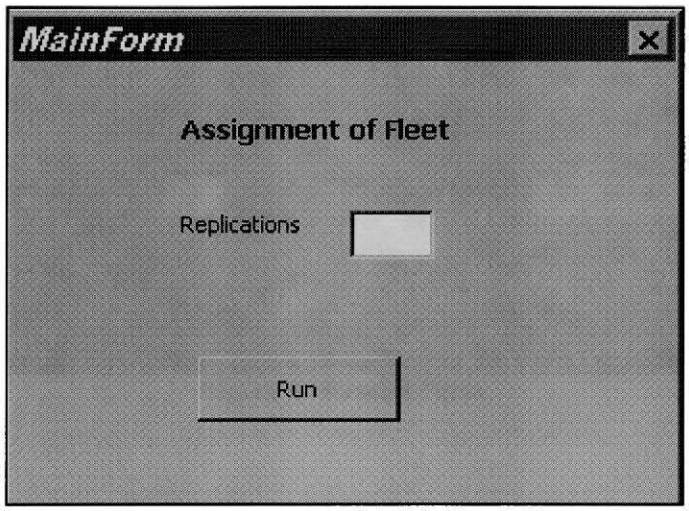

Figure 5: Main Form

Full automation can be achieved if the following four components are built:

1) An Interface between the heuristic and the operational systems of reservations, rentals and Inventory.
2) A direct feed to input demand projections from forecasting applications to the heuristic.
3) A complete connection between the optimization, simulation, rules and user interface modules.
4) A fully automated module to translate the optimization output into simulation input.

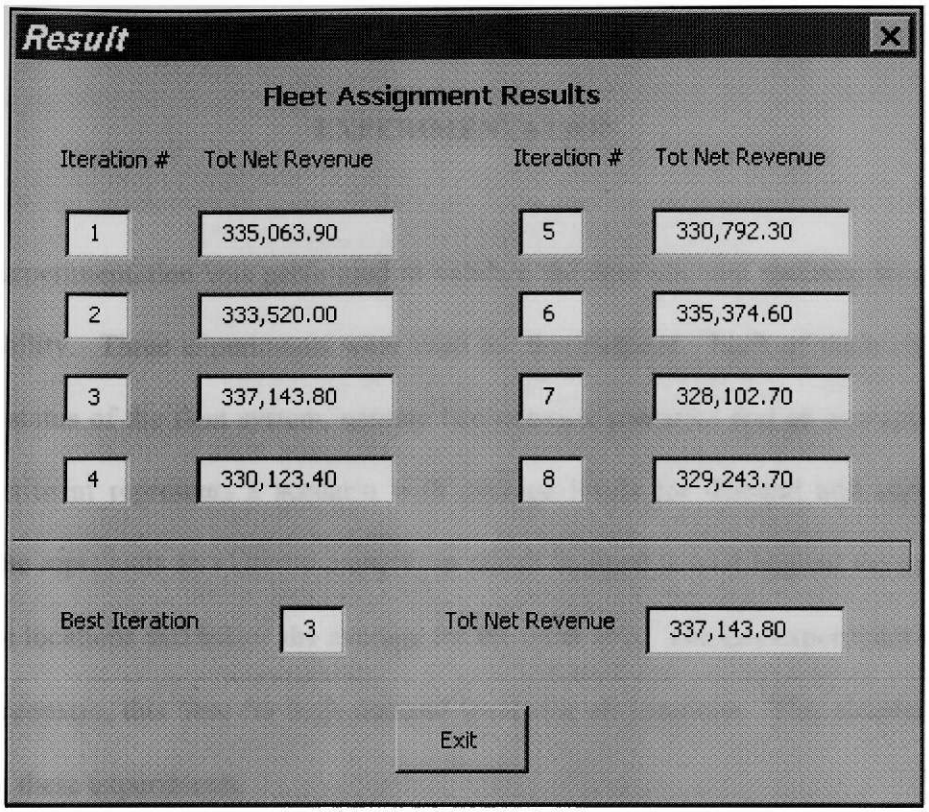

Figure 6: Results Form.

These four components can be built in different languages; however, it is recommended to build number three and four in a language that can communicate with the optimization and simulation modules. In the current implementation this can be achieved by incorporating Lingo in the VBA for Arena code.

One possibility that was not explored was the use of a language such as Visual Basic to serve as an umbrella under which the optimization tool and the simulation tool operate. This possibility is worth exploring if a commercial version of the prototype is to be developed.

CHAPTER 5:

EXPERIMENTATION

Experimentation was performed to validate the heuristic and measure its usefulness and reliability. Three experiments were used for this purpose. Each of them represents a different status of the fleet system, namely two extreme scenarios and an average one. The first experiment represents a scenario with average levels for demand and supply. The second one represents an extreme scenario in which demand is well beyond the average for one of the locations and below the average for the other two. The last experiment is another extreme scenario, this time for high demand levels for all locations. This chapter provides details on these experiments.

5.1 Description of the Experiments

The level of usability for the heuristic resides in its capability to appropriately react at the different situations that may occur in real life. By testing it in an average situation as well as in extreme ones, it was possible to assess its validity and scope. All three experiments represent the same problem size, but they have different settings with respect to fleet capacity and the demand to be satisfied.

Table 28 provides the values for parameters that are common to all three experiments. Some of the parameters that require further clarification are location, car classes, and LOS. There are three locations, namely Miami, Fort Lauderdale, and Key West. Two car classes are considered: Class 1 and Class 2. There are two levels of LOS: one day

and three days. Table 29 gives the initial fleet for each location-car class combination. The lists of prices common to all three experiments are shown in Table 30 to Table 32.

Parameters	Values
Locations (rental stations)	3
Number of car classes	2
Length of stay (LOS)	2
Length of planning horizon	7 days
Number of movements in planning horizon	1
Average rental time	Normal(15,5)
Customer waiting time limit	60 minutes
Car breakdown percents	Weibull 0.10
Repair time	Normal(1440,120)
Car movements	Generated
Booking limits	Generated

Table 28: Parameters for Experiments.

Locations	Car Class 1	Car Class 2	Addition Car Class 1	Addition Car Class 2
Miami	150	497	500	500
Fort Lauderdale	293	822	300	300
Key West	29	29		

Table 29: Initial Fleet and Planned Additions

Car Class	Day of the week	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Economic	One day	28	32	25	28	28	29	26
	Three day	22	21	21	23	24	23	22
Medium	One day	46	37	33	37	38	43	39
	Three day	33	29	28	30	33	34	34

Table 30: Prices for Miami.

Car Class	Day of the week	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Economic	One day	26	22	25	26	27	26	21
	Three day	17	17	19	20	21	21	17
Medium	One day	33	32	30	35	35	40	29
	Three day	26	22	25	26	27	28	27

Table 31: Prices for Fort Lauderdale.

Car Class	Day of the week	Sun	Mon	Tue	Wed	Thu	Fri	Sat
Economic	One day	16	16	31	24	24	19	17
	Three day	23	14	16	13	29	14	15
Medium	One day	12	18	12	23	21	18	26
	Three day	20	7	13	22	25	19	25

Table 32: Prices for Key West.

For each experiment, the average net revenue obtained was compared against historical net revenue using the following hypothesis:

$$H_o : \mu_E \leq \mu_0$$

$$H_a : \mu_E > \mu_o$$

Where,

μ_0 = Historical average net revenue.

μ_E = Average net revenue from the heuristic.

The significance level of $\alpha = 0.05$ was the same for all experiments. Because the experiments yielded ten observations each, the actual tests were done using the t test with 9 degrees of freedom.

To obtain the values of μ_0 for each experiment (Table 33), the actual pricing and rental data was masked and grouped accordingly with the selected system size of three locations, two car classes, and two length of stays. The demand for the different scenarios was projected using historical trends in a triangular distribution. Its values are given in the sections that discuss each experiment. However, the process to establish these values was the same for all experiments. To estimate the demand, it is worth noting that making short-term decisions in this industry requires the handling and analysis of large amounts of data; and obtaining summaries is just part of the data needed; projections of RU demand to account for future behavior are also required. For this effort, the projections are estimated using a triangular probability function, in conduction with average historical percentages for the different demand groups.

Experiment	Scenario	Total Net Revenue (μ_0)
I	Average demand for all three locations	$ 280,335
II	High demand for one location, low demand for the rest	$ 323,576
III	High demand for all three locations.	$ 476,767

Table 33: Historical Net Revenue.

There are four major steps in the projection process as shown in Table 34. A triangular distribution was used to generate the Total Unconstrained demand, (TC) by location and time period. Its parameters are based in the historical low, average, and high values for TU. The mode of the distributions is the historical average; the lower and upper bounds are derived from the historical low and high levels of TU. The random variables are obtained around the three parameters within comparable probabilistic regions of about 5% to

7%. Random variables around the lower bound are observations for the low demand scenario, random variable around the mode are observations for the average scenario and finally, random variables around the upper bound are observations for the high demand scenario. The projection is done once and it was coded in VBA for Arena. Appendix E contains the different code components.

1	Generate TU for each day of the week using an appropriate distribution.
2	Obtain RU applying no-show and cancellation percentages to TU
3	Group RU by car class and length of stay
4	Apply arrival time and over stay percentages.

Table 34: Steps to obtain Projections of TU.

Realized Unconstrained demand is obtained applying the appropriate no-show and cancellation percentages to TU. Note that no-show and cancellation percentages vary by location and car class (see Table 7 to Table 18). Car class and length of stay historical percentages are applied to RU. Finally, step 4 distributes RU by time of day and determines breakdowns using another set of percentages. The percentages for arrival patterns are the historical ones while the ones for breakdowns are projections (Table 28).

5.2 Experiment I

Experiment I evaluates the heuristic in a scenario that has average levels of demand. The projected RU demands that it uses are shown in Table 35 to Table 37. The initial and final fleet movements are in Table 38. The net revenue obtained from each run and the average are shown in Table 39. Finally, Table 40 shows the historical and experimental total RC quantity.

Car Class	LOS	Run	Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	1	1	76	106	107	1	1	3	85
		2	76	106	107	1	1	3	84
		3	76	107	106	1	1	3	84
		4	76	106	106	1	1	3	85
		5	76	106	107	1	1	3	84
		6	76	106	107	1	1	3	84
		7	76	107	107	1	1	3	85
		8	76	106	106	1	1	3	84
		9	76	106	106	1	1	3	84
		10	76	106	107	1	1	3	85
	3	1	70	70	63	30	42	65	122
		2	71	71	63	32	40	63	121
		3	70	71	62	32	41	63	121
		4	70	71	62	31	42	61	122
		5	70	70	63	31	40	63	121
		6	71	71	63	31	40	64	121
		7	70	71	63	30	39	62	122
		8	70	71	62	32	42	61	121
		9	70	70	62	31	41	63	121
		10	71	71	63	31	40	64	122
2	1	1	41	76	63	1	1	4	64
		2	41	77	63	1	1	3	64
		3	41	77	62	1	1	3	63
		4	41	77	62	1	1	3	64
		5	41	76	63	1	1	3	64
		6	41	77	63	1	1	4	63
		7	41	77	63	1	1	3	64
		8	41	77	62	1	1	3	64
		9	41	76	62	1	1	3	63
		10	41	77	63	1	1	4	64
	3	1	38	51	37	29	33	86	92
		2	38	51	37	31	31	83	92
		3	38	52	37	31	32	84	91
		4	38	51	37	30	33	81	92
		5	38	51	37	30	31	84	92
		6	38	51	37	30	31	85	91
		7	38	52	37	29	31	82	92
		8	38	51	37	31	33	81	92
		9	38	51	37	30	32	84	91
		10	38	51	37	30	31	85	92

Table 35: Fort Lauderdale Average Demand.

Car Class	LOS	Run	Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	1	1	83	99	102	2	1	4	56
		2	83	99	101	2	1	4	56
		3	83	100	101	2	1	4	56
		4	83	100	101	2	1	4	56
		5	82	99	102	2	1	4	56
		6	83	100	101	2	1	4	56
		7	83	99	101	2	1	4	56
		8	83	99	101	2	1	4	56
		9	83	100	102	2	1	4	56
		10	82	99	101	2	1	4	56
	3	1	160	161	110	111	136	181	223
		2	161	162	109	109	135	178	224
		3	161	162	110	110	135	179	223
		4	161	162	110	111	137	180	222
		5	160	162	110	111	135	181	224
		6	160	163	109	109	136	178	223
		7	161	161	110	110	137	179	224
		8	161	162	110	111	135	180	223
		9	161	162	110	111	135	181	222
		10	160	162	109	110	136	178	224
2	1	1	109	107	98	3	2	5	68
		2	110	107	97	3	2	5	69
		3	110	108	97	3	2	5	68
		4	110	108	97	3	2	5	68
		5	109	107	98	3	2	5	68
		6	109	108	97	3	2	5	68
		7	110	107	97	3	2	5	69
		8	110	108	97	3	2	5	68
		9	110	108	98	3	2	5	68
		10	109	107	97	3	2	5	68
2	3	1	212	175	106	131	222	230	273
		2	213	175	105	128	220	227	274
		3	214	176	105	129	221	228	272
		4	213	176	106	130	223	229	272
		5	212	175	106	131	220	230	274
		6	212	176	105	128	222	227	273
		7	213	175	105	129	223	228	274
		8	214	176	106	130	220	229	272
		9	213	176	106	131	220	230	272
		10	212	175	105	129	222	227	274

Table 36: Miami Average Demand.

Car Class	LOS	Run	Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	1	1	2	3	8	0	0	0	3
		2	2	3	8	0	0	0	3
		3	2	3	8	0	0	0	3
		4	2	3	8	0	0	0	3
		5	2	3	8	0	0	0	3
		6	2	3	8	0	0	0	3
		7	2	3	8	0	0	0	3
		8	2	3	8	0	0	0	3
		9	2	3	8	0	0	0	3
		10	2	3	8	0	0	0	3
	3	1	1	2	0	1	1	3	2
		2	1	2	0	1	1	3	2
		3	1	2	0	1	1	3	2
		4	1	2	0	1	1	3	2
		5	1	2	0	1	1	3	2
		6	1	2	0	1	1	3	2
		7	1	2	0	1	1	3	2
		8	1	2	0	1	1	3	2
		9	1	2	0	1	1	3	2
		10	1	2	0	1	1	3	2
2	1	1	6	4	2	1	0	0	6
		2	6	4	2	1	0	0	6
		3	6	4	2	1	0	0	6
		4	6	4	2	1	0	0	6
		5	6	4	2	1	0	0	6
		6	6	4	2	1	0	0	6
		7	6	4	2	1	0	0	6
		8	6	4	2	1	0	0	6
		9	6	4	2	1	0	0	6
		10	6	4	2	1	0	0	6
	3	1	3	3	0	1	3	1	3
		2	3	3	0	1	3	1	3
		3	3	3	0	1	3	1	3
		4	3	3	0	1	3	1	3
		5	3	3	0	1	3	1	3
		6	3	3	0	1	3	1	3
		7	3	3	0	1	3	1	3
		8	3	3	0	1	3	1	3
		9	3	3	0	1	3	1	3
		10	3	3	0	1	3	1	3

Table 37: Key West Average Demand.

	Source Location	Target Location	Car Class	Time	Quantity	Cost
Initial	1	3	1	0.5	28	20
	1	3	2	0.5	39	20
Final	1	3	1	0.5	28	20
	1	3	2	0.5	39	20

Table 38: Initial and final fleet movement alternatives.

Run	Net Revenue
1	$293,076
2	$292,555
3	$292,151
4	$290,834
5	$292,661
6	$292,915
7	$291,835
8	$291,288
9	$291,737
10	$292,520
Average	$292,157
Standard Deviation	726.4

Table 39: Net Revenue from the Experiment

	location	Total RC quantity
Historical	Total	4100
Experiment	Total	4431

Table 40: Historical and Experimental Total Realized Constrained demand quantity.

The hypothesis to test is:

$$H_0 : T \leq \$280{,}335$$

$$H_a : T > \$280{,}335$$

The statistic for this test is T, and it is given by

$$T = \frac{\overline{X} - \mu_0}{s / \sqrt{n}}$$

The rejection region for this test is $t > 1.833$ since $t_{0.05,9} = 1.833$.

Using the numeric values from the experiment, $\hat{\mu} = \overline{X} = \$280{,}335$ and $\hat{\sigma} = s = 726.4$ and thus T is

$$T = \frac{(\$292{,}157.14 - \$280{,}335)}{\frac{726.4}{\sqrt{10}}} = 51.466$$

Since $T > 1.833$, the null hypothesis is rejected. Thus, there is sufficient evidence to suggest that the revenue from the plan generated by the heuristic is higher than the historical one.

5.3 Experiment II

Experiment II evaluates the heuristic in an extreme scenario of low demand for Fort Lauderdale and Key West and high demand for Miami. The projected RU demands that it uses are shown in Table 43 to Table 45. The initial and final fleet movement alternatives suggested by the optimization module are shown in Table 41 (the simulation selected the optimal one as the best one). The net revenue obtained from each run and the average are shown in Table 42. Finally, Table 46 shows the historical and experimental total RC quantity.

The hypothesis to test is:

$$H_0 : T \leq \$323{,}576$$

$$H_a : T > \$323{,}576$$

The rejection region for this test is $T > 1.833$ since $t_{0.05,9} = 1.833$.

Using the numeric values from the experiment, $\hat{\mu} = \overline{X} = \$323{,}576$, $\overline{\sigma} = s = 4531.812$ and thus T is

$$T = \frac{(\$336{,}465.27 - \$323{,}576)}{\frac{4531.812}{\sqrt{10}}} = 8.994$$

Since $T > 1.833$, the null hypothesis is rejected. Thus, there is sufficient evidence to suggest that the revenue from the plan generated by the heuristic is higher than the historical one.

	Source Location	Target Location	Car Class	Time	Quantity	Cost
Initial	1	3	2	0.5	191	20
	1	3	2	0.5	191	20
Final	2	3	1	0.5	14	30
	2	3	2	0.5	14	30

Table 41: Initial and Final Fleet movement alternatives.

Run	Net Revenue
1	340,103
2	328,324
3	338,773
4	337,949
5	341,832
6	332,069
7	340,280
8	330,669
9	337,462
10	337,192
Average	336,465
Standard Deviation	4531.812

Table 42: Net Revenue from Experiments

Car Class	LOS	Run	Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	1	1	0	1	1	0	1	0	0
		2	0	1	1	0	1	1	0
		3	0	1	1	0	1	1	0
		4	0	1	1	0	1	1	0
		5	0	1	0	0	1	0	0
		6	0	1	1	0	1	1	0
		7	0	1	1	0	1	1	0
		8	0	1	1	0	1	1	0
		9	0	1	1	0	1	0	0
		10	0	1	1	0	1	1	0
	3	1	39	44	25	12	20	15	51
		2	32	27	28	20	12	21	43
		3	35	50	34	16	16	28	60
		4	24	44	31	12	21	24	55
		5	39	27	18	19	14	14	51
		6	32	50	25	16	19	19	43
		7	35	44	28	13	21	27	60
		8	24	27	34	19	14	24	55
		9	39	30	31	16	19	14	51
		10	32	51	29	13	21	19	43
2	1	1	0	0	0	0	1	1	0
		2	0	0	1	0	0	1	0
		3	0	1	1	0	1	1	0
		4	0	0	1	0	1	1	0
		5	0	0	0	0	0	1	0
		6	0	1	0	0	1	1	0
		7	0	0	1	0	1	1	0
		8	0	0	1	0	0	1	0
		9	0	0	1	0	1	1	0
		10	0	1	1	0	1	1	0
	3	1	33	24	24	9	17	18	47
		2	27	15	27	15	11	26	40
		3	30	27	32	12	13	34	55
		4	21	24	30	9	18	29	50
		5	33	15	17	15	12	17	47
		6	27	27	24	12	16	23	40
		7	30	24	27	10	18	33	55
		8	21	15	32	15	12	29	50
		9	33	16	30	12	16	17	47
		10	27	27	28	10	18	23	40

Table 43: Fort Lauderdale - low Demand.

