Order Number 1335812

# Developing and validating a simulation model for emergency vehicle locations

Chen, Jen-Ming, M.S.

The University of Arizona, 1988

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

# DEVELOPING AND VALIDATING A SIMULATION MODEL FOR EMERGENCY VEHICLE LOCATIONS

by

Jen-Ming Chen

---

A Thesis Submitted to the Faculty of the

DEPARTMENT OF SYSTEMS AND INDUSTRIAL ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

MASTER OF SCIENCE
WITH A MAJOR IN INDUSTRIAL ENGINEERING

In the Graduate College

THE UNIVERSITY OF ARIZONA

1988

## STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at the University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: Jen-ming Chen

## APPROVAL BY THESIS DIRECTOR

This thesis has been approved on the date shown below:

Jeffrey B. Goldberg
Assistant Professor of Systems
and Industrial Engineering

10/20/88
Date

## ACKNOWLEDGMENTS

For his continued assitance and guidance over the past year and especially throughout the period of my thesis work, I would like to express my deepest appreciation to my advisor, Dr. Jeffrey B. Goldberg. Without his constructive criticism and advice, this thesis would not have progressed as smoothly as it did up through its completion.

I would also like to thank the Systems and Industrial Engineering Department committee members, Dr. Paul Sanchez and Dr. Julia Higle, for their advice and suggestions that were essential and invaluable to finish the final stages of the work.

Finally, I wish to express my profound gratitude to my parents for their love, support, and for all that they have given me.

## TABLE OF CONTENTS

## TABLE OF CONTENTS—Continued

## LIST OF ILLUSTRATIONS

## LIST OF ILLUSTRATIONS—Continued

## LIST OF TABLES

## ABSTRACT

This thesis deals with the problem of locating emergency ambulances in an urban area. We developed a simulation model to analyze possible improvements in ambulance service. A new point-to-point travel time model is introduced in our simulation. Validating the model proved to be a difficult task and is discussed in detail. Our model has been applied to the Tucson Emergency Medical Service system.

# CHAPTER 1
# INTRODUCTION

An urban Emergency Medical Service (EMS) system provides ambulances to respond to requests for service which can occur at any time and any place throughout a city. As large medical centers become more capable, the performance of an EMS system becomes increasingly important. In the design of such systems, there are several factors to be considered, such as setup and operating costs, ambulance base locations, workload balance and system performance. One of the most important factors is service quality, which is measured by the system performance and is mainly determined by ambulance base locations.

## 1.1 Problem Environment

Emergency medical service has been provided by the city of Tucson to the public for a long time. Just as in many cities, the EMS system in Tucson is part of the Fire Department.

With the growth of the city, the service is now available to more than 365,000 persons 24 hours a day, every day of the year. In 1986, the city's ambulance service responded to approximately 20,000 calls for emergency assistance. In 1987, the demand for EMS increased 5 percent. As a result of the growing work load,

together with an increasing concern about the service quality of the system, the Fire Department administration began a study of its EMS system to determine where to locate current ambulances to achieve a higher service level.

Currently, the EMS system consists of 7 ambulance bases, each with 1 ambulance, and 7 medical centers. Besides that, there are 8 fire stations for potential ambulance bases. A map of the city, with current and potential base locations, is shown in Figure 1.1, where each number represents a particular "zone" with a potential base, and the open bases are circled. We define the term "zone" as a geographical district which is bounded by city streets, freeways, railroads, rivers or any other type of landmarks.

The EMS system must respond to emergency calls from the general public and give medical service in a timely fashion. Requests for emergency ambulance service are made by the public via telephone (911 service) to the Police Department, Communications Division. Upon receipt of a call, the dispatcher on duty will notify the nearest idle ambulance, i.e., an ambulance that is available for service, to perform emergency assistance service. If there are no idle ambulances, the call must either wait in queue for service or a private ambulance service must be called.

Since response time is a critical measure of service quality during an emergency, the goal of the Tucson EMS system is to have a paramedic arrive on scene within 8 minutes of call origination. Due to limited resources, all calls cannot be serviced within the 8 minute limit and are serviced in a first-come-first-served manner. So the problem becomes one of picking up as many calls within 8 minutes as possible. We use the term "successful pickup" as a call that is reached within

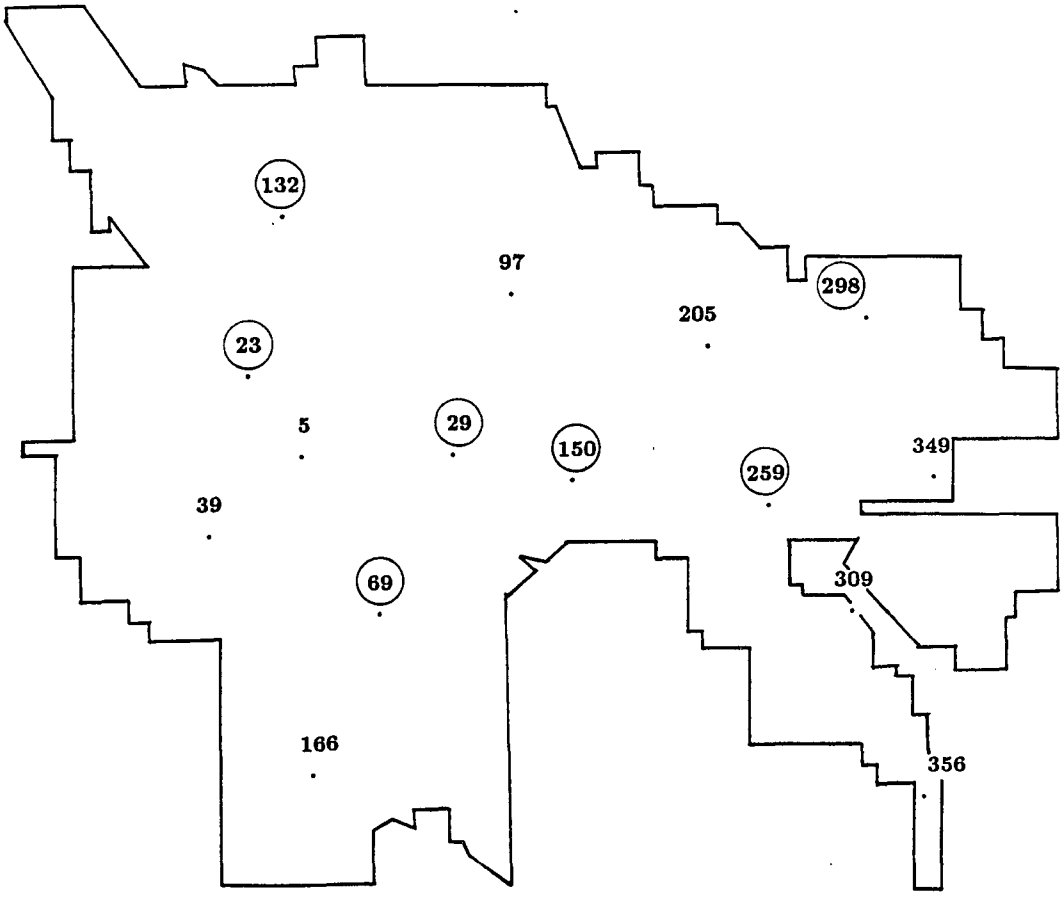



**Figure 1.1 A Map of Tucson City with Current and Potential Base Locations, (Current Open Bases are Circled)**

8 minutes from the time of call origination. In addition, we define the following terminologies which are used throughout this thesis.

1) System success rate: the percentage of calls system wide that are picked up successfully.
2) Zone success rate: the percentage of calls originating in a particular zone that are picked up successfully.

Total call service time for an ambulance includes dispatching time, travel time to the call location, service time at the scene, possible transport time to the hospital and possible time at the hospital. A detailed event sequence of ambulance service during a call is shown in Figure 1.2. In addition, Figure 1.3 provides time definitions for ambulance service. Dispatching time in the Tucson system requires between 30 and 45 seconds, and is generally small relative to travel time. Throughout this thesis, we have assumed that the time required to dispatch a call is constant. Total service time varies highly with accident location and has an average of 30 minutes for the empirical data we have seen. Travel time to the accident scene averages 20% of the total service time. The service time, not including travel time to the scene, varies highly by zone since some zones are further than others from hospitals, and different types of emergencies occur in different zones.

Ambulances are equipped with two-way radios, which enable them to communicate with the Communications Division. After the assigned ambulance transports the patient to a hospital, the ambulance becomes idle. It can then be dispatched again either directly from the hospital or while enroute back to the base. One should note that queueing rarely occurs in a well designed and well operated system. Clearly, if the call must wait in the queue, it will generally not be reached

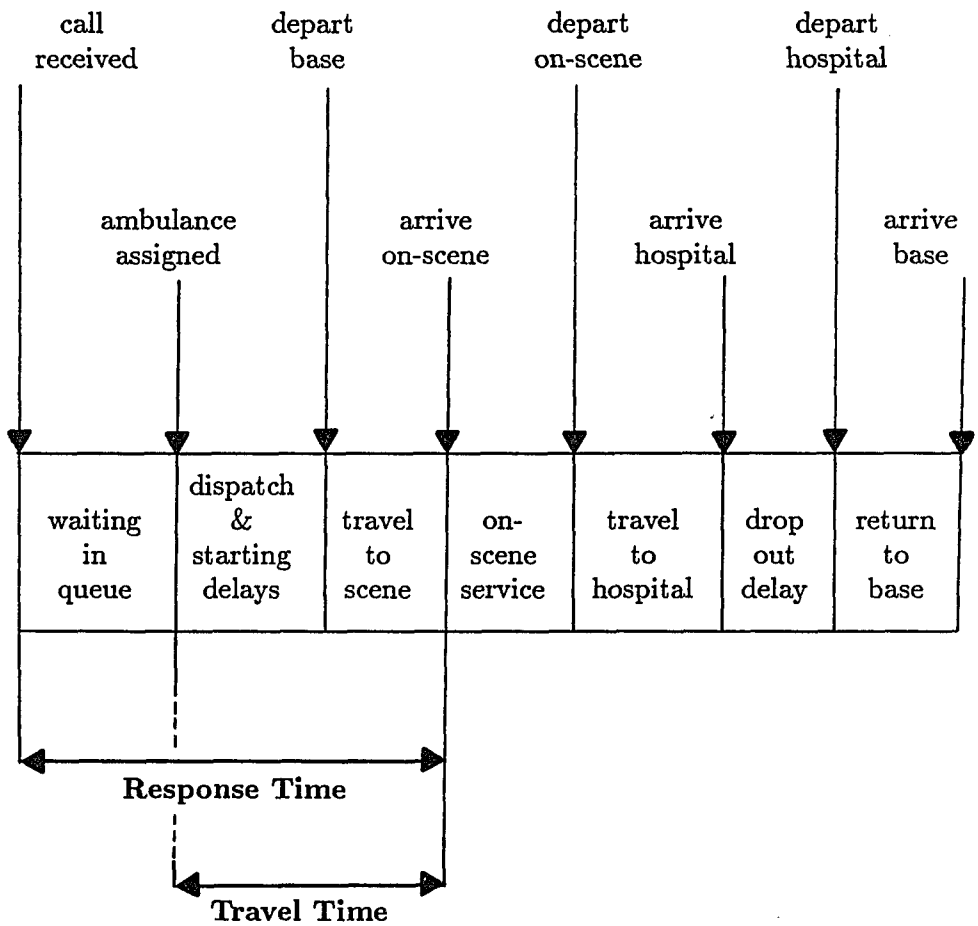
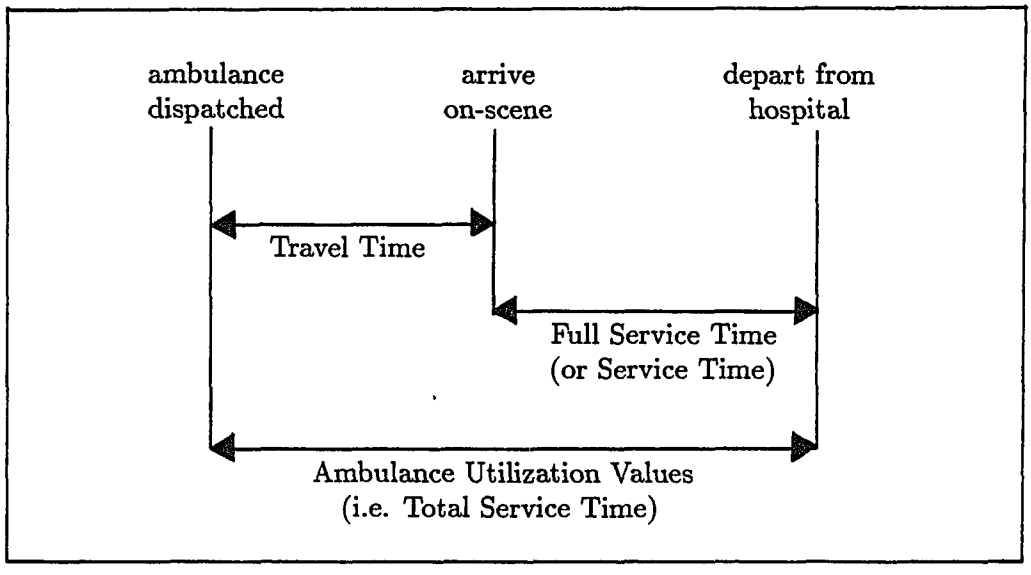
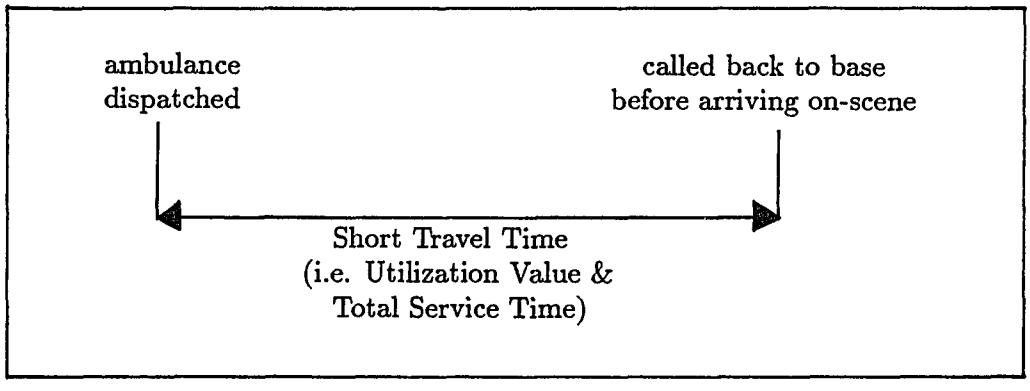



**Figure 1.2 The Event Sequence of Ambulance Service During a Call**




**(a) Actual Emergency Call**




**(b) False Alarm Call**

**Figure 1.3 Time Definitions for Ambulance Service During a Call**

within 8 minutes. Ambulance utilizations of 0.15 are common. In accordance with the low utilization, the majority of calls are serviced by an ambulance starting from its base.

Each ambulance is manned by 3 pairs of paramedics. In a given week, two of the pairs rotate 24 hour shifts while the third pair is on vacation. The pair on vacation rotates weekly so that every pair works 2 of 3 weeks. Given this schedule, it is clear that employee burnout and fatigue should be a factor in determining where to locate ambulance bases. Therefore, bases are constrained to fire stations, where the paramedics can rest and perform routine maintenance and resupply while not on call. A larger set of locations (such as convenience store parking lots) may allow more calls to be serviced within 8 minutes, but the effect on employees is too detrimental to be permitted. Thus, in the Tucson EMS system, we only consider fire stations as potential ambulance bases.

Ambulances at a given location are generally assigned to cover a particular geographic district in the vicinity of that location. To facilitate our analysis, the urban area is divided into zones. Both demand and travel times are predicted based on the zone structure. For example, the travel time from each potential base site to a zone must be estimated to determine the amount of time required to service a call in that particular zone. It is generally assumed that all demand for a zone occurs at the zone population center. If the zones are too large, poor travel time approximations will persist. So, one must not structure extremely large zones.

On the other hand, the zones cannot be too small because the amount of data analysis and computational work required to solve the models will be a function of the number of zones. In chapter 3, we will discuss zone structure in Tucson. From

our experience, it is clear that zone structuring has great implications concerning the validity of models for the problem.

## 1.2 Objectives

The facility location problem can have many objectives. Its primary objective is to maximize the expected percentage of calls reached within a certain time limit. In addition, because of political pressures, there is a desire to give adequate service over the entire area. That is, a set of locations where 90% of the calls are reached in time with some zones having only a 50% success rate, may be inferior to a set of locations where 89% of the calls are reached in time where the lowest zone success rate is 75%. From these objectives, it is clear that a model for determining base locations must yield estimates of the system success rate as well as individual zone success rates.

In this thesis, we will focus on maximizing the overall system success rate, together with balancing both ambulance workload and zone performance. In particular, we will present an approach by means of computer simulation for predicting the performances of the EMS system with different sets of ambulance bases. We will also concentrate on the validation of the simulation model.

## 1.3 Organization

This thesis is organized into 7 chapters. We discuss the problem environment in this chapter and review past research in chapter 2. In chapter 3, we present data collection and its analysis, and the development of the travel time model. Both regression and residual analyses are also included in that chapter. Simulation

of the system is discussed in chapter 4. Model validation is shown in chapter 5. In particular, we developed a detailed validation of the model by discussing criteria such as ambulance utilization, system performance and dispatch patterns. Once we convinced the administration that the model is validated, it was used along with actual data from the Tucson Paramedic System to evaluate potential changes in the current base deployment. Three case studies are presented in chapter 6. Conclusions from our study, and directions for further work in vehicle location problems are discussed in chapter 7.

# CHAPTER 2
# LITERATURE REVIEW

## 2.1 Existing Models

The existing models in facility location problems can be classified into two approaches: queueing approaches and set covering approaches.

Larson ([20], 1974) developed a Hypercube Queueing model that uses some basic Markov Chain theory to solve the problems of a multi-server queueing system with distinguishable servers. The model contains an efficient enumeration procedure to evaluate steady state probabilities, and has the ability to look at a wide variety of output measures. However, weaknesses include assumptions that service times have identical exponential distribution, and are not a function of both the call location and the ambulance that serves the call. In addition, the state space of the Markov Chain grows exponentially with the number of open bases, so computations and data collection can be overwhelming.

To remedy the computational problems, Larson ([21], 1975) later developed a Hypercube Approximating approach, which uses the following assumptions.

1) Exactly one response unit is assigned to each call.
2) The service time is a negative exponential distribution with mean $1/\mu$.

3) The service time is independent of the identity of the server, the location of the customer, and the history of the system.

4) Calls are serviced in a first-come-first-served (FCFS) manner.

The model is:

$$1-\rho_i = \Big[\, 1 + \sum_{j\epsilon G_i^1} \lambda_j + \sum_{j\epsilon G_i^2} \lambda_j Q(n,\rho,1)\rho_{n_{j1}} + \sum_{j\epsilon G_i^3} \lambda_j Q(N,\rho,2)\rho_{n_{j1}}\rho_{n_{j2}} + \cdots + \sum_{j\epsilon G_i^N} \lambda_j Q(N,\rho,N-1)\rho_{n_{j1}}\rho_{n_{j2}}\cdots\rho_{n_{j(N-1)}} + \lambda_D/(1-\rho_i)\,\Big]^{-1}, \quad i = 1,2,\cdots,N \qquad [1]$$

where

$\rho$ = utilization factor for infinite-capacity system,

$\rho_i$ = fraction of time that unit $i$ is busy servicing calls,

$\rho_{ik}$ = fraction of dispatches that send unit $i$ to geographical atom (i.e., "zone" in this thesis) $k$,

$$\lambda_D = \begin{cases} 0, & \text{for zero-line capacity case,} \\ \lambda P_N/N, & \text{for infinite-line capacity case,} \end{cases}$$

$=$ delayed dispatches from a queue,

$\lambda$ = mean rate at which calls for service are generated from within the region,

$P_k$ = probability that exactly $k$ servers are busy, $k = 0,1,\cdots,N$,

$\lambda_k$ = mean rate at which calls for service are generated from atom $k$, $\lambda f_k = \lambda_k$,

$f_k$ = fraction of region-wide workload generated from within atom $k$,

$N$ = number of servers or response units,

$G_i^k$ = set of atoms for which unit $i$ is the $k$st. preference, $k = 0, 1, \cdots, N$,

$Q(N, \rho, j)$ = "correction factor" for computing probability that the $(j+1)$st. selected server is the first available server, given a total of $N$ servers and a utilization factor $\rho$, $j = 0, 1, \cdots, N-1$.

Equation [1] represents a set of $N$ simultaneous nonlinear equations in the $\rho_i$'s that can be solved iteratively. Even though this paper still contains assumptions that the service times are identically distributed over the entire area, it made two main contributions: 1) the ambulance utilizations can be estimated by solving equation [1] whose size depends on the number of open bases, and 2) one can approximately characterize the dependence structure of the utilizations as a function of the number of busy ambulances. Larson found that the approximation is generally within 2 or 3% of the exact values generated by the Hypercube Queueing model, and that the performance gets better as system utilization decreases since the correction factors are less important under low utilizations.

Berman, Larson and Chiu ([4], 1985) proposed an M/G/1 network operating queueing model to look for the optimal location for a single server. This paper is in part motivated by Larson's Hypercube Queueing model. In addition, they developed a travel time model to remedy Larson's assumption that the service times are identically distributed. Berman assumed that mean travel time varies proportionately with mean travel distance, and that the on-scene service times are i.i.d. random variables. The travel time and on-scene service time assumptions are

more realistic. However, the single server model is restrained to fewer real world applications.

Set covering approaches have been discussed by many researchers. Toregas et. al. ([28], 1971) formulated a 0–1 integer program to decide on the minimum number of bases to use so that every zone was covered. That is, for each zone, the expected travel time from at least one open base was within the specified time limit.

Church and ReVelle ([7], 1974) withdrew the coverage requirement while adding a demand factor. Their model was to locate at most $n$ bases so that the maximum number of calls could be reached within the specified limit. Both of these papers assume that an ambulance that is based within the specified time limit can always cover the call. They do not take ambulance utilization nor stochastic travel times into account. If an ambulance is busy, it may not be able to reach a call on time, even though it is within the specified time limit.

Daskin's paper ([18], 1983) rectifies this shortcoming by formulating an objective that represents the expected number of covered calls, and is based on the number of ambulances that cover a particular zone. In this paper, the following assumptions are made.

1) All ambulances have identical utilizations, (denoted $\rho$).
2) The utilization value is independent of the state of system.
3) The travel time is deterministic.

The Maximum EXpected Covering Location Problem (MEXCLP) model is:

$$Max \quad \sum_{k=1}^{N}\sum_{j=1}^{M}(1-\rho)\rho^{j-1}h_k y_{jk} = \sum_{k}\sum_{j} w_j h_k y_{jk} \qquad [2a]$$

$$Subject\ to \quad \sum_{j=1}^{M} y_{jk} - \sum_{i=1}^{N} a_{ki} X_i \leq 0, \quad \forall\, k \qquad [2b]$$

$$\sum_{i=1}^{N} X_i \leq M \qquad [2c]$$

$$X_i = 0, 1, \cdots, M, \quad \forall\, i \qquad [2d]$$

$$y_{jk} = [0, 1], \quad \forall\, j, k \qquad [2e]$$

in which

$w_j =$ $(1-\rho)\rho^{j-1}$, $j = 1, 2, \cdots, M$,

$h_k =$ demand generated at node $k$,

$y_{jk} = \begin{cases} 1, & \text{if node } k \text{ is covered by at least } j \text{ facilities,} \\ 0, & \text{if node } k \text{ is covered by less than } j \text{ facilities,} \end{cases}$

$a_{ki} = \begin{cases} 1, & \text{if a facility at } i \text{ covers demands at } k, \\ 0, & \text{if a facility at } i \text{ does not cover demands at } k, \end{cases}$

$X_i =$ number of facilities located at node $i$,

$\rho =$ the probability that a facility is working, (i.e., utilization),

$M =$ number of facilities to be located,

$N =$ number of nodes in the network.

The objective function [2a] represents the expected number of calls serviced within the specified time limit. The independence assumption is critical in this

computation, since utilizations are multiplied to obtain the probability that at least one of $j$ ambulances that cover zone $k$ will be idle. Constraint [2b] is the logical constraint that defines the covering by open bases. Constraint [2c] limits the number of open bases.

Daskin's objective is based on ideas similar to Larson's system of nonlinear equations in Hypercube Approximating model. He developed a pairwise interchange heuristic for solving the model for a single $\rho$ value and extended the procedure to find good solutions for all $\rho$. He noted that the solution remained relatively stable for wide ranges of $\rho$.

There are two major shortcomings in Daskin's model. As stated above, Daskin assumed that each ambulance in the system has the same utilization. Load balancing is a desired objective, but it may not be optimal or even feasible. Actually, the utilizations are a function of the entire set of locations and the dispatching policy. Second, Daskin assumed that travel times are deterministic. This is not true in general.

Saydam and Mcknew ([25], 1985) transformed Daskin's model into a separable convex program. The new model has two advantages over Daskin's model. First, by using branch and bound, this model guarantees an optimal solution. Second, using standard linear programming packages, the model generates its solutions quickly, which is advantageous when dealing with large problems. The main weakness of this model is that it is unable to guarantee integer solutions when using a linear programming relaxation.

To complete our review of existing models, we note a paper that blends the queueing and set covering approaches. Goldberg et. al. ([15], 1988) developed a nonlinear integer programming model which considers both stochastic travel times and unequal ambulance utilizations. The model uses a travel time distribution for each potential base–zone pair. Due to the stochastic property, the proposed model remedies the previous papers' deficiencies where travel times were assumed to be deterministic. However, it has only been validated on systems with low utilization, since it assumes that all ambulances are dispatched from base. In addition, it assumes that the probability that an ambulance is busy is unaffected by the state of the system, i.e., it ignores base interdependency.

## 2.2 Applications

Facility location problems have been studied in various cities since 1960. Savas ([24], 1969) was the pioneer (see Weaver [30], 1980), using computer simulation to analyze the possible improvements in the EMS system of New York city. The author reported that for a given number of ambulances, the lowest overall average response time could be achieved by completely dispersing the ambulances. After examining the cost-effectiveness of several alternatives, Savas found that eight dispersed ambulances are as effective as ten ambulances in a satellite system at about one-fourth the incremental cost per call.

Fitzsimmons ([13], 1973) proposed a Computerized Ambulance Location Logic (CALL) algorithm to find the deployment of ambulances that minimizes mean travel time. In accordance with this objective, the final deployment substantially increased the success rate as well. The CALL methodology successfully improved

the EMS systems in both San Fernando Valley Area of Los Angeles and Melbourne, Australia.

Weaver and Church ([30], 1980) developed a multi-criteria approach which not only reduced the overall average response time, but also balanced ambulance station workload in Knox County, Tennessee. At the same time, Schilling et. al. ([26], 1980) applied mathematical and computer models to the Baltimore City Fire Department in Maryland.

Eaton et. al. ([12], 1985) reformulated a Maximal Covering Location (MCL) model by Church and ReVelle ([7], 1974), to maximize the total demand that can be covered by a fixed number of facilities. This model, applied in Austin, Texas, has saved 3.4 million (1984 dollars) in setup cost and 1.2 million (1984 dollars) in annual operating cost.

As a conclusion to this literature review, we should note that assumptions are made in all model developments, and these models have to be adapted for specific real world applications. However, the many benefits, as stated earlier in this section, make this research invaluable.

# CHAPTER 3
# MODEL DEVELOPMENT

The framework of the ambulance location models is based on a geographical partition. Our region of interest, Tucson, is divided into 405 zones of which 247 have positive demand for emergency service. This construction corresponds to zones used by county traffic planners to estimate travel times and is also compatible with our historical data. In this chapter, we discuss the data requirements of these models, and the development of travel time models to predict the EMS system performance.

## 3.1 Data Collection & Analysis

We contend that ambulance location models predict system performance better when empirical data, as opposed to fitted stationary theoretical distributions, are used. This is due to non-stationarities in both call locations and call frequencies in the real world system. For example, calls come into the system at different rates during a day, and the locations of calls change as the population location changes during the day. Therefore, unless one can capture the peculiarities of the data, a single model is not sufficient to characterize the data. If sufficient data is available, then it will be better to use the empirical data as opposed to fitting many distributions to the data. On the other hand, data required for studying the sensitivity of a solution with increasing demands or with decreasing service times

generally does not exist, and theoretical distributions have to be fitted to existing data.

Data for our model was collected from January 1, 1986, to June 30, 1986. The data set contained 10,650 calls for emergency service. Approximately 1,350 of those calls were false alarms, where the ambulance left the base and then was called back before arriving on scene. For each call, the following data were collected:

- zone where the call originated,
- ambulance that serviced the call,
- time the call was assigned to an ambulance,
- time the assigned ambulance arrived on scene, and
- time the assigned ambulance became available again to service calls.

According to the location of each call, we classified the data by zone number. The time data was precise to the nearest minute. Zone location and call arrival time were used to generate the demand processes of the system, including both actual emergency calls and false alarm calls. We classified a call as a false alarm if the time the ambulance arrived on scene for that call was not recorded. Otherwise, it was counted as an actual emergency call.

Information concerning the ambulance that answered the call was used to determine dispatching patterns. Travel times for the current set of bases were computed by taking the difference between the on-scene arrival time and the dispatch time. Service times were determined as the period from the on-scene arrival time to the time the ambulance was available again. For false alarm calls, short travel times were computed as the time duration between dispatch time and the time the

ambulance was available again. We estimated the percentage of system demand in each zone by taking the fraction of calls system wide that occured in each zone. Also, we computed the average service time per call for each zone by taking the average value of service times for all calls that occured in that zone. For validation, we determined if our dispatching assumptions were correct by looking at the dispatch patterns of historical calls and the utilizations of each ambulance. We also compared actual dispatch patterns with those generated by simulation using the currently open bases.

## 3.2 Travel Time Model

To evaluate if a set of bases is better than another, it is crucial to develop a system wide travel time model. The purpose of the model is to yield a travel time distribution for each potential base–zone pair. In this section, we discuss two approaches based on linear regression analysis. One is the method proposed by Volz ([29], 1971) and the other is an extension that first predicts mean travel times, and then uses the empirical residuals to predict the variance. The strength of both approaches is that we do not require the travel times to follow any particular distribution.

### 3.2.1 Regression Analysis

The basis of our analysis is that different road types will generate different travel speeds, and hence will result in different travel times. We divided the Tucson road network into 4 road types described below:

- $D_1$ distances of freeways, such as I–10, I–19,

- $D_2$ distances of main roads which have 5 or 7 lanes,
- $D_3$ distances of non-main roads which have 2 or 3 lanes,
- $D_4$ distances of local roads.

For each base–zone pair, the shortest path was computed and the number of miles for each road type was measured. As we assumed in chapter 1, we measured the travel distances from each base to each zone population center, as opposed to zone geographical center. So for each call, we had a travel distance on each road type and the actual travel time of that call.

One method to estimate travel times is to regress the travel distances against the actual travel time for each call. The model becomes:

$$\hat{t}_{ij} = \beta_0 + \beta_1 D_1 + \beta_2 D_2 + \beta_3 D_3 + \beta_4 D_4 + \epsilon \qquad [3a]$$

where $\hat{t}_{ij}$ is the predicted travel time between base $i$ and zone $j$. The regression coefficients represent the amount of travel time per mile for each road type above the constant term $\beta_0$. The constant term is the dispatching time and the delay in starting. The $\epsilon$ represents the residual between the predictor and the empirical data. This term can be estimated using a standard mean squared residual statistic and is discussed in the next section.

Another approach is to regress travel distances against average travel time for each base–zone pair, and then use both the predictions for mean travel time and the actual data to estimate error term variance. In our analysis, only base–zone pairs with 10 or more calls were used in the regression model. This means that 7,867 of the 9,296 actual emergency calls were used in our model. Weighted

linear regression, where the weights were the number of base–zone trips, yielded the following model:

$$\bar{t}_{ij} = \beta_0 + \beta_1 D_1 + \beta_2 D_2 + \beta_3 D_3 + \beta_4 D_4 + \epsilon \qquad [3b]$$

where the dependent variable $\bar{t}_{ij}$, is the predicted mean travel time between base $i$ and zone $j$ for each base–zone pair. However, the independent variables in this model are kept the same as those in the previous model. So, each call between a particular base–zone pair will have an identical dependent variable value in this regression calculation, as opposed to the highly varying values that are used in the first regression model.

In both approaches, [3a] and [3b], we did not take "ambulance effect" or "base location effect" into account, such as

$$\hat{t}_{kj} = \beta_{0_k} + \beta_{1_k} D_1 + \beta_{2_k} D_2 + \beta_{3_k} D_3 + \beta_{4_k} D_4 + \epsilon, \quad k = 1, 2, \cdots, N \qquad [4a]$$

or

$$\hat{t}_{ij} = \beta_0 + \beta_1 D_1 + \beta_2 D_2 + \beta_3 D_3 + \beta_4 D_4 + \beta_5 L + \epsilon \qquad [4b]$$

where $k$ represents individual ambulance or base location, $L$ is an indicator for location and $N$ is the total number of ambulances, (i.e., total number of current open bases in Tucson EMS system). We could not regress our model by using either equation [4a] or [4b], because we did not have available travel time data between each zone and the currently closed bases, (i.e., no ambulances are deployed in those bases).

### 3.2.2 Residual Analysis

The residual statistics also played a major role in the travel time model development. We computed the residuals by taking the differences between actual travel times and the values generated by the regression model. All of the 9,296 actual emergency calls were used in our calculations. The predicted residuals were then tabulated in a histogram and fitted to a theoretical distribution. This analytical procedure was applied in all regression models which are generated by both approaches stated in the previous section.

When discretizing the continuous curve, careful steps must be taken regardless of whether a theoretical or empirical distribution is used for the residual term. In our model, we used 240 equally spaced intervals to approximate the continuous distribution. Once the correct interval was found, linear interpolation was then applied to estimate the fraction within the interval. For example, each interval in the histogram of prediction errors represents 15 seconds. If the actual fraction went one third of the way into the interval, we would add 5 seconds to the travel time residual.

From all the travel time models determined, we found that the residuals were normally distributed with mean 0 minute, and the common standard deviation ranged from 2.2 to 2.4 minutes. (A detailed plot is shown in Figure 3.1, where $X$ is the total travel distance, $Y$ is the residual between actual travel time and the predicted value). The high standard deviation is attributed to the following reasons. First, the actual travel times were reported only to the nearest minute. Second, since we did not consider the time spent waiting in the queue in our regression model, some very high deviations existed. Third, driving in an urban area naturally had

a high time variance due to different traffic conditions during the day. We did not split the day into shorter periods and determine travel time models for those periods. This will certainly reduce the variance and we leave it to later analysis. Fourth, we assumed that all ambulances started serving calls from their home bases only, not enroute back to the base from a prior call. Fifth, we measured the travel distances from each base to each zone population center and not to the exact call location.

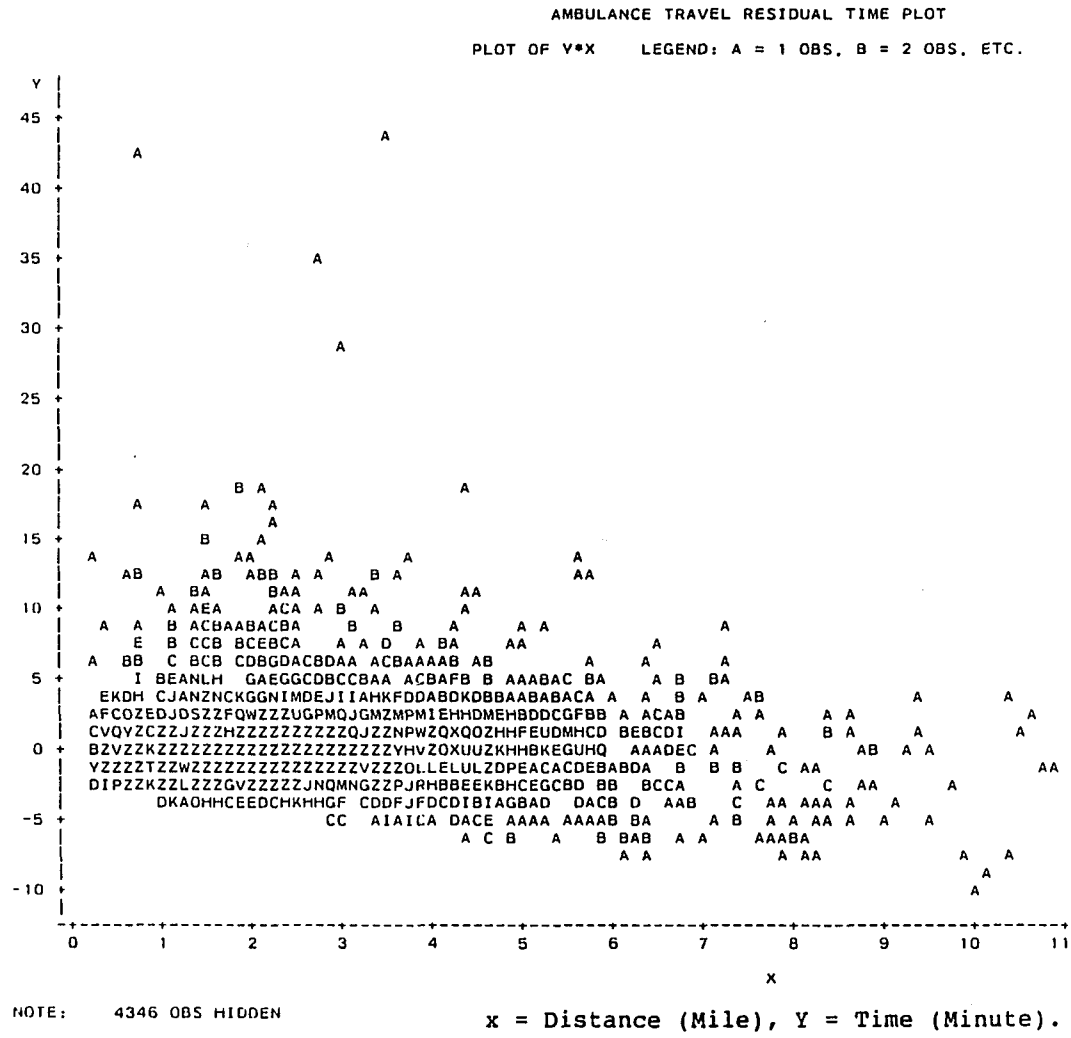



**Figure 3.1 A Plot of Travel Time Residuals**

# CHAPTER 4
# SIMULATION

The main purpose of the simulation model is to predict both the system and zone success rates of the EMS system with a particular set of ambulance locations. Besides that, the model output includes ambulance utilizations and dispatching statistics. The primary concern of any simulation in general is that it mirrors the behavior of the actual system. In this chapter, we explain the assumptions and the dynamics of the simulation used in our model. An experimental design is also included in this chapter.

## 4.1 Assumptions

In order to achieve a more tractable problem structure, several assumptions are required in our model.

1) Ambulances do not break down. They service calls 24 hours a day, every day of the year.
2) Ambulances can take calls right after service is completed.
3) Call pattern is the same as it was in 1986.
4) When a call is received, the nearest idle ambulance is dispatched.
5) The assigned ambulance follows the shortest path to service a call.

6) The capacities of hospitals are unlimited.
7) The service discipline is in a first-come-first-served (FCFS) manner.
8) All the ambulances assigned originate from home bases.
9) Exactly one ambulance is assigned to each call that is serviced.
10) Both the travel time and service time are independent of the indentity of the ambulance and the base location.
11) Travel time variance is independent of the travel distance and the time of day.

## 4.2 Methodology

The discrete event simulation was coded in Pascal and run on a VAX 11/780. (The main program is included in Appendix A). A detailed description of the simulation model and its input procedures are given in this section with the aid of figures and graphs. A brief diagram of the simulation model is presented in Figure 4.1.

### 4.2.1 Input Descriptions

For model validation, we used empirical call arrival times and service times, as stated in section 3.1, to generate arrival rates, service rates and service disciplines. The travel time procedure was the only randomness in the simulation model. Once an ambulance has been selected for a call, its travel time must be determined. No matter which travel time model is used, the procedure to generate the travel time random variable is the same. First, we substitute the appropriate base–zone distances into the regression equation (Eq. [3a] or [3b]), to obtain an initial travel

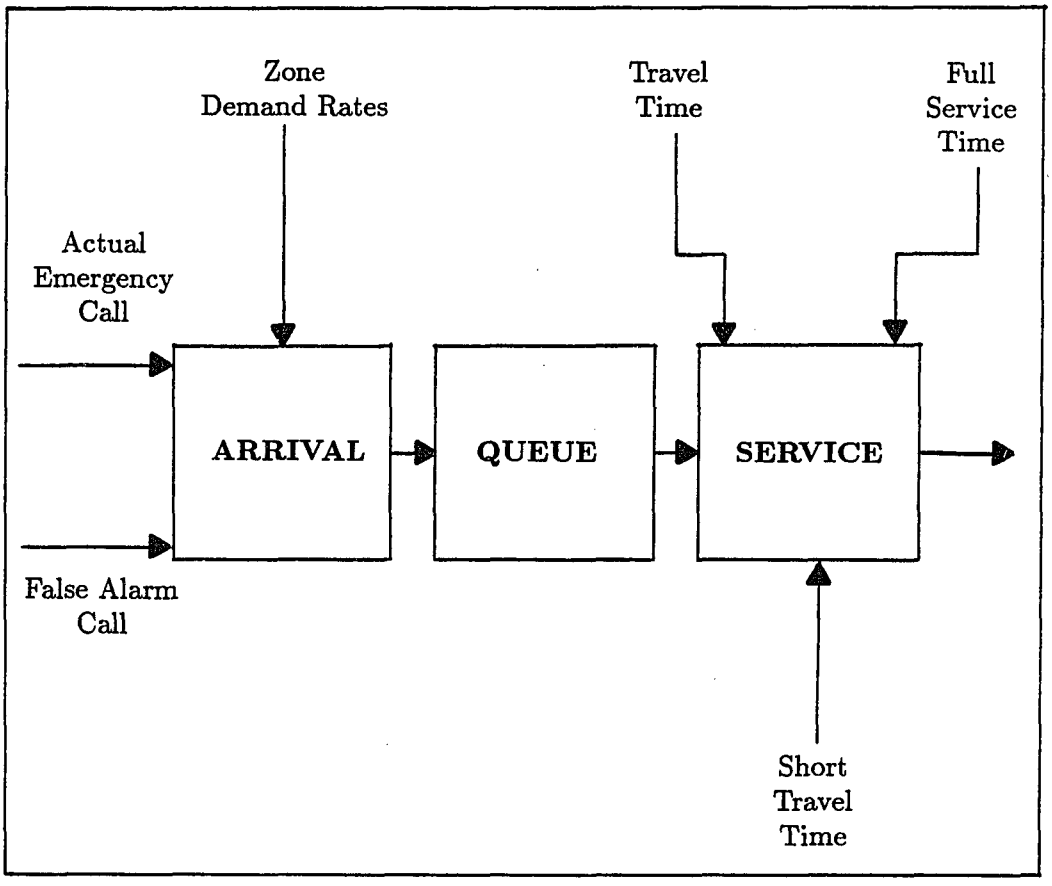



**Figure 4.1 A Brief Diagram of the Simulation Model**

time value. Next, we generate a random variable either from a residual distribution or from a theoretical fit of that distribution. The initial value is then added to the residual term.