Car Class	LOS	Run	Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	1	1	269	279	259	184	272	349	272
		2	258	269	253	187	281	336	269
		3	262	274	268	181	277	331	275
		4	256	272	255	192	289	349	266
		5	269	271	263	184	272	336	272
		6	258	285	259	187	282	331	269
		7	262	274	253	181	288	349	275
		8	256	271	268	192	277	336	266
		9	269	285	255	181	272	331	277
		10	258	274	263	187	282	334	266
	3	1	165	125	111	184	160	214	232
		2	158	121	109	187	165	206	229
		3	161	123	115	181	163	203	235
		4	157	122	109	192	170	214	226
		5	165	122	113	184	160	206	232
		6	158	128	111	187	166	203	229
		7	161	123	109	181	169	214	235
		8	157	122	115	192	162	206	226
		9	165	128	109	181	160	203	236
		10	158	123	113	187	166	205	227
2	1	1	303	279	239	200	333	393	284
		2	291	269	234	202	343	379	280
		3	296	274	248	196	339	373	287
		4	289	272	236	208	353	393	276
		5	303	271	243	200	332	379	284
		6	291	285	239	202	345	373	280
		7	296	274	234	196	351	393	287
		8	289	271	248	208	338	379	276
		9	303	285	236	197	332	373	288
		10	291	274	243	203	345	377	277
	3	1	186	125	102	200	196	241	242
		2	178	121	100	202	202	232	238
		3	181	123	106	196	199	229	244
		4	177	122	101	208	207	241	235
		5	186	122	104	200	195	232	242
		6	178	128	102	202	203	229	238
		7	181	123	100	196	206	241	244
		8	177	122	106	208	198	232	235
		9	186	128	101	197	195	229	245
		10	178	123	104	203	203	231	236

Table 44: Miami high Demand.

Car Class	LOS	Run	Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	1	1	0	0	0	0	0	0	0
		2	0	0	1	0	0	0	0
		3	0	0	0	0	0	0	0
		4	0	0	0	0	0	0	0
		5	0	0	0	0	0	0	0
		6	0	0	1	0	0	0	0
		7	0	0	0	0	0	0	0
		8	0	0	0	0	0	0	0
		9	0	0	1	0	0	0	0
		10	0	0	0	0	0	0	0
	3	1	0	1	0	0	1	0	0
		2	0	2	1	0	0	0	0
		3	0	1	0	0	1	0	0
		4	0	1	0	0	1	0	0
		5	0	1	0	0	1	0	0
		6	0	2	1	0	1	0	0
		7	0	1	0	0	1	0	0
		8	0	1	0	0	1	0	0
		9	0	1	1	0	1	0	0
		10	0	1	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0
		2	0	0	1	0	0	0	0
		3	0	0	0	0	0	0	0
		4	0	0	0	0	0	0	0
		5	0	0	0	0	0	0	0
		6	0	0	1	0	0	0	0
		7	0	0	0	0	0	0	0
		8	0	0	0	0	0	0	0
		9	0	0	1	0	0	0	0
		10	0	0	0	0	0	0	0
	3	1	1	3	0	0	1	0	2
		2	2	3	1	0	0	0	2
		3	2	1	0	0	1	0	2
		4	1	3	0	0	1	0	0
		5	2	3	0	0	1	0	2
		6	2	3	1	0	1	0	2
		7	1	1	0	0	1	0	0
		8	2	3	0	0	1	0	2
		9	2	3	1	0	1	0	2
		10	1	2	0	0	0	0	0

Table 45: Key West - low Demand.

	location	Total RC quantity
Historical	Total	5832
Experiment	Total	6333

Table 46: Historical and Experimental Total Realized Constrained demand quantity.

5.4 Experiment III

Experiment III evaluates the heuristic in an extreme scenario of high demand for all the cities. The projected RU demand that it uses is shown from Table 50 to Table 52. The initial and final fleet movement alternatives suggested by the optimization model are shown in Table 47. The net revenue obtained from each run and the average are shown in Table 48. Finally, Table 49 shows the historical and experimental total RU demand quantity.

The hypothesis to test is:

$$H_0 : T \leq \$476{,}767$$

$$H_a : T > \$476{,}767$$

The rejection region for this test is $T < 1.833$ since $t_{0.05,9} = 1.833$.

Using the numeric values from the experiment, $\hat{\mu} = \overline{X} = \$485{,}043$, $\overline{\sigma} = s = 2988.639$ and thus T is

$$T = \frac{(\$485{,}043.41 - \$476{,}767)}{\frac{2988.639}{\sqrt{10}}} = 8.757$$

Since $T > 1.833$, the null hypothesis is rejected. Thus, there is sufficient evidence to suggest that the revenue from the plan generated by the heuristic is higher than the historical one.

	Source Location	Target Location	Car Class	Time	Quantity	Cost
Initial	1	3	1	0.5	87	20
	1	3	2	0.5	60	20
Final	1	3	2	0.5	191	20

Table 47: Initial and Final fleet movement alternatives.

Run	Net Revenue
1	$ 486,338
2	$ 480,478
3	$ 487,571
4	$ 486,133
5	$ 484,392
6	$ 485,243
7	$ 483,195
8	$ 480,438
9	$ 489,681
10	$ 486,966
Average	$ 485,043
Standard Deviation	2988.639

Table 48: Historical Net Revenue.

	location	Total RC quantity
Historical	Total	9951
Experiment	Total	10511

Table 49: Historical and Experimental Total Realized Constrained demand quantity.

Car Class	LOS	Run	Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	1	1	195	215	161	294	309	392	244
		2	199	222	157	302	315	407	258
		3	202	212	163	290	321	397	253
		4	192	215	159	294	310	414	248
		5	195	222	161	302	327	402	242
		6	199	212	157	290	306	388	244
		7	202	215	168	301	317	396	258
		8	192	213	157	289	310	414	253
		9	195	223	158	296	327	402	248
		10	199	216	164	301	306	388	242
	3	1	88	57	43	93	103	184	110
		2	89	59	42	95	105	192	116
		3	91	56	43	92	107	187	114
		4	86	57	42	93	103	195	111
		5	88	59	43	95	109	189	109
		6	89	56	42	92	102	183	110
		7	91	57	45	95	106	186	116
		8	86	57	42	91	103	195	114
		9	88	59	42	93	109	189	111
		10	89	57	43	95	102	183	109
2	1	1	120	207	149	283	321	347	208
		2	122	213	145	291	328	361	220
		3	124	204	150	279	334	352	215
		4	118	207	146	283	323	367	211
		5	120	213	149	291	340	356	206
		6	122	204	145	279	319	344	208
		7	124	207	155	289	330	351	220
		8	118	204	145	278	323	367	215
		9	120	214	146	284	340	356	211
		10	122	207	151	289	319	344	206
	3	1	54	55	39	89	107	163	93
		2	55	57	38	92	109	170	99
		3	56	54	40	88	111	166	97
		4	53	55	39	89	108	173	95
		5	54	57	39	92	113	168	93
		6	55	54	38	88	106	162	93
		7	56	55	41	91	110	165	99
		8	53	54	39	88	108	173	97
		9	54	57	39	90	113	168	95
		10	55	55	40	91	106	162	93

Table 50: Fort Lauderdale high Demand.

Car Class	LOS	Run	Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	1	1	269	279	259	184	272	349	272
		2	258	269	253	187	281	336	269
		3	262	274	268	181	277	331	275
		4	256	272	255	192	289	349	266
		5	269	271	263	184	272	336	272
		6	258	285	259	187	282	331	269
		7	262	274	253	181	288	349	275
		8	256	271	268	192	277	336	266
		9	269	285	255	181	272	331	277
		10	258	274	263	187	282	334	266
	3	1	165	125	111	184	160	214	232
		2	158	121	109	187	165	206	229
		3	161	123	115	181	163	203	235
		4	157	122	109	192	170	214	226
		5	165	122	113	184	160	206	232
		6	158	128	111	187	166	203	229
		7	161	123	109	181	169	214	235
		8	157	122	115	192	162	206	226
		9	165	128	109	181	160	203	236
		10	158	123	113	187	166	205	227
2	1	1	303	279	239	200	333	393	284
		2	291	269	234	202	343	379	280
		3	296	274	248	196	339	373	287
		4	289	272	236	208	353	393	276
		5	303	271	243	200	332	379	284
		6	291	285	239	202	345	373	280
		7	296	274	234	196	351	393	287
		8	289	271	248	208	338	379	276
		9	303	285	236	197	332	373	288
		10	291	274	243	203	345	377	277
	3	1	186	125	102	200	196	241	242
		2	178	121	100	202	202	232	238
		3	181	123	106	196	199	229	244
		4	177	122	101	208	207	241	235
		5	186	122	104	200	195	232	242
		6	178	128	102	202	203	229	238
		7	181	123	100	196	206	241	244
		8	177	122	106	208	198	232	235
		9	186	128	101	197	195	229	245
		10	178	123	104	203	203	231	236

Table 51: Miami high Demand.

Car Class	LOS	Run	Sun	Mon	Tue	Wed	Thu	Fri	Sat
1	1	1	11	14	9	18	20	21	12
		2	11	14	9	19	20	21	13
		3	11	15	10	18	21	20	12
		4	11	15	9	19	20	20	12
		5	11	14	9	18	21	21	13
		6	11	14	10	19	20	21	12
		7	11	15	9	18	21	20	12
		8	12	15	9	19	20	20	13
		9	11	15	10	18	20	21	12
		10	11	14	9	19	21	21	12
	3	1	3	2	3	4	3	7	2
		2	3	2	2	4	3	7	2
		3	3	2	3	3	3	7	2
		4	3	2	2	4	3	7	2
		5	3	2	3	4	3	7	2
		6	3	2	3	4	3	7	2
		7	3	2	2	3	3	7	2
		8	3	2	3	4	3	7	2
		9	3	2	3	4	3	7	2
		10	3	2	2	4	3	7	2
2	1	1	9	16	8	13	23	16	12
		2	9	15	8	14	22	15	12
		3	9	16	9	13	23	15	11
		4	9	16	8	14	22	15	11
		5	9	16	8	13	23	16	12
		6	9	15	9	14	22	15	11
		7	9	16	8	13	23	15	11
		8	9	16	8	14	23	15	12
		9	9	16	9	13	22	16	11
		10	9	16	8	14	24	15	11
	3	1	2	2	2	3	3	5	2
		2	2	2	2	3	3	5	2
		3	2	2	2	2	3	5	2
		4	2	2	2	3	3	5	2
		5	2	2	2	3	3	5	2
		6	2	2	2	3	3	5	2
		7	2	2	2	2	3	5	2
		8	2	2	2	3	3	5	2
		9	2	2	2	3	3	5	2
		10	2	2	2	3	3	5	2

Table 52: Key West high Demand.

5.5 Analysis and Discussion

The null hypothesis is rejected in all three experiments and the alternative hypothesis is accepted (Table 53). There is enough evidence to indicate that the heuristic generates higher returns for the company. The different movement alternatives and optimal constrained demand suggested by the heuristic produced higher net revenue than the current assignment and over booking processes.

In Experiment I, for the average scenario of demand, the heuristic best return's average was in the third run where alternative one was the highest, and the heuristic's more common iteration was the 6^{th} one. The suggestions generated by the heuristic make sense intuitively since it picked Fort Lauderdale as the car giver location. Fort Lauderdale produces less revenue than Miami. The realized constrained demand as a result of the suggested alternative is higher by 7.4% than the historical constrained demand in this type of scenario, this makes possible the acceptance of more reservations and, therefore, more opportunity for rentals.

In Experiment II for the extreme scenario in which Miami has a peak in demand and the other two do not, the heuristic's best return average was in the eight run, and the most common iteration was the sixth one. Its suggestions are also appropriate. It recommended the movement of almost half of the cars in Key West to Miami. Miami had not only a peak in demand but a high valued one; therefore, transferring cars from the low demand locations was a sound decision alternative, despite the high transfer cost of Key West. Additionally, its optimal constrained demand was higher by 7.9% than the historical one, leading to more reservations and rentals.

In Experiment III, for the extreme scenario in which all the cities peak in demand, the heuristic best return average was in the sixth run and its most common iteration for the best alternative was the fourth one. The heuristic was selective and mostly suggested movements from Fort Lauderdale to Miami. Key West had a peak in demand too and the cost of move its cars and the loss in rentals was not justified. The optimal constrained demand was higher by 5.3% than the historical process and again identified more business opportunities given the current resources.

	Experiment	Null Hypothesis	T statistic	$t_{0.05,9}$	Decision
$\alpha = 0.05$	I	$T \leq 280,335$	51.466	1.833	Reject
	II	$T \leq 323,576$	8.994	1.833	Reject
	III	$T \leq 476,767$	8.757	1.833	Reject
$\alpha = 0.005$	I	$T \leq 280,335$	51.466	1.38	Reject
	II	$T \leq 323,576$	8.994	1.38	Reject
	III	$T \leq 476,767$	8.757	1.38	Reject
$\alpha = 0.1$	I	$T \leq 280,335$	51.466	3.28	Reject
	II	$T \leq 323,576$	8.994	3.28	Reject
	III	$T \leq 476,767$	8.757	3.28	Reject

Table 53: Experiment Results.

CHAPTER 6:

SUMMARY

6.1 Contributions

Currently, the Car Rental Industry is in need of the appropriate tools for fleet allocation. Most of the time companies make projections based on historical trends and decide based on them. Some companies have used various types of simulation with minimal stochastic components. The present work designed a tool that searches for alternatives and tests them. It provides industry with a simulation based heuristic that gets good solutions that have passed successfully a thorough simulation phase and, therefore, have less chance for failure.

The heuristic has integrated an integer-programming model with simulation to reach its goal. From Integer-programming, the first set of alternatives are obtained, they constitute the initial point in its search for the best solution. Its optimization model differs from current ones being used in the car rental industry in its representation of the fleet and Realized Unconstrained demand (RU). Fleet is modeled as a dynamic element that changes in value across the different periods of time. Its initial value is the only fixed amount being considered. RU demand is kept variable. Common optimization models in used, model fleet with running amounts that vary by day and limit the possible variability of RU and RC (Realized Constrained demand) consequently. The heuristic's optimization formulation only uses the initial fleet. It seeks to reproduce the changing behavior of the fleet as the result of decisions and events that occurred in previous periods.

With Simulation, on the other hand, the heuristic has the mechanism to test each alternative. It is the appropriate tool for the search process because of its ability to represent the stochastic and dynamic behavior of the system. The model implemented in the prototype contains a representation of the arrival pattern and the fleet breakdown; it also has an interface to the rules. Information flows back and forth in the search for the best solution.

Finally, the heuristic's set of rules capture opportunities for better performance by observing the behavior of the system and identifying ranges for movement alternatives. In a static analysis not all possibilities for improvement can be recognized; it is within the operation of the system where more alternatives can be drawn.

To implement this work's heuristic in a production environment a company needs to accomplish the following four tasks:

1. Determine the appropriate system size.

   The number of cities, movements, LOS, car classes, and periods of time determine the system size. The number of cities can ultimately be adjusted by simulating different combinations using the heuristic. It is recommended that the number of LOS, car classes and periods of time are the same as in the real system.

2. Implement the heuristic for production scale.

   Expand limits set for system size, automate all components of the heuristic and its interfaces with demand projections, reservations, fleet inventory and others.

3. Determine parameters and input information for the heuristic.

   RU, cost, and distributions for breakdown and LOS.

4. Implement alternatives.

   To implement a decision, RC from the heuristic needs to be translated into RU. RU will constitute the booking limit for the company's reservation system. On the other hand, fleet movements need to be scheduled in the quantity and time determined in the heuristic.

In summary, the contributions of this work are:

- The design and prototyping of a simulation based heuristic for fleet assignment, at the operational level, in the Car Rental Industry.
- The use of a systemic approach in the heuristic's design.
- The integration of optimization, simulation and a set of rules.
- The development of an optimization model representing rentals as variables, and fleet across periods of time as function of previous periods.
- A simulation model that incorporates relevant stochastic elements and interfaces with rules to search for the best allocation of fleet.
- A set of rules of thumb that seek to identify opportunities for improvement beneath the dynamics of the system.

6.2 Extensions to this Effort

From the experiments, there is enough evidence that this heuristic is a good solution for the problem of assignment of fleet in the short term planning horizon. Its applicability in the car rental industry was prototyped and its performance indicated that it generates better

revenue returns. These positive results encourage further studies and its implementation in a production scale. Research may be done in the areas of:

- *Rules expansion*: To increase the rules' scope as system size grows, to implement more searching mechanisms, and to consider more alternatives.
- *Modeling of the number of cities as another variable*: The best value for the number of cities to pool can be a heuristic variable.
- *Modeling self-adjusting parameters for LOS, Arrival pattern, breakdown, etc.*: Parameters that get updated by statistical routines every time new key information becomes available.
- *Integration of tactical and strategic assignment of fleet with the heuristic*: The incorporation of fleet acquisitions, depreciation, returns, and tactical movement across regions.
- *Complete automation of the heuristic*: The development of additional code to integrate the optimization and simulation models (Lingo with Arena).
- *Creation of interfaces to operational systems*: The heuristic needs to be automatically linked to the operational systems through interfaces. Systems like Reservations, Rentals, Demand Projections, Fleet, Cost and Budgeting need to be connected to the heuristic and their information transmitted to it.
- *Incorporation of a module that interacts with the decision-makers*: A module that gets movement alternatives/booking limits from the users and evaluates and reports the revenue they produce. With this additional development, the heuristic will be a tool for what if analysis.

- *Statistical measurement of heuristic outcomes and their comparison against actual results*: This development would record the heuristic's performance and control its quality.

As we see from the above, the area of dynamic stochastic problems in the Car Rental Industry will benefit from more research and implementation of solutions. In a highly competitive market the correct use and deployment of resources will make the big difference in profitability, market share and survival.

The Simulation-based heuristic's design and prototype is a contribution to this area. Its aim has been to present a suitable tool for the fleet allocation decision process. Its further research and development promises gains for the Car Rental Industry as well as for others with perishable assets and stochastic and dynamic behaviors.

LIST OF REFERENCES

Bitran, G. R. and S. V. Mondschein (1995), "An Application of Yield Management to the Hotel Industry Considering Multiple Day stays," Operations Research, 43 (3), 427-443.

Bitran, G. R. and S. Dasu (1992), "Ordering Policies in an Environment of Stochastic Yields and Substitutable Demands," Operations Research, 40 (5), 999-1017.

Bixby, R. E., J. W. Gregory, I. J. Lustig, R. E. Marsten, and D. F. Shanno (1992), "Very large-scale Linear Programming: a Case Study in Combining Interior Point and Simplex Methods," Operations Research, 40 (5), 885-897.

Carroll, W. J. and R. C. Grimes (1995), "Evolutionary Change in Product Management: Experiences in the Car Rental Industry," Interfaces, 25 (5), 84-104.

Ciancimino, A., G. Inzerillo, S. Lucidi, and L. Palagi (1999), "A Mathematical Programming Approach for the Solution of the Railway Yield Management Problem," Transportation Science, 33 (2), 168-181.

Cook, T. M. (1998), "SABRE SOARS," ORMS Today, June 1998, 26-31.

Edelstein, M. and M. Melnyk (1977), "The Pool Control System," Interfaces, 8 (1) (Part 2), 21-36.

Eom, H. B. and S. M. Lee (1990), "A Survey of Decision Support System Applications," Interfaces, 20 (3), 65-79.

Geraghty, M. K. and E. Johnson (1997), "Revenue Management Saves National Car Rental," Interfaces, 27 (1), 107-127.