For further studies beyond those which can use actual data, (e.g., the case studies in chapter 6), we developed 6 input distributions instead of using actual data, to drive our simulation. The empirical distributions are:

- Distribution of arrival for actual emergency calls. (Figure 4.2)
- Distribution of arrival for false alarm calls. (Figure 4.3)
- Residual (error term) distribution of travel times. (Figure 4.4)
- Distribution of service times for actual emergency calls with the validated model. (Figure 4.5)
- Distribution of short travel times for false alarm calls. (Figure 4.6)
- Distribution of zone demand rates with the original model. (Figure 4.7)

We fitted the first two inputs mentioned above to theoretical probability models. The hypothesized models were exponentially distributed with means 28.02 and 194.36 minutes for actual emergency and false alarm inter-arrival time data, respectively. Tables 4.1 and 4.2 and Figures 4.8 and 4.9 show the results of the Chi-square and Kolmogorov-Smirnov tests. In the Chi-square tests, since the empirical data were recorded only to the nearest minute, it was difficult to choose the intervals (denoted $k$) so that there is equal probability in each interval. We chose the probabilities to be greater than and nearest to 5 percent, to minimize bias in the tests. On the other hand, the Kolmogorov-Smirnov tests do not require us to group the data in a particular way, and therefore eliminate the problem of interval specification. We chose our intervals to be one and five minute increments for

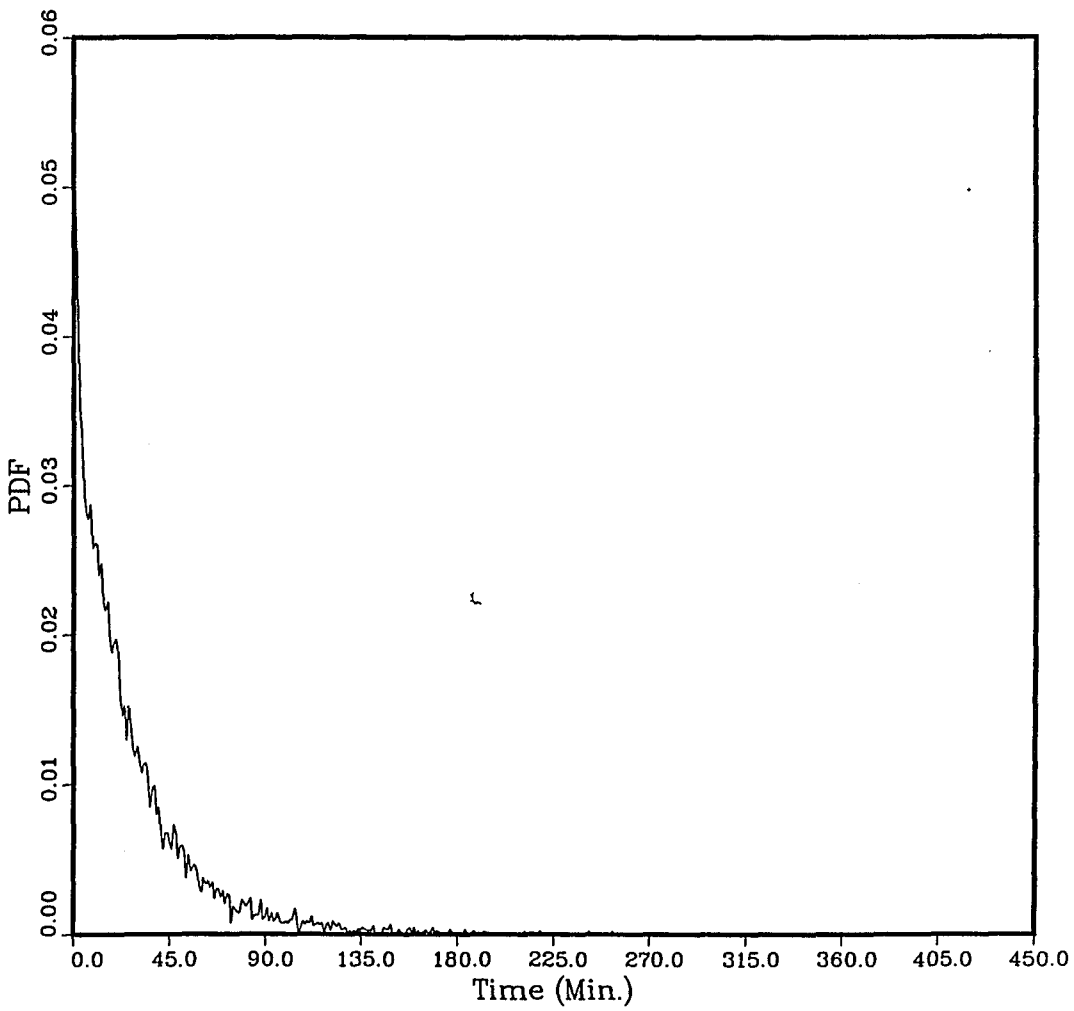



**Figure 4.2 Distribution of Arrival for Actual Emergency Calls (Empirical Data)**

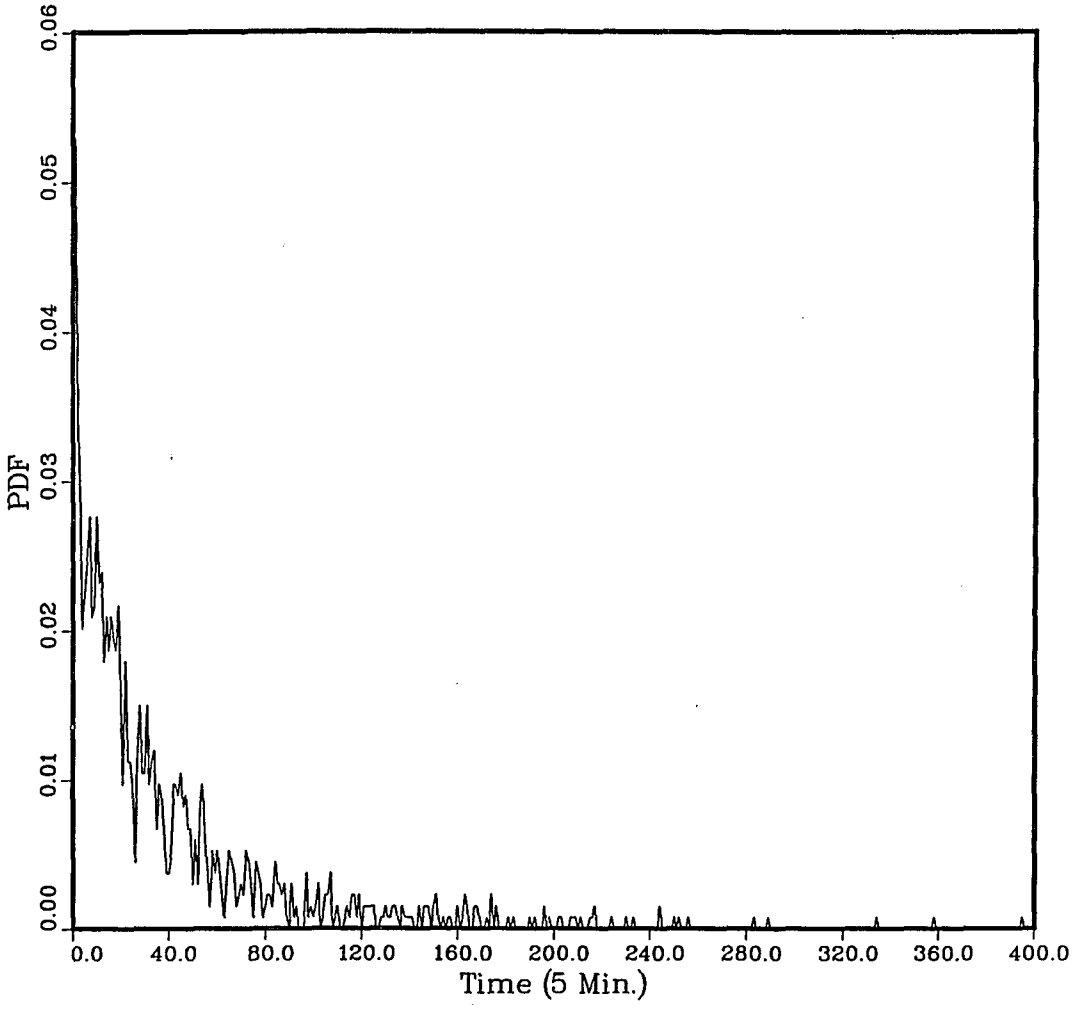



**Figure 4.3 Distribution of Arrival for False Alarm Calls (Empirical Data)**

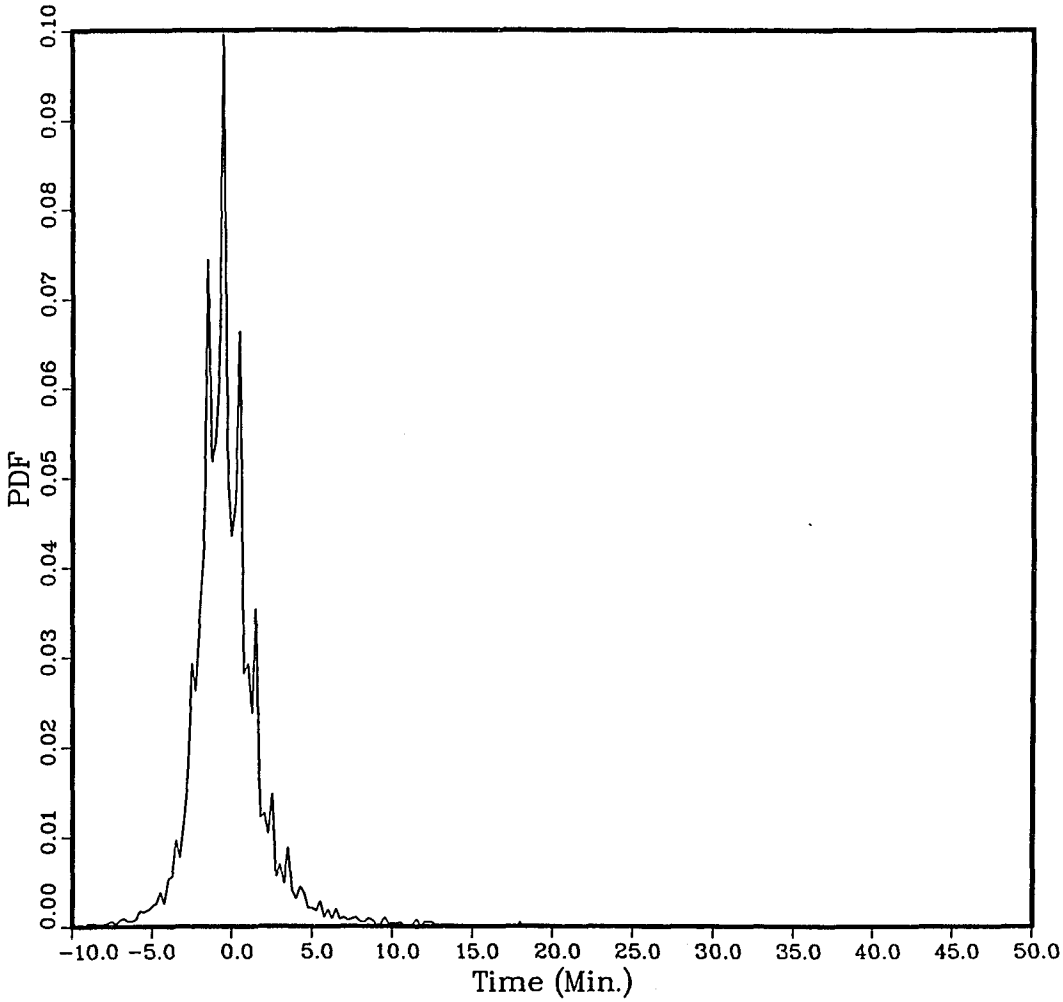



**Figure 4.4 Distribution of Travel Time Residuals (Empirical Data)**

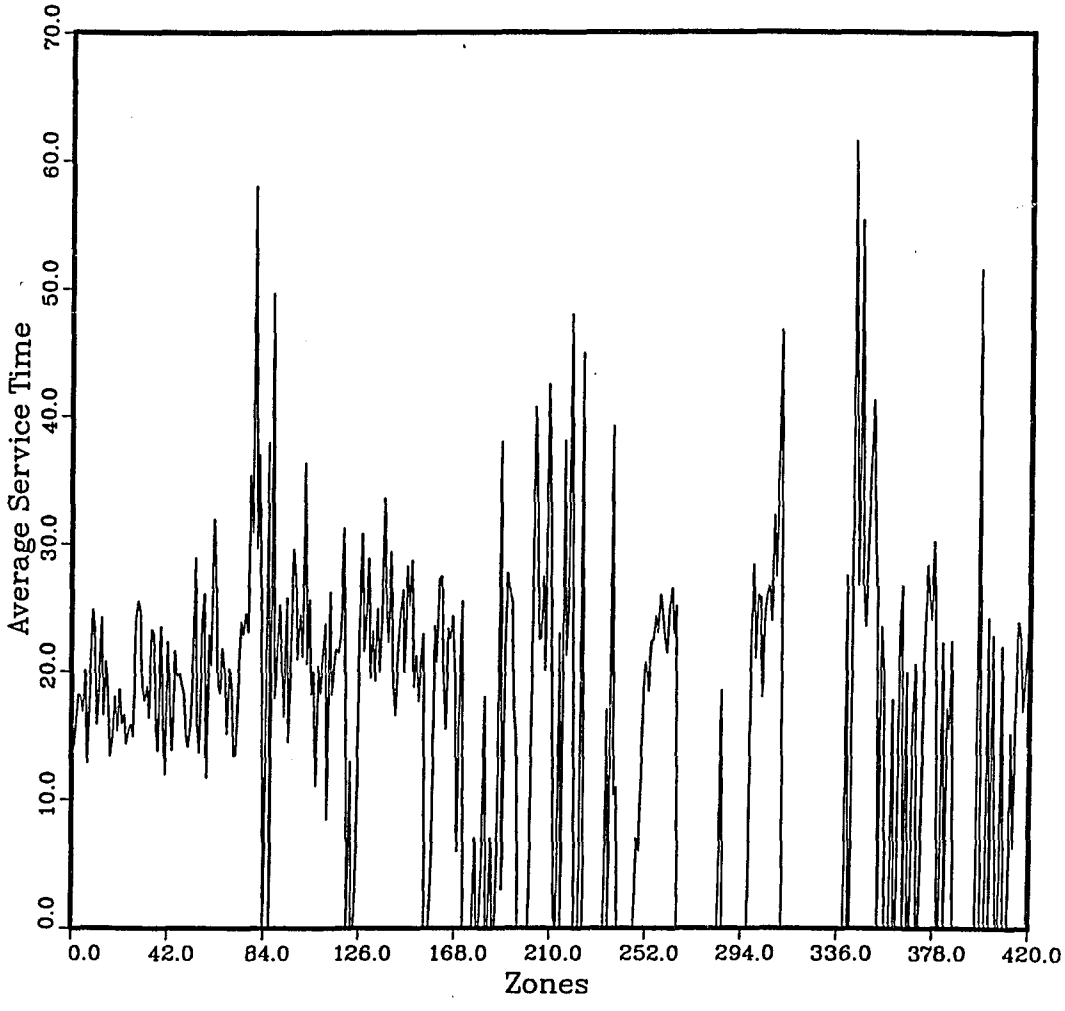



**Figure 4.5 Distribution of Full Service Times (Empirical Data, Validated Model)**

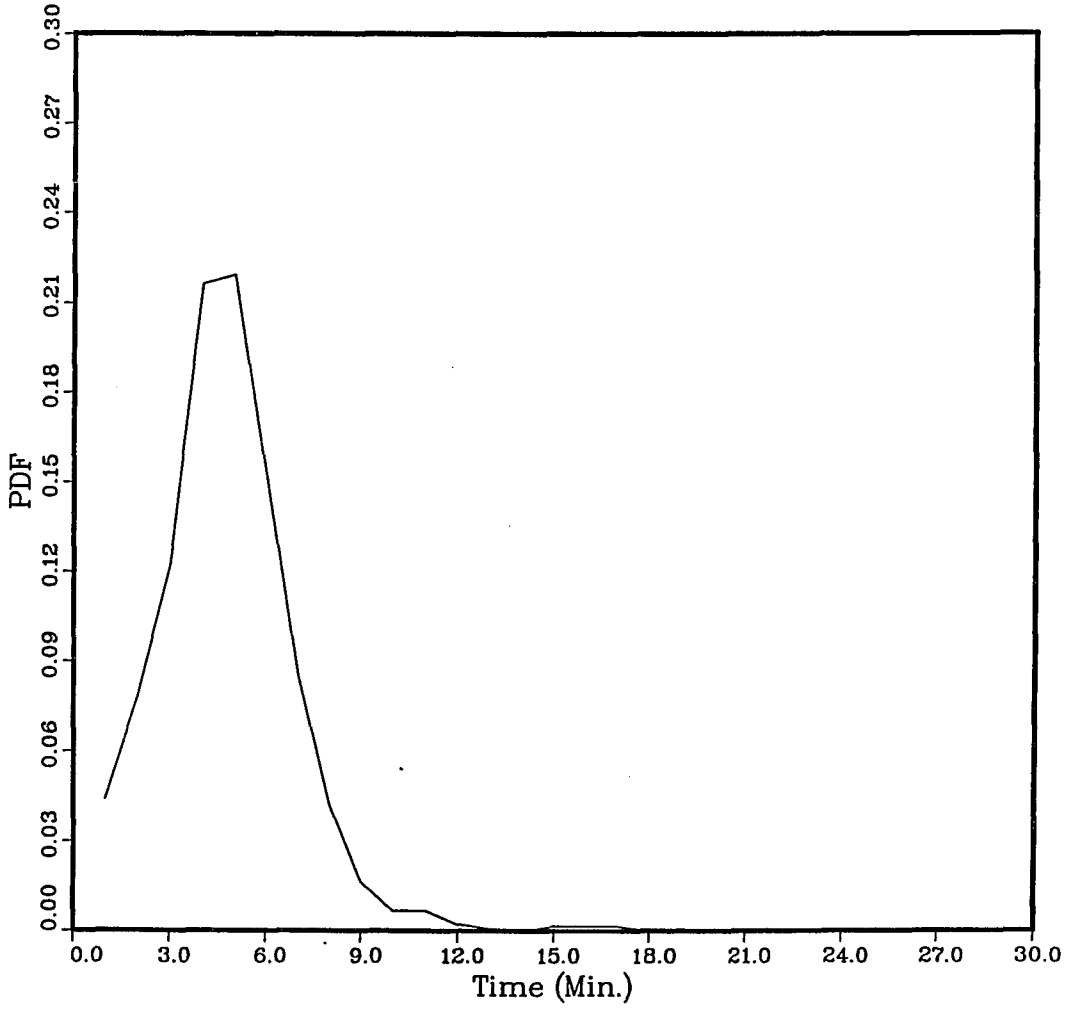



**Figure 4.6 Distribution of Short Travel Times (Empirical Data)**

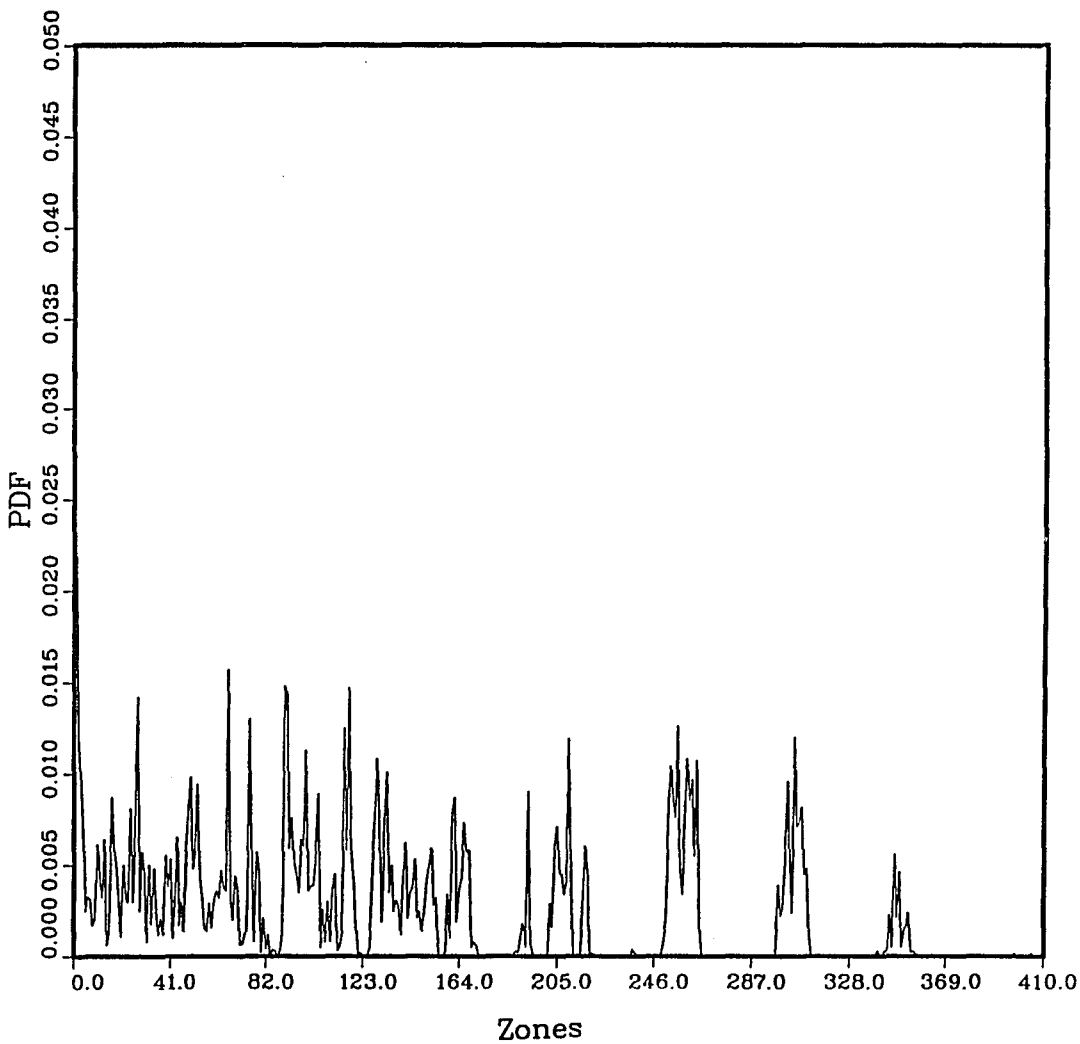



**Figure 4.7 Distribution of Zone Demand Rates (Empirical Data, Original Model)**

actual emergency and false alarm interarrival time data respectively, to conform to the empirical data.

For both the Chi-square and Kolmogorov-Smirnov tests, the actual emergency time data yielded small statistical values and indicated a good fit. However, results for the false alarm time data showed large statistical values and were very close to the critical values from the tables. Results from both tests generated statistics that were smaller than the critical values and therefore, we failed to reject the hypotheses ($H_0$) at the $\alpha = 0.01$ level. (Note that we would reject $H_0$ for certain larger values of $\alpha$ such as 0.05). When these input distributions were used in our model, the service times were still dependent on the call locations.

### 4.2.2 Algorithms

The basic structure of the simulation is a multi-server queueing model. The model starts with an empty and idle system. This initial state certainly creates initialization bias; however the bias is slight, since the entire system is often idle. The simulation runs until it exhausts either the input data set of 10,650 calls (when using actual data), or the amount of time, 260,640 minutes (when using the input distributions). The number of calls and the amount of time represent 6 months of actual operation. Replication is the only variance reduction technique used in our model. The algorithms for our simulation define a procedure to keep track of the states of counters, to update the simulation clock and to schedule new events. The simulation consists of 24 subroutines which are linked to the main program. A flow chart of the simulation is shown in Figure 4.10.

**Table 4.1 Statistical Values of Chi-square Tests for Interarrival Time Data**

| | Actual Emergency Calls | False Alarm Calls |
|---|---|---|
| Number of Intervals ($k$) | 17 | 18 |
| Number of Observations ($n$) | 500 | 500 |
| $\chi$ | 5.1957 | 29.098 |
| $\chi_{k-1,0.99}$ | 32.000 | 33.409 |
| Reject | No | No |

**Table 4.2 Statistical Values of Kolmogorov-Smirnov Tests for Interarrival Time Data**

| | Actual Emergency Calls | False Alarm Calls |
|---|---|---|
| Number of Intervals ($k$) | 450 | 400 |
| $Max\ D$ Located in Interval ($k$) | 21 | 20 |
| $Max\ D$ | 0.0321 | 0.0803 |
| $D_{k-1,0.99}$ | 0.0768 | 0.0815 |
| Reject | No | No |

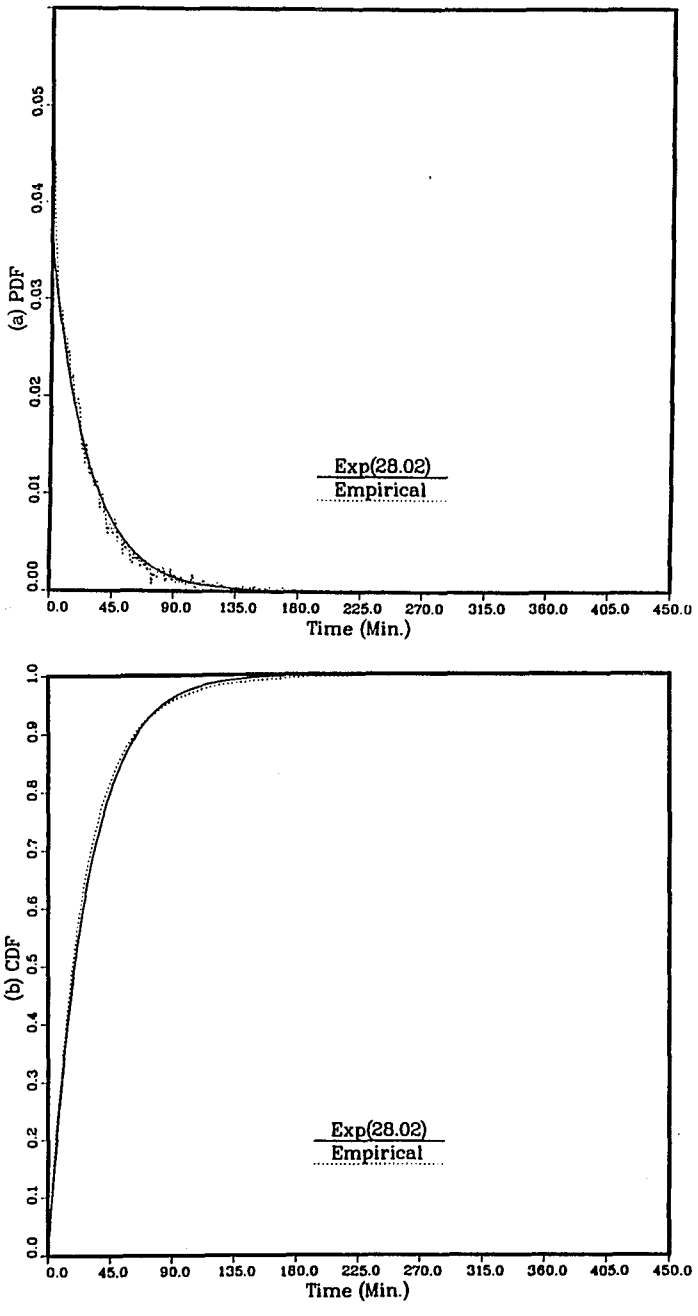



**Figure 4.8 Distributions of Actual Emergency Arrival (Theoretical vs. Empirical)**

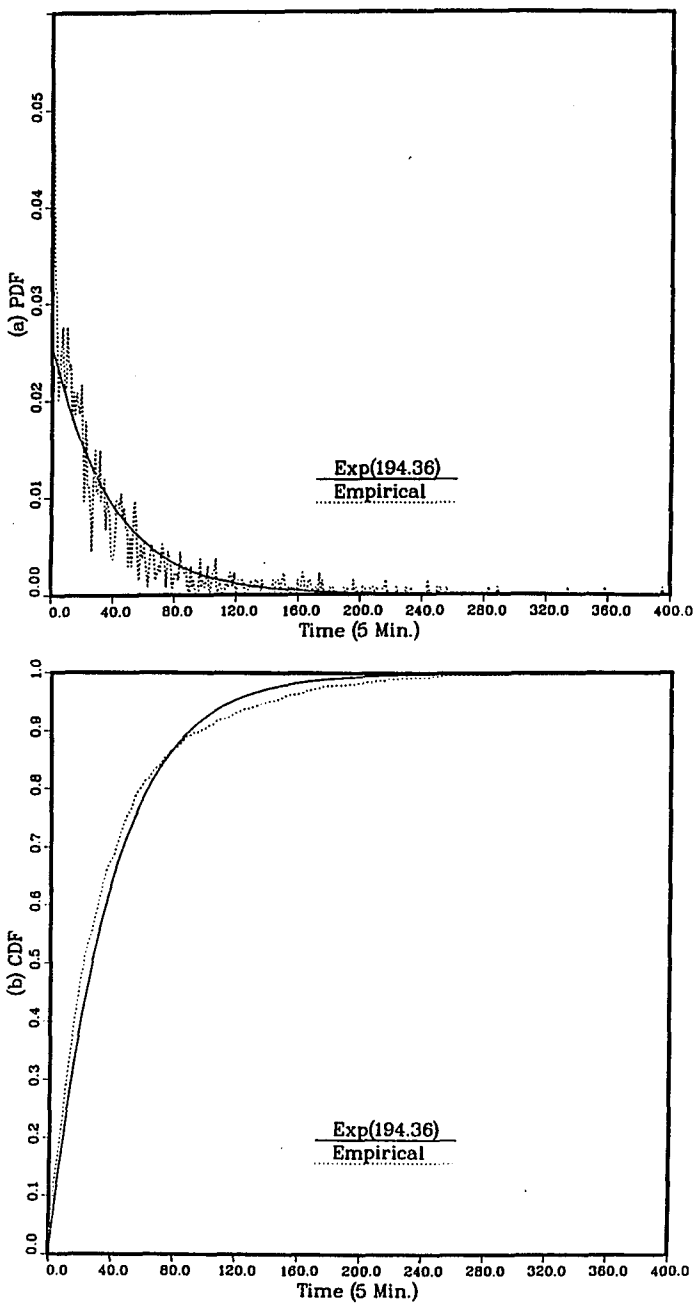



**Figure 4.9 Distributions of False Alarm Arrival (Theoretical vs. Empirical)**

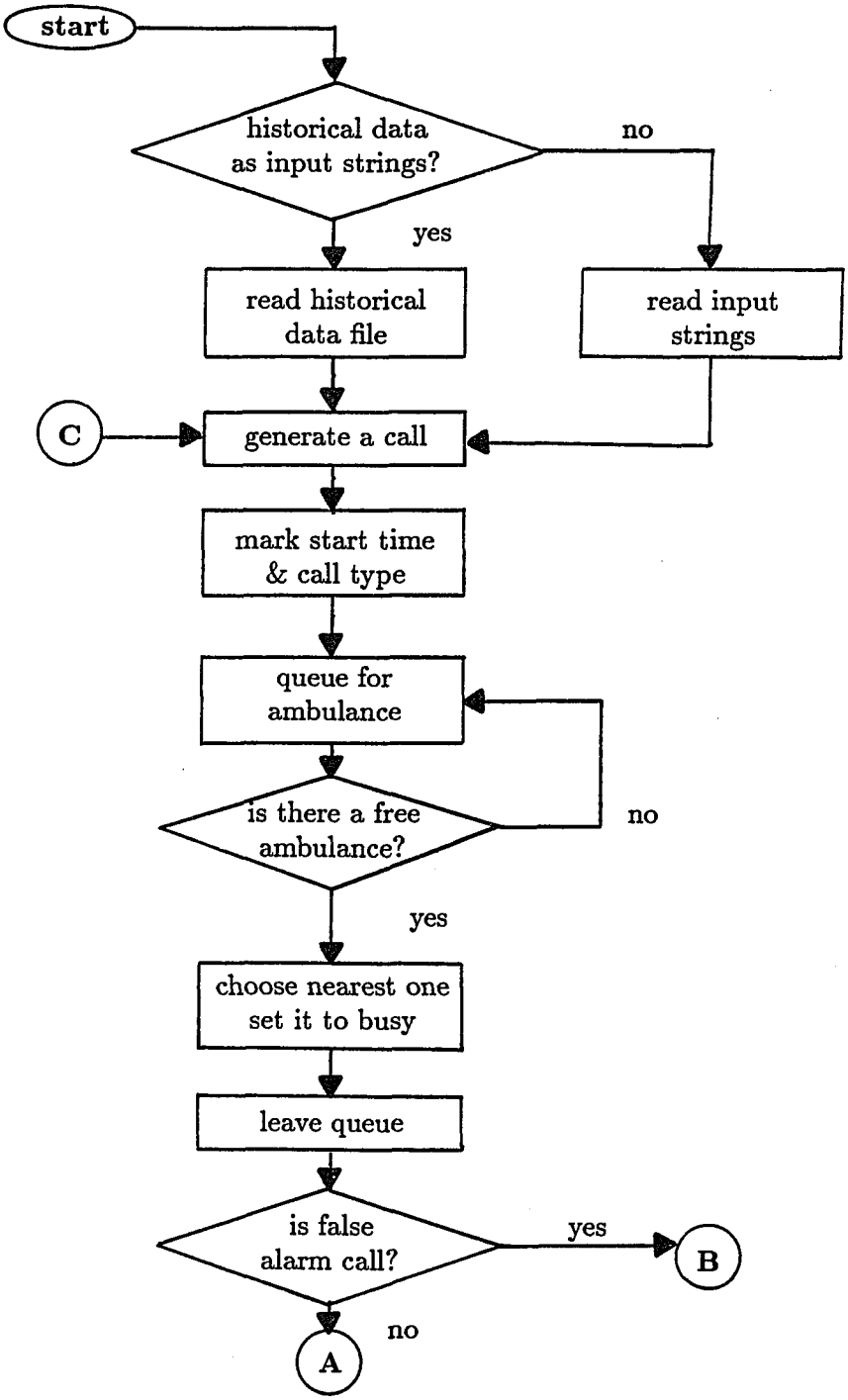



**Figure 4.10 Flow Chart of the Simulation**

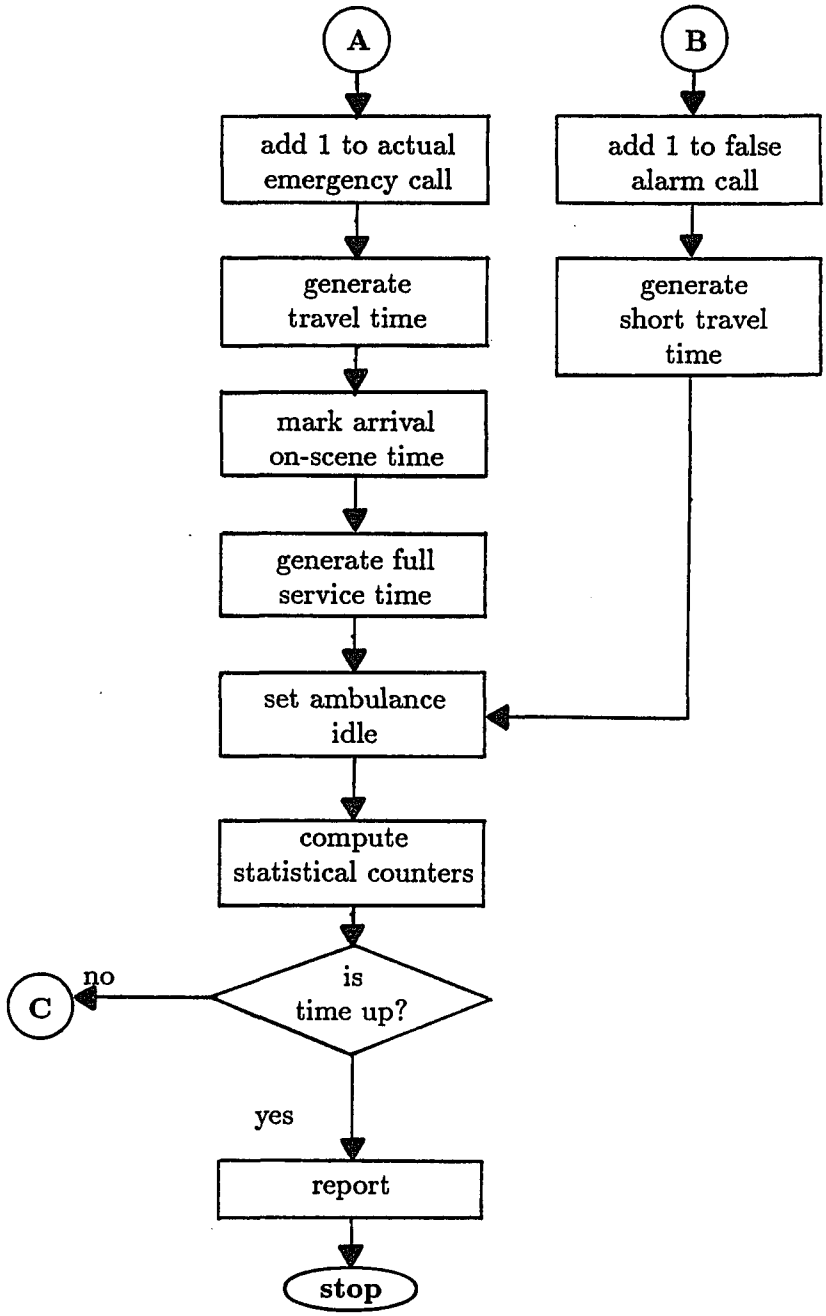



**Figure 4.10 Flow Chart of the Simulation–Continued**

## 4.3 Experimental Design

In addition to the actual call data, the input distributions and the travel time model, the simulation requires the ambulance base locations. A $t$-statistic table is used to compute confidence intervals, for the output statistics. The model has the ability to produce a wide variety of output statistics. The output of the simulation represents average values using 100 replications. The sample size of 100 runs yielded a worst case confidence interval of $\pm$ 8% (90% level) for the highest variance output statistic, the percentage of calls reached on time for each zone. If we only consider zones with 20 or more calls, then the worst case confidence interval was reduced to $\pm$ 1.5%. These confidence intervals were generated with runs that used actual arrivals and service times, in our experimental runs. Travel times were generated using regression and residual models. Each 100 replication experiment required approximately 25 minutes of the VAX 11/780 CPU time. The user can limit the number of replications by running until the confidence intervals are small enough or until a maximum number of replications has been executed.

# CHAPTER 5
# VALIDATION

We compared and validated our simulation model against both Goldberg's analytic model (denoted A-Model, and stated in chapter 2) and the actual data by using the current open bases. The A-Model was coded in PL/I and run on an AT&T UNIX PC. A brief description of the A-Model is in Appendix B. The actual data were analyzed on a VAX 11/780.

## 5.1 Validation Criteria

The first test performed was a validation to determine how close the simulation model estimated the utilizations and success rates of the current system. Particularly, we were concerned with the dispatching patterns. This is especially crucial when we attempt to evaluate proposed systems. If we are not dispatching similarly to how an actual system operates, then we do not have the ability to use the model as a predictive tool. A standard approach, such as measuring the overall success rate of the system, will not validate these location models since many factors can cancel each other out in the current system, but have significant impact in proposed systems.

Another key point for validation is to determine how close the model's performance must be to the actual system before we can claim model validity. For example, in our system, each 1% of system pickup success rate represents approximately 100 calls over a period of 6 months. Due to the potentially high impact of each call, any small errors are preceived as critical. Unfortunately, the errors and approximations are unavoidable when applying a model to real world situations. Therefore, model validation must come down to a group decision between decision-makers and the administration.

## 5.2 Validation Procedure

A procedure for validation is presented in Figure 5.1. Initially, we ran the simulation model, compared and analyzed the validation criteria, and then had to decide whether the model was validated. If the model was not validated, we then changed the zone structure by dividing some of the zones into smaller ones in step 2. In step 3, we re-measured the travel distances from each base to the newly created zones. By using the regression and residual analyses stated in section 3.2, we generated a travel time model in step 4. Following that, by comparing the actual dispatched ambulance to each zone with the closest ambulance to that zone according to our travel time model, we either adjusted the travel distances or went back to step 1. We made sure the closest ambulance to a particular zone is exactly the same one dispatched in the real system. Totally, it took three iterations of model development and data adjustment to validate the simulation model.

We developed the initial travel time model by using approximately 50% of the potential base–zone pairs. The second regression approach (Eq. [3b]), was used

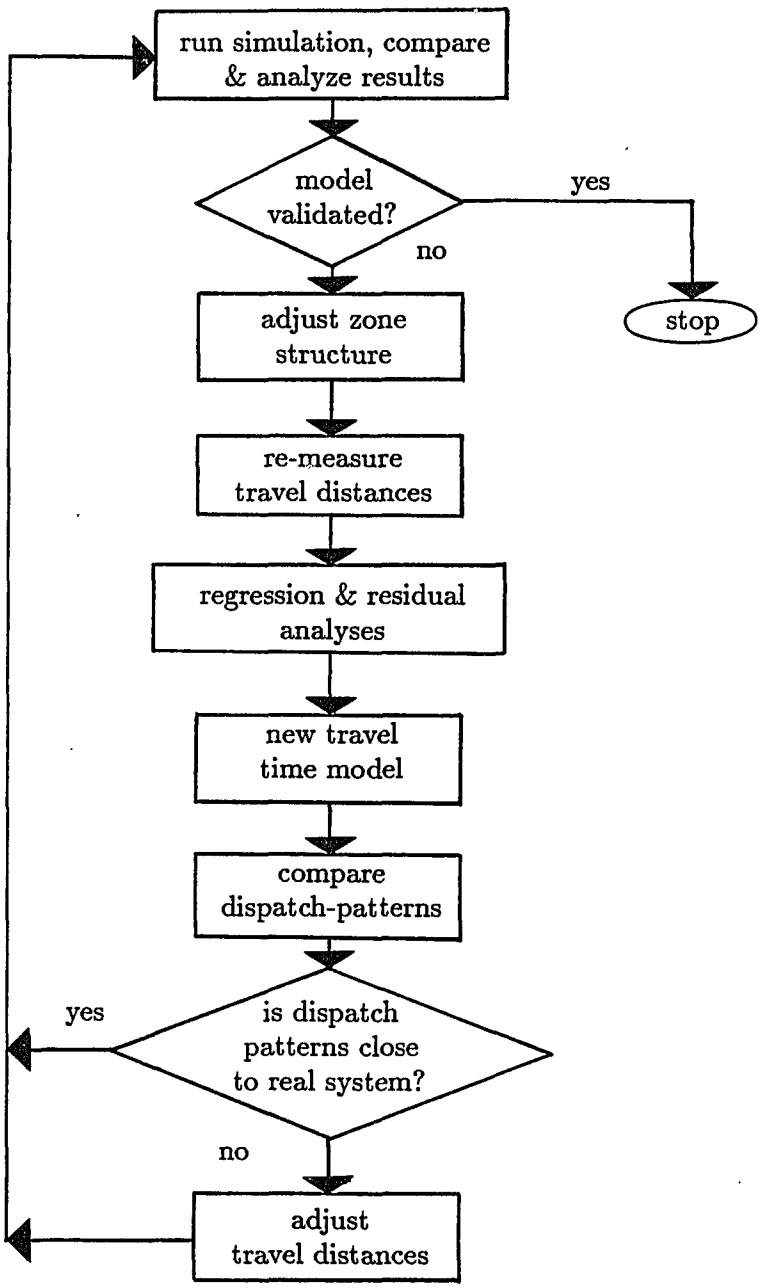



**Figure 5.1 Procedure for Model Validation**

since the data indicated that the spread in travel times for any single pair was large. We generated the following model:

$$\bar{t}_{ij} = 3.535 + 0.631D_1 + 0.676D_2 + 0.717D_3 + 0.811D_4 + \epsilon.$$

The predicted errors (denoted $\epsilon$) were computed and then fitted to a distribution which was normally distributed with mean 0 and standard deviation of 2.175 minutes. We simulated the system, but the results were not satisfactory. This is due to the discrepancies in both the dispatch patterns (Table 5.2) and the success rates (Figure 5.2) between our simulation output and actual data.

Since the initial model was not validated, our next effort was to develop all base–zone paths and re-run the regression calculations. We ran both regression approaches and developed the following models:

$$\hat{t}_{ij} = 3.500 + 0.688D_1 + 0.636D_2 + 0.733D_3 + 0.696D_4 + \epsilon$$

where $\epsilon$ is normally distributed with mean 0 and standard deviation of 2.39 minutes, and

$$\bar{t}_{ij} = 2.947 + 0.807D_1 + 0.889D_2 + 1.004D_3 + 1.121D_4 + \epsilon$$

where $\epsilon$ is also normally distributed with mean 0 and standard deviation of 2.41 minutes. We simulated the system by using both travel time models, but the results were still disappointing. In the next section, we discuss and analyze the results in detail.

## 5.3 Results Discussion & Analysis

Validation test results for the original model are contained in Tables 5.1 and 5.2, and Figures 5.2 and 5.3. The results of the utilization section of the analysis

**Table 5.1 Utilization Values for the Current System**

| Base Location | Actual Data | Analytic Model | Simulation Model |
|---|---|---|---|
| 23 | 0.157 | 0.155 | 0.151 |
| 29 | 0.119 | 0.151 | 0.148 |
| 69 | 0.168 | 0.163 | 0.169 |
| 132 | 0.128 | 0.127 | 0.130 |
| 150 | 0.128 | 0.120 | 0.124 |
| 259 | 0.141 | 0.158 | 0.151 |
| 298 | 0.122 | 0.115 | 0.116 |

are given in Table 5.1, and the results concerning success rate and call statistics for the system and each ambulance are in Table 5.2. Figure 5.2 contains plots of the success rates for all zones with 10 or more calls over the 6 months period. Zones with extremely small demand rates are subject to large variations in performance and do not reflect overall model validity. Figure 5.3 represents the differences in the success rates between the simulation model and the A-Model.

### 5.3.1 Utilization Phase

From Table 5.1 and Figure 5.3, we can see that the utilizations generated by the simulation model closely match with those generated by the A-Model. The

**Table 5.2 Success Rates and Zone Statistics for the Current System**

| | Actual Data | | A-Model | | Simulation Model | |
|---|---|---|---|---|---|---|
| Base Location | Success Rates | Total Calls | Success Rates | Total Calls | Success Rates | Total Calls |
| System | 0.920 | 9296 | 0.897 | 9296 | 0.895 | 9296 |
| 23 | 0.928 | 1923 | 0.912 | 1721 | 0.913 | 1710 |
| 29 | 0.932 | 1112 | 0.890 | 1480 | 0.898 | 1414 |
| 69 | 0.922 | 1578 | 0.889 | 1464 | 0.883 | 1538 |
| 132 | 0.938 | 1166 | 0.928 | 1176 | 0.926 | 1192 |
| 150 | 0.921 | 1228 | 0.887 | 1105 | 0.879 | 1145 |
| 259 | 0.906 | 1233 | 0.885 | 1365 | 0.888 | 1309 |
| 298 | 0.886 | 1056 | 0.891 | 979 | 0.881 | 988 |

small differences can be attributed to the base interdependency not accounted for in the A-Model and the use of a single false alarm time. Similar insights between the simulation model and the A-Model can be derived from Table 5.2.