Ladany, S. P. (1976), "Dynamic Operating Rules for Motel Reservations," Decision Sciences, 7 (4), 829-840.

McGill, J. I. and G. J. Van Ryzin (1999), "Revenue Management: Research Overview and Prospects," Transportation Science, 33 (2), 233-256.

Rexing, B., C. Barnhart, T. Kniker, A. Jarrah, and N. Krishnamurthy (2000), "Airline Fleet Assignment with Time Windows," Transportation Science, 34 (1), 1-20.

Richter, H. (1989), " Thirty Years of Airline Operations Research," Interfaces, 19 (4), 3-9.

Rothstein, M. (1974), "Hotel Overbooking as a Markovian Sequential Decision Process," Decision Sciences, 5, 389-404.

Subramanian, R., R. P. Scheff, Jr., J. D. Quillinan, D. S. Wiper, R. E. Marsten (1994), "Coldstart: Fleet Assignment at Delta Air Lines," Interfaces, 24 (1), 104-120.

Subramanian, J., S. Stidham, Jr., and C. J. Lautenbacher (1999), "Airline Yield Management with Overbooking, Cancellations, and No-Shows," Transportation Science, 33 (2), 147-167.

Weatherford, L. R. and S. E. Bodily (1992), "A taxonomy and Research Overview of Perishable-Asset Revenue Management: Yield Management, Overbooking, and Pricing," Operations Research, 40 (5), 831-844.

Williams, F. E. (1977), "Decision Theory and the Innkeeper: An Approach for Setting Hotel Reservation Policy," Interfaces, 7 (4), 18-30.

APPENDIX A

It contains historical data for Fort Lauderdale and Key West for three different scenarios of low, average and high demand.

Time period	Realized Constrained Demand (Rentals)	No Shows	Cancellations	Total Constrained Demand (Reservations)	Turndowns	Total Unconstrained Demand
Sun	53	14	6	73	13	86
Mon	56	13	3	72	17	89
Tue	49	22	7	78	12	90
Wed	22	6	1	29	9	38
Thu	24	5	1	30	11	41
Fri	35	14	4	53	23	76
Sat	84	19	14	117	20	137

Table 54: Demand for Fort Lauderdale by day of week – First week.

Time period	0-2 am	2-4 am	4-6 am	6-8 am	8-10 am	10-12pm	12-2 pm	2-4 pm	4-6 pm	6-8 pm	8-10 pm	10-12 pm
Sun	0	0	0	0.06	0.09	0.31	0.17	0.06	0.11	0.09	0.07	0.04
Mon	0.02	0	0	0.02	0.07	0.14	0.11	0.12	0.07	0.12	0.18	0.16
Tue	0	0	0	0	0.06	0.18	0.12	0.14	0.16	0.16	0.08	0.12
Wed	0.04	0	0	0	0.09	0.09	0.17	0.17	0.09	0.04	0.13	0.17
Thu	0.04	0	0.04	0	0	0.13	0.17	0.04	0.13	0.17	0.17	0.13
Fri	0.03	0.05	0	0.03	0.05	0.13	0.13	0.08	0.11	0.24	0.08	0.08
Sat	0.02	0.02	0	0.01	0.05	0.08	0.08	0.28	0.15	0.24	0.02	0.03

Table 55: Demand for Fort Lauderdale – Arrival Pattern – First week.

Time period	No Show %	Cancellation %	Realized Unconstrained Demand	One day length of stay %	Three days length of stay %
Sun	0.19	0.08	62	0.00	1.00
Mon	0.18	0.04	69	0.02	0.98
Tue	0.28	0.09	57	0.02	0.98
Wed	0.21	0.03	29	0.00	1.00
Thu	0.17	0.03	33	0.04	0.96
Fri	0.26	0.08	50	0.03	0.97
Sat	0.16	0.12	98	0.00	1.00

Table 56: Demand for Fort Lauderdale – Percentages - First week.

Time period	RU demand for Car Class 1 & 1 day Los	RU demand for Car Class 1 & 3 days Los	RU demand for Car Class 2 & 1 day Los	RU demand for Car Class 2 & 3 day Los
Sun	0	34	0	29
Mon	1	44	0	24
Tue	1	28	1	27
Wed	0	17	0	12
Thu	1	17	1	14
Fri	1	22	1	27
Sat	0	51	0	48

Table 57: Demand for Fort Lauderdale by car class and length of stay – First week.

Time period	Realized Constrained Demand (Rentals)	No Shows	Cancellations	Total Constrained Demand (Reservations)	Turndowns	Total Unconstrained Demand
Sun	200	97	36	333	42	375
Mon	296	115	21	432	16	448
Tue	256	91	37	384	23	407
Wed	51	19	4	74	18	92
Thu	61	24	10	95	21	116
Fri	118	34	22	174	53	227
Sat	326	134	43	503	60	563

Table 58: Demand for Fort Lauderdale by day of week – Second week.

Time period	0-2 am	2-4 am	4-6 am	6-8 am	8-10 am	10-12pm	12-2 pm	2-4 pm	4-6 pm	6-8 pm	8-10 pm	10-12 pm
Sun	0.01	0	0	0.02	0.03	0.24	0.13	0.15	0.13	0.13	0.07	0.06
Mon	0.03	0.01	0	0.01	0.04	0.17	0.23	0.21	0.06	0.1	0.08	0.07
Tue	0.03	0.01	0	0.02	0.05	0.14	0.14	0.21	0.14	0.15	0.07	0.06
Wed	0.02	0	0	0.04	0	0.13	0.13	0.26	0.08	0.13	0.11	0.09
Thu	0.06	0	0	0.03	0.1	0.13	0.21	0.08	0.13	0.11	0.11	0.03
Fri	0	0	0	0	0.03	0.16	0.14	0.24	0.09	0.13	0.12	0.09
Sat	0.03	0.01	0	0.01	0.03	0.14	0.17	0.24	0.13	0.13	0.05	0.06

Table 59: Demand for Fort Lauderdale – Arrival Pattern – Second week.

Time period	No Show %	Cancellation %	Realized Unconstrained Demand	One day length of stay %	Three days length of stay %
Sun	0.29	0.11	225	0.52	0.48
Mon	0.27	0.05	307	0.60	0.40
Tue	0.24	0.10	271	0.63	0.37
Wed	0.26	0.05	63	0.04	0.96
Thu	0.25	0.11	74	0.02	0.98
Fri	0.20	0.13	154	0.04	0.96
Sat	0.27	0.09	364	0.41	0.59

Table 60: Demand for Fort Lauderdale – Percentages – Second week.

Time period	RU demand for Car Class 1 & 1 day Los	RU demand for Car Class 1 & 3 days Los	RU demand for Car Class 2 & 1 day Los	RU demand for Car Class 2 & 3 day Los
Sun	76	70	41	38
Mon	106	71	77	51
Tue	106	63	63	37
Wed	1	31	1	30
Thu	1	40	1	32
Fri	3	62	3	84
Sat	84	121	64	91

Table 61: Demand for Fort Lauderdale by car class and length of stay – Second week.

Time period	Realized Constrained Demand (Rentals)	No Shows	Cancellations	Total Constrained Demand (Reservations)	Turndowns	Total Unconstrained Demand
Sun	420	202	61	683	69	752
Mon	510	177	100	787	42	829
Tue	363	177	62	602	37	639
Wed	689	283	113	1085	104	1189
Thu	751	383	181	1315	190	1505
Fri	921	487	226	1634	340	1974
Sat	587	216	115	918	136	1054

Table 62: Demand for Fort Lauderdale by day of week – Third week.

Time period	0-2 am	2-4 am	4-6 am	6-8 am	8-10 am	10-12pm	12-2 pm	2-4 pm	4-6 pm	6-8 pm	8-10 pm	10-12 pm
Sun	0.05	0	0.01	0.02	0.04	0.17	0.2	0.15	0.1	0.12	0.08	0.07
Mon	0.02	0	0	0.02	0.08	0.15	0.15	0.17	0.13	0.09	0.08	0.1
Tue	0.04	0	0	0.02	0.07	0.22	0.15	0.19	0.09	0.06	0.07	0.09
Wed	0.02	0	0	0.01	0.07	0.12	0.18	0.17	0.12	0.1	0.1	0.11
Thu	0.03	0	0	0.01	0.05	0.18	0.2	0.12	0.11	0.09	0.09	0.1
Fri	0.04	0.01	0	0.02	0.06	0.14	0.18	0.16	0.1	0.12	0.08	0.08
Sat	0.07	0	0	0.02	0.06	0.14	0.19	0.16	0.11	0.1	0.08	0.06

Table 63: Demand for Fort Lauderdale – Arrival Pattern – Third week.

Time period	No Show %	Cancellation %	Realized Unconstrained Demand	One day length of stay %	Three days length of stay %
Sun	0.30	0.09	462	0.69	0.31
Mon	0.22	0.13	537	0.79	0.21
Tue	0.29	0.10	386	0.79	0.21
Wed	0.26	0.10	756	0.76	0.24
Thu	0.29	0.14	859	0.75	0.25
Fri	0.30	0.14	1111	0.68	0.32
Sat	0.24	0.13	673	0.69	0.31

Table 64: Demand for Fort Lauderdale – Percentages – Third week.

Time period	RU demand for Car Class 1 & 1 day Los	RU demand for Car Class 1 & 3 days Los	RU demand for Car Class 2 & 1 day Los	RU demand for Car Class 2 & 3 day Los
Sun	197	88	120	54
Mon	217	58	209	55
Tue	160	43	148	39
Wed	295	93	283	90
Thu	315	105	329	110
Fri	398	188	353	166
Sat	248	111	210	95

Table 65: Demand for Fort Lauderdale by car class and length of stay – Third week.

Time period	Realized Constrained Demand (Rentals)	No Shows	Cancellations	Total Constrained Demand (Reservations)	Turndowns	Total Unconstrained Demand
Sun	1	0	1	2	1	3
Mon	3	0	0	3	1	4
Tue	2	0	0	2	0	2
Wed	0	0	0	0	0	0
Thu	2	0	0	2	0	2
Fri	0	0	0	0	0	0
Sat	1	1	0	2	1	3

Table 66: Demand for Key West by day of week – First week.

Time period	0-2 am	2-4 am	4-6 am	6-8 am	8-10 am	10-12pm	12-2 pm	2-4 pm	4-6 pm	6-8 pm	8-10 pm	10-12 pm
Sun	0	0	0	0	0	0	0	0	0	1	0	0
Mon	0	0	0	0	0.33	0	0	0.33	0.33	0	0	0
Tue	0	0	0	0	0	0	0	0	0	1	0	0
Wed	0	0	0	0	0	0	0	0	0	0	0	0
Thu	0	0	0	0	0	0	0	0	1	0	0	0
Fri	0	0	0	0	0	0	0	0	0	0	0	0
Sat	0	0	0	0	0	0	1	0	0	0	0	0

Table 67: Demand for Key West – Arrival Patterns – First week.

Time period	No Show %	Cancellation %	Realized Unconstrained Demand	One day length of stay %	Three days length of stay %
Sun	0	0.5	2	0.00	1.00
Mon	0	0	4	0.00	1.00
Tue	0	0	2	0.50	0.50
Wed	0	0	0	0.00	0.00
Thu	0	0	2	0.00	1.00
Fri	0	0	0	0.00	0.00
Sat	0.5	0	2	0.00	1.00

Table 68: Demand for Key West – Percentages - First week.

Time period	RU demand for Car Class 1 & 1 day Los	RU demand for Car Class 1 & 3 days Los	RU demand for Car Class 2 & 1 day Los	RU demand for Car Class 2 & 3 day Los
Sun	0	0	0	2
Mon	0	1	0	3
Tue	1	0	0	1
Wed	0	0	0	0
Thu	0	1	0	1
Fri	0	0	0	0
Sat	0	0	0	2

Table 69: Demand for Key West by car class and length of stay – First week.

Time period	Realized Constrained Demand (Rentals)	No Shows	Cancellations	Total Constrained Demand (Reservations)	Turndowns	Total Unconstrained Demand
Sun	12	4	6	22	0	22
Mon	10	2	2	14	1	15
Tue	9	3	4	16	3	19
Wed	3	1	0	4	0	4
Thu	3	0	1	4	1	5
Fri	3	1	3	7	3	10
Sat	12	2	8	22	2	24

Table 70: Demand for Key West by day of the week – Second week.

Time period	0-2 am	2-4 am	4-6 am	6-8 am	8-10 am	10-12pm	12-2 pm	2-4 pm	4-6 pm	6-8 pm	8-10 pm	10-12 pm
Sun	0	0	0	0	0	0.08	0.15	0.15	0.31	0.15	0.15	0
Mon	0	0	0	0	0	0.2	0.3	0.2	0.1	0.1	0.1	0
Tue	0	0	0	0	0	0.11	0.11	0.11	0	0.33	0.33	0
Wed	0	0	0	0	0	0	0.33	0.67	0	0	0	0
Thu	0	0	0	0.33	0	0.33	0	0.33	0	0	0	0
Fri	0	0	0	0	0	0	0	0.33	0.33	0.33	0	0
Sat	0	0	0	0	0.08	0	0.38	0.15	0.15	0.15	0.08	0

Table 71: Demand for Key West – Arrival Pattern – Second week.

Time period	No Show %	Cancellation %	Realized Unconstrained Demand	One day length of stay %	Three days length of stay %
Sun	0.18	0.27	12	0.69	0.31
Mon	0.14	0.14	11	0.60	0.40
Tue	0.19	0.25	11	1.00	0.00
Wed	0.25	0.00	3	0.33	0.67
Thu	0.00	0.25	4	0.00	1.00
Fri	0.14	0.43	4	0.00	1.00
Sat	0.09	0.36	13	0.69	0.31

Table 72: Demand for Key West – Percentages - Second week.

Time period	RU demand for Car Class 1 & 1 day Los	RU demand for Car Class 1 & 3 days Los	RU demand for Car Class 2 & 1 day Los	RU demand for Car Class 2 & 3 day Los
Sun	2	1	6	3
Mon	2	2	4	3
Tue	9	0	2	0
Wed	0	1	1	1
Thu	0	1	0	3
Fri	0	3	0	1
Sat	3	2	6	2

Table 73: Demand for Key West by car class and length of stay – Second week.

Time period	Realized Constrained Demand (Rentals)	No Shows	Cancellations	Total Constrained Demand (Reservations)	Turndowns	Total Unconstrained Demand
Sun	25	8	12	45	0	45
Mon	31	9	16	56	8	64
Tue	19	5	6	30	6	36
Wed	36	13	11	60	4	64
Thu	49	15	18	82	0	82
Fri	40	17	14	71	12	83
Sat	22	15	11	48	11	59

Table 74: Demand for Key West by day of the week – Third week.

Time period	0-2 am	2-4 am	4-6 am	6-8 am	8-10 am	10-12pm	12-2 pm	2-4 pm	4-6 pm	6-8 pm	8-10 pm	10-12 pm
Sun	0	0	0	0.04	0.11	0.11	0.15	0.33	0.11	0.15	0	0
Mon	0	0	0	0.06	0.06	0.13	0.29	0.13	0.13	0.16	0.03	0
Tue	0	0	0	0.11	0.05	0.11	0.16	0.26	0.05	0.21	0.05	0
Wed	0	0	0	0.05	0.05	0.16	0.24	0.24	0.11	0.16	0	0
Thu	0	0	0	0.08	0.16	0.1	0.22	0.16	0.14	0.08	0.08	0
Fri	0	0	0	0	0.13	0.08	0.2	0.25	0.18	0.13	0.05	0
Sat	0	0	0	0	0.09	0.17	0.17	0.17	0.26	0.13	0	0

Table 75: Demand for Key West – Arrival Patterns – Third week.

Time period	No Show %	Cancellation %	Realized Unconstrained Demand	One day length of stay %	Three days length of stay %
Sun	0.18	0.27	25	0.81	0.19
Mon	0.16	0.29	35	0.87	0.13
Tue	0.17	0.20	23	0.79	0.21
Wed	0.22	0.18	38	0.84	0.16
Thu	0.18	0.22	49	0.88	0.12
Fri	0.24	0.20	47	0.75	0.25
Sat	0.31	0.23	27	0.87	0.13

Table 76: Demand for Key West – Percentages - Third week.

Time period	RU demand for Car Class 1 & 1 day Los	RU demand for Car Class 1 & 3 days Los	RU demand for Car Class 2 & 1 day Los	RU demand for Car Class 2 & 3 day Los
Sun	11	3	9	2
Mon	15	2	16	2
Tue	9	3	9	2
Wed	18	4	13	3
Thu	20	3	23	3
Fri	20	7	15	5
Sat	12	2	11	2

Table 77: Demand for Key West by car class and length of stay – Third week.

APPENDIX B

It contains the Lingo code for the optimization model and an example of its input file.

```
MODEL:
TITLE Car_Rental;
!-------------------------------------------------------;
! This models maximizes assignment of cars across different;
! Rental stations;
! Data for this model is read from car_rental.ldt;
!-------------------------------------------------------;
SETS:
!-------------------------------------------------------;

! The set of locations;
LOCATION/ @FILE( 'CAR_RENTAL.LDT')/;

! The set of car classes;
CAR_CLASS/ @FILE( 'CAR_RENTAL.LDT')/ : HIERARCHY;

! The set of cars in each location;
CARS( LOCATION, CAR_CLASS ): INIFLEET;

! The set of Lenght of Stay;
LOS/ @FILE( 'CAR_RENTAL.LDT')/ : DAYS;

! The set of time periods;
TIME / @FILE( 'CAR_RENTAL.LDT')/;

! The set of transfers between cities;
TRANSFER( LOCATION, LOCATION) | &2 #NE# &1 : COST;

! The set of transfers by car classes;
MOVEMENT( TRANSFER, CAR_CLASS ): MOVES;

! The set of Car assignments, same car or an upgrade
CAR_ASSIGNED( CAR_CLASS, CAR_CLASS ) |
              HIERARCHY( &1) #LE# HIERARCHY( &2 ) : INICAR;

CAR_ASSIGNED / @FILE( 'CAR_RENTAL.LDT' )/ : ORIGCAR, UPGCAR;

! The set of Demand;
DEMAND( LOCATION, CAR_CLASS, LOS, TIME) : CUSTOMERS, PRICE;

! The set of Rentals;
RENTAL( LOCATION, CAR_ASSIGNED, LOS, TIME) : RENTER;

! Total Fleet at the beginning of each day;
DAILYFLEET(LOCATION, CAR_CLASS, TIME) : DAYFLEET;

ENDSETS

!-------------------------------------------------------;
! Misc. Variables;
!-------------------------------------------------------;

! Set NP = no. of time periods in the problem;
NP = @SIZE( TIME);

@FOR( RENTAL (I,J,K,L): @GIN( RENTER( I,J,K,L)));
@FOR( MOVEMENT (I,J,K): @GIN( MOVES(I,J,K)));

!-------------------------------------------------------;
! Maximization Function;
! Car Rentals need to be maximized and the cost of movement;
! between cities needs to be minimized;
!-------------------------------------------------------;
[OBJECTIVE] MAX = @SUM ( RENTAL(I,J,K,L ):
                 RENTER (I,J,K,L) * DAYS(K) * PRICE(I,ORIGCAR(J),K,L )) -
                         @SUM ( MOVEMENT( M, N, O):
                         MOVES (M,N,O) * COST(M,N));
!-------------------------------------------------------;
! Constraints;
```

```
!-------------------------------------------------------;

!------;
!Demand;
!------;

 @FOR( DEMAND (I,J,K,L) : [SATISFIED_DEMAND] CUSTOMERS (I,J,K,L)>=
     @SUM( RENTAL (I,M,K,L) | ORIGCAR( M ) #EQ#J:RENTER(I,M,K,L)));

!------;
!Supply;
!------;

 @FOR (DAILYFLEET (I,J,T) : [USED_FLEET] DAYFLEET (I,J,T) >=
     @SUM(RENTAL (I,M,K,T) | UPGCAR(M) #EQ# J: RENTER (I,M,K,T)));

 @FOR (DAILYFLEET (I,J,T) :
                          DAYFLEET (I,J,T) =
                                  INIFLEET (I,J) +
                                  @SUM(MOVEMENT(M,I,J) : MOVES(M,I,J)) -
                                  @SUM(MOVEMENT(I,M,J) : MOVES(I,M,J)) -
         @SUM( RENTAL(I,M,K,P)
           | UPGCAR(M) #EQ# J #AND# P #LT# T: RENTER (I,M,K,P)) +
         @SUM( RENTAL(I,M,K,P)
           | UPGCAR(M) #EQ# J #AND# P #LT# T #AND# DAYS(K) #LE# (T-P)
                 : RENTER(I,M,K,P)));


!-------------------------------------------------------;
DATA:
!-------------------------------------------------------;

! Get the Car Class Hierarchy;
HIERARCHY= @FILE('CAR_RENTAL.LDT');

! Get Initial Fleet at each location;
INIFLEET= @FILE('CAR_RENTAL.LDT');

! Get the Days;
DAYS = @FILE('CAR_RENTAL.LDT');

! Get the Transfer Cost;
COST = @FILE('CAR_RENTAL.LDT');

! Get the Initial car;
ORIGCAR, UPGCAR = @FILE('CAR_RENTAL.LDT');

! Get demand at different locations,
car classes, los and time;
CUSTOMERS, PRICE = @FILE('CAR_RENTAL.LDT');

ENDDATA
END
```

An example of the Input file:

```
! Locations list;
   F, ! Fort Lauderdale;
   M, ! Miami;
   K~ ! Key West;

! Car Classes list;
   E, ! Economic;
   M~ ! Medium;

! Lenght of Stay list;
   D, ! Daily;
   T~ ! Three day rental;

! The set of periods;
  1..7~


! This is an alternative;
 ! E     M ;
 ! E;    EE,      EM,
 ! M;             MM ~


 ! The Car Class Hierarchy;
 1, 2~

  ! The Initial number of cars;
  ! in each location;
  !F; 650, 907,
  !M; 593, 1022,
  !K; 29, 29~

  ! The Lenght of stay;
  1,3 ~

 ! The transfer cost;
!20, 20, 30, 30, 30, 30 ~
1000000, 1000000, 1000000, 1000000, 1000000, 1000000 ~

  ! The Original and Upgrade car class;
 1 1
 1 2
 2 2~

 ! Demand - Customers and Price;
 !
 ! LOC   CAR      LOS      TIME     customers        price;
 ! F              E        D        1;               207      26
 ! F              E        D        2;               232      22
 ! F              E        D        3;               163      25
 ! F              E        D        4;               327      26
 ! F              E        D        5;               322      27
 ! F              E        D        6;               457      26
 ! F              E        D        7;               273      21
 ! F              E        T        1;               86       17
 ! F              E        T        2;               55       17
 ! F              E        T        3;               47       19
 ! F              E        T        4;               88       20
 ! F              E        T        5;               105      21
 ! F              E        T        6;               199      21
 ! F              E        T        7;               104      17
 ! F              M        D        1;               118      33
 ! F              M        D        2;               203      32
 ! F              M        D        3;               153      30
```

```
! F  M  D  4;  284  35
! F  M  D  5;  334  35
! F  M  D  6;  373  40
! F  M  D  7;  207  29
! F  M  T  1;  60   26
! F  M  T  2;  62   22
! F  M  T  3;  40   25
! F  M  T  4;  96   26
! F  M  T  5;  111  27
! F  M  T  6;  189  28
! F  M  T  7;  113  27
! M  E  D  1;  287  28
! M  E  D  2;  282  32
! M  E  D  3;  271  25
! M  E  D  4;  193  28
! M  E  D  5;  291  28
! M  E  D  6;  366  29
! M  E  D  7;  289  26
! M  E  T  1;  151  22
! M  E  T  2;  128  21
! M  E  T  3;  108  21
! M  E  T  4;  185  23
! M  E  T  5;  154  24
! M  E  T  6;  251  23
! M  E  T  7;  246  22
! M  M  D  1;  276  46
! M  M  D  2;  272  37
! M  M  D  3;  227  33
! M  M  D  4;  192  37
! M  M  D  5;  302  38
! M  M  D  6;  357  43
! M  M  D  7;  262  39
! M  M  T  1;  209  33
! M  M  T  2;  135  29
! M  M  T  3;  119  28
! M  M  T  4;  211  30
! M  M  T  5;  238  33
! M  M  T  6;  318  34
! M  M  T  7;  300  34
! K  E  D  1;  12   16
! K  E  D  2;  15   16
! K  E  D  3;  9    31
! K  E  D  4;  17   24
! K  E  D  5;  22   24
! K  E  D  6;  18   19
! K  E  D  7;  10   17
! K  E  T  1;  3    23
! K  E  T  2;  2    14
! K  E  T  3;  3    16
! K  E  T  4;  5    13
! K  E  T  5;  2    29
! K  E  T  6;  8    14
! K  E  T  7;  4    15
! K  M  D  1;  10   12
! K  M  D  2;  14   18
! K  M  D  3;  8    12
! K  M  D  4;  15   23
! K  M  D  5;  23   21
! K  M  D  6;  15   18
! K  M  D  7;  12   26
! K  M  T  1;  2    20
! K  M  T  2;  4    7
! K  M  T  3;  3    13
! K  M  T  4;  2    22
! K  M  T  5;  4    25
! K  M  T  6;  4    19
! K  M  T  7;  2    25
```

APPENDIX C

It contains the Arena model.

Model Frame

```
customers       CREATE,          1,0.00001:,1;
getattributes   READ,            rdemand,free:
                                 v_alocation_rdemand,
                                 v_carclass_rdemand,
                                 v_los_rdemand,
                                 v_adatetime_rdemand,
                                 v_customers_rdemand,
                                 v_dailyrate_rdemand;
entersys        ASSIGN:          a_dailyrate=v_dailyrate_rdemand:
                                 a_alocation=v_alocation_rdemand:
                                 a_carreserved=v_carclass_rdemand:
                                 a_objecttype=2:
                                 a_los=v_los_rdemand:
                                 a_adatetime=v_adatetime_rdemand:
                                 a_customers=v_customers_rdemand:
                                 a_dow=(a_adatetime-1)/1440 + 1:
                                 a_idatetime=a_adatetime:
                                 a_ilos=a_los;
set_time        ASSIGN:          a_adatetime=a_adatetime + ED(4+3*(a_dow-1)+a_alocation);
read_cust       DUPLICATE:       1,station_door:NEXT(getattributes);

station_door    BRANCH,          1:
                                 If,a_customers > 0,42$,Yes:
                                 Else,nothing,Yes;
42$             DELAY:           a_adatetime - tnow;
arrival_of_cust DUPLICATE:       a_customers - 1;
52$             BRANCH,          1,10:
                                 If,a_carreserved == 1,T1,Yes:
                                 If,a_carreserved == 2,81$,Yes;
T1              BRANCH,          1,10:
                                 If,NQ(e_base + 1 + 26) > 0,80$,Yes:
                                 If,NQ(e_base + 2 + 26) > 0,81$,Yes:
                                 Else,80$,Yes;
80$             ASSIGN:          a_car_about_to_rent=1;
q11             COUNT:           e_base + 1,1;
49$             BRANCH,          1,10:
                                 If,a_alocation == 1,CustCity1forCar1,Yes:
                                 If,a_alocation == 2,CustCity2forCar1,Yes:
                                 If,a_alocation == 3,CustCity3forCar1,Yes;
CustCity1forCar1 QUEUE,          q_custCity1forCar1:MARK(a_timewaitforcar):DETACH;
CustCity2forCar1 QUEUE,          q_custCity2forCar1:MARK(a_timewaitforcar):DETACH;
CustCity3forCar1 QUEUE,          q_custCity3forCar1:MARK(a_timewaitforcar):DETACH;
81$             ASSIGN:          a_car_about_to_rent=2;
q12             COUNT:           e_base + 2,1;
50$             BRANCH,          1,10:
                                 If,a_alocation == 1,CustCity1forCar2,Yes:
                                 If,a_alocation == 2,CustCity2forCar2,Yes:
                                 If,a_alocation == 3,CustCity3forCar2,Yes;
CustCity1forCar2 QUEUE,          q_custCity1forCar2:MARK(a_timewaitforcar):DETACH;
CustCity2forCar2 QUEUE,          q_custCity2forCar2:MARK(a_timewaitforcar):DETACH;
CustCity3forCar2 QUEUE,          q_custCity3forCar2:MARK(a_timewaitforcar):DETACH;
nothing         DISPOSE;


cars            CREATE,          1,0.000000000001:1,1;
fleetinfo       READ,            afleet,free:
                                 v_alocation_afleet,
                                 v_carclass_afleet,
                                 v_quantity_afleet;
0$              ASSIGN:          a_alocation=v_alocation_afleet:
                                 a_carrented=v_carclass_afleet:
                                 a_quantity=v_quantity_afleet:
                                 a_objecttype=1;
```

```
duplicatefleet DUPLICATE:    a_quantity,111$:NEXT(fleetinfo);

111$           COUNT:        e_base+174+a_carrented,1;
stationLot     ROUTE:        0.0,a_alocation;

               MATCH,:       CustCity1forCar1,51$:
                             carsCity1Car1;
51$            ASSIGN:       a_carrented=1;
31$            ROUTE:        0.0,a_alocation + 3;

               MATCH,:       CustCity1forCar2,53$:
                             carsCity1Car2;
53$            ASSIGN:       a_carrented=2:NEXT(31$);


8$             CREATE,       1,1:1,1;
infleet        READ,         ifleet,free:
                             v_alocation_ifleet,
                             v_carclass_ifleet,
                             v_adatetime_ifleet,
                             v_quantity_ifleet;
9$             ASSIGN:       a_alocation=v_alocation_ifleet:
                             a_carrented=v_carclass_ifleet:
                             a_adatetime=v_adatetime_ifleet:
                             a_quantity=v_quantity_ifleet:
                             a_objecttype=1;
32$            DUPLICATE:    a_quantity,43$:NEXT(infleet);

43$            DELAY:        a_adatetime - tnow;
44$            ROUTE:        0.0,a_alocation;


10$            CREATE,       1,1.0001:1,1;
11$            READ,         dfleet,free:
                             v_alocation_dfleet,
                             v_carclass_dfleet,
                             v_adatetime_dfleet,
                             v_quantity_dfleet;
12$            ASSIGN:       a_alocation=v_alocation_dfleet:
                             a_carrented=v_carclass_dfleet:
                             a_adatetime=v_adatetime_dfleet:
                             a_quantity=v_quantity_dfleet:
                             a_objecttype=1;
19$            DUPLICATE:    a_quantity,36$:NEXT(11$);

36$            ROUTE:        0.0,e_baseindex + 6;


17$            CREATE,       1,0.00015:1,1;
moveinfo       READ,         mfleet,free:
                             v_alocation_mfleet,
                             v_rlocation_mfleet,
                             v_carclass_mfleet,
                             v_adatetime_mfleet,
                             v_quantity_mfleet,
                             v_cost_mfleet;
18$            ASSIGN:       a_alocation=v_alocation_mfleet:
                             a_rlocation=v_rlocation_mfleet:
                             a_carrented=v_carclass_mfleet:
                             a_adatetime=v_adatetime_mfleet:
                             a_quantity=v_quantity_mfleet:
                             a_costmovement=v_cost_mfleet;
59$            DUPLICATE:    1,48$:NEXT(moveinfo);

48$            DELAY:        a_adatetime - tnow;
38$            BRANCH,       1:
                             If,v_moving(a_alocation,a_carrented) == 1,37$,Yes:
                             Else,39$,Yes;
37$            WAIT:         e_baseindex,1;
```

```
39$             ASSIGN:         v_moving(a_alocation,a_carrented)=1;
20$             DUPLICATE:      a_quantity,46$;
56$             ROUTE:          0.0,checkmoves;

46$             ROUTE:          0.0,e_baseindex + 12;

28$             STATION,        LotCity1-LotCity3;
29$             COUNT:          e_baseindex + 29,1;
1$              BRANCH,         1:
                                If,a_alocation == 1 .AND. a_carrented == 1,carsCity1Car1,Yes:
                                If,a_alocation == 1 .AND. a_carrented == 2,carsCity1Car2,Yes:
                                If,a_alocation == 2 .AND. a_carrented == 1,carsCity2Car1,Yes:
                                If,a_alocation == 2 .AND. a_carrented == 2,carsCity2Car2,Yes:
                                If,a_alocation == 3 .AND. a_carrented == 1,carsCity3Car1,Yes:
                                If,a_alocation == 3 .AND. a_carrented == 2,carsCity3Car2,Yes;
carsCity1Car1 QUEUE,            q_carsCity1Car1:DETACH;
carsCity1Car2 QUEUE,            q_carsCity1Car2:DETACH;
carsCity2Car1 QUEUE,            q_carsCity2Car1:DETACH;
carsCity2Car2 QUEUE,            q_carsCity2Car2:DETACH;
carsCity3Car1 QUEUE,            q_carsCity3Car1:DETACH;
carsCity3Car2 QUEUE,            q_carsCity3Car2:DETACH;

Rental_process STATION,         CounterCity1-CounterCity3;
2$              COUNT:          a_alocation + 8,1;
3$              DELAY:          norm(15,5);
4$              TALLY:          1,interval(a_timewaitforcar),1;
assign_breakdown ASSIGN:        a_breakdown=cont(0.000001,1,1,0);
83$             BRANCH,         1:
                                If,a_breakdown == 1,90$,Yes:
                                Else,91$,Yes;
90$             ASSIGN:         a_los=0.5*(1+ED(25 + a_dow)) * a_los;
84$             DELAY:          a_los;
54$             DUPLICATE:      1,5$;
6$              COUNT:          a_alocation + 41,1;
88$             BRANCH,         1:
                                If,a_breakdown == 1,92$,Yes:
                                Else,7$,Yes;
92$             COUNT:          c_breakdowns,1;
89$             ASSIGN:         v_timeend=tnow:
                                v_losindays=a_los/1440:
                                v_totlosindays=v_totlosindays + v_losindays:
                                v_revenue=-2 * a_dailyrate * v_losindays:
                                v_totrevenue=v_totrevenue + v_revenue:
                                v_totnetrevenue=v_totnetrevenue - v_totcostmovement;
exitsystem      DISPOSE;

7$              ASSIGN:         v_timeend=tnow:
                                v_losindays=a_los/1440:
                                v_totlosindays=v_totlosindays + v_losindays:
                                v_revenue=a_dailyrate * v_losindays:
                                v_totrevenue=v_totrevenue + v_revenue:
                                v_totnetrevenue=v_totrevenue - v_totcostmovement;
102$            BRANCH,         1:
                                If,aint(a_ilos/1440)==1,94$,Yes:
                                Else,103$,Yes;
94$             BRANCH,         1:
                                If,aint(a_idatetime/1440) ==0,95$,Yes:
                                If,aint(a_idatetime/1440)==1,96$,Yes:
                                If,aint(a_idatetime/1440)==2,97$,Yes:
                                If,aint(a_idatetime/1440)==3,98$,Yes:
                                If,aint(a_idatetime/1440)==4,99$,Yes:
                                If,aint(a_idatetime/1440)==5,100$,Yes:
                                Else,101$,Yes;
95$             COUNT:          48+(a_alocation - 1)*3 + (a_carreserved + a_carrented -
2),1:NEXT(exitsystem);

96$             COUNT:          57 + (a_alocation - 1)*3 + (a_carreserved + a_carrented -
2),1:NEXT(exitsystem);
```

```
97$             COUNT:          66+ (a_alocation - 1)*3 + (a_carreserved + a_carrented -
2),1:NEXT(exitsystem);

98$             COUNT:          75 + (a_alocation - 1)*3 + (a_carreserved + a_carrented -
2),1:NEXT(exitsystem);

99$             COUNT:          84 + (a_alocation - 1)*3 + (a_carreserved + a_carrented -
2),1:NEXT(exitsystem);

100$            COUNT:          93 + (a_alocation - 1)*3 + (a_carreserved + a_carrented -
2),1:NEXT(exitsystem);

101$            COUNT:          102 + (a_alocation - 1)*3 + (a_carreserved + a_carrented -
2),1:NEXT(exitsystem);

103$            BRANCH,         1:
                                If,aint(a_idatetime/1440) ==0,104$,Yes:
                                If,aint(a_idatetime/1440)==1,105$,Yes:
                                If,aint(a_idatetime/1440)==2,106$,Yes:
                                If,aint(a_idatetime/1440)==3,107$,Yes:
                                If,aint(a_idatetime/1440)==4,108$,Yes:
                                If,aint(a_idatetime/1440)==5,109$,Yes:
                                Else,110$,Yes;
104$            COUNT:          112+(a_alocation - 1)*3 + (a_carreserved + a_carrented -
2),1:NEXT(exitsystem);

105$            COUNT:          121 + (a_alocation - 1)*3 + (a_carreserved + a_carrented -
2),1:NEXT(exitsystem);

106$            COUNT:          130 + (a_alocation - 1)*3 + (a_carreserved + a_carrented -
2),1:NEXT(exitsystem);

107$            COUNT:          139 + (a_alocation - 1)*3 + (a_carreserved + a_carrented -
2),1:NEXT(exitsystem);

108$            COUNT:          148 + (a_alocation - 1)*3 + (a_carreserved + a_carrented -
2),1:NEXT(exitsystem);

109$            COUNT:          157 + (a_alocation - 1)*3 + (a_carreserved + a_carrented -
2),1:NEXT(exitsystem);

110$            COUNT:          166 + (a_alocation - 1)*3 + (a_carreserved + a_carrented -
2),1:NEXT(exitsystem);

5$              COUNT:          a_alocation + 44,1;
57$             ASSIGN:         a_dailyrate=0:
                                a_carreserved=0:
                                a_objecttype=1:
                                a_los=0:
                                a_adatetime=0:
                                a_customers=0:
                                a_idatetime=0:
                                a_ilos=0;
85$             BRANCH,         1:
                                If,a_breakdown == 1,86$,Yes:
                                Else,30$,Yes;
86$             DELAY:          norm(1440,120);
87$             ASSIGN:         a_breakdown=0;
30$             ROUTE:          0.0,a_alocation;

91$             ASSIGN:         a_los=1 * a_los;
will_work_fine  DELAY:          a_los:NEXT(54$);


35$             STATION,        DownfleetCity1Car1-DownFleetCity3Car2;
45$             DELAY:          a_adatetime - tnow;
13$             QUEUE,          M;
14$             SCAN:           NQ(e_baseindex + 26) > 0;
16$             COUNT:          e_baseindex + 23,1;
```

```
15$             REMOVE:         1,e_baseindex + 26,downfleetexit;
downfleetexit   DISPOSE;


47$             STATION,        MoveProcCity1Car1-MoveProcCity3Car2;
21$             QUEUE,          e_baseindex + 12;
22$             SCAN:           NQ( e_baseindex + 26 ) > 0;
23$             REMOVE:         1,e_baseindex + 26,33$;
individual_car_movement QUEUE, e_baseindex + 18:DETACH;
33$             DISPOSE;

                MATCH,:         CustCity2forCar1,51$:
                                carsCity2Car1;
                MATCH,:         CustCity2forCar2,53$:
                                carsCity2Car2;
                MATCH,:         CustCity3forCar1,51$:
                                carsCity3Car1;
                MATCH,:         CustCity3forCar2,53$:
                                carsCity3Car2;

55$             STATION,        Checkmoves;
60$             QUEUE,          e_baseindex + 32;
61$             SCAN:           (NQ(e_baseindex + 18) == a_quantity) .OR. ( tnow - a_adatetime > 720);
62$             WHILE:          NQ(e_baseindex + 18) > 0;
27$             COUNT:          e_baseindex + 17,1;
63$             REMOVE:         NQ(e_baseindex + 18),e_baseindex + 18,82$;
65$             ENDWHILE;
34$             ASSIGN:         v_movefleetflag(a_alocation,a_carrented)=0:
                                v_alocation_mfleet=0:
                                v_rlocation_mfleet=0:
                                v_carclass_mfleet=0:
                                v_adatetime_mfleet=0:
                                v_quantity_mfleet=0;
cleanincompletemoves WHILE:     NQ(e_baseindex + 12 )  > 0;
26$             COUNT:          e_baseindex + 11,1;
24$             REMOVE:         NQ( e_baseindex + 12 ),e_baseindex + 12,todispose;
25$             ENDWHILE;
41$             ASSIGN:         v_moving(a_alocation,a_carrented)=0;
40$             SIGNAL:         e_baseindex;
todispose       DISPOSE;

82$             ASSIGN:         a_objecttype=1:
                                a_alocation=a_rlocation:
                                v_totcostmovement=v_totcostmovement + a_costmovement;
64$             ROUTE:          0.0,a_rlocation;


Clock           CREATE,         1,0:1420;
58$             DISPOSE;


66$             CREATE,         1,0.0001:,1;
76$             ASSIGN:         v_carslocsid=1;
67$             WHILE:          v_carslocsid <= v_totcarslocs;
69$             SEARCH,         v_carslocsid,1,NQ(v_carslocsid):tnow - a_adatetime > 45;
68$             WHILE:          J <> 0;
70$             REMOVE:         J,v_carslocsid,countrenegedcustomers;
71$             SEARCH,         v_carslocsid,1,NQ(v_carslocsid):tnow - a_adatetime;
73$             ENDWHILE;
78$             BRANCH,         1:
                                If,v_carslocsid == v_totcarslocs,79$,Yes:
                                Else,77$,Yes;
79$             DELAY:          1;
77$             ASSIGN:         v_carslocsid=v_carslocsid + 1;
74$             ENDWHILE;
75$             DELAY:          60:NEXT(76$);

countrenegedcustomers COUNT:    e_base + a_car_about_to_rent + 35,1;
113$            BRANCH,         1:
```

```
                    If,a_alocation==1 .and. a_carreserved == 1,112$,Yes:
                    If,a_alocation==1  .and. a_carreserved == 2,114$,Yes:
                    If,a_alocation == 2 .and. a_carreserved == 1,115$,Yes:
                    If,a_alocation==2 .and. a_carreserved == 2,116$,Yes:
                    If,a_alocation==3 .and. a_carreserved == 1,117$,Yes:
                    Else,118$,Yes;
112$      ASSIGN:   v_dvl11=v(50) + (a_los/1440)*a_dailyrate;
119$      BRANCH,   1:
                    If,anint(a_idatetime/1440) == 1,120$,Yes:
                    If,anint(a_idatetime/1440) == 2,121$,Yes:
                    If,anint(a_idatetime/1440) == 3,122$,Yes:
                    If,anint(a_idatetime/1440) == 4,123$,Yes:
                    If,anint(a_idatetime/1440) == 5,124$,Yes:
                    If,anint(a_idatetime/1440) == 6,125$,Yes:
                    Else,126$,Yes;
120$      ASSIGN:   v(e_rulesindex)=1;
93$       WRITE,    reneged,"%f %f %f %f %f %f %f %f %f %f\n":
                    NC(e_base + a_car_about_to_rent + 35),
                    a_alocation,
                    a_carreserved,
                    a_los,
                    a_idatetime,
                    anint(a_idatetime/1440),
                    a_adatetime,
                    a_dow,
                    a_dailyrate,
                    (a_los/1440)*a_dailyrate;
72$       DISPOSE;

121$      ASSIGN:   v(6+e_rulesindex)=1:NEXT(93$);

122$      ASSIGN:   V(12+e_rulesindex)=1:NEXT(93$);

123$      ASSIGN:   V(18 + e_rulesindex)=1:NEXT(93$);

124$      ASSIGN:   V(24 + e_rulesindex)=1:NEXT(93$);

125$      ASSIGN:   V(30 + e_rulesindex)=1:NEXT(93$);

126$      ASSIGN:   V(36 + e_rulesindex)=1:NEXT(93$);

114$      ASSIGN:   v_dvl12=v(51) + (a_los/1440)*a_dailyrate:NEXT(119$);

115$      ASSIGN:   v_dvl21=v(52) + (a_los/1440)*a_dailyrate:NEXT(119$);

116$      ASSIGN:   v_dvl22=v(53) + (a_los/1440)*a_dailyrate:NEXT(119$);

117$      ASSIGN:   v_dvl31=v(54) + (a_los/1440)*a_dailyrate:NEXT(119$);

118$      ASSIGN:   v_dvl32=v(55) + (a_los/1440)*a_dailyrate:NEXT(119$);


127$      CREATE,   1,0:,1;
128$      DELAY:    14999;
131$      TALLY:    3,v_totnetrevenue,1;
132$      TALLY:    4,v(56) + v(62) + v(68) + v(74) + v(80) + v(86) + v(92),1;
133$      TALLY:    5,v(57) + v(63) + v(69) + v(75) + v(81) + v(87) + v(93),1;
134$      TALLY:    6,v(58) + v(64) + v(70) + v(76) + v(82) + v(88) + v(94),1;
135$      TALLY:    7,v(59) + v(65) + v(71) + v(77) + v(83) + v(89) + v(95),1;
136$      TALLY:    8,v(60) + v(66) + v(72) + v(78) + v(84) + v(90) + v(96),1;
137$      TALLY:    9,v(61) + v(67) + v(73) + v(79) + v(85) + v(91) + v(97),1;
138$      TALLY:    10,v(50),1;
139$      TALLY:    11,v(51),1;
140$      TALLY:    12,v(52),1;
141$      TALLY:    13,v(53),1;
142$      TALLY:    14,v(54),1;
143$      TALLY:    15,v(55),1;
130$      WRITE,    rulesstat,"%f %f %f %f %f %f %f %f %f %f\n":
                    v_dvl11/7,
```

```
                        v_dvl12/7 + v_dvl11/7,
                        v_dvl21/7,
                        v_dvl22/7 + v_dvl21/7,
                        v_dvl31/7,
                        v_dvl32/7 + v_dvl31/7,
                        v_totrevenue,
                        v_totnetrevenue,
                        v_totcostmovement;
129$            DISPOSE;
```