### 5.3.2 Performance Phase

As expected, the simulation model did not match the actual system values exactly. System performance was about 2.2% off. Figure 5.2 graphically shows how

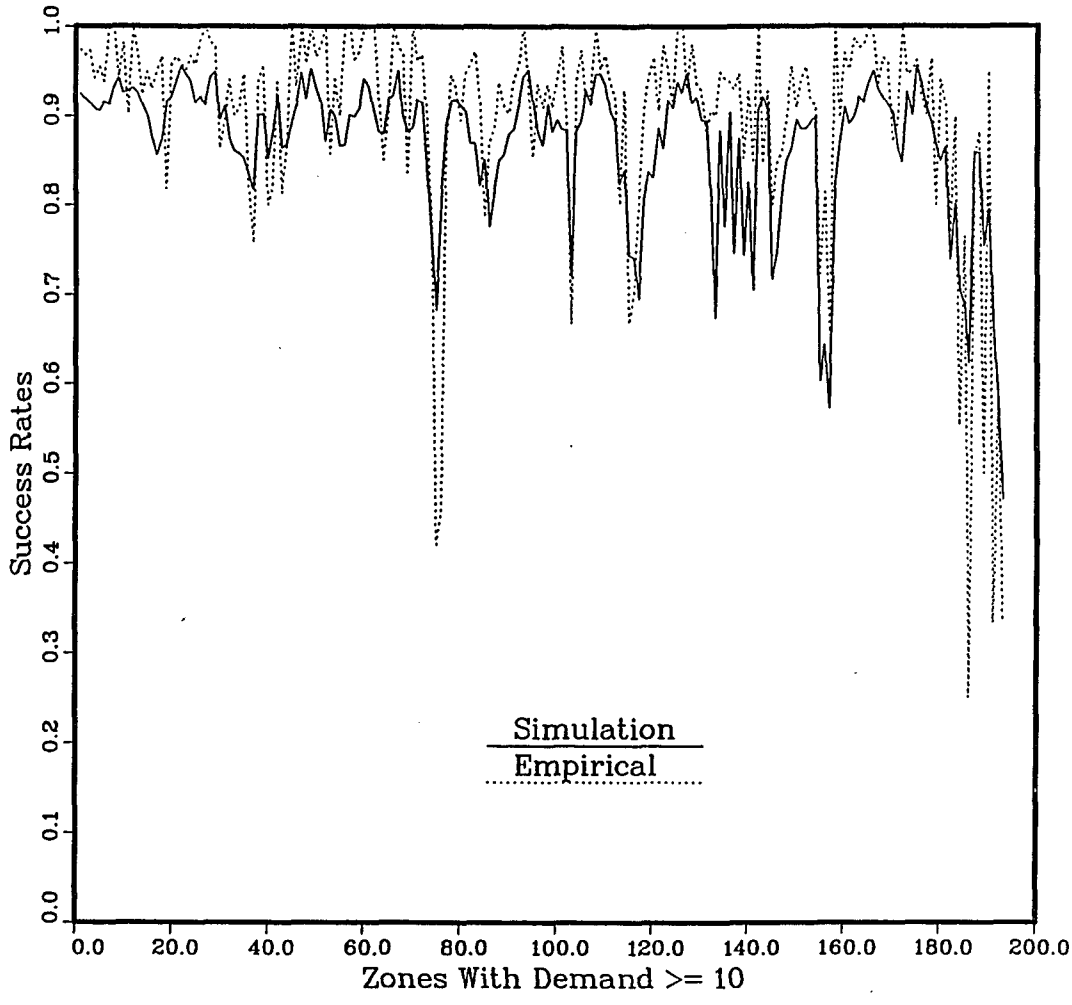



**Figure 5.2 Success Rates, Simulation vs. Empirical**

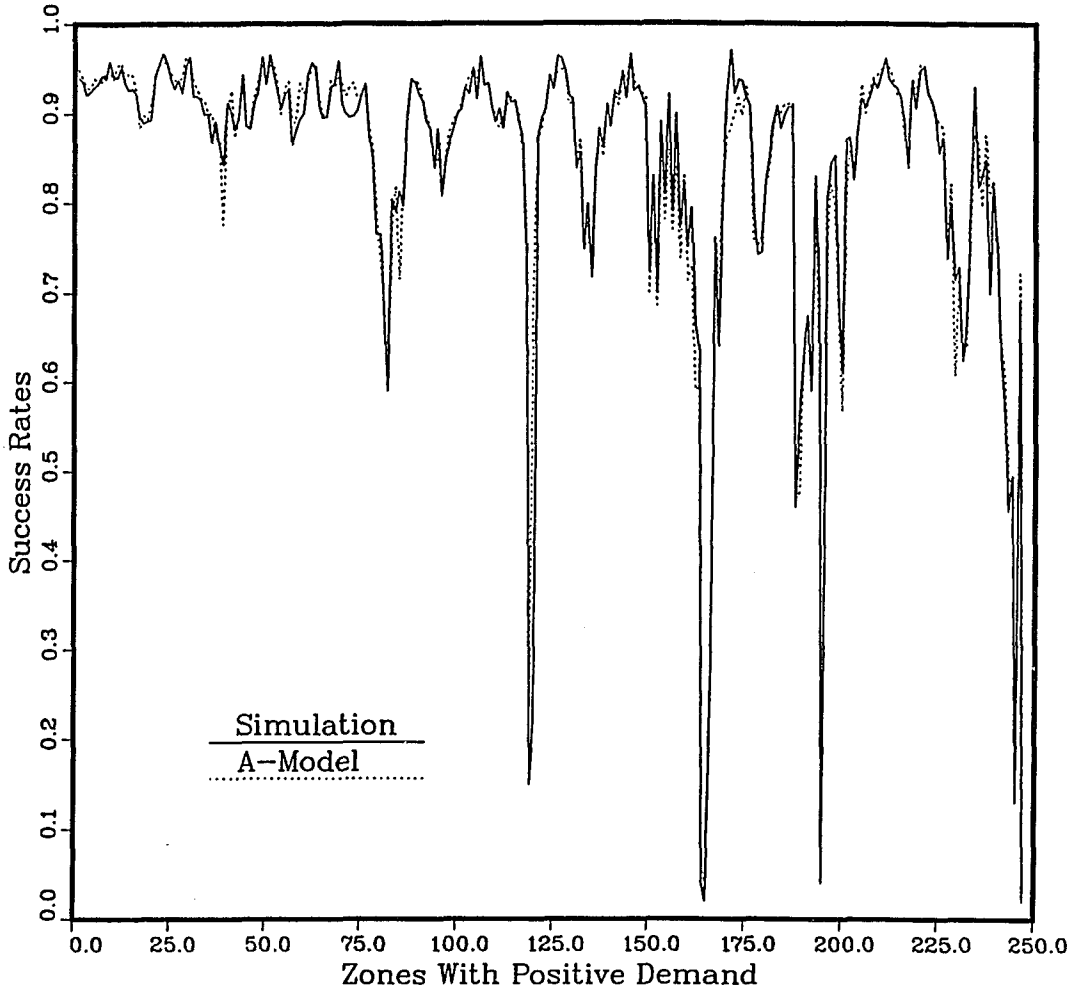



**Figure 5.3 Success Rates, Simulation vs. A-Model**

different the zone success rates are. As noted before, for zones with 1 or 2 calls (not shown on the graph), there can be large differences in performance. In the real system, for example, there were zones with 1 call that was successfully picked up even though the probability of successful pickup was very small. This is due to good fortune and the possibility that the ambulance was not dispatched from its base, but from a location closer to the call.

Another potential cause of the difference between the plots was the distribution of the travel time residuals. Through approximation of the residual distribution with the normal distribution, we overestimated critical probabilities. This is demonstrated in Figure 5.4, where we ploted the theoretical cumulative distribution against the actual cumulative distribution of the residuals. The higher complementary cumulative probabilities for the 2.5 to 3.5 minute residual values of the theoretical Normal lead to more instances of residuals in the range where calls are unsuccessfully reached. This follows since the average travel time is approximately 5.1 minutes and we add the generated residual to the average base–zone travel time. So, the use of Normal distributions tends to underestimate the success rate for the case.

In later models we used the empirical distribution of the residuals with 0.25 minute intervals. Also, in the first model, we used a coarse grid (0.1) when evaluating the random variable realizations. In later models we used a finer grid and a linear interpolation to determine travel time realizations.

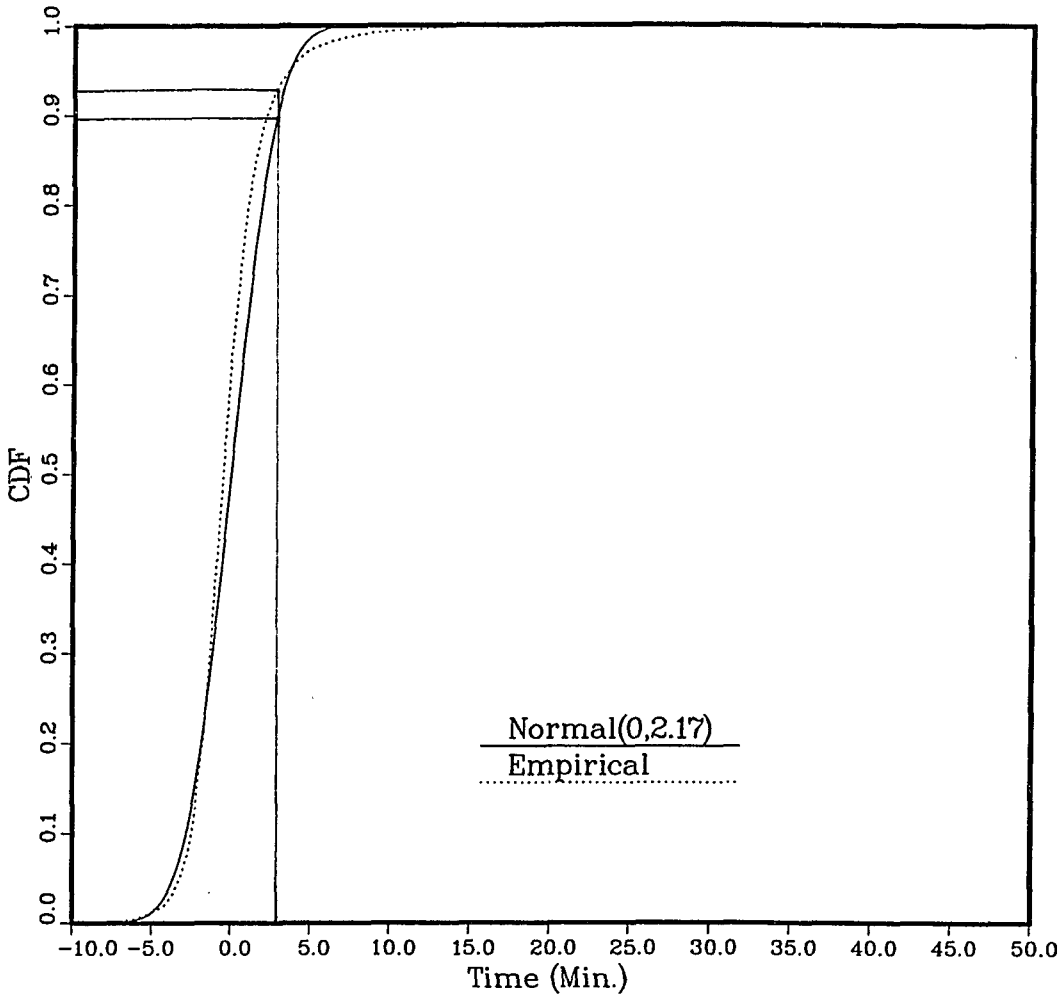



**Figure 5.4 Cumulative Residual Distributions (Theoretical vs. Empirical)**

### 5.3.3 Dispatch Phase

As shown in Table 5.2, there were significant differences in dispatching, especially between bases at zones 23 and 29. Upon closer examination of the simulation results, we found that of the 247 zones with positive demands, the dispatching patterns of the simulation model were significantly different from the actual system 25% of the time.

By way of discussion with the client, we discovered various causes. First, the road network in 1986 was different from the current network. In particular, a new road was built that allowed an ambulance based at zone 29 to service calls on the south side of town. Since our travel time computations included that road, we assigned substantially more calls to that base than the real system did. After discussions with the client, it was decided that the current road network was appropriate to use.

Second, there were zones where the real system essentially split the demand for a zone between two bases. It is clear that our model could never do this for low ambulance utilization values since we always assigned the nearest idle ambulance to respond the call. Upon further investigation, it was discovered that these split zones were equi-distant between two or more open bases. To remedy the situation, we made the zones smaller so that there was a clear choice for the most appropriate base for each zone. However, changing the zone structure required a lot of additional work as described in Figure 5.1. We initially considered random assignment of ambulances from those equi-distant bases to the zones. But we found that this approach could not be used due to the following reasons.

1) Since ambulances took longer time to travel to the population center of a bigger zone than to the center of the closer half of that zone (i.e., when the zone was split into two smaller ones), we were overestimating the travel time by using random assignment instead of splitting those zones.
2) In potential systems, we would not know the correct percentage of calls to split. So, we could not generate the correct percentage out of the empirical values.

Third, we were greatly underestimating the travel time required in the downtown section of the city. This was caused by ignoring some one-way streets and classifying downtown traffic as local traffic. The widths of the downtown streets were similar to those of local streets, however the traffic volume was greatly increased. To remedy this problem, we adjusted the local distances in routes that used the downtown streets.

Finally, we found that there were inherent biases in the dispatching where the closest base (according to our travel time model), was not used. This was especially evident in the downtown area, in the far east side of town (bases at zones 259 and 298), and in the far north side of town. We adjusted for these biases by "stretching" the map appropriately when making dispatching decisions, and "unstretching" when computing the success probabilities or the travel time realizations.

## 5.4 Validated Model

By using the processes described in section 5.2, after splitting the zones, adjusting travel time distances, and doing the regression analysis, we generated the

**Table 5.3 Utilization Values for the Current System (Validated Model)**

| Base Location | Actual Data | Analytic Model | Simulation Model |
|---|---|---|---|
| 23 | 0.157 | 0.172 | 0.165 |
| 29 | 0.119 | 0.137 | 0.133 |
| 69 | 0.168 | 0.163 | 0.169 |
| 132 | 0.128 | 0.127 | 0.129 |
| 150 | 0.128 | 0.128 | 0.124 |
| 259 | 0.141 | 0.136 | 0.136 |
| 298 | 0.122 | 0.125 | 0.126 |

final travel time model:

$$\bar{t}_{ij} = 2.838 + 0.792D_1 + 0.929D_2 + 1.006D_3 + 1.153D_4 + \epsilon.$$

In this model, there were 277 zones with positive demand. (See Figure 5.5). The distribution of travel time residuals ( $\epsilon$ ) for this model is given in Figure 5.6. We again compared the simulation model against the A-Model and the actual data. The results are in Tables 5.3 and 5.4 and Figures 5.7 and 5.8.

Comparison of the results from our model with those from the A-Model shows a close match in both dispatching patterns and success rates. However, when the results of our model are compared to the actual data, we find that the success

**Table 5.4 Success Rates and Zone Statistics for the Current System (Validated Model)**

| | Actual Data | | A-Model | | Simulation Model | |
|---|---|---|---|---|---|---|
| Base Location | Success Rates | Total Calls | Success Rates | Total Calls | Success Rates | Total Calls |
| System | 0.920 | 9296 | 0.904 | 9296 | 0.910 | 9296 |
| 23 | 0.928 | 1923 | 0.916 | 1881 | 0.923 | 1871 |
| 29 | 0.932 | 1112 | 0.904 | 1336 | 0.913 | 1280 |
| 69 | 0.922 | 1578 | 0.904 | 1452 | 0.902 | 1539 |
| 132 | 0.938 | 1166 | 0.939 | 1166 | 0.937 | 1185 |
| 150 | 0.921 | 1228 | 0.905 | 1133 | 0.895 | 1137 |
| 259 | 0.906 | 1233 | 0.914 | 1190 | 0.914 | 1201 |
| 298 | 0.886 | 1056 | 0.891 | 1069 | 0.886 | 1083 |

rates are still about 1.2% off. This is due to inevitable reasons that were stated in section 3.2.2. In addition, there are also discrepancies in the dispatching patterns, but all those can be explained by changes in road network or an inconsistency in the dispatching in the real system. Despite the discrepancies in success rates and dispatching patterns, both the administration and the clients were satisfied that the simulation model was validated.