Experiment Frame

```
PROJECT,      Shortterm,Sonia R. Anorga,07/13/2000,Yes;

ATTRIBUTES:   1,a_dailyrate,0:
              2,a_alocation,0:
              3,a_carreserved,0:
              4,a_objecttype,0:
              5,a_los,0:
              6,a_adatetime,0:
              7,a_customers,0:
              8,a_timecounter,0:
              9,a_timewaitforcar:
              10,a_carrented,0:
              11,a_quantity,0:
              12,a_rlocation,0:
              13,a_ddatetime,0:
              14,a_car_about_to_rent,0:
              15,a_costmovement,0:
              16,a_dow,0:
              17,a_breakdown,0:
              18,a_idatetime:
              19,a_ilos;

FILES:        1,rdemand,"rdemand.txt",Sequential(),Free Format,Dispose,No,Hold:
              2,afleet,"afleet.txt",Sequential(),Free Format,Dispose,No,Hold:
              3,ifleet,"ifleet.txt",Sequential(),Free Format,Dispose,No,Hold:
              4,dfleet,"dfleet.txt",Sequential(),Free Format,Dispose,No,Hold:
              5,mfleet,"mfleet.txt",Sequential(),Free Format,Dispose,No,Hold:
              6,reneged,"reneged.txt",Sequential(),Free Format,Dispose,No,Hold:
              7,rulesstat,"rulesstat.txt",Sequential(),Free Format,Dispose,No,Hold;

VARIABLES:    1,v_timeend:
              2,v_alocation_afleet,0:
              3,v_carclass_afleet,0:
              4,v_quantity_afleet:
              5,v_alocation_rdemand:
              6,v_carclass_rdemand:
              7,v_los_rdemand:
              8,v_adatetime_rdemand:
              9,v_customers_rdemand:
              10,v_dailyrate_rdemand:
              11,v_alocation_ifleet,0:
              12,v_carclass_ifleet,0:
              13,v_adatetime_ifleet,0:
              14,v_quantity_ifleet,0:
              15,v_alocation_dfleet,0:
              16,v_carclass_dfleet,0:
              17,v_adatetime_dfleet,0:
              18,v_quantity_dfleet,0:
              19,v_alocation_mfleet,0:
              20,v_rlocation_mfleet,0:
              21,v_carclass_mfleet,0:
              22,v_adatetime_mfleet,0:
              23,v_quantity_mfleet,0:
              24,v_cost_mfleet,0:
              25,v_losindays,0:
              26,v_totlosindays,0:
              27,v_revenue,0:
              28,v_totrevenue,0:
              29,v_MaxBatches,10000:
              30,v_moving(3,2),0:
              36,v_movecounts,200000:
              37,v_movefleetflag(3,2),0:
              43,v_locations,3:
              44,v_carclasses,2:
              45,v_totcarslocs,6:
```

```
                46,v_carslocsid,0:
                47,v_costmovement,0:
                48,v_totcostmovement,0:
                49,v_totnetrevenue,500000:
                50,v_dvl11,0:
                51,v_dvl12,0:
                52,v_dvl21,0:
                53,v_dvl22,0:
                54,v_dvl31,0:
                55,v_dvl32,0:
                56,v_ndl_111,0:
                57,v_ndl_121,0:
                58,v_ndl_211,0:
                59,v_ndl_221,0:
                60,v_ndl_311,0:
                61,v_ndl_321,0:
                62,v_ndl_112,0:
                63,v_ndl_122,0:
                64,v_ndl_212,0:
                65,v_ndl_222,0:
                66,v_ndl_312,0:
                67,v_ndl_322,0:
                68,v_ndl_113,0:
                69,v_ndl_123:
                70,v_v70,0:
                71,v_v71:
                72,v_v72:
                73,v_v73:
                74,v_v74:
                75,v_v75:
                76,v_v76,0:
                77,v_v77,0:
                78,v_v78,0:
                79,v_v79,0:
                80,v_v80:
                81,v_v81:
                82,v_v82:
                83,v_v83,0:
                84,v_v84,0:
                85,v_v85,0:
                86,v_v86:
                87,v_v87:
                88,v_v88,0:
                89,v_v89,0:
                90,v_v90,0:
                91,v_v91,0:
                92,v_v92:
                93,v_v93:
                94,v_v94,0:
                95,v_v95,0:
                96,v_v96,0:
                97,v_v97,0:
                98,v_v98,0;

QUEUES:         1,q_custCity1forCar1,FirstInFirstOut:
                2,q_custCity1forCar2,FirstInFirstOut:
                3,q_custCity2forCar1,FirstInFirstOut:
                4,q_custCity2forCar2,FirstInFirstOut:
                5,q_custCity3forCar1,FirstInFirstOut:
                6,q_custCity3forCar2,FirstInFirstOut:
                7,q_waitdfleetCity1Car1,FirstInFirstOut:
                8,q_waitdfleetCity1Car2,FirstInFirstOut:
                9,q_waitdfleetCity2Car1,FirstInFirstOut:
                10,q_waitdfleetCity2Car2,FirstInFirstOut:
                11,q_waitdfleetCity3Car1,FirstInFirstOut:
                12,q_waitdfleetCity3Car2,FirstInFirstOut:
                13,q_waitmoveCity1Car1,FirstInFirstOut:
                14,q_waitmoveCity1Car2,FirstInFirstOut:
                15,q_waitmoveCity2Car1,FirstInFirstOut:
```

```
            16,q_waitmoveCity2Car2,FirstInFirstOut:
            17,q_waitmoveCity3Car1,FirstInFirstOut:
            18,q_waitmoveCity3Car2,FirstInFirstOut:
            19,q_moveCity1Car1,FirstInFirstOut:
            20,q_moveCity1Car2,FirstInFirstOut:
            21,q_moveCity2Car1,FirstInFirstOut:
            22,q_moveCity2Car2,FirstInFirstOut:
            23,q_moveCity3Car1,FirstInFirstOut:
            24,q_moveCity3Car2,FirstInFirstOut:
            25,q_notused1,FirstInFirstOut:
            26,q_waitentersys,FirstInFirstOut:
            27,q_carsCity1Car1,FirstInFirstOut:
            28,q_carsCity1Car2,FirstInFirstOut:
            29,q_carsCity2Car1,FirstInFirstOut:
            30,q_carsCity2Car2,FirstInFirstOut:
            31,q_carsCity3Car1,FirstInFirstOut:
            32,q_carsCity3Car2,FirstInFirstOut:
            33,q_orderCity1Car1,FirstInFirstOut:
            34,q_orderCity1Car2,FirstInFirstOut:
            35,q_orderCity2Car1,FirstInFirstOut:
            36,q_orderCity2Car2,FirstInFirstOut:
            37,q_orderCity3Car1,FirstInFirstOut:
            38,q_orderCity3Car2,FirstInFirstOut:
            39,q_waitcheckoutcity1,FirstInFirstOut:
            40,q_waitcheckoutcity2,FirstInFirstOut:
            41,q_waitcheckoutcity3,FirstInFirstOut;

RESOURCES:  1,r_free1,Capacity(1,),-,Stationary:
            2,r_checkscan,Capacity(1,),-,Stationary:
            3,r_rentalCity1,Capacity(200,),-,Stationary:
            4,r_rentalCity2,Capacity(200,),-,Stationary:
            5,r_rentalCity3,Capacity(200,),-,Stationary;

STATIONS:   1,LotCity1:
            2,LotCity2:
            3,LotCity3:
            4,CounterCity1:
            5,CounterCity2:
            6,CounterCity3:
            7,DownfleetCity1Car1:
            8,DownfleetCity1Car2:
            9,DownfleetCity2Car1:
            10,DownfleetCity2Car2:
            11,DownfleetCity3Car1:
            12,DownFleetCity3Car2:
            13,MoveProcCity1Car1:
            14,MoveProcCity1Car2:
            15,MoveProcCity2Car1:
            16,MoveProcCity2Car2:
            17,MoveProcCity3Car1:
            18,MoveProcCity3Car2:
            19,MoveConCity1Car1:
            20,MoveConCity1Car2:
            21,MoveConCity2Car1:
            22,MoveConCity2Car2:
            23,MoveConCity3Car1:
            24,MoveConCity3Car2:
            25,checkmoves;

COUNTERS:   1,c_custCity1forCar1,,Replicate:
            2,c_custCity1forCar2,,Replicate:
            3,c_CustCity2forCar1,,Replicate:
            4,c_custCity2forCar2,,Replicate:
            5,c_CustCity3forCar1,,Replicate:
            6,c_CustCity3forCar2,,Replicate:
            7,c_notused1,,Replicate:
            8,c_notused2,,Replicate:
            9,c_rentalsCity1,,Replicate:
            10,c_rentalsCity2,,Replicate:
```

```
11,c_rentalsCity3,,Replicate:
12,c_notmovedCity1Car1,,Replicate:
13,c_notmovedCity1Car2,,Replicate:
14,c_notmovedCity2Car1,,Replicate:
15,c_notmovedCity2Car2,,Replicate:
16,c_notmovedCity3Car1,,Replicate:
17,c_notmovedCity3Car2,,Replicate:
18,c_movedCity1Car1,,Replicate:
19,c_movedCity1Car2,,Replicate:
20,c_movedCity2Car1,,Replicate:
21,c_movedCity2Car2,,Replicate:
22,c_movedCity3Car1,,Replicate:
23,c_movedCity3Car2,,Replicate:
24,c_dfleetCity1Car1,,Replicate:
25,c_dfleetCity1Car2,,Replicate:
26,c_dfleetCity2Car1,,Replicate:
27,c_dfleetCity2Car2,,Replicate:
28,c_dfleetCity3Car1,,Replicate:
29,c_dfleetCity3Car2,,Replicate:
30,c_carsCity1Car1,,Replicate:
31,c_carsCity1Car2,,Replicate:
32,c_carsCity2Car1,,Replicate:
33,c_carsCity2Car2,,Replicate:
34,c_carsCity3Car1,,Replicate:
35,c_carsCity3Car2,,Replicate:
36,c_renegeCity1Car1,,Replicate:
37,c_renegeCity1Car2,,Replicate:
38,c_renegeCity2Car1,,Replicate:
39,c_renegeCity2Car2,,Replicate:
40,c_renegeCity3Car1,,Replicate:
41,c_renegeCity3Car2,,Replicate:
42,c_numcustleavecity1,,Replicate:
43,c_numcustleavecity2,,Replicate:
44,c_numcustleavecity3,,Replicate:
45,c_numcheckincity1,,Replicate:
46,c_numcheckincity2,,Replicate:
47,c_numcheckincity3,,Replicate:
48,c_city1res1rent1tim1,,Replicate:
49,c_city1res1rent2tim1,,Replicate:
50,c_city1res2rent2tim1,,Replicate:
51,c_city2res1rent1tim1,,Replicate:
52,c_city2res1rent2tim1,,Replicate:
53,c_city2res2rent2tim1,,Replicate:
54,c_city3res1rent1tim1,,Replicate:
55,c_city3res1rent2tim1,,Replicate:
56,c_city3res2rent2tim1,,Replicate:
57,c_city1res1rent1tim2,,Replicate:
58,c_city1res1rent2tim2,,Replicate:
59,c_city1res2rent2tim2,,Replicate:
60,c_city2res1rent1tim2,,Replicate:
61,c_city2res1rent2tim2,,Replicate:
62,c_city2res2rent2tim2,,Replicate:
63,c_city3res1rent1tim2,,Replicate:
64,c_city3res1rent2tim2,,Replicate:
65,c_city3res2rent2tim2,,Replicate:
66,c_city1res1rent1tim3,,Replicate:
67,c_city1res1rent2tim3,,Replicate:
68,c_city1res2rent2tim3,,Replicate:
69,c_city2res1rent1tim3,,Replicate:
70,c_city2res1rent2tim3,,Replicate:
71,c_city2res2rent2tim3,,Replicate:
72,c_city3res1rent1tim3,,Replicate:
73,c_city3res1rent2tim3,,Replicate:
74,c_city3res2rent2tim3,,Replicate:
75,c_city1res1rent1time4,,Replicate:
76,c_city1res1rent2time4,,Replicate:
77,c_city1res2rent2time4,,Replicate:
78,c_city2res1rent1time4,,Replicate:
79,c_city2res1rent2time4,,Replicate:
```

```
80,c_city2res2rent2time4,,Replicate:
81,c_city3res1rent1time4,,Replicate:
82,c_city3res1rent2time4,,Replicate:
83,c_city3res2rent2time4,,Replicate:
84,c_city1res1rent1time5,,Replicate:
85,c_city1res1rent2time5,,Replicate:
86,c_city1res2rent2time5,,Replicate:
87,c_87,,Replicate:
88,c_88,,Replicate:
89,c_89,,Replicate:
90,c_90,,Replicate:
91,c_91,,Replicate:
92,c_92,,Replicate:
93,c_93,,Replicate:
94,c_94,,Replicate:
95,c_95,,Replicate:
96,c_96,,Replicate:
97,c_97,,Replicate:
98,c_98,,Replicate:
99,c_99,,Replicate:
100,c_100,,Replicate:
101,c_101,,Replicate:
102,c_102,,Replicate:
103,c_103,,Replicate:
104,c_104,,Replicate:
105,a_105,,Replicate:
106,a_106,,Replicate:
107,c_107,,Replicate:
108,c_108,,Replicate:
109,c_109,,Replicate:
110,c_110,,Replicate:
111,c_breakdowns,,Replicate:
112,c_112,,Replicate:
113,c_113,,Replicate:
114,c_114,,Replicate:
115,c_115,,Replicate:
116,c_116,,Replicate:
117,c_117,,Replicate:
118,c_118,,Replicate:
119,c_119,,Replicate:
120,c_120,,Replicate:
121,c_121,,Replicate:
122,c_122,,Replicate:
123,c_123,,Replicate:
124,c_124,,Replicate:
125,c_125,,Replicate:
126,c_126,,Replicate:
127,c_127,,Replicate:
128,c_128,,Replicate:
129,c_129,,Replicate:
130,c_130,,Replicate:
131,c_131,,Replicate:
132,c_132,,Replicate:
133,c_133,,Replicate:
134,c_134,,Replicate:
135,c_135,,Replicate:
136,c_136,,Replicate:
137,c_137,,Replicate:
138,c_138,,Replicate:
139,c_139,,Replicate:
140,c_140,,Replicate:
141,c_141,,Replicate:
142,c_142,,Replicate:
143,c_143,,Replicate:
144,c_144,,Replicate:
145,c_145,,Replicate:
146,c_146,,Replicate:
147,c_147,,Replicate:
148,c_148,,Replicate:
```

```
            149,c_149,,Replicate:
            150,c_150,,Replicate:
            151,c_151,,Replicate:
            152,c_152,,Replicate:
            153,c_153,,Replicate:
            154,c_154,,Replicate:
            155,c_155,,Replicate:
            156,c_156,,Replicate:
            157,c_157,,Replicate:
            158,c_158,,Replicate:
            159,c_159,,Replicate:
            160,c_160,,Replicate:
            161,c_161,,Replicate:
            162,c_162,,Replicate:
            163,c_163,,Replicate:
            164,c_164,,Replicate:
            165,c_165,,Replicate:
            166,c_166,,Replicate:
            167,c_167,,Replicate:
            168,c_168,,Replicate:
            169,c_169,,Replicate:
            170,c_170,,Replicate:
            171,c_171,,Replicate:
            172,c_172,,Replicate:
            173,c_173,,Replicate:
            174,c_174,,Replicate:
            175,c_initfleet_11,,Replicate:
            176,c_initfleet_12,,Replicate:
            177,c_initfleet21,,Replicate:
            178,c_initfleet_22,,Replicate:
            179,c_initfleet_31,,Replicate:
            180,c_initfleet_32,,Replicate;

TALLIES:    1,t_timewaitforcar:
            2,t_timecounter:
            3,t_netrevenue:
            4,t_dlostloc1car1:
            5,t_dlostloc1car2:
            6,t_dlostloc2car1:
            7,t_dlostloc2car2:
            8,t_dlostloc3car1:
            9,t_dlostloc3car2:
            10,t_dvl11:
            11,t_dvl12:
            12,t_dvl21:
            13,t_dvl22:
            14,t_dvl31:
            15,t_dvl32;

DSTATS:     1,NQ(q_carsCity1Car1):
            2,NQ(q_carsCity1Car2):
            3,NQ(q_carsCity2Car1):
            4,NQ(q_carsCity2Car2):
            5,NQ(q_carsCity3Car1):
            6,NQ(q_carsCity3Car2):
            7,NQ(q_custCity1forCar1):
            8,NQ(q_custCity1forCar2):
            9,NQ(q_custCity2forCar1):
            10,NQ(q_custCity2forCar2):
            11,NQ(q_custCity3forCar1):
            12,NQ(q_custCity3forCar2):
            13,NR(r_rentalCity1):
            14,NR(r_rentalCity2):
            15,NR(r_rentalCity3);

OUTPUTS:    1,v(26),"runningdays.dat",Total running days:
            2,v(28),"totrevenue.dat",Total rental revenue:
            3,v(48),"totcostmovement.dat",Total cost of movement:
            4,v(49),"totnetrevenue.dat",Total Net Revenue;
```

```
REPLICATE,     10,0,15000,Yes,Yes,0.0;

EXPRESSIONS:  1,e_baseindex,2*(a_alocation - 1 ) + a_carrented:
              2,e_base,2*(a_alocation - 1):
              3,e_rbaseindex,2*(a_rlocation - 1) + a_carrented:
              4,e_custtype,3*(a_alocation - 1) + a_carreserved + a_carrented:
              5,e_city1dow1,

cont(0.05,0,0.06,240,0.08,360,0.12,480,0.29,600,0.49,720,0.64,840,0.74,960,0.86,1080,0.94,1200,1,132
0):
              6,e_city2dow1,cont(0.04,360,0.15,480,0.26,600,0.41,720,0.74,840,0.85,960,1,1080):
              7,e_city3dow1,

cont(0.01,0,0.02,120,0.04,240,0.1,360,0.16,480,0.27,600,0.42,720,0.53,840,0.72,960,0.83,1080,0.93,12
00,1,1320):

8,e_city1dow2,cont(0.02,0,0.04,360,0.12,480,0.27,600,0.42,720,0.59,840,0.72,960,0.81,1080,0.89,1200,
1,1320):

9,e_city2dow2,cont(0.06,360,0.12,480,0.25,600,0.54,720,0.67,840,0.8,960,0.96,1080,1,1200  ):
              10,e_city3dow2,

cont(0.04,0,0.06,120,0.08,240,0.1,360,0.16,480,0.3,600,0.44,720,0.57,840,0.74,960,0.86,1080,0.91,120
0,1,1320):

11,e_city1dow3,cont(0.04,0,0.06,360,0.13,480,0.35,600,0.5,720,0.69,840,0.78,960,0.84,1080,0.91,1200,
1,1320):

12,e_city2dow3,cont(0.11,360,0.16,480,0.27,600,0.43,720,0.69,840,0.74,960,0.95,1080,1,1200):
              13,e_city3dow3,

cont(0.04,0,0.06,120,0.08,240,0.1,360,0.16,480,0.3,600,0.44,720,0.57,840,0.74,960,0.86,1080,0.91,120
0,1,1320):

14,e_city1dow4,cont(0.02,0,0.03,360,0.1,480,0.22,600,0.4,720,0.57,840,0.69,960,0.79,1080,0.89,1200,1
,1320):
              15,e_city2dow4,cont(0.05,360,0.1,480,0.26,600,0.5,720,0.74,840,0.85,960,1,1080  ):
              16,e_city3dow4,

cont(0.04,0,0.07,240,0.12,360,0.17,480,0.3,600,0.43,720,0.55,840,0.72,960,0.84,1080,0.91,1200,1,1320
):

17,e_city1dow5,cont(0.03,0,0.04,360,0.09,480,0.27,600,0.47,720,0.59,840,0.7,960,0.79,1080,0.88,1200,
1,1320):

18,e_city2dow5,cont(0.08,360,0.24,480,0.34,600,0.56,720,0.72,840,0.86,960,0.94,1080,1,1200):
              19,e_city3dow5,

cont(0.02,0,0.03,120,0.06,240,0.1,360,0.15,480,0.28,600,0.44,720,0.56,840,0.69,960,0.83,1080,0.91,12
00,1,1320):
              20,e_city1dow6,

cont(0.04,0,0.05,120,0.07,360,0.13,480,0.27,600,0.45,720,0.61,840,0.71,960,0.83,1080,0.91,1200,1,132
0):
              21,e_city2dow6,cont(0.13,480,0.21,600,0.41,720,0.66,840,0.84,960,0.97,1080,1,1200):
              22,e_city3dow6,

cont(0.05,0,0.07,120,0.09,240,0.13,360,0.19,480,0.32,600,0.45,720,0.58,840,0.74,960,0.83,1080,0.91,1
200,1,1320):

23,e_city1dow7,cont(0.07,0,0.09,360,0.15,480,0.29,600,0.48,720,0.64,840,0.75,960,0.85,1080,0.93,1200
,1,1320):
              24,e_city2dow7,cont(0.09,480,0.26,600,0.43,720,0.6,840,0.86,960,1,1080):
              25,e_city3dow7,

cont(0.04,0,0.05,120,0.08,240,0.12,360,0.17,480,0.29,600,0.42,720,0.54,840,0.73,960,0.87,1080,0.94,1
200,1,1320):
              26,e_staydow1,cont(0.35,0.2,0.46,-0.2,1,0):
```

```
              27,e_staydow2,cont(0.45,0.2,0.57,-0.2,1,0):
              28,e_staydow3,cont(0.39,0.2,0.51,-0.2,1,0):
              29,e_staydow4,cont(0.51,0.2,0.63,-0.2,1,0):
              30,e_staydow5,cont(0.43,0.2,0.55,-0.2,1,0):
              31,e_staydow6,cont(0.49,0.2,0.61,-0.2,1,0):
              32,e_staydow7,cont(0.5,0.2,0.62,-0.2,1,0):
              33,e_baseindexres,2*(a_alocation-1) + a_carreserved + 49:
              34,e_rulesindex,2*(a_alocation-1) + a_carreserved + 55;

REPORTS:      1,report1,"report1.txt",report1,,Unsorted,Free:
              2,report2,"report2.txt",report2,,Unsorted,Free;

REPORTLINES:  1,line1,report1,"end of simulation at time %f\n",v_timeend:
              2,line2,report1,"total running days: %f \n",v_totlosindays:
              3,line3,report1,"total revenue: %f \n",v_totrevenue:
              4,line4,report1," locations * carclasses : %f\n",v_totcarslocs:
              5,line5,report1," Tot net revenue: %f\n",v_totnetrevenue:
              6,line6,report1,"daily rev lost in loc 1 for absence of car 1: %f\n",v_dvl11/7:
              7,line7,report1,"daily rev lost in loc 1 for absence of car 2: %f\n",v_dvl11/7 +
v_dvl12/7:
              8,line8,report1,"daily rev lost in loc 2 for absence of car 1: %f\n",v_dvl21/7:
              9,line9,report1,"daily rev lost in loc 2 for absence of car 2: %f\n",v_dvl21/7 +
v_dvl22/7:
              10,line10,report1,"daily rev lost in loc 3 for absence of car 1: %f\n",v_dvl31/7:
              11,line11,report1,"daily rev lost in loc 3 for absence of car 2: %f\n",v_dvl31/7 +
v_dvl32/7:
              12,line12,report1,"loc1 car1 %f\n",V (56) + V (62) + V (68) + V (74) + V (80) + V
(86):
              13,line13,report1,"loc1car2 %f\n",X (2) + X (8) + X (14) + X (21) + X (28) + X (35):
              14,line14,report1,"loc3car1 %f\n",V (60) + V (66) + V (72) + V (78) + V (84) + V (90):
              15,line15,report2,"%f,%f,%f,%f,%f,%f\n",v_dvl11/7,v_dvl12/7 +
v_dvl11/7,v_dvl21/7,v_dvl22/7 + v_dvl21/7,v_dvl31/7,
              v_dvl32/7 + v_dvl31/7;
```

APPENDIX D

It contains an example of the Arena model's input files.

Demand File

System name: Rdemand.txt

Field	Description	Data type
v_alocation_rdemand	Location	Integer
v_carclass_rdemand	Car Class	Integer
v_los_rdemand	Time Period	Integer
v_adatetime_rdemand	RC demand	Integer
v_customers_rdemand	Price	Integer
v_dailyrate_rdemand	LOS	Integer

Table 78: Demand file structure.

Values:

```
1       2       4320    1       51      26
2       2       4320    1       2       20
3       2       4320    1       168     33
1       1       4320    1       83      17
2       1       4320    1       3       23
3       1       4320    1       149     22
1       2       1440    1       113     33
2       2       1440    1       9       12
3       2       1440    1       275     46
1       1       1440    1       185     26
2       1       1440    1       11      16
3       1       1440    1       243     28
1       2       4320    1441    53      22
2       2       4320    1441    2       7
3       2       4320    1441    113     29
1       1       4320    1441    55      17
2       1       4320    1441    2       14
3       1       4320    1441    113     21
1       2       1440    1441    198     32
2       2       1440    1441    16      18
3       2       1440    1441    252     37
1       1       1440    1441    206     22
2       1       1440    1441    15      16
3       1       1440    1441    252     32
1       2       4320    2881    37      25
2       2       4320    2881    2       13
3       2       4320    2881    101     28
1       1       4320    2881    40      19
2       1       4320    2881    3       16
3       1       4320    2881    110     21
1       2       1440    2881    139     30
2       2       1440    2881    8       12
3       2       1440    2881    237     33
1       1       1440    2881    151     25
2       1       1440    2881    9       31
3       1       1440    2881    256     25
1       2       4320    4321    84      26
2       2       4320    4321    3       22
3       2       4320    4321    193     30
1       1       4320    4321    88      20
```

2	1	4320	4321	4	13
3	1	4320	4321	178	23
1	2	1440	4321	267	35
2	2	1440	4321	13	23
3	2	1440	4321	193	37
1	1	1440	4321	277	26
2	1	1440	4321	18	24
3	1	1440	4321	178	28
1	2	4320	5761	110	27
2	2	4320	5761	3	25
3	2	4320	5761	144	33
1	1	4320	5761	106	21
2	1	4320	5761	3	29
3	1	4320	5761	18	24
1	2	1440	5761	331	35
2	2	1440	5761	19	21
3	2	1440	5761	324	38
1	1	1440	5761	318	27
2	1	1440	5761	21	24
3	1	1440	5761	265	28
1	2	4320	7201	172	28
2	2	4320	7201	5	19
3	2	4320	7201	227	34
1	1	4320	7201	194	21
2	1	4320	7201	7	14
3	1	4320	7201	201	23
1	2	1440	7201	366	40
2	2	1440	7201	15	18
3	2	1440	7201	198	43
1	1	1440	7201	305	26
2	1	1440	7201	20	19
3	1	1440	7201	174	29
1	2	4320	8641	95	27
2	2	4320	8641	2	25
3	2	4320	8641	236	34
1	1	4320	8641	111	17
2	1	4320	8641	2	15
3	1	4320	8641	226	22
1	2	1440	8641	211	29
2	2	1440	8641	11	26
3	2	1440	8641	155	39
1	1	1440	8641	248	21
2	1	1440	8641	13	17
3	1	1440	8641	126	26

Initial Fleet File

System Name: afleet.txt

Field	Description	Field Type
v_alocation_afleet	Location	Integer
v_carclass_afleet	Car Class	Integer
v_quantity_afleet	Quantity	Integer

Table 79: Initial Fleet layout.

Values:

```
1          1          404
1          2          594
2          1          25
2          2          25
3          1          466
3          2          896
```

Movement File

System Name: mfleet.txt

Field	Description	Data Type
v_alocation_mfleet	Source location	integer
v_rlocation_mfleet	Target location	integer
v_carclass_mfleet	Car class	integer
v_adatetime_mfleet	Time of movement	number
v_quantity_mfleet	Quantity to move	integer
v_cost_mfleet	Cost of movement	integer

Table 80: Movement file layout.

Values:

```
1     3     2     0.5   191   20
2     3     1     0.5   14    30
2     3     2     0.5   14    30
```

APPENDIX E

It contains the VBA code within the Arena Model.