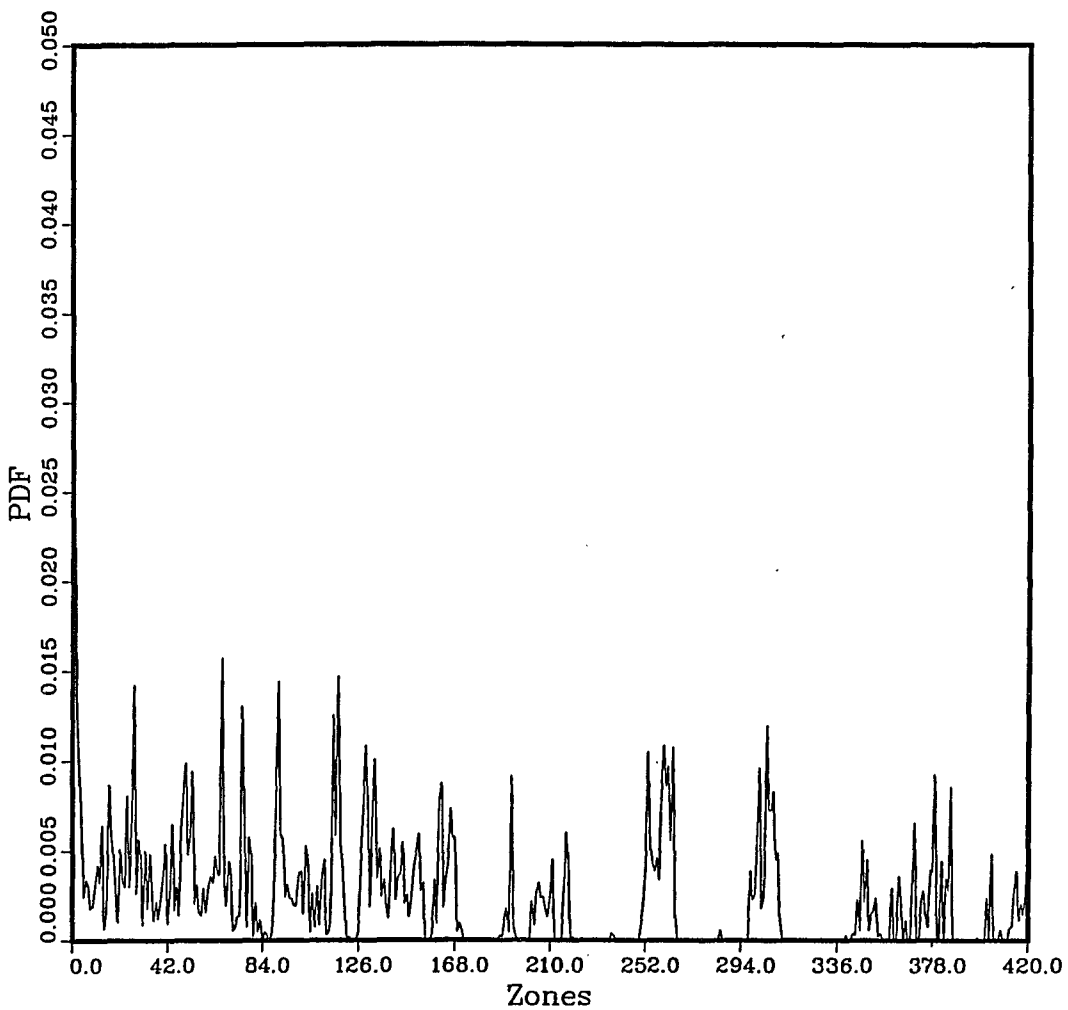



**Figure 5.5 Distribution of Zone Demand Rates for the Validated Model (Empirical Data)**

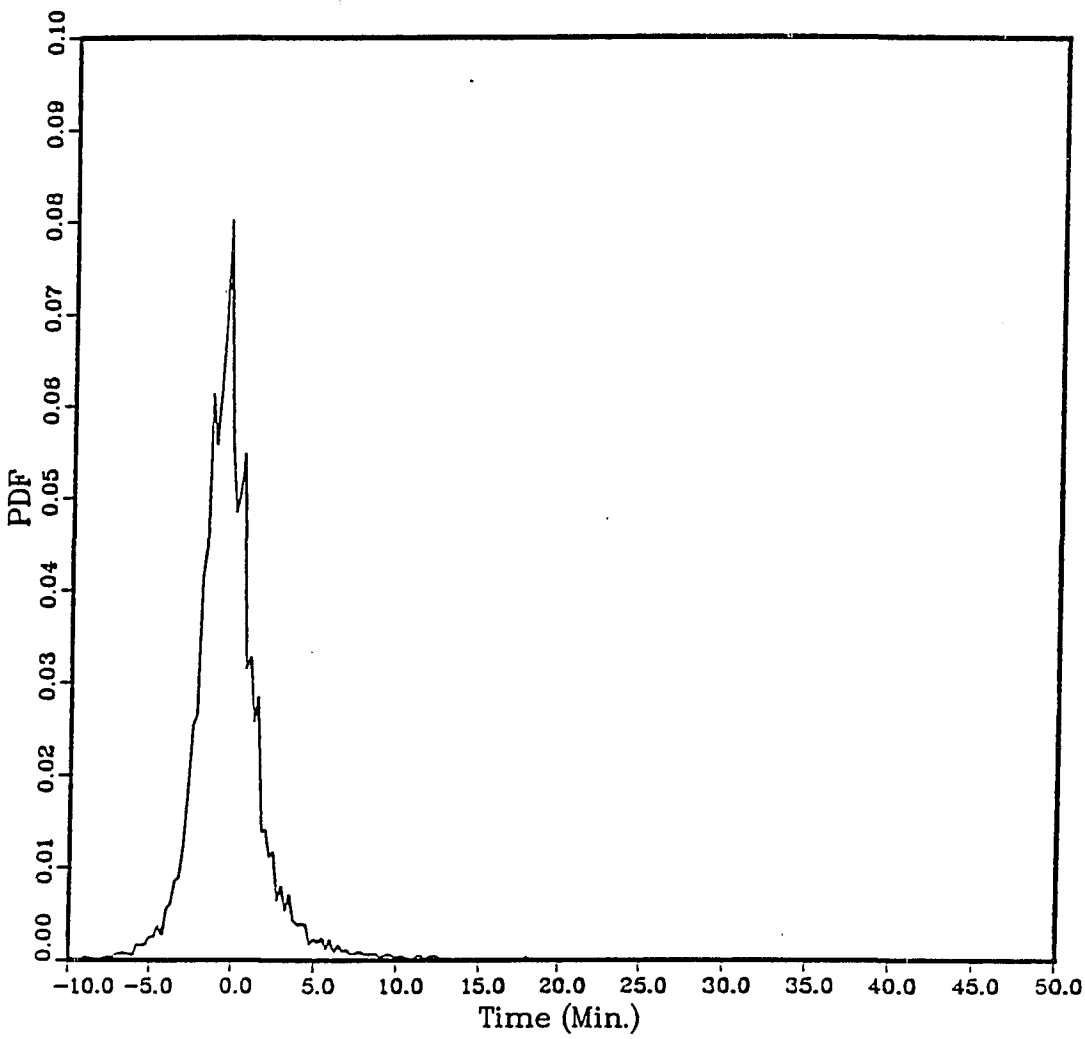



**Figure 5.6 Residual Distribution of the Validated Model (Empirical Data)**

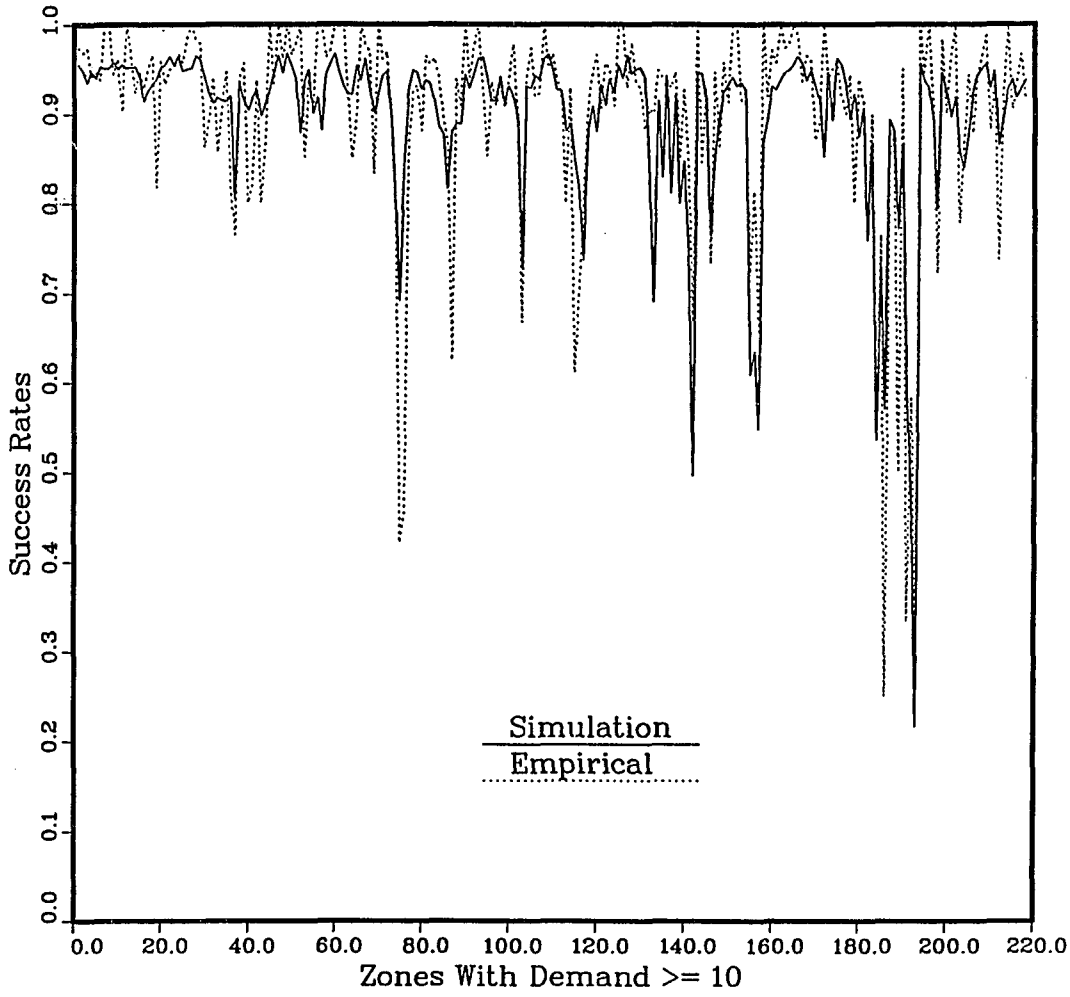



**Figure 5.7 Success Rates of the Validated Model (Simulation vs. Empirical)**

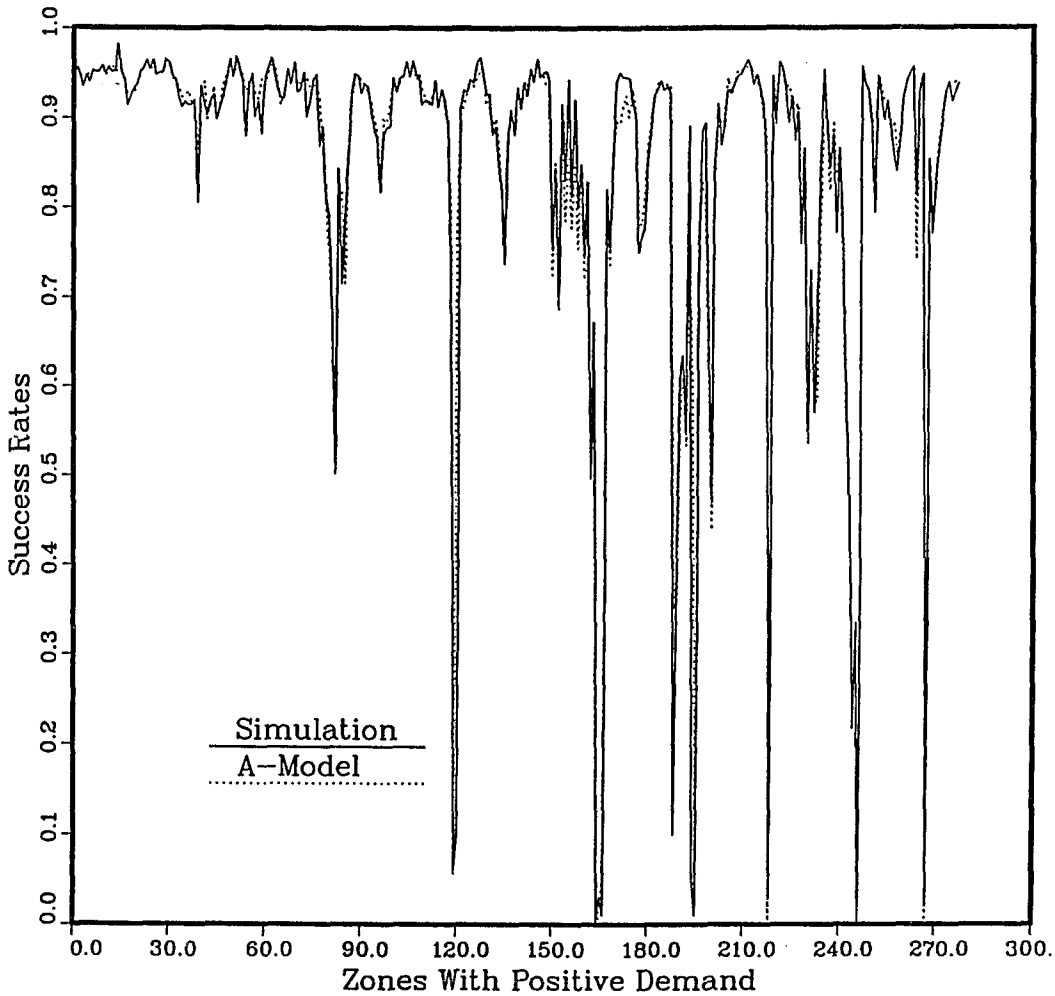



**Figure 5.8 Success Rates of the Validated Model (Simulation vs. A-Model)**

# CHAPTER 6
# MODEL IMPLEMENTATION

The simulation model has been used for three case studies. The first study was to predict the system performance with increasing demands using the current bases. The second study was to evaluate two alternative systems that would improve service performance to the northern sections of the city. Finally, the third study was to use the interchange heuristic proposed by Goldberg ([15], 1988), to determine if a better system could be found.

## 6.1 Case Study 1

With the expectation of future growth in the Tucson area and population, demand for the EMS system will increase yearly. To ensure successful evaluation of proposed systems, it is necessary that our model predicts system performance with increasing demand rates. In this study, we have predicted the system performance up to the year 1990 by using the demand rate in 1986 as base, and by increasing demand 5% per year uniformly across the whole area, while the other inputs to the simulation remain the same, as stated in section 4.2.1. The results are shown in Table 6.1 and Figure 6.1.

**Table 6.1 Future Performances of the Validated Model for the Current System (Using the Fitted Input Distributions)**

| Year | Total Calls | Utilization Values | Success Rates |
|---|---|---|---|
| 1986 | 9,132 | 0.125 | 0.894 |
| 1987 | 9,588 | 0.132 | 0.889 |
| 1988 | 10,068 | 0.140 | 0.883 |
| 1989 | 10,571 | 0.150 | 0.876 |
| 1990 | 11,100 | 0.162 | 0.868 |

In Figure 6.1, the solid line shows how the system overall success rate declines as demand increases. On the other hand, as demand increases, ambulance utilization also increases, as shown by the dotted line. One should note that both functions are not linear due to increased demand rates as well as longer mean travel time. For example, if the utilization is higher, the probability of the closest ambulance responding to a call is lower.

## 6.2 Case Study 2

Two alternative systems suggested by the administration were tested in this section. Both systems were identical to the existing system, except a new base at zone 97 was opened, while an existing base was closed. In system 1, the base at zone 29 was closed, while in system 2, the base at zone 150 was closed. The

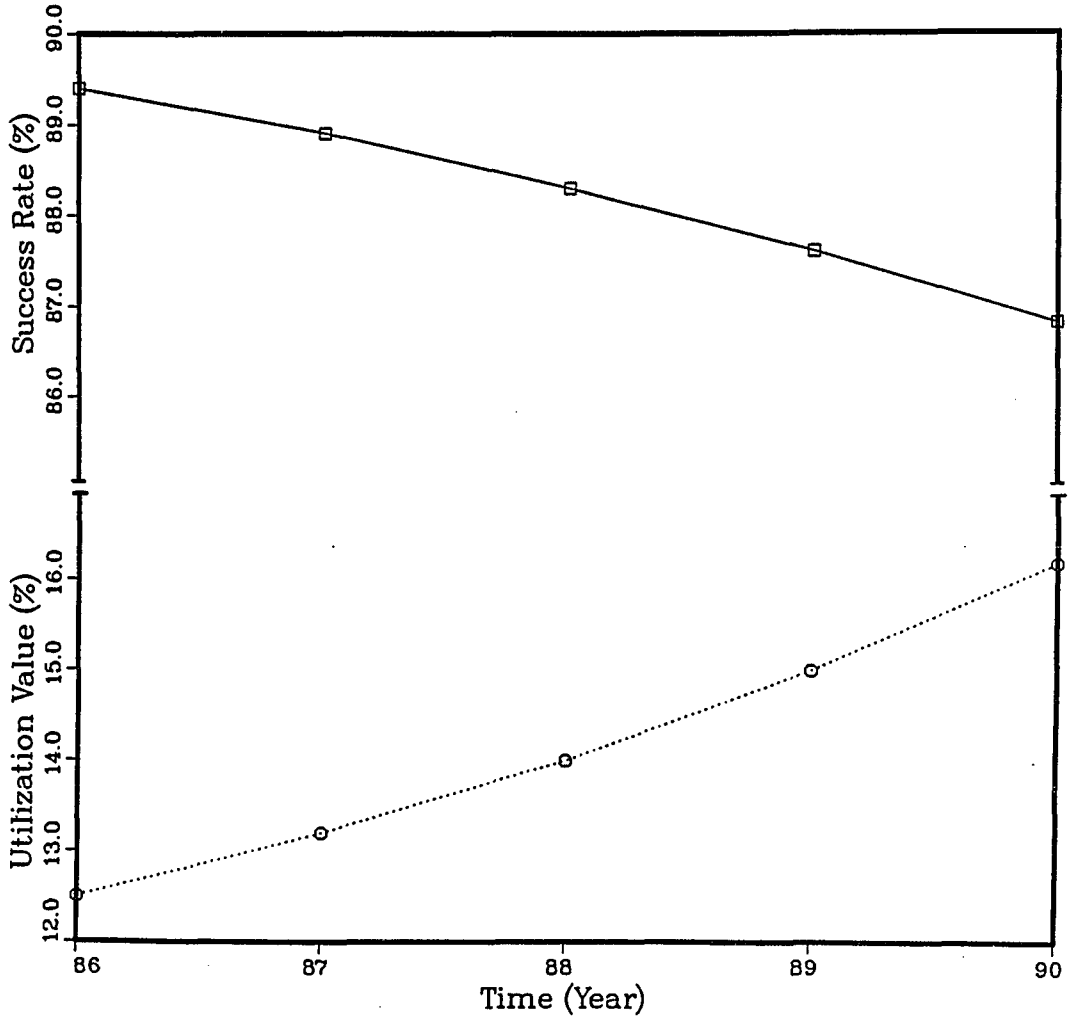



**Figure 6.1 Future Performances of the Validated Model (Current System)**

**Table 6.2 Utilization Values for the Proposed System (Validated Model)**

| Base Location | Current System | 29 out 97 in | 150 out 97 in |
|---|---|---|---|
| 23 | 0.165 | 0.178 | 0.160 |
| 29 | 0.133 | - | 0.140 |
| 69 | 0.169 | 0.186 | 0.170 |
| 97 | - | 0.126 | 0.118 |
| 132 | 0.129 | 0.118 | 0.113 |
| 150 | 0.124 | 0.117 | - |
| 259 | 0.136 | 0.135 | 0.154 |
| 298 | 0.126 | 0.123 | 0.127 |

simulation results for these two systems are presented in Tables 6.2 and 6.3, and Figure 6.2.

From the tables, the performances of the two systems were similar. Both proposed systems increased service level in the northern zones. Although not shown in the tables, the zone with the worst performance (zones with 20 or more calls) was the same in both systems, i.e., zone 353 on the far southeast. Both systems performed at less than a 90% success rate in the southeast zones. System wide, the two systems performed at approximately the same level (0.4% system success rate

**Table 6.3 Success Rates and Zone Statistics for the Proposed Systems (Validated Model)**

| | 29 out & 97 in | | 150 out & 97 in | |
|---|---|---|---|---|
| Base Location | Success Rates | Total Calls | Success Rates | Total Calls |
| System | 0.908±0.000 | 9296 | 0.912±0.000 | 9296 |
| 23 | 0.918±0.001 | 2023 | 0.918±0.001 | 1821 |
| 29 | - | - | 0.903±0.001 | 1395 |
| 69 | 0.899±0.001 | 1731 | 0.898±0.001 | 1544 |
| 97 | 0.925±0.001 | 1130 | 0.929±0.001 | 1045 |
| 132 | 0.933±0.001 | 1094 | 0.938±0.001 | 1045 |
| 150 | 0.882±0.001 | 1060 | - | - |
| 259 | 0.914±0.001 | 1195 | 0.917±0.001 | 1359 |
| 298 | 0.887±0.001 | 1060 | 0.881±0.002 | 1090 |

difference). Even with all these similarities, there was a strong preference to close the base at zone 150.

The reasons for closing the base at zone 150 were due mainly to secondary objectives such as balancing workload and balancing zone performance. When the base at zone 29 was closed, two of the ambulance utilizations went above 18% and the performance of the ambulance at zone 150 decreased to 87%. So when the base

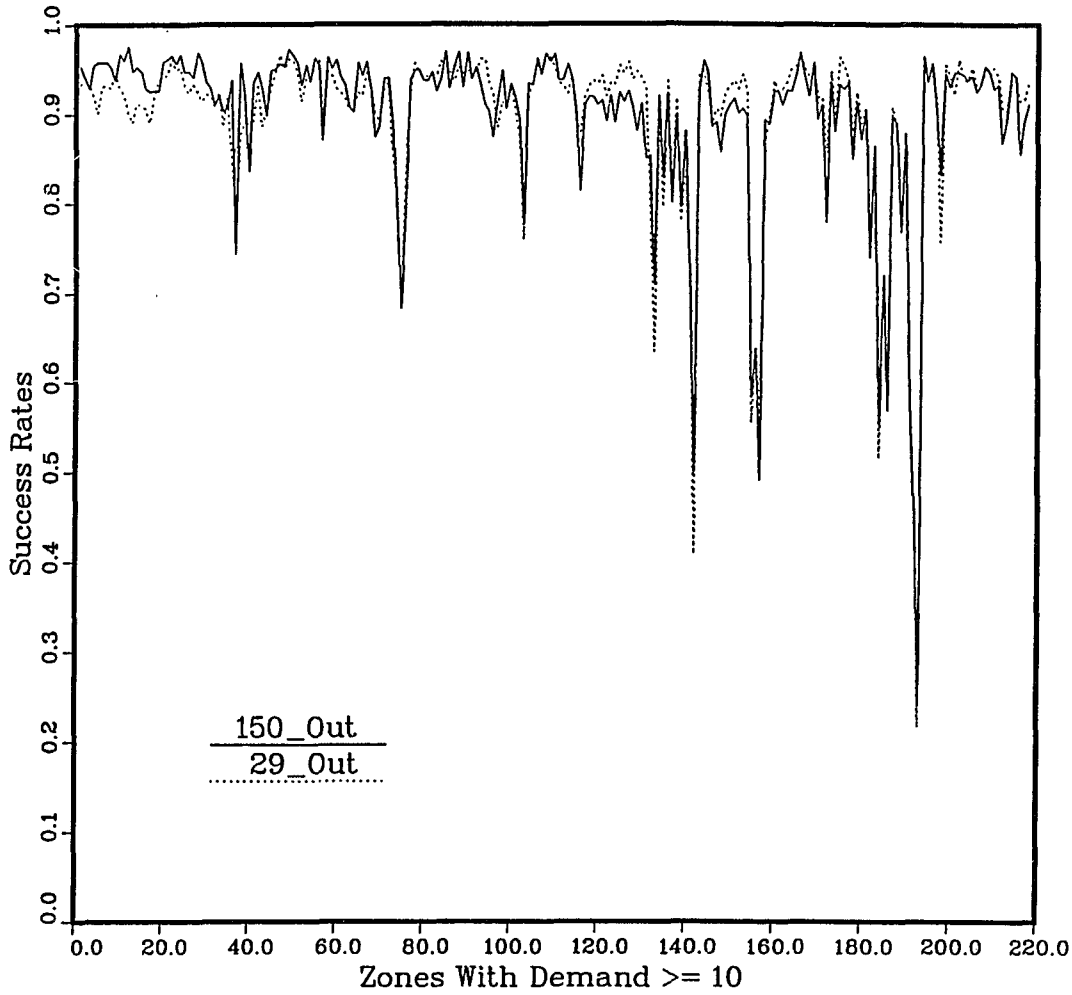



**Figure 6.2 Success Rates of the Validated Model (Proposed Systems)**

at zone 29 was closed, the base at zone 150 did not pick up all of the slack. This occurs since the base at zone 150 was rarely the primary ambulance base. These calls tend to have longer travel times since it was often used as a backup. Note that in the current system its performance is significantly lower than other bases that serve 30% and 40% more calls. When the base at zone 150 was closed, the remaining bases and the new base at zone 97 shared the resulting work load better.

## 6.3 Case Study 3

In the final study, Goldberg's interchange heuristic was used to find a better solution than the current system. A brief description of the heuristic is in Appendix C. The current system was used as a starting solution and all pairwise changes were tested. Each major iteration of the heuristic required testing 56 systems and nearly 1 hour of the UNIX PC CPU time. The process stopped after 4 major iterations with the following open bases:

23, 29, 69, 132, 205, 259, and 349.

The performance statistics of our simulation model, using the heuristic open bases above, are in Table 6.4 and the plots of zone success rates of the heuristic system versus the current system are in Figure 6.3.

System wide, the heuristic solution successfully reaches approximately 80 more calls than the current system. This may be justification enough for the change, however there is a more striking comparison. When considering all zones with 20 or more calls (approximately 85% of the demand), the worst zone in the current system is zone 353 (southeast) with a success rate of 0.240. Zone 353 is better served in the heuristic solution, since a base at zone 349 is opened. The worst zone

**Table 6.4 Performances for the Heuristic Solution (Validated Model)**

| Base Location | Utilization Values | Success Rates | Total Calls |
|---|---|---|---|
| System | - | 0.915 | 9,296 |
| 23 | 0.166 | 0.922 | 1,889 |
| 29 | 0.149 | 0.913 | 1,459 |
| 69 | 0.171 | 0.899 | 1,562 |
| 132 | 0.130 | 0.933 | 1,189 |
| 205 | 0.126 | 0.924 | 1,166 |
| 259 | 0.153 | 0.920 | 1,346 |
| 349 | 0.086 | 0.891 | 682 |

becomes zone 218 (far south) with a success rate of 0.531. Therefore, not only does the heuristic solution provide better performance system wide, it also provides a more balanced service level in each zone.

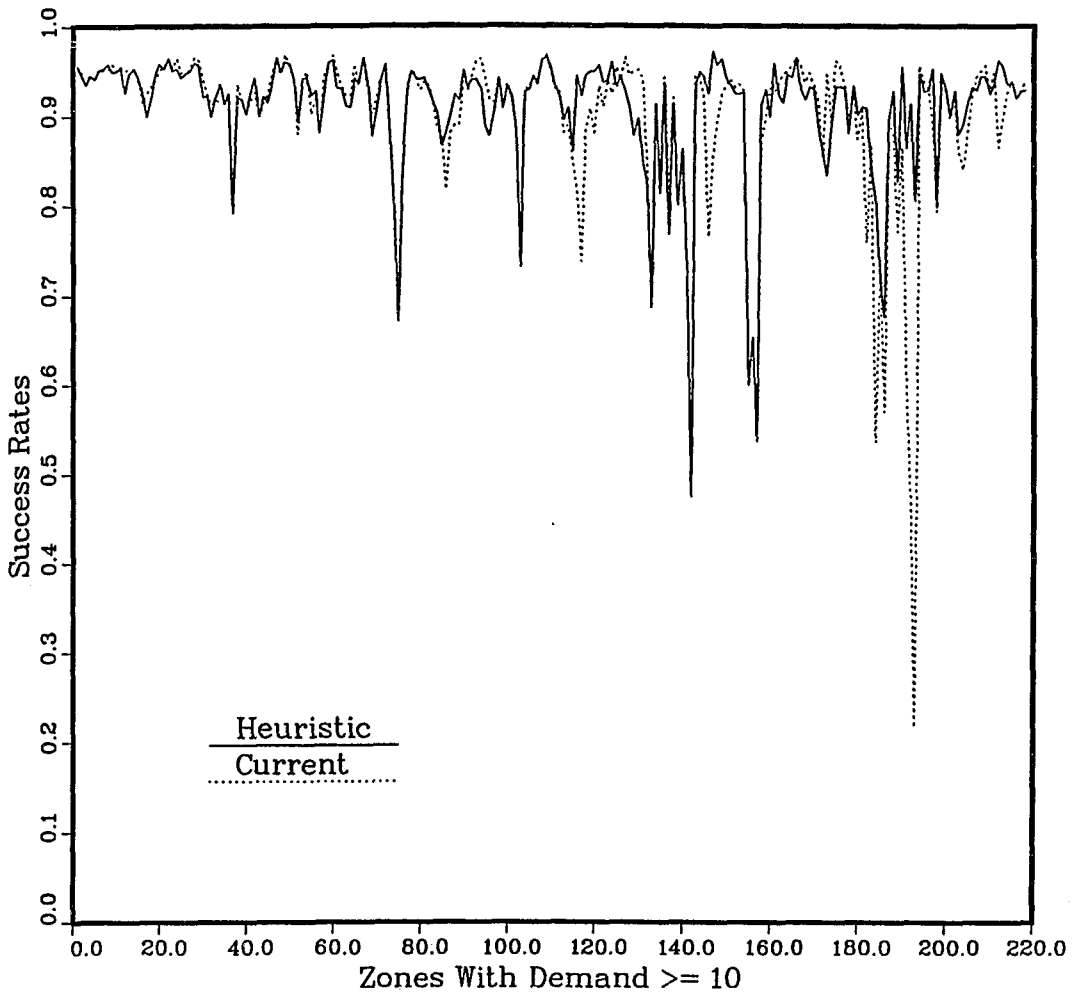



**Figure 6.3 Success Rates of the Validated Model (Heuristic Solution vs. Current System)**

# CHAPTER 7
# CONCLUSIONS

## 7.1 Conclusions

In this thesis, we presented concrete steps to develop a simulation model for evaluating locations for EMS vehicle bases. The strength of the model is its ability to consider the fact that travel time and service time vary with call location. In addition, the travel time model can also be applied to many urban facility location problems besides the EMS system, such as police car and fire engine deployments.

On the other hand, it is very costly and time-consuming to obtain geographically referenced data on call frequencies and travel times. Aside from these problems, our major difficulty was in validating the simulation model. This is because model validation is rarely documented in the large majority of literature on vehicle location problems. As a point of interest, the validation phase of this study took approximately 6 months, twice the amount of time required to develop the initial model.

## 7.2 Future Directions

From our experiences, future work considerations should include developing an efficient method to obtain geographically referenced data on call frequencies

(probably a computer-implemented data base). Better travel time models (possibly by time of day) can also be developed. Other directions to be considered are; allowing dispatching from locations other than open bases (dispatches enroute or from hospitals), developing dispatching priorities for each zone, and allowing more than one ambulance to be assigned to a call that is serviced.

## APPENDIX A

```
program ambulance(input, output);

const
    number_of_ambulances = 7;
    number_of_zones = 420;
    number_of_hospitals = 7;
    critical_time = 8.00;
    arrival_intervals = 450;
    travel_intervals = 240;
                                        { travel time residual distribution }
    service_intervals = 420;
    false_length_intervals = 30;
    false_arrival_intervals = 400;

type
    event_type = (emergency_call,
        ambulance_arrives_at_scene,
        ambulance_leaves_hospital,
        ambulance_returns_to_base);

    event_info_record = record
        time : real;                                  { time of event }
        event : event_type;                           { type of event }
        zone,                                         { zone number }
        number : integer;                             { ambulance number }
        false_alarm_time,                             { short service time }
        service_time : real;                          { full service time }
        false_alarm : boolean;          { true if the call is a false alarm }
    end;

    event_link = êvent_node;                          { event list structure }
    event_node = record
        event_info : event_info_record;
        next : event_link;
    end;
```

```
    ambulance_node = record
        busy : boolean;                                  { busy or idle }
        location,                                { current location of ambulance }
        emergency_location,                        { location of emergency call }
        hospital_location,                       { location of nearest hospital }
        base_location : 1..number_of_zones;
                                                   { ambulance base location }
        time_of_emergency,                           { time call was received }
        max_response_time,
        on_line_time,
        utilization,
        cumulative_response_time : real;               { sum of response times }
        number_of_false_alarms,
        number_of_responses,                         { total number of responses }
        number_under_critical_time : integer;
                                                  { responses under critical time }
    end;

var
    ambulance : array[1..number_of_ambulances] of ambulance_node;
    event_list,
    emergency_call_queue : event_link;
    travel_time : array[1..number_of_zones, 1..number_of_zones] of real;
    hospital : array[1..number_of_hospitals] of 1..number_of_zones;
    call_file,
    error_file : text;
    next_false_alarm_time,
    halt_time,
    queue_time,
    max_queue_time : real;

    arrival_empirical : array[1..arrival_intervals] of real;
    false_arrival_empirical : array[1..false_arrival_intervals] of real;
    travel_empirical : array[1..travel_intervals] of real;
    service_empirical : array[1..service_intervals] of real;
    false_length_empirical : array[1..false_length_intervals] of real;
    demand : array[1..number_of_zones] of real;

    queue_entries,
    calls : integer;
    historical : boolean;                          { use historical data or not }
    zone_data : array[1..number_of_zones, 1..2] of integer;
    Z : integer;                                { seed of random number generator }
```

```
    done : boolean;
    clock,
    half_length,
    max_half,
    cut_off : real;
    runs,
    batch_size,
    min_calls : integer;
    t : array[0..32] of real;

    total_zone_stats : array[1..number_of_zones, 1..3] of real;
                                                { sum of x, sum of x squared, n }
    total_am_stats : array[1..number_of_ambulances + 1, 1..2]
            of ambulance_node;                  { sum of x, sum x squared }
    amb_zone_stat,
    gamb_zone_stat : array[1..number_of_zones,
                    1..number_of_ambulances, 1..3] of integer;
              { stats for following calls picked up by each ambu in each zone }
    b0, b1, b2, b3, b4 : real;                  { regression coefficients }

function rand(var IX : integer) : real;

const
    A = 16807;
    B15 = 32768;
    B16 = 65536;
    P = 2147483647;

var
    XHI, XALO,
    LEFTLO, FHI, K : integer;

begin
    XHI := (IX div B16);
    XALO := (IX - XHI * B16) * A;
    LEFTLO := (XALO div B16);
    FHI := XHI * A + LEFTLO;
    K := (FHI div B15);
    IX := (((XALO - LEFTLO * B16) - P) + (FHI - K * B15) * B16) + K;
    while (IX < 0) do
        IX := IX + P;
    IX := A * IX mod P;
    rand := IX * 4.656612875E-10
```

```
end;

function expon(mean : integer) : real;

var
    U : real;

begin
    U := rand(Z);
    expon := -mean * Ln(U);
end;

function unif(a, b : real) : real;

begin
    unif := a + (b - a) * rand(Z);
end;

function U_normal(var IX : integer) : real;

    { Generate unit normal deviate by composition method of ahrens and }
    { dieter. The area under normal curve is divided into 5 different }
    { areas. }
    { IX : random number seed. }
    { non_negative : the generated normal distribution must be }
    { non_negative, }
    { i.e. the unit_normal > -mean / standard deviate. }

const
    non_negative = -3;

label
    160, 210, 260, 310, 340, 500;

var
    U, UO, trptmp, unit : real;

begin
    repeat
        U := rand(ix);
        UO := rand(ix);

        if (U >= 0.919544) then goto 160
        else Unit := 2.40376 * (UO + U * 0.825339) - 2.11403;
```

```
            goto 500;

            160: if (U < 0.965487) then
                 goto 210
            else
                 repeat
                      trptmp := sqrt(4.46911 − 2 * ln(rand(ix)))
                 until
                      (trptmp * rand(ix) <= 2.11403);
            goto 340;

            210: if (U < 0.949991) then
                 goto 260
            else
                 repeat
                      trptmp := 1.8404 + rand(ix) * 0.273629
                 until
                      (0.398942 * exp(− trptmp * trptmp/2) −0.443299
                      + trptmp * 0.209694 >= rand(ix) * 4.27026E-02);
            goto 340;

            260: if (U < 0.925852) then
                 goto 310
            else
                 repeat
                      trptmp := 0.28973 + rand(ix) * 1.55067
                 until
                      (0.398942 * exp(−trptmp * trptmp/2) −0.443299
                      + trptmp * 0.209694 >= rand(ix) * 1.59745E-02);
            goto 340;

            310: repeat
                 trptmp := rand(ix) * 0.28973
            until (0.398942 * exp(trptmp * trptmp/2) −0.382545
                 >= rand(ix) * 1.63977E-02);

            340: if (UO > 0.5) then
                 Unit := trptmp
            else
                 Unit := −trptmp;

       500: until (Unit > non_negative);
       U_normal := unit;
end;
```

```
function normal(mean, sigma : real) : real;

begin
    normal := mean + U_normal(Z) * sigma;
end;

function pois(lambda : integer) : integer;

var
    A, B : real;
    i : integer;

begin
    A := exp(-lambda);
    B := rand(Z);
    i := 0;
    while (B >= A) do
    begin
        B := B * rand(Z);
        i := i + 1;
    end;
    pois := i;
end;

procedure enter_list(var list : event_link; info : event_info_record);

    { This procedure adds the info node to the correct place according }
    { to time in the list passed in. }

var
    window, node : event_link;
    found : boolean;

begin
    if list = nil then
    begin
        new(list);
        list^.event_info := info;
    end
    else if info.time <= list^.event_info.time then
    begin
        new(node);
        node^.event_info := info;
        node^.next := list;
```

```
            list := node;
        end
        else if list^.next = nil then
        begin
            new(node);
            node^.event_info := info;
            node^.next := nil;
            list^.next := node;
        end
        else
        begin
            window := list;
            found := false;
            while not found do
                if window^.next = nil then
                    found := true
                else if info.time <= window^.next^.event_info.time then
                    found := true
                else
                    window := window^.next;

        new(node);
        node^.event_info := info;
        node^.next := window^.next;
        window^.next := node;
        end;

end;                                                    { procedure enter_list }

procedure leave_list(var list:event_link; var info:event_info_record);

        { This procedure pops the list passed in. }

var
        node : event_link;

begin
        node := list;
        list := list^.next;
        info := node^.event_info;
        dispose(node);

end;                                                    { procedure leave_list }
```

```
function random_travel_error : real;

var
      i : integer;
      x : real;
      found : boolean;

begin
      x := rand(Z);
      i := 1;
      found := false;
      while (i <= travel_intervals) and not found do
            if x <= travel_empirical[i] then
                  found := true
            else
                  i := i + 1;

      if i = 1 then
            random_travel_error := -10.0
      else if i >= travel_intervals) and not found do
            random_travel_error := 50.0
      else
            random_travel_error := -10.0 + 0.25 * ((i-1) +
            ((x-travel_empirical [i-1]) / (travel_empirical [i] -
            travel_empirical [i-1])));
                              { random_travel_error := normal(-0.116, 2.418) }

end;                                          { function random_travel_error }

function random_interarrival_time : real;

var
      i : integer;
      x : real;
      found : boolean;

begin
      x := rand(Z);
      i := 1;
      found := false;
      while (i <= arrival_intervals) and not found do
            if x <= arrival_empirical [i] then
                  found := true
```

```
            else
                i := i + 1;
        random_interarrival_time := i;
                                    { random_interarrival_time := expon(28.02) }
                                  { combined_interarrival_time := expon(24.49) }
end;                                          { function random_interarrival_time }

function random_zone : integer;

var
        i : integer;
        x : real;
        found : boolean;

begin
        x := rand(Z);
        i := 1;
        found := false;
        while (i <= number_of_zones) and not found do
            if x <= demand [i] then
                found := true
            else
                i := i + 1;
        random_zone := i;
end;                                                  { function random_zone }

function random_service_length(i : integer) : real;

begin

        random_service_length:= service_empirical [i];

end;                                          { function random_service_length }

function random_false_interarrival_time : real;

var
        i : integer;
        x : real;
        found : boolean;

begin
        x := rand(Z);
```

```
        i := 1;
        found := false;
        while (i <= false_arrival_intervals) and not found do
            if x <= false_arrival_empirical [i] then
                found := true
            else
                i := i + 1;
        random_false_interarrival_time := 5 * i;
                                { random_false_interarrival_time:= expon(194.36) }
end;                                        { function random_false_interarrival_time }

function random_false_alarm_length : real;

var
        i : integer;
        x : real;
        found : boolean;

begin

        x := rand(Z);
        i := 1;
        found := false;
        while (i <= false_length_intervals) and not found do
            if x <= false_length_empirical [i] then
                found := true
            else
                i := i + 1;
        random_false_alarm_length := i;
                            { random_false_alarm_length := lognormal(4.82,2.29) }
end;                                        { function random_false_alarm_length }

function time_transform(date, time : integer) : real;

var
        hours_elasped : integer;
                                        { hours elasped before the current date }
begin
        date := date div 100;                                   { erase year }
        case date div 100 of
            0 : hours_elasped := 0;
```

```
            1 : hours_elasped := 0;                          { months before }
            2 : hours_elasped := 31 * 24;
            3 : hours_elasped := 59 * 24;
            4 : hours_elasped := 90 * 24;
            5 : hours_elasped := 120 * 24;
            6 : hours_elasped := 151 * 24;
            7 : hours_elasped := 181 * 24
        end;                                                 { case }
        hours_elasped := hours_elasped + 24
            * ((date mod 100) - 1);                          { days before }
        time_transform := (time mod 100) + 60
            * ((time div 100) + hours_elasped);
        if time_transform <= 0 then
            time_transform := 0;
end;                                          { function time_transform }

function generate_next_emergency_call
            (var info : event_info_record) : boolean;

        { This function uses either historical data or input strings }
        { for generating arrivals. }

var
        line : varying [35] of char;
        base, z, d1, t1, d2, t2, d3, t3 : integer;

begin
        if historical then
        begin
            if not eof(call_file) then
            begin
                readln(call_file, line, base, t1, d1, t2, d2, t3, d3, z);
                while z = 0 do
                    readln(call_file, line, base, t1, d1, t2, d2, t3, d3, z);

                with info do
                begin
                    time := time_transform(d1,t1);
                    zone := z;
                    if d2 = 0 then
                    begin
                        false_alarm := true;
                        false_alarm_time := time_transform(d3,t3)
```

```
                                    -time_transform(d1,t1);
                        end
                        else
                        begin
                            false_alarm := false;
                            service_time := time_transform(d3,t3)
                                -time_transform(d2,t2);
                        end;
                        event := emergency_call;
                    end;
                    generate_next_emergency_call := true;
                end
                else
                begin
                    close(call_file);
                    generate_next_emergency_call := false;
                end
            end
            else
            begin
                with info do
                begin
                    time := time + random_interarrival_time;
                    zone := random_zone;
                    if time > next_false_alarm_time then
                    begin
                        false_alarm := true;
                        false_alarm_time := random_false_alarm_length;
                        next_false_alarm_time := next_false_alarm_time
                            + random_false_interarrival_time;
                    end
                    else false_alarm := false;
                    service_time := random_service_length(zone);
                    event := emergency_call;
                end;
                generate_next_emergency_call := true;
            end;
        end;                                { function generate_next_emergency_call }

function assign_ambulance_to_call(var number : integer;
                                  zone : integer) : boolean;
```

```
      { This function assigns the nearest idle ambulance to the call }
      { at the zone and returns its number in the variable. If all }
      { ambulances are busy, the function returns false. }

var
      i : integer;
      minimum : real;

begin
      minimum := maxint;
      for i := 1 to number_of_ambulances do
            if not ambulance [i].busy then
                  if travel_time[ambulance [i].location,zone] < minimum then
                  begin
                        minimum := travel_time[ambulance [i].location,zone];
                        number := i;
                  end;
      assign_ambulance_to_call := minimum <> maxint;
end;                                          { function assign_ambulance_to_call }

function assign_hospital(zone : integer) : integer;

      { This function returns the zone number of the closest hospital.}

var
      i, closest : integer;
      minimum : real;

begin
      minimum := maxint;
      for i := 1 to number_of_hospitals do
            if travel_time[zone, hospital [i]] < minimum then
            begin
                  closest := hospital [i];
                  minimum := travel_time[zone, hospital [i]];
            end;
      assign_hospital := closest;
end;                                                { function assign_hospital }

procedure execute_event(event_info : event_info_record);

var
      t_time, error, current_time : real;
      am_number : integer;
```

```
begin
    case event_info.event of
    emergency_call :
    begin
        calls := calls + 1;
        current_time := event_info.time;
        if assign_ambulance_to_call(event_info.number,event_info.zone) then
        begin
            with event_info do
            begin
                current_time := event_info.time;
                ambulance [number].busy := true;
                ambulance [number].on_line_time := time;
                if false_alarm then
                begin
                    ambulance [number].number_of_false_alarms :=
                    ambulance [number].number_of_false_alarms + 1;
                    time := time + false_alarm_time;

                                        { * added by J. Goldberg 4/8/88 * }
                    amb_zone_stat [zone, number, 1] :=
                        amb_zone_stat [zone, number, 1] + 1 ;
                    amb_zone_stat [zone, number, 3] :=
                        amb_zone_stat [zone, number, 3] + 1 ;
                                        { * end of lines added on 4/8/88 * }

                    event := ambulance_returns_to_base;
                end
                else
                begin
                    ambulance [number].emergency_location := zone;
                    ambulance [number].time_of_emergency := time;

                                        { * added by J. Goldberg 4/8/88 * }
                    amb_zone_stat [zone, number, 1] :=
                        amb_zone_stat [zone, number, 1] + 1 ;
                                        { * end of lines added on 4/8/88 * }

                    t_time := travel_time[ambulance
                        [number].location, ambulance
                        [number].emergency_location];

                    error := random_travel_error;
                    while t_time + error <= 0 do
```

```
                        error := random_travel_error;
                    time := time + t_time + error;
                    event := ambulance_arrives_at_scene;
                end;
            end;

            enter_list(event_list, event_info);
        end
        else
        begin
            enter_list(emergency_call_queue, event_info);
            queue_entries := queue_entries + 1;
        end;

        event_info.time := current_time;
        if generate_next_emergency_call(event_info) then
            enter_list(event_list, event_info);
end;

ambulance_arrives_at_scene :
begin
        with event_info do
        begin
            ambulance [number].location :=
                ambulance [number].emergency_location;
                                        { calculate statistics for system }
            t_time:= time - ambulance [number].time_of_emergency;

            if ambulance [number].max_response_time < t_time then
                ambulance [number].max_response_time := t_time;

            ambulance [number].cumulative_response_time :=
                ambulance [number].cumulative_response_time
                + t_time;
            ambulance [number].number_of_responses :=
                ambulance [number].number_of_responses + 1;

            if t_time <= critical_time then
            begin
                ambulance [number].number_under_critical_time :=
                    ambulance [number].number_under_critical_time
                    + 1;

                zone_data [zone, 1]:= zone_data [zone, 1] + 1;
```

```
                                        { * added by J. Goldberg on 4/8/88 * }
                    amb_zone_stat [zone, number, 2] :=
                        amb_zone_stat [zone, number, 2] + 1 ;
                                            { * end of lines added on 4/8/88 * }

            end;

            zone_data [zone, 2] := zone_data [zone, 2] + 1;
            time := time + service_time;

            event := ambulance_returns_to_base;
        end;
        enter_list(event_list, event_info);
end;

ambulance_leaves_hospital :
begin
        with event_info do
        begin
            ambulance [number].location :=
                ambulance [number].hospital_location;

            t_time := travel_time [ambulance [number].location,
                ambulance [number].base_location];

            error := random_travel_error;
            while t_time + error <= 0 do
                error := random_travel_error;
            time := time + t_time + error;
            event := ambulance_returns_to_base;
        end;

        enter_list(event_list, event_info);
end;

ambulance_returns_to_base :
begin
        with event_info do
            ambulance [number].utilization :=
                ambulance [number].utilization +
                time - ambulance [number].on_line_time;

        ambulance [event_info.number].location :=
            ambulance [event_info.number].base_location;

        if emergency_call_queue <> nil then
```

```
begin
    am_number := event_info.number;
    current_time := event_info.time;
    leave_list(emergency_call_queue, event_info);
    with event_info do
    begin
        number := am_number;
        if current_time − time > 0.0 then
        begin
            if current_time − time > max_queue_time then
                max_queue_time := current_time − time;
            queue_time := queue_time + current_time − time;
        end
        else queue_entries := queue_entries − 1;
                                                { no actual wait }

        if false_alarm then
        begin
            ambulance [number].number_of_false_alarms :=
                ambulance [number].number_of_false_alarms
                + 1;

                        { * lines added by J. Goldberg 4/8/88 * }
            amb_zone_stat [zone, number, 1] :=
                amb_zone_stat [zone, number, 1] + 1 ;
            amb_zone_stat [zone, number, 3] :=
                amb_zone_stat [zone, number, 3] + 1 ;
                            { * end of lines added on 4/8/88 * }

            time := current_time + false_alarm_time;
            event := ambulance_returns_to_base;
        end
        else
        begin
            ambulance [number].emergency_location := zone;
            ambulance [number].time_of_emergency := time;

                        { * lines added by J. Goldberg 4/8/88 * }
            amb_zone_stat [zone, number, 1] :=
                amb_zone_stat [zone, number, 1] + 1 ;
                            { * end of lines added 4/8/88 * }

            time := current_time;
            t_time := travel_time[ambulance [number].location,
```

```
                                    ambulance [number].emergency_location];
                                error := random_travel_error;
                                while t_time + error <= 0 do
                                    error := random_travel_error;

                                time := time + t_time + error;
                                event := ambulance_arrives_at_scene;