Arena Objects

```
This Document
Option Explicit
'
  Dim ModelObjects As Arena.SIMAN
'
  Dim TheModel As Arena.Model

Private Function ModelLogic_RealTimeTerminate() As Long

End Function

Private Sub ModelLogic_RunBegin()
Dim ObjectIndex, l, c, s, t, w, i, j, k, prevprkey, prkey, location, time, week, tu As Integer
Dim price(2, 1, 1, 6), factkey, sample, iaux, lf, lt, NumOfReps As Integer
Dim nsh(2, 6, 2), can(2, 6, 2), car(2, 6, 2, 1), los(2, 6, 2, 1), aux(5) As Double
Dim tunc(2, 6, 24), runc(2, 6, 24) As Double

Dim sMove, sMovebkp, sProj, sPrice, sFactor, sSample, STemp, sCost As String
Dim vLoc1, vLoc2, vCarClass, vQtyCar As Integer
Dim vTime, vCost As Double
'
' Connecting to the model,
'
Set TheModel = ThisDocument.Model
'
' SIMAN Objects are not yet accessible
'
'=====================================================
' If this is the very first time iteration, then

'=====================================================

If IterationNumber < 1 Then
   Call Create_proj
   Call Create_Proj_Files
   Call Create_Opt_Inputr
   Call Mod_Opt_Inputr
End If

If IterationNumber < 1 Then
  ' MsgBox " RunBegin "
   Call Init_Heuristic
   MaxIteration = 8 ' 9
'
' Displaying the form to collect user's inputs
' the MainForm form retains control of the execution until it is hidden
' The form is hidden when the user clicks on the OK command button
'
   Load MainForm
   MainForm.Show
'
' Getting the value for the initial resource provided by the user
' which was entered by the user in the TEXTBOX with the name InitResCap in MainForm
'
'   MaxIteration = valf(MainForm.NumIter.Text)
   NumOfReps = vAL(MainForm.NumReps.Text)
'
   IterationNumber = 1

'  Saves original moves

   sMove = "mfleet.txt"
   Open sMove For Input As #1
   sMovebkp = "mfleetbkp" + Format$(IterationNumber, "00") + ".txt"
   Open sMovebkp For Output As #2
```

```
    Do While Not EOF(1)
         Input #1, vLoc1, vLoc2, vCarClass, vTime, vQtyCar, vCost
         Print #2, vLoc1; vLoc2; vCarClass; vTime; vQtyCar; vCost
         vLoc1 = vLoc1 - 1
         vLoc2 = vLoc2 - 1
         vCarClass = vCarClass - 1
         mv_opt(vLoc1, vLoc2, vCarClass, 0) = 1  'exists
         mv_opt(vLoc1, vLoc2, vCarClass, 1) = vLoc1 'from location
         mv_opt(vLoc1, vLoc2, vCarClass, 2) = vLoc2 ' to location
         mv_opt(vLoc1, vLoc2, vCarClass, 3) = vCarClass  ' car class to be transferred
         mv_opt(vLoc1, vLoc2, vCarClass, 4) = vQtyCar ' qty to be transferred
         mv_opt(vLoc1, vLoc2, vCarClass, 5) = vCost ' Cost
    Loop

    Close #1
    Close #2
'   Get cost
    sCost = "cost.txt"
    Open sCost For Input As #1
    For lf = 0 To 2
         For lt = 0 To 2
             If Not EOF(1) Then
                 If lt = lf Then
                     cost(lf, lt) = 0
                 Else
                     Input #1, i, j, cost(lf, lt)
                 End If
             End If
         Next lt
    Next lf
    Close #1

' Finding the REPLICATE element
'
   ObjectIndex = TheModel.Modules.Find(smFindTag, "ReplicateElement")
'
' Connecting to the REPLICATE element
'
   Set ReplicateModule = TheModel.Modules(ObjectIndex)
'
' Setting the value of the number of replications
'
   ReplicateModule.Data("NumReps") = NumOfReps
   ReplicateModule.UpdateShapes

   Unload MainForm
End If ' IterationNumber was less than 1

If IterationNumber > 1 Then
    sMove = "mfleet.txt"
    Open sMove For Input As #1
    sMovebkp = "mfleetbkp" + Format$(IterationNumber, "00") + ".txt"
    Open sMovebkp For Output As #2
    Do While Not EOF(1)
         Input #1, vLoc1, vLoc2, vCarClass, vTime, vQtyCar, vCost
         Print #2, vLoc1; vLoc2; vCarClass; vTime; vQtyCar; vCost
    Loop
    Close #1
    Close #2
End If ' IterationNumber greater than 1

'MsgBox "RunBegin; iteration #" & IterationNumber'
End Sub


Private Sub ModelLogic_RunBeginReplication()
Dim j As Integer
Dim drevenue As Double
```

```
'
Set TheModel = ThisDocument.Model

vNRep = ModelObjects.RunCurrentReplication
vMrep = ModelObjects.RunMaximumReplications

If IterationNumber = MaxIteration And vNRep = 1 Then
   MsgBox "Last iteration; to run for " & vMrep & " Replications now"
End If

End Sub

Private Sub ModelLogic_RunBeginSimulation()

'MsgBox "This is run begin simulation"

Set ModelObjects = TheModel.SIMAN
Call Init_Simulation

End Sub


Private Sub ModelLogic_RunEnd()
    Dim i, j, w, z As Integer
    Dim STemp, sMove As String
    Dim TotNet, vTotRev, vTotNetRev, vTotCostMov, vDVLmax As Double
    Dim vaux(9), vDDLflag(2, 1, 6), k, l, c, vrandom, vDDLmax, vDCLmax, vDVLloc As Integer
    Dim listloc(1, 2), listmvinc(1, 2, 3) As Integer
'================================================================
'Prepares for a new Iteration, if next is the last one then will
'run the best Iteration again
'================================================================
If IterationNumber < MaxIteration Then
   'iterate again
   IterationNumber = IterationNumber + 1
   TheModel.Go
End If

If IterationNumber = MaxIteration Then
  'Clear global variables
  Call Clear_Run
End If

End Sub

Private Sub ModelLogic_RunEndReplication()
    Dim i, j, w, z, list_index As Integer
    Dim STemp As String
    Dim vCounter(2, 1) As Integer
    Dim vaux(9), k, l, c As Integer
    Dim vTotRev, vTotNetRev, vTotCostMov, Tot1, Tot2, Tot3, drevenue As Double

' get the value of NREP and MREP to compare them if last iteration
   vNRep = ModelObjects.RunCurrentReplication
   vMrep = ModelObjects.RunMaximumReplications
'MsgBox "This is run End replication"

   Set TheModel = ThisDocument.Model
   Set ModelObjects = TheModel.SIMAN
   Set Out = TheModel.SIMAN

   list_index = vNRep - 1

   ' j = TheModel.Modules.Find(smFindTag, "modvars")

   ' If j > 0 Then
   '     Set VariableModule = TheModel.Modules(j)
   ' Else
   '     MsgBox " Could not find variables element"
```

```
    ' End If
    '
    ' drevenue = vAL(VariableModule.Data("value(1,49)"))
    '
    ' Stop

If IterationNumber = 1 Then
        '================================================================
        'Gets 4 values for heuristic.  3 values are obtained at the end of
        'each replication.  When last replication occurs the 4th value is
        'extracted as well as averages for the three previous ones.
        '================================================================
        '1 - Daily Customer lost
        For l = 0 To 2
            For c = 0 To 1
                w = l * 2 + c
                vDCL_list(l, c, list_index) = CInt(ModelObjects.CounterValue(36 + w) / 7)
            Next c
        Next l
        vDCL_list(0, 1, list_index) = vDCL_list(0, 1, list_index) + vDCL_list(0, 0, list_index)
'Adjust for upgrades
        vDCL_list(1, 1, list_index) = vDCL_list(1, 1, list_index) + vDCL_list(1, 0, list_index)
        vDCL_list(2, 1, list_index) = vDCL_list(2, 1, list_index) + vDCL_list(2, 0, list_index)
        '2 - Total Days Lost
        j = 3
        For l = 0 To 2
            For c = 0 To 1
                j = j + 1
                vDDL_list(l, c, list_index) = vAL(Out.TallyAverage(j))
            Next c
        Next l
        '3 - Daily Revenue Lost
        j = 9
        For l = 0 To 2
            For c = 0 To 1
                j = j + 1
                vDVL_list(l, c, list_index) = vAL(Out.TallyAverage(j)) / 7
            Next c
        Next l
    If vNRep = vMrep Then
        '================================================================
        'load remaining 4th value and obtains averages needed for heuristic
        '================================================================
        '4 - Net Initial Fleet
        For l = 0 To 2
            For c = 0 To 1
                w = l * 2 + c
                vCounter(l, c) = ModelObjects.CounterValue(175 + w) - ModelObjects.CounterValue(18 +
w)
                If vCounter(l, c) >= 0 Then vALF(l, c) = vCounter(l, c) Else vALF(l, c) = 0
            Next c
        Next l
        ' Gets averages
        Tot1 = 0
        Tot2 = 0
        Tot3 = 0
        For l = 0 To 2
            For c = 0 To 1
                For i = 0 To list_index
                    Tot1 = Tot1 + vDCL_list(l, c, i)
                    Tot2 = Tot2 + vDDL_list(l, c, i)
                    Tot3 = Tot3 + vDVL_list(l, c, i)
                Next i
                vDCL(l, c) = Tot1 / vMrep
                vDDL(l, c) = Tot2 / vMrep
                vDVL(l, c) = Tot3 / vMrep
            Next c
        Next l
        'write in Simresult
```

```
        STemp = "Simresult.dat"
        Open STemp For Output As #1
        Write #1, "AL "; vALF(0, 0); vALF(0, 1); vALF(1, 0); vALF(1, 1); vALF(2, 0); vALF(2, 1)
        Write #1, "DCL "; vDCL(0, 0); vDCL(0, 1); vDCL(1, 0); vDCL(1, 1); vDCL(2, 0); vDCL(2, 1)
        Write #1, "DVL "; vDVL(0, 0); vDVL(0, 1); vDVL(1, 0); vDVL(1, 1); vDVL(2, 0); vDVL(2, 1)
        Write #1, "DDL "; vDDL(0, 0); vDDL(0, 1); vDDL(1, 0); vDDL(1, 1); vDDL(2, 0); vDDL(2, 1)
        Close #1
   End If
End If 'Iteration Number 1 and last replication

'================================================================
'Gets the total Net Revenue from replication
'================================================================

TotNetRev_list(list_index) = vAL(Out.TallyAverage(3))

'================================================================
'Stops Simulation when it is the last replication
'================================================================
If vNRep = vMrep Then
   TheModel.End
End If

End Sub

Private Sub ModelLogic_RunEndSimulation()
    Dim TotNet As Double
    Dim i, j, list_max_index As Integer
    Dim w, z As Integer
    Dim STemp, sMove As String
    Dim vTotRev, vTotNetRev, vTotCostMov, vDVLmax As Double
    Dim vaux(9), vDDLflag(2, 1, 6), k, l, c, vrandom, vDDLmax, vDCLmax, vDVLloc As Integer
    Dim listloc(1, 2), listmvinc(1, 2, 3) As Integer


'MsgBox "This is run End Simulation"

'================================================================
'Gets average net revenue from run
'================================================================
TotNet = 0
list_max_index = vMrep - 1
For i = 0 To list_max_index
    TotNet = TotNet + TotNetRev_list(i)
Next i
AvgNetRev = TotNet / vMrep

'================================================================
'Evaluates which is the best iteration so far
'Gets maximum Net Revenue
'================================================================
If AvgNetRev > vBestNetRev Then
   vBestNetRev = AvgNetRev
   '  vBestRev = vTotRev
   '  vBestCostMov = vTotCostMov
   vBestIteration = IterationNumber
End If

'================================================================
'Runs Heuristic during first Iteration
'================================================================
If IterationNumber = 1 Then
    ' Establishes heuristic values
    '1- Get O()
    '----------
    vDVLmax = -99999999
    vDVLloc = -1
    vDCLmax = -99999999
    vDDLmax = -99999999
```

```
    vrandom = 0
    For c = 0 To 1
        o(c) = -1
        'get location with max daily revenue lost
        For l = 0 To 2
            If vDVL(l, c) > vDVLmax And vDCL(l, c) <> 0 Then
                vDVLmax = vDVL(l, c)
                vDVLloc = l
                vDCLmax = vDCL(l, c)
                vDDLmax = vDDL(l, c)
                Randomize
                vrandom = Int(2 * Rnd + 1)
            End If
        Next l
        If vDVLloc <> -1 Then
        'solve ties
            For l = 0 To 2
                If l <> vDVLloc Then
                    If vDVL(l, c) = vDVLmax And vDCL(l, c) > vDCLmax Then
                        vDCLmax = vDCL(l, c)
                        vDVLloc = l
                        vDDLmax = vDDL(l, c)
                    ElseIf vDVL(l, c) = vDVLmax And vDCL(l, c) = vDCLmax And vDDL(l, c) >= vDDLmax
Then
                            If vDDL(l, c) > vDDLmax Then
                                vDDLmax = vDDL(l, c)
                                vDVLloc = l
                            Else
                                If vrandom > 1 Then vDVLloc = l
                            End If
                    End If
                End If
            Next l
            o(c) = vDVLloc
        End If
    Next c
    '2- Get listloc(c,l)
    '-------------------
    vDVLmax = -99999999
    vDVLloc = -1
    vDCLmax = -99999999
    vDDLmax = -99999999
    vrandom = 0
    For c = 0 To 1
        For l = 0 To 2
            listloc(c, l) = -1
            For k = 0 To 3
                listmvinc(c, l, k) = 0
            Next k
        Next l
    Next c
    ' get ordered list of locations with available fleet
    For c = 0 To 1
      If o(c) <> -1 Then
        For l = 0 To 2
            If l <> o(c) And vALF(l, c) > 0 Then
                If vDVL(l, c) > vDVLmax Then
                    vDVLloc = l
                    vDVLmax = vDVL(l, c)
                    vDCLmax = vDCL(l, c)
                    vDDLmax = vDDL(l, c)
                    Randomize
                    vrandom = Int(2 * Rnd + 1)
                End If
            End If
        Next l
        If vDVLloc <> -1 Then
            'solve ties
            For l = 0 To 2
```

```
                If l <> vDVLloc And l <> o(c) And vALF(l, c) > 0 Then
                    If vDVL(l, c) = vDVLmax And vDCL(l, c) > vDCLmax Then
                        vDCLmax = vDCL(l, c)
                        vDVLloc = l
                        vDDLmax = vDDL(l, c)
                    ElseIf vDVL(l, c) = vDVLmax And vDCL(l, c) = vDCLmax And vDDL(l, c) >= vDDLmax
Then
                            If vDDL(l, c) > vDDLmax Then
                                vDDLmax = vDDL(l, c)
                                vDVLloc = l
                            Else
                                If vrandom > 1 Then vDVLloc = l
                            End If
                    End If
                End If
            Next l
            'get list and move increments for each location with available fleet
            listloc(c, 0) = vDVLloc
            For l = 0 To 2
                If l <> vDVLloc And l <> o(c) Then   'And valf(l, c) > 0
                    listloc(c, 1) = l
                End If
            Next l
            listmvinc(c, 0, 0) = 0
            If vALF(vDVLloc, c) <= vDCL(o(c), c) Then
                listmvinc(c, 0, 2) = vALF(vDVLloc, c)
                listmvinc(c, 0, 1) = CInt(vALF(vDVLloc, c) / 2)
                listmvinc(c, 0, 3) = 1 ' exists
            Else
                listmvinc(c, 0, 2) = vDCL(o(c), c)
                listmvinc(c, 0, 1) = CInt(vDCL(o(c), c) / 2)
                listmvinc(c, 0, 3) = 1 ' exists
            End If
            listmvinc(c, 1, 0) = 0
            If vALF(listloc(c, 1), c) <= vDCL(o(c), c) Then
                listmvinc(c, 1, 2) = vALF(listloc(c, 1), c)
                listmvinc(c, 1, 1) = CInt(vALF(listloc(c, 1), c) / 2)
                listmvinc(c, 1, 3) = 1 ' exists
            Else
                listmvinc(c, 1, 2) = vDCL(o(c), c)
                listmvinc(c, 1, 1) = CInt(vDCL(o(c), c) / 2)
                listmvinc(c, 1, 3) = 1 ' exists
            End If

        End If ' listloc(c,0) <> -1 exists
      End If ' o(c) <> -1  exists
    Next c
    '3- Select 9 different alternatives for movement
    '-----------------------------------------------
    For i = 0 To 9
        For j = 0 To 13
            If j < 4 Then
                mv_heu_amt(i, j) = 0
            End If
            mv_heu(i, j) = 0
        Next j
    Next i
    j = 0
    For c = 0 To 1
        If o(c) <> -1 And listmvinc(c, 0, 3) = 1 Then
            For i = 0 To 2
                If Not (c = 1 And i = 0 And j = 2) Then 'skips first increment in last car class
because it is zero
                    If c = 0 Then
                        j = i
                    Else
                        j = j + 1
                    End If
                    mv_heu(j, 0) = listloc(c, 0)    ' listloc(c,0) from location
```

```
                    mv_heu(j, 1) = o(c)              '   o(c)    to location
                    mv_heu(j, 2) = c                 '   c car class
                    mv_heu(j, 3) = listmvinc(c, 0, i) ' listmvinc(c,0,0) increment zero
                    mv_heu(j, 4) = cost(listloc(c, 0), o(c)) 'cost(listloc(c,0),o(c))
                    mv_heu(j, 5) = 1                         ' only increment in one car class
                End If
            Next i
        End If
    Next c

    If j = 4 Then
        ' 3 combinations
        mv_heu(5, 0) = mv_heu(1, 0)
        mv_heu(5, 1) = mv_heu(1, 1)
        mv_heu(5, 2) = mv_heu(1, 2)
        mv_heu(5, 3) = mv_heu(1, 3)
        mv_heu(5, 4) = mv_heu(1, 4)
        mv_heu(5, 5) = 2
        mv_heu(5, 6) = mv_heu(3, 0)
        mv_heu(5, 7) = mv_heu(3, 1)
        mv_heu(5, 8) = mv_heu(3, 2)
        mv_heu(5, 9) = mv_heu(3, 3)
        mv_heu(5, 10) = mv_heu(3, 4)
        '
        mv_heu(6, 0) = mv_heu(1, 0)
        mv_heu(6, 1) = mv_heu(1, 1)
        mv_heu(6, 2) = mv_heu(1, 2)
        mv_heu(6, 3) = mv_heu(1, 3)
        mv_heu(6, 4) = mv_heu(1, 4)
        mv_heu(6, 5) = 2
        mv_heu(6, 6) = mv_heu(4, 0)
        mv_heu(6, 7) = mv_heu(4, 1)
        mv_heu(6, 8) = mv_heu(4, 2)
        mv_heu(6, 9) = mv_heu(4, 3)
        mv_heu(6, 10) = mv_heu(4, 4)
        '
        mv_heu(7, 0) = mv_heu(2, 0)
        mv_heu(7, 1) = mv_heu(2, 1)
        mv_heu(7, 2) = mv_heu(2, 2)
        mv_heu(7, 3) = mv_heu(2, 3)
        mv_heu(7, 4) = mv_heu(2, 4)
        mv_heu(7, 5) = 2
        mv_heu(7, 6) = mv_heu(4, 0)
        mv_heu(7, 7) = mv_heu(4, 1)
        mv_heu(7, 8) = mv_heu(4, 2)
        mv_heu(7, 9) = mv_heu(4, 3)
        mv_heu(7, 10) = mv_heu(4, 4)
        MaxIteration = 8    '7 + 1 because it starts at zero
    ElseIf j = 2 Then
            MaxIteration = 3
    Else
        MaxIteration = 1
    End If
End If 'Iteration number 1

' SIMAN functions are not accessible any more because SIMAN has stop running
' Some ARENA object are still accessible though
'
'=================================================================
'load statistics to mv_heu array from last iteration - index starts at zero
'=================================================================
'mv_heu_amt(IterationNumber - 1, 0) = vTotRev
mv_heu_amt(IterationNumber - 1, 1) = AvgNetRev
'mv_heu_amt(IterationNumber - 1, 2) = vTotCostMov

'=================================================================
'Prepares for a new Iteration, if next is the last one then will
'run the best Iteration again
'=================================================================
```

```
If IterationNumber = MaxIteration Then
   'prepares best movement file
   Call Init_mv_opt_file
   If vBestIteration = 1 Then
        Call Load_mv_opt_bkp
    Else
        Call Load_mv_opt_file(vBestIteration - 1)
        Call Write_mv_file
   End If
ElseIf IterationNumber < MaxIteration Then
   'use next movement values from heuristic for next iteration
   'prepare movement file
   Call Init_mv_opt_file
   Call Load_mv_opt_file(IterationNumber)
   Call Write_mv_file
End If

'================================================================
'Shows results if last Iteration
'================================================================
If IterationNumber = MaxIteration Then
' Transfering the values from VBA variables to the label objects in the Results form
   Load Results
   If MaxIteration > 1 Then
        Results.TotNetRev0.Text = Format$(mv_heu_amt(0, 1), "####,##0.00")
        Results.Iteration0.Text = "1"
   End If
   If MaxIteration = 2 Then
        Results.TotNetRev1.Visible = False
        Results.Iteration1.Visible = False
        Results.TotNetRev2.Visible = False
        Results.Iteration2.Visible = False
        Results.TotNetRev3.Visible = False
        Results.Iteration3.Visible = False
        Results.TotNetRev4.Visible = False
        Results.Iteration4.Visible = False
        Results.TotNetRev5.Visible = False
        Results.Iteration5.Visible = False
        Results.TotNetRev6.Visible = False
        Results.Iteration6.Visible = False
        Results.TotNetRev7.Visible = False
        Results.Iteration7.Visible = False
   End If
   If MaxIteration > 3 Then
        Results.TotNetRev1.Text = Format$(mv_heu_amt(1, 1), "####,##0.00")
        Results.Iteration1.Text = "2"
        Results.TotNetRev2.Text = Format$(mv_heu_amt(2, 1), "####,##0.00")
        Results.Iteration2.Text = "3"
   End If
   If MaxIteration = 4 Then
        Results.TotNetRev3.Visible = False
        Results.Iteration3.Visible = False
        Results.TotNetRev4.Visible = False
        Results.Iteration4.Visible = False
        Results.TotNetRev5.Visible = False
        Results.Iteration5.Visible = False
        Results.TotNetRev6.Visible = False
        Results.Iteration6.Visible = False
        Results.TotNetRev7.Visible = False
        Results.Iteration7.Visible = False
   End If
   If MaxIteration > 4 Then
        Results.TotNetRev3.Text = Format$(mv_heu_amt(3, 1), "####,##0.00")
        Results.Iteration3.Text = "4"
        Results.TotNetRev4.Text = Format$(mv_heu_amt(4, 1), "####,##0.00")
        Results.Iteration4.Text = "5"
        Results.TotNetRev5.Text = Format$(mv_heu_amt(5, 1), "####,##0.00")
        Results.Iteration5.Text = "6"
```

```
        Results.TotNetRev6.Text = Format$(mv_heu_amt(6, 1), "####,##0.00")
        Results.Iteration6.Text = "7"
        Results.TotNetRev7.Text = Format$(mv_heu_amt(7, 1), "####,##0.00")
        Results.Iteration7.Text = "8"
    End If
    Results.TotNetRev.Text = Format$(vBestNetRev, "####,##0.00")
    Results.BestIteration.Text = CStr(vBestIteration)
    Results.Show
' When it returns here is because the user clicked on OK
    Unload Results
End If

End Sub
```

Heuristic Modules:

```
Sub Create_proj()
Dim sProjParm, sProj, sPbounds, key As String
Dim a_min(2, 6), a_mode(2, 6), a_max(2, 6), a_avg(2, 6) As Double
Dim min, mode, max, avg, a, b, c As Double
Dim min_lbound, min_ubound, mode_lbound, mode_ubound, max_lbound, max_ubound, vrandom As Double
Dim location, time, l, t, min_cnt, mode_cnt, max_cnt As Integer
Dim min_val(2, 6, 24), mode_val(2, 6, 24), max_val(2, 6, 24) As Integer
Dim p1, p2, p3, p4, p5 As Double

sProjParm = "projparms.prn"
Open sProjParm For Input As #3
Do While Not EOF(3)
        Input #3, location, time, min, mode, max, avg
        a_min(location - 1, time - 1) = min
        a_mode(location - 1, time - 1) = mode
        a_max(location - 1, time - 1) = max
        a_avg(location - 1, time - 1) = avg
Loop
Close #3

sPbounds = "proj.prn"
Open sPbounds For Output As #2
For l = 0 To 2
    For t = 0 To 6
        min = a_min(l, t)
        mode = a_mode(l, t)
        max = a_max(l, t)
        avg = a_avg(l, t)
        a = min - Sqr(0.01 * (mode - min) * (max - min))   '1 value
        c = max + Sqr(0.01 * (max - mode) * (max - min))   '2 value
        b = mode
        min_lbound = a
        min_ubound = a + Sqr(2) * (min - a)                '3 value
        p1 = 1 - ((c - max) ^ 2 / ((c - b) * (c - a))) + ((min - a) ^ 2 / ((b - a) * (c - a)))
        If p1 > 1 Then p1 = 1
        p2 = 1 - ((c - max) ^ 2 / ((c - b) * (c - a))) - ((min - a) ^ 2 / ((b - a) * (c - a)))
        max_ubound = c - Sqr((1 - p1) * (c - b) * (c - a))
        max_lbound = c - Sqr((1 - p2) * (c - b) * (c - a))
        p5 = (b - a) / (c - a)                              '4 value
        mode_lbound = a + Sqr((b - a) ^ 2 - (min - a) ^ 2)
        mode_ubound = c - Sqr((c - b) * ((c - a) - (((b - a) ^ 2 + (min - a) ^ 2) / (b - a))))
        Print #2, l + 1, t + 1, "a: ", a, min_lbound, min, min_ubound, mode_lbound, mode,
mode_ubound, max_lbound, max, max_ubound, "c: ", c, avg
        min_cnt = 0
        mode_cnt = 0
        max_cnt = 0
        Do While (min_cnt < 25 Or mode_cnt < 25 Or max_cnt < 25)
            Randomize
            vrandom = Rnd
```

```
            If vrandom = 0 Then
                x = a
                If min_cnt < 25 Then
                    min_val(l, t, min_cnt) = CInt(x)
                    min_cnt = min_cnt + 1
                End If
            End If
            If vrandom <= p5 And vrandom > 0 Then
                x = a + Sqr(vrandom * (b - a) * (c - a))
                If x <= min_ubound Then
                    If min_cnt < 25 Then
                        min_val(l, t, min_cnt) = CInt(x)
                        min_cnt = min_cnt + 1
                    End If
                ElseIf x >= mode_lbound Then
                    If mode_cnt < 25 Then
                        mode_val(l, t, mode_cnt) = CInt(x)
                        mode_cnt = mode_cnt + 1
                    End If
                End If
            End If
            If vrandom > p5 And vrandom < 1 Then
                x = c - Sqr((1 - vrandom) * (c - b) * (c - a))
                If x <= mode_ubound Then
                  If mode_cnt < 25 Then
                    mode_val(l, t, mode_cnt) = CInt(x)
                    mode_cnt = mode_cnt + 1
                  End If
                ElseIf x >= max_lbound And x <= max_ubound Then
                    If max_cnt < 25 Then
                        max_val(l, t, max_cnt) = CInt(x)
                        max_cnt = max_cnt + 1
                    End If
                End If
            End If
        Loop
    Next t
Next l
Close #2

'writes
sProj = "projections.prn"
Open sProj For Output As #3
For l = 0 To 2
    For t = 0 To 6
        For min_cnt = 0 To 24
            key = l + 1 & t + 1 & 1
            Print #3, key, l + 1, t + 1, 1, min_val(l, t, min_cnt)
        Next min_cnt
        For mode_cnt = 0 To 24
            key = l + 1 & t + 1 & 2
            Print #3, key, l + 1, t + 1, 2, mode_val(l, t, mode_cnt)
        Next mode_cnt
        For max_cnt = 0 To 24
            key = l + 1 & t + 1 & 3
            Print #3, key, l + 1, t + 1, 3, max_val(l, t, max_cnt)
        Next max_cnt
    Next t
Next l
Close #3

End Sub
Sub Create_Proj_Files()
Dim ObjectIndex, l, c, s, t, w, prevprkey, prkey, location, time, week, tu As Integer
Dim price(2, 1, 1, 6), factkey, sample, iaux As Integer
Dim nsh(2, 6, 2), can(2, 6, 2), car(2, 6, 2, 1), los(2, 6, 2, 1), aux(5) As Double
Dim tunc(2, 6, 24), runc(2, 6, 24) As Double

Dim sMove, sMovebkp, sProj, sPrice, sFactor, sSample, STemp As String
```

```
Dim vLoc1, vLoc2, vCarClass, vQtyCar As Integer
Dim vTime, vCost As Double

If IterationNumber < 1 Then

prevprkey = 0
i = 0
sProj = "projections.prn"
Open sProj For Input As #3
Do While Not EOF(3)
        Input #3, prkey, location, time, week, tu
        If prkey <> prevprkey Then
           If i > 0 Then Close #4
           i = i + 1
           prevprkey = prkey
           STemp = "prkey" + Format$(prevprkey, "0") + ".dat"
           Open STemp For Output As #4
        End If
         Write #4, prkey; location; time; week; tu
Loop
Close #3
Close #4

End If

End Sub

Sub Create_Opt_Inputr()
Dim ObjectIndex, l, c, s, t, w, prevprkey, prkey, location, time, week, tu As Integer
Dim price(2, 1, 1, 6), factkey, sample, iaux As Integer
Dim nsh(2, 6, 2), can(2, 6, 2), car(2, 6, 2, 1), los(2, 6, 2, 1), aux(5) As Double
Dim tunc(2, 6, 24), runc(2, 6, 24) As Double
Dim city(2), carclass(1), lofstay(1) As String

Dim sMove, sMovebkp, sProj, sPrice, sFactor, sSample, STemp, sHeader, caux, cend As String
Dim vLoc1, vLoc2, vCarClass, vQtyCar As Integer
Dim vTime, vCost As Double
Dim e
Dim InputData

city(0) = "F"
city(1) = "K"
city(2) = "M"
carclass(0) = "E"
carclass(1) = "M"
lofstay(0) = "D"
lofstay(1) = "T"

If IterationNumber < 1 Then
   'read price
   sPrice = "price.prn"
   Open sPrice For Input As #5
   For l = 0 To 2
        For c = 0 To 1
            For s = 0 To 1
                For t = 0 To 6
                    Input #5, price(l, c, s, t)
                Next t
            Next s
        Next c
    Next l
    Close #5
    ' read factors
    sFactor = "factors.prn"
    Open sFactor For Input As #5
    Do While Not EOF(5)
        Input #5, factkey, l, t, w, aux(0), aux(1), aux(2), aux(3), aux(4), aux(5)
        l = l - 1
        t = t - 1
```

```
        w = w - 1
        nsh(l, t, w) = aux(0)
        can(l, t, w) = aux(1)
        car(l, t, w, 0) = aux(2)
        car(l, t, w, 1) = aux(3)
        los(l, t, w, 0) = aux(4)
        los(l, t, w, 1) = aux(5)
    Loop
    Close #5
    ' Creation of 75 files for experiments
    For w = 0 To 2
        For l = 0 To 2 ' Load projections  for experiment w into variables
            For t = 0 To 6
                location = l + 1
                time = t + 1
                week = w + 1
                STemp = "prkey" + Format$(location, "0") + Format$(time, "0") + Format$(week, "0") +
".dat"
                Open STemp For Input As #5
                For sample = 0 To 24
                    Input #5, prkey, location, time, week, tu
                    location = location - 1
                    time = time - 1
                    tunc(location, time, sample) = tu
                Next sample
                Close #5
            Next t
        Next l          'End loading of proj for exp w into vars
        'Create 25 files for experiment w
        For sample = 0 To 24
            sSample = "sample" + Format$(w, "0") + Format$(sample, "0") + ".dat"
            Open sSample For Output As #7
            'Writes to Optimal Input file, one location at a time
            l = 0
                For t = 0 To 6
                    'obtain realized unconstrained demand
                    runc(l, t, sample) = tunc(l, t, sample) * (1 - nsh(l, t, w) - can(l, t, w))
                    For c = 0 To 1
                        For s = 0 To 1
                            aux(0) = runc(l, t, sample) * car(l, t, w, c) * los(l, t, w, s)
                            iaux = CInt(aux(0))
                            caux = "!"
                            cend = "~"
                            'Write #7, l + 1, c + 1, s + 1, iaux, price(l, c, s, t)
                            If l = 1 And t = 6 And c = 1 And s = 1 Then
                                Print #7, l, c, s, t + 1, iaux, price(l, c, s, t)
                            Else
                                Print #7, l, c, s, t + 1, iaux, price(l, c, s, t)
                            End If
                        Next s
                    Next c
                Next t
            l = 2
                For t = 0 To 6
                    'obtain realized unconstrained demand
                    runc(l, t, sample) = tunc(l, t, sample) * (1 - nsh(l, t, w) - can(l, t, w))
                    For c = 0 To 1
                        For s = 0 To 1
                            aux(0) = runc(l, t, sample) * car(l, t, w, c) * los(l, t, w, s)
                            iaux = CInt(aux(0))
                            caux = "!"
                            cend = "~"
                            'Write #7, l + 1, c + 1, s + 1, iaux, price(l, c, s, t)
                            If l = 1 And t = 6 And c = 1 And s = 1 Then
                                Print #7, l, c, s, t + 1, iaux, price(l, c, s, t)
                            Else
                                Print #7, l, c, s, t + 1, iaux, price(l, c, s, t)
                            End If
                        Next s
```

```
                    Next c
                Next t
            l = 1
                For t = 0 To 6
                    'obtain realized unconstrained demand
                    runc(l, t, sample) = tunc(l, t, sample) * (1 - nsh(l, t, w) - can(l, t, w))
                    For c = 0 To 1
                        For s = 0 To 1
                            aux(0) = runc(l, t, sample) * car(l, t, w, c) * los(l, t, w, s)
                            iaux = CInt(aux(0))
                            caux = "!"
                            cend = "~"
                            'Write #7, l + 1, c + 1, s + 1, iaux, price(l, c, s, t)
                            If l = 1 And t = 6 And c = 1 And s = 1 Then
                                Print #7, l, c, s, t + 1, iaux, price(l, c, s, t)
                            Else
                                Print #7, l, c, s, t + 1, iaux, price(l, c, s, t)
                            End If
                        Next s
                    Next c
                Next t
            Close #7
        Next sample
    Next w      'Next experiment w
 End If ' IterationNumber < 1

End Sub

Sub Mod_Opt_Inputr()
Dim l, c, s, t, w, sample As Integer
Dim sSample, sData, sHeader As String
Dim caux, cend As String
Dim i_list(2, 1, 1, 6, 5) As Integer
Dim city(2), carclass(1), lofstay(1) As String

city(0) = "F"
city(1) = "K"
city(2) = "M"
carclass(0) = "E"
carclass(1) = "M"
lofstay(0) = "D"
lofstay(1) = "T"

For w = 0 To 2
   For sample = 0 To 24
        sSample = "sample" + Format$(w, "0") + Format$(sample, "0") + ".dat"
        Open sSample For Input As #7
        For l = 0 To 2
            For t = 0 To 6
                For c = 0 To 1
                     For s = 0 To 1
                        Input #7, i_list(l, c, s, t, 0), i_list(l, c, s, t, 1), i_list(l, c, s, t,
2), i_list(l, c, s, t, 3), i_list(l, c, s, t, 4), i_list(l, c, s, t, 5)
                     Next s
                Next c
            Next t
        Next l
        Close #7
        'write a new one
        sData = "data" + Format$(w, "0") + Format$(sample, "0") + ".dat"
        Open sData For Output As #9
        sHeader = "optinput.ldt"
        Open sHeader For Input As #8   ' Open file for input.
        'header
        Do While Not EOF(8) ' Check for end of file.
            Line Input #8, InputData   ' Read line of data.
            ' Debug.Print InputData   ' Print to Debug window.
            Print #9, InputData
        Loop
```

```
        Close #8    ' Close file.
        For l = 0 To 2
            For c = 0 To 1
                For s = 0 To 1
                    For t = 0 To 6
                        caux = "!"
                        cend = "~"
                        If l = 2 And t = 6 And c = 1 And s = 1 Then
                           Print #9, caux; city(i_list(l, c, s, t, 0)), carclass(i_list(l, c, s, t,
1)), lofstay(i_list(l, c, s, t, 2)), i_list(l, c, s, t, 3), ";", i_list(l, c, s, t, 4), i_list(l, c,
s, t, 5), cend
                        Else
                           Print #9, caux; city(i_list(l, c, s, t, 0)), carclass(i_list(l, c, s, t,
1)), lofstay(i_list(l, c, s, t, 2)), i_list(l, c, s, t, 3), ";", i_list(l, c, s, t, 4), i_list(l, c,
s, t, 5)
                        End If
                    Next t
                Next s
            Next c
        Next l
        Close #9
    Next sample
Next w


End Sub
Sub Init_mv_opt_file()
Dim lf, lt, c, i As Integer

For lf = 0 To 2
    For lt = 0 To 2
        For c = 0 To 1
            For i = 0 To 5
                mv_opt_file(lf, lt, c, i) = mv_opt(lf, lt, c, i)
            Next i
        Next c
    Next lt
Next lf
End Sub

Sub Load_mv_opt_file(iteration)
Dim heu_part1, heu_part2, lf, lt, c As Integer

For lf = 0 To 2
    For lt = 0 To 2
        For c = 0 To 1
            If mv_opt_file(lf, lt, c, 0) = 1 Then
                'exists in optimal file
                If (mv_heu(iteration, 0) = lf And mv_heu(iteration, 1) = lt And mv_heu(iteration, 2)
= mv_opt_file(lf, lt, c, 3)) Then
                    'exist in original movement alternative from optimization step
                    mv_opt_file(lf, lt, c, 4) = mv_opt_file(lf, lt, c, 4) + mv_heu(iteration, 3)
                    mv_opt_file(lf, lt, c, 5) = mv_heu(iteration, 4)
                    heu_part1 = 1
                End If
                '
                If mv_heu(iteration, 5) = 2 Then
                    'iteration has alternatives for two car classes
                    If (mv_heu(iteration, 6) = lf And mv_heu(iteration, 7) = lt And
mv_heu(iteration, 8) = mv_opt_file(lf, lt, c, 3)) Then
                        'exist in original movement alternative from optimization step
                        mv_opt_file(lf, lt, c, 4) = mv_opt_file(lf, lt, c, 4) + mv_heu(iteration, 9)
                        mv_opt_file(lf, lt, c, 5) = mv_heu(iteration, 10) 'cost
                        heu_part2 = 1
                    End If
                End If
            Else ' Does not exist in optimal file
                If (mv_heu(iteration, 0) = lf And mv_heu(iteration, 1) = lt And mv_heu(iteration, 2)
= c) Then
```

```
                    mv_opt_file(lf, lt, c, 0) = 1
                    mv_opt_file(lf, lt, c, 1) = lf
                    mv_opt_file(lf, lt, c, 2) = lt
                    mv_opt_file(lf, lt, c, 3) = c
                    mv_opt_file(lf, lt, c, 4) = mv_heu(iteration, 3)
                    mv_opt_file(lf, lt, c, 5) = mv_heu(iteration, 4)
                    heu_part1 = 1
                End If
                '
                If mv_heu(iteration, 5) = 2 Then
                    'iteration has alternatives for two car classes
                    If (mv_heu(iteration, 6) = lf And mv_heu(iteration, 7) = lt And
mv_heu(iteration, 8) = c) Then
                        'exist in original movement alternative from optimization step
                        mv_opt_file(lf, lt, c, 0) = 1
                        mv_opt_file(lf, lt, c, 1) = lf
                        mv_opt_file(lf, lt, c, 2) = lt
                        mv_opt_file(lf, lt, c, 3) = c
                        mv_opt_file(lf, lt, c, 4) = mv_heu(iteration, 9)
                        mv_opt_file(lf, lt, c, 5) = mv_heu(iteration, 10) 'cost
                        heu_part2 = 1
                    End If
                End If
            End If ' does not exist in optimal file
        Next c
    Next lt
Next lf

End Sub

Sub Write_mv_file()
Dim lf, lt, c, i As Integer
Dim sMove As String

sMove = "mfleet.txt"
Open sMove For Output As #1

For lf = 0 To 2
    For lt = 0 To 2
        For c = 0 To 1
            If mv_opt_file(lf, lt, c, 0) = 1 Then
                'exists
                Print #1, mv_opt_file(lf, lt, c, 1) + 1, mv_opt_file(lf, lt, c, 2) + 1,
mv_opt_file(lf, lt, c, 3) + 1, 0.5, mv_opt_file(lf, lt, c, 4), mv_opt_file(lf, lt, c, 5)
            End If
        Next c
    Next lt
Next lf

Close #1
End Sub

Sub Load_mv_opt_bkp()
Dim sMove, sMovebkp As String
Dim vLoc1, vLoc2, vCarClass, vTime, vQtyCar, vCost As Integer

   sMovebkp = "mfleetbkp01.txt"
   Open sMovebkp For Input As #1
   sMove = "mfleet.txt"
   Open sMove For Output As #2

   Do While Not EOF(1)
        Input #1, vLoc1, vLoc2, vCarClass, vTime, vQtyCar, vCost
        Print #2, vLoc1, vLoc2, vCarClass, vTime, vQtyCar, vCost
   Loop

Close #1
Close #2
```

```
End Sub
Sub Init_Simulation()
Dim i, j As Integer

For i = 0 To 42  ' Number of replications
   TotNetRev_list(i) = 0
  Next i
End Sub

Sub Init_Heuristic()
Dim i, l, c, lf, lt As Integer

   'clear values
    For l = 0 To 2
       For c = 0 To 1
            vALF(l, c) = 0
            vDCL(l, c) = 0
            vDVL(l, c) = 0
            vDDL(l, c) = 0
            For i = 0 To 42
                vDCL_list(l, c, i) = 0
                vDVL_list(l, c, i) = 0
                vDDL_list(l, c, i) = 0
            Next i
        Next c
    Next l

     For lf = 0 To 2
        For lt = 0 To 2
            For c = 0 To 1
                For i = 0 To 5
                    mv_opt(lf, lt, c, i) = -1
                Next i
            Next c
        Next lt
   Next lf

End Sub

Sub Clear_Run()

  IterationNumber = 0
  MaxIteration = 9
  vBestIteration = 0
  vBestNetRev = -9999999
  vBestRev = -9999999
  vBestCostMov = 99999999

End Sub

Variables
Public TheModel As Arena.Model
Public TallyModule As Arena.Module
Public CreateModule As Arena.Module
Public CounterModule As Arena.Module
Public ResourceModule As Arena.Module
Public VariableModule As Arena.Module
Public ReplicateModule As Arena.Module
Public SetsModule As Arena.Module
Public ReportLineModule As Arena.Module
Public Out As SIMAN
Public vLocation(7) As Integer
Public e As Double
Public appname As String
Public vALF(2, 1) As Integer
Public vDCL(2, 1) As Integer
Public vDVL(2, 1) As Double
Public vDDL(2, 1) As Integer
Public vDDL_list(2, 1, 42), vDCL_list(2, 1, 42) As Integer
```

```
Public vDVL_list(2, 1, 42) As Double
Public o(1) As Integer
Public cost(2, 2) As Integer
Public mv_opt(2, 2, 1, 5) As Integer
Public mv_opt_file(2, 2, 1, 5) As Integer
Public mv_heu(9, 13) As Integer
Public mv_heu_amt(9, 3) As Double
Public vBestNetRev, vBestRev, vBestCostMov, AvgNetRev As Double
Public IterationNumber, MaxIteration, NumOfReps, vBestIteration As Integer
Public vTotRev, vTotNetRev, vTotCostMov As Double
Public TotNetRev_list(42) As Double
Public vNRep, vMrep As Integer
```