                            end;
                        end;

                        enter_list(event_list, event_info);
                    end
                    else
                        ambulance [event_info.number].busy := false;

            end;
            end;  case

end;                                                    { procedure execute_event }

function t_stat(n : integer) : real;

begin
    if n >= 32 then t_stat := t [31]
    else t_stat := t [n-1];

end;                                                    { function t_stat }

function sq_root(n : real) : real;

begin
    if n < 0 then writeln(error_file, n:0:15);
    sq_root := sqrt(abs(n));

end;                                                    { function sq_root }

procedure report;

var
    result_file : text;
    result_name : packed array[1..40] of char;
    i, j, jmin : integer;
    number, under, number_fa : integer;
    time, s, m, n, total_u, temp : real;
```

```
    zone_sort1 : array[1..number_of_zones, 1..4] of real;
    zone_sort2 : array[1..number_of_zones, 1..5] of real;

begin
    writeln; writeln;
    write('Result File Name (<CR> for default) >> ');
    readln(result_name);
    writeln; writeln;
    if result_name = '' then
    begin
        writeln('Results will be written in file RESULTS.DAT');
        result_name := 'results.dat';
    end;

    open(file_variable := result_file, file_name := result_name,
        history := new, access_method := sequential);
    rewrite(result_file);

    writeln(result_file); writeln(result_file);

    writeln(result_file,'                                BATCH RESULTS');
    writeln(result_file,'                               Number of Runs = ',
        runs:0);
    writeln(result_file);
    writeln(result_file);

    for i := 1 to number_of_ambulances do
    begin
        writeln(result_file);
        writeln(result_file,'       Ambulance ',i:1,
            ' at Zone ',ambulance [i].base_location : 1);
        if total_am_stats [i,1].number_of_responses = 0 then
            writeln(result_file,'       No responses')
        else
        begin
            s := total_am_stats [i,2].number_of_responses/runs -
                (total_am_stats [i,1].number_of_responses/runs) ** 2;

            writeln(result_file,'       Number of Responses = ',
                total_am_stats [i,1].number_of_responses/runs :0:3,'
                +/- ', t_stat(runs) * sq_root(s/runs) :0:3);

            s := total_am_stats [i,2].max_response_time/runs -
                (total_am_stats [i,1].max_response_time/runs) ** 2;
```

```
            writeln(result_file,'       Maximum Response Time = ',
                total_am_stats [i,1].max_response_time/runs:0:3,' +/- ',
                t_stat(runs) * sq_root(s/runs):0:3);

            s := total_am_stats [i,2].cumulative_response_time/runs -
                (total_am_stats [i,1].cumulative_response_time/runs)
                ** 2;

            writeln(result_file,'       Average Response Time = ',
            total_am_stats [i,1].cumulative_response_time/runs:0:3,'
            +/- ',t_stat(runs) * sq_root(s/runs):0:3);

            s := total_am_stats [i,2].on_line_time/runs -
                (total_am_stats [i,1].on_line_time/runs) ** 2;

            writeln(result_file,'       Percentage of Responses Under ',
                'Critical Time = ',
                total_am_stats [i,1].on_line_time/runs:0:3,'
                +/- ',t_stat(runs) * sq_root(s/runs):0:3);

            s := total_am_stats [i,2].number_of_false_alarms/runs -
                (total_am_stats [i,1].number_of_false_alarms/runs) ** 2;

            writeln(result_file,'       Number of False Alarms = ',
                total_am_stats [i,1].number_of_false_alarms/runs:0:3,'
                +/- ',t_stat(runs) * sq_root(s/runs):0:3);

            s := total_am_stats [i,2].utilization/runs -
                (total_am_stats [i,1].utilization/runs) ** 2;

            writeln(result_file,'       Utilization = ',
                total_am_stats [i,1].utilization/runs:0:3,'
                +/- ',t_stat(runs) * sq_root(s/runs):0:3);

        end;
    end;

    writeln(result_file); writeln(result_file);
    number := number_of_ambulances + 1;
    s := total_am_stats [number, 2].number_of_responses/runs -
        (total_am_stats [number, 1].number_of_responses/runs) ** 2;

    writeln(result_file,'   AVERAGE TOTAL NUMBER OF RESPONSES = ',
        total_am_stats [number, 1].number_of_responses/runs:0:3,'
        +/- ',t_stat(runs) * sq_root(s/runs):0:3);
```

```
s := total_am_stats [number, 2].cumulative_response_time/runs -
     (total_am_stats [number, 1].cumulative_response_time/runs) ** 2;
writeln(result_file,'      AVERAGE OVERALL RESPONSE TIME = ',
     total_am_stats [number, 1].cumulative_response_time/runs:0:3,'
     +/- ',t_stat(runs) * sq_root(s/runs):0:3);

s := total_am_stats [number, 2].on_line_time/runs -
     (total_am_stats [number, 1].on_line_time/runs) ** 2;
writeln(result_file,'      AVERAGE OVERALL PERCENTAGE UNDER
     CRITICAL TIME = ',
     total_am_stats [number, 1].on_line_time/runs:0:3,'
     +/- ',t_stat(runs) * sq_root(s/runs):0:3);

s := total_am_stats [number, 2].number_of_false_alarms/runs -
     (total_am_stats [number, 1].number_of_false_alarms/runs) ** 2;
writeln(result_file,'      AVERAGE TOTAL NUMBER OF
     FALSE ALARMS = ',
     total_am_stats [number, 1].number_of_false_alarms/runs:0:3,'
     +/- ',t_stat(runs) * sq_root(s/runs):0:3);

s := total_am_stats [number, 2].utilization/runs -
     (total_am_stats [number, 1].utilization/runs) ** 2;

writeln(result_file,'      AVERAGE OVERALL UTILIZATION = ',
     total_am_stats [number, 1].utilization/runs:0:3,'
     +/- ',t_stat(runs) * sq_root(s/runs):0:3);

for i := 1 to number_of_zones do
begin
     zone_sort1 [i,1] := total_zone_stats [i,1];
     zone_sort1 [i,2] := total_zone_stats [i,2];
     zone_sort1 [i,3] := total_zone_stats [i,3];
     zone_sort1 [i,4] := i;
end;

for i := 1 to number_of_zones do
begin
     jmin := i;
     for j := i + 1 to number_of_zones do
          if zone_sort1 [j,1] > zone_sort1 [jmin,1] then jmin := j;

     temp := zone_sort1 [jmin, 1];
     zone_sort1 [jmin, 1] := zone_sort1 [i,1];
     zone_sort1 [i,1]:= temp;
```

```
        temp := zone_sort1 [jmin, 2];
        zone_sort1 [jmin, 2] := zone_sort1 [i,2];
        zone_sort1 [i,2] := temp;

        temp := zone_sort1 [jmin, 3];
        zone_sort1 [jmin, 3] := zone_sort1 [i,3];
        zone_sort1 [i,3] := temp;

        temp := zone_sort1 [jmin, 4];
        zone_sort1 [jmin, 4] := zone_sort1 [i,4];
        zone_sort1 [i,4] := temp;
end;

for i := 1 to number_of_zones do
begin
        zone_sort2 [i,1] := total_zone_stats [i,1];
        zone_sort2 [i,2] := total_zone_stats [i,2];
        zone_sort2 [i,3] := total_zone_stats [i,3];
        zone_sort2 [i,4] := i;
        zone_sort2 [i,5] := zone_data [i,2];
end;

for i := 1 to number_of_zones do
begin
        jmin := i;
        for j := i + 1 to number_of_zones do
                if zone_sort2 [j,5] > zone_sort2 [jmin,5] then jmin := j;

        temp := zone_sort2 [jmin, 1];
        zone_sort2 [jmin, 1] := zone_sort2 [i,1];
        zone_sort2 [i,1] := temp;

        temp := zone_sort2 [jmin, 2];
        zone_sort2 [jmin, 2] := zone_sort2 [i,2];
        zone_sort2 [i,2] := temp;

        temp := zone_sort2 [jmin, 3];
        zone_sort2 [jmin, 3] := zone_sort2 [i,3];
        zone_sort2 [i,3] := temp;

        temp := zone_sort2 [jmin, 4];
        zone_sort2 [jmin, 4] := zone_sort2 [i,4];
        zone_sort2 [i,4] := temp;

        temp := zone_sort2 [jmin, 5];
```

```
        zone_sort2 [jmin, 5] := zone_sort2 [i,5];
        zone_sort2 [i,5] := temp;
end;

writeln(result_file); writeln(result_file); writeln(result_file);
writeln(result_file,'                                                ',
        'PERCENTAGES UNDER CRITICAL TIME');
writeln(result_file,'                                                ',
              '  ','According to Zone');
writeln(result_file,'                                                ',
              '  ','Cut-off at ',cut_off:0:3);

writeln(result_file); writeln(result_file);

write(result_file,'          Sorted by Zone                ');
write(result_file,'       Sorted by Performance');
writeln(result_file,'                  Sorted by Demand');
writeln(result_file);

for i := 1 to number_of_zones do
begin
        if total_zone_stats [i,3] > 0 then
        begin
                n := total_zone_stats [i,3];
                m := total_zone_stats [i,1] / n;
                s := total_zone_stats[i,2] / n - m ** 2;
                write(result_file,'    Zone ',i:3,' = ',m:0:3,' +/- ',
                        t_stat(round(n)) * sq_root(s/n):0:3,' ');
        end
        else write(result_file,'    Zone ',i:3,' was under the limit. ');
        write(result_file,'              ');

        if zone_sort1 [i,3] > 0 then
        begin
                n := zone_sort1 [i,3];
                m := zone_sort1 [i,1] / n;
                s := zone_sort1 [i,2] / n - m ** 2;
                write(result_file,'Zone ',round(zone_sort1 [i,4]):3,' = ',
                        m:0:3,' +/- ',t_stat(round(n)) * sq_root(s/n):0:3);

        end
        else write(result_file,'Zone ',round(zone_sort1 [i,4]):3,
                ' was under limit. ');
        write(result_file,'              ');
```

```
            if zone_sort2 [i,3] > 0 then
            begin
                  n := zone_sort2 [i,3];
                  m := zone_sort2 [i,1] / n;
                  s := zone_sort2 [i,2] / n - m ** 2;
                  write(result_file,'Zone ',round(zone_sort2 [i,4]):3,' = ',
                        m:0:3,' +/- ',t_stat(round(n)) * sq_root(s/n):0:3);

                  if m < cut_off then write(result_file,'  ***');
            end
            else write(result_file,'Zone ',round(zone_sort2 [i,4]):3,
                  ' was under limit.');
            writeln(result_file);
      end;

      writeln(result_file);
      writeln(result_file);
      for i :=1 to number_of_zones do
            for j := 1 to number_of_ambulances do
            begin
                  gamb_zone_stat [i,j,1] := (gamb_zone_stat [i,j,1] div runs) ;
                  gamb_zone_stat [i,j,2] := (gamb_zone_stat [i,j,2] div runs) ;
                  gamb_zone_stat [i,j,3] := (gamb_zone_stat [i,j,3] div runs) ;
            end;
      for i :=1 to number_of_zones do
      begin
      write(result_file,i:4,'  ':2) ;
      for j := 1 to number_of_ambulances do
            write(result_file, gamb_zone_stat [i, j, 1]:3,' ':1,
            gamb_zone_stat [i,j,2]:3,' ':1,gamb_zone_stat [i,j,3]:3,' : ':3);
      writeln(result_file);
      end;

      close(result_file);
end;                                                    { procedure report }

procedure initialize;

var
      i, j, k, l : integer;
      infile : text;
      time, t1, t2, t3, t4 : real;
      event_info : event_info_record;
```

```
    ch : char;

begin
    for i := 1 to number_of_zones do
    begin
        zone_data [i,1]:= 0; zone_data [i,2]:= 0;
    end;

    writeln;
    writeln('Initializing Ambulances');
    open(file_variable := infile, file_name := 'base.dat',
        history := old, access_method := sequential);
    reset(infile);

                                    { * lines added by J. Goldberg 4/8/88 * }
    for i := 1 to number_of_zones do
        for j :=1 to number_of_ambulances do
        begin
            amb_zone_stat [i, j, 1] := 0 ;
            amb_zone_stat [i, j, 2] := 0 ;
            amb_zone_stat [i, j, 3] := 0 ;
        end ;
                                        { * end of lines added on 4/8/88 * }

    for i := 1 to number_of_ambulances do
        with ambulance [i] do
        begin
            readln(infile, base_location);
            location := base_location;          { ambulance starts at base }
            busy := false;                    { ambulance starts not busy }
            cumulative_response_time := 0;          { ambulance statistics }
            max_response_time := - 1;
            utilization := 0;
            number_of_responses := 0;
            number_of_false_alarms := 0;
            number_under_critical_time := 0;
        end;
    close(infile);
    calls := 0;
    queue_time := 0;
    queue_entries := 0;
    max_queue_time := - 1;

    emergency_call_queue := nil;
```

```
        event_list := nil;

        if historical then
        begin
            open(file_variable := call_file, file_name := '[emsproj]ems3.sort',
                history := old, access_method := sequential);
            reset(call_file);

            if generate_next_emergency_call(event_info) then
                enter_list(event_list,event_info)
        end
        else
        begin
            next_false_alarm_time := random_false_interarrival_time;
            if generate_next_emergency_call(event_info) then
                enter_list(event_list, event_info);
        end;

        writeln;
        writeln('Starting Simulation -- Run ',runs:0,' of ', batch_size:0);
        if max_half <> - 1 then
            writeln('Maximum Half - Length = ', max_half:0:3);
        writeln;

end;                                                    { procedure initialize }

procedure run_simulation(var clock : real);

var
        count : integer;
        event_info : event_info_record;

begin
        clock := 0; count := 1;
        initialize;

        while (clock <= halt_time) and (event_list <> nil) do
        begin
            leave_list(event_list, event_info);
            clock := event_info.time;
            if round(clock) div round(halt_time/25) >= count then
            begin
                writeln('Time in Simulation is ', clock:10:3,
                    '   (',count * 4:0,'%)');
```

```
                count := count + 1;
            end;
            execute_event(event_info);
        end;

end;                                        { procedure run_simulation }

procedure read_termination_conditions;

var
    i, j, k, l : integer;
    infile : text;
    time, t1, t2, t3, t4 : real;
    event_info : event_info_record;
    ch : char;

begin
    writeln; write('Historical Data (Y/N) ? >> ');
    readln(ch); writeln;
    if (ch = 'Y') or (ch = 'y') then
        historical := true
    else historical := false;

    write('Random number seed (0 for default) >> ');
    readln(Z); writeln;
    if Z = 0 then Z := 12345;
    write('What halt time ? >> ');
    readln(halt_time); writeln;
    write('What cut-off percentage ? >> ');
    readln(cut_off); writeln;
    writeln; writeln; writeln('TERMINATION CONDITIONS'); writeln;
    write('What is the maximum number of runs ? >> ');
    readln(batch_size);
    writeln;
    write('What is the number of calls needed',
        'for examination ? >> ');
    readln(min_calls); writeln;
    write('What is the half-length size ? >> ');
    readln(half_length);
    writeln; writeln; writeln; writeln;

    writeln; writeln('Reading t Table');
    open(file_variable := infile, file_name := 't.dat',
        history := old, access_method := sequential);
```

```
reset(infile);

for i := 1 to 31 do readln(infile, t[i]);
close(infile);

                                { * Read lines added on 3/17/88 by Mousa * }
open(file_variable := infile, file_name := 'travc.dat',
     history := old, access_method := sequential);
reset(infile);
readln(infile, b0, b1, b2, b3, b4);
writeln(output,'Reading travel time coefficients...');
close(infile);
                                                { * end reading extra data * }

                                        { * lines added by J. Goldberg 4/8/88 * }
for i := 1 to number_of_zones do
     for j :=1 to number_of_ambulances do
     begin
          gamb_zone_stat [i, j, 1] := 0 ;
          gamb_zone_stat [i, j, 2] := 0 ;
          gamb_zone_stat [i, j, 3] := 0 ;
     end ;
                                                { * end of lines added on 4/8/88 * }

for i := 1 to number_of_ambulances + 1 do
     for j := 1 to 2 do
          with total_am_stats[i,j] do
          begin
               on_line_time := 0;
               cumulative_response_time := 0;
               max_response_time := 0;
               utilization := 0;
               number_of_responses := 0;
               number_of_false_alarms := 0;
               number_under_critical_time := 0;
          end;

for i := 1 to number_of_zones do
     for j := 1 to 3 do total_zone_stats [i,j] := 0;

writeln('Reading Hospital Bases');
open(file_variable := infile, file_name := '[.prb]hospital.dat',
     history := old, access_method := sequential);
reset(infile);
```

```
for i := 1 to number_of_hospitals do readln(infile, hospital [i]);
close(infile);

writeln('Reading Travel Times');
for i := 1 to number_of_zones
      do for j := 1 to number_of_zones do
      travel_time [i,j] := 12.000;

open(file_variable := infile,
      file_name := '[emsproj.working]amroutes3.dat',
      history := old, access_method := sequential);
reset(infile);

while not eof(infile) do
begin
      readln(infile, i, j, t1, t2, t3, t4);
      time := b0 + b1 * t1 + b2 * t2 + b3 * t3 + b4 * t4;

      if travel_time [i,j] > time then
      begin
            travel_time [i,j] := time;
      end;
end;
close(infile);

writeln('Reading Travel Time Error Empirical Distribution');
open(file_variable := infile, file_name := '[emsproj.working]resid.prb',
      history := old, access_method := sequential);
reset(infile);

for i := 1 to travel_intervals do readln(infile, travel_empirical [i]);
close(infile);
if not historical then
begin
      writeln('Reading Crisis Arrival Empirical Distribution');
      open(file_variable := infile, file_name := '[.prb]arrival.prb',
            history := old, access_method := sequential);
      reset(infile);

      for i := 1 to arrival_intervals do
            readln(infile, arrival_empirical [i]);
      close(infile);

      writeln('Reading Zone Weightings');
      open(file_variable := infile, file_name := '[.prb]demand.prb',
```

```
            history := old, access_method := sequential);
        reset(infile);

        for i := 1 to number_of_zones do readln(infile, demand [i]);
        close(infile);
        writeln('Reading Service Time Empirical Distribution');
        open(file_variable := infile, file_name := '[.prb]service.prb',
        history := old, access_method := sequential);
        reset(infile);

        for i := 1 to service_intervals do
            readln(infile, service_empirical [i]);
        close(infile);

        writeln('Reading False Alarm Arrival Empirical Distribution');
        open(file_variable := infile,file_name := '[.prb]falseint.prb',
            history := old, access_method := sequential);
        reset(infile);

        for i := 1 to false_arrival_intervals do
            readln(infile, false_arrival_empirical [i]);
        close(infile);

        writeln('Reading False Alarm Duration Empirical Distribution');
        open(file_variable := infile, file_name := '[.prb]falselen.prb',
            history := old, access_method := sequential);
        reset(infile);

        for i := 1 to false_length_intervals do
            readln(infile, false_length_empirical [i]);
        close(infile);
    end;
    max_half := -1;

end;                                    { procedure read_termination_conditions }

function check_termination_conditions(clock : real) : boolean;

var
    i, j : integer;
    done : boolean;
    sigma, ts : real;
    temp : ambulance_node;
```

```
begin
    with temp do
    begin
        number_of_responses := 0;
        max_response_time := 0;
        cumulative_response_time := 0;
        on_line_time := 0;
        number_of_false_alarms := 0;
        utilization := 0;
    end;

                                        { * lines added by J. Goldberg 4/8/88 * }
    for i := 1 to number_of_zones do
        for j :=1 to number_of_ambulances do
        begin
            gamb_zone_stat [i,j,1] :=
                gamb_zone_stat [i,j,1] + amb_zone_stat [i,j,1] ;
            gamb_zone_stat [i,j,2] :=
                gamb_zone_stat [i,j,2] + amb_zone_stat [i,j,2] ;
            gamb_zone_stat [i,j,3] :=
                gamb_zone_stat [i,j,3] + amb_zone_stat [i,j,3] ;
        end ;
                                            { * end of lines added on 4/8/88 * }

    for i := 1 to number_of_ambulances do
    begin
        total_am_stats [i,1].number_of_responses :=
            total_am_stats [i,1].number_of_responses +
            ambulance [i].number_of_responses;

        temp.number_of_responses :=
            temp.number_of_responses +
            ambulance [i].number_of_responses;

        total_am_stats [i,2].number_of_responses :=
            total_am_stats [i,2].number_of_responses +
            ambulance [i].number_of_responses ** 2;

        total_am_stats [i,1].max_response_time :=
            total_am_stats [i,1].max_response_time +
            ambulance [i].max_response_time;

        temp.max_response_time :=
            temp.max_response_time +
```

```
        ambulance [i].max_response_time;

total_am_stats [i,2].max_response_time :=
        total_am_stats [i,2].max_response_time +
        ambulance [i].max_response_time ** 2;

total_am_stats [i,1].cumulative_response_time :=
        total_am_stats [i,1].cumulative_response_time +
        ambulance [i].cumulative_response_time/
        ambulance [i].number_of_responses;

temp.cumulative_response_time :=
        temp.cumulative_response_time +
        ambulance [i].cumulative_response_time/
        ambulance [i].number_of_responses;

total_am_stats [i,2].cumulative_response_time :=
        total_am_stats [i,2].cumulative_response_time +
        (ambulance [i].cumulative_response_time/
        ambulance [i].number_of_responses) ** 2;

                                        { percentage under critical time }
total_am_stats [i,1].on_line_time :=
        total_am_stats [i,1].on_line_time +
        ambulance [i].number_under_critical_time/
        ambulance [i].number_of_responses;

temp.on_line_time := temp.on_line_time +
        ambulance [i].number_under_critical_time/
        ambulance [i].number_of_responses;

total_am_stats [i,2].on_line_time :=
        total_am_stats [i,2].on_line_time +
        (ambulance [i].number_under_critical_time/
        ambulance [i].number_of_responses) ** 2;

total_am_stats [i,1].number_of_false_alarms :=
        total_am_stats [i,1].number_of_false_alarms +
        ambulance [i].number_of_false_alarms;

temp.number_of_false_alarms :=
        temp.number_of_false_alarms +
        ambulance [i].number_of_false_alarms;

total_am_stats [i,2].number_of_false_alarms :=
        total_am_stats [i,2].number_of_false_alarms +
```

```
            ambulance [i].number_of_false_alarms ** 2;

        total_am_stats [i,1].utilization :=
            total_am_stats [i,1].utilization +
            ambulance [i].utilization/clock;

        temp.utilization := temp.utilization +
            ambulance [i].utilization/clock;

        total_am_stats [i,2].utilization :=
            total_am_stats [i,2].utilization +
            (ambulance [i].utilization/clock) ** 2;
end;

total_am_stats[number_of_ambulances + 1,1].number_of_responses :=
total_am_stats[number_of_ambulances + 1,1].number_of_responses +
    temp.number_of_responses;

total_am_stats[number_of_ambulances + 1, 2].number_of_responses :=
total_am_stats[number_of_ambulances + 1, 2].number_of_responses +
    temp.number_of_responses ** 2;

total_am_stats[number_of_ambulances + 1, 1].max_response_time :=
total_am_stats[number_of_ambulances + 1, 1].max_response_time +
    temp.max_response_time;

total_am_stats[number_of_ambulances + 1, 2].max_response_time :=
total_am_stats[number_of_ambulances + 1, 2].max_response_time +
    temp.max_response_time ** 2;

total_am_stats[number_of_ambulances+1,1].cumulative_response_time :=
total_am_stats[number_of_ambulances+1,1].cumulative_response_time+
    temp.cumulative_response_time/number_of_ambulances;

total_am_stats[number_of_ambulances+1,2].cumulative_response_time:=
total_am_stats[number_of_ambulances+1,2].cumulative_response_time+
    (temp.cumulative_response_time/number_of_ambulances) ** 2;

total_am_stats[number_of_ambulances+1,1].on_line_time :=
total_am_stats[number_of_ambulances+1,1].on_line_time +
    temp.on_line_time/number_of_ambulances;

total_am_stats[number_of_ambulances+1,2].on_line_time :=
total_am_stats[number_of_ambulances+1,2].on_line_time +
    (temp.on_line_time/number_of_ambulances) ** 2;
```

```
total_am_stats[number_of_ambulances+1,1].number_of_false_alarms :=
total_am_stats[number_of_ambulances+1,1].number_of_false_alarms +
      temp.number_of_false_alarms;

total_am_stats[number_of_ambulances+1,2].number_of_false_alarms :=
total_am_stats[number_of_ambulances+1,2].number_of_false_alarms +
      temp.number_of_false_alarms ** 2;

total_am_stats[number_of_ambulances+1,1].utilization :=
total_am_stats[number_of_ambulances+1,1].utilization +
      temp.utilization/number_of_ambulances;

total_am_stats[number_of_ambulances + 1, 2].utilization :=
total_am_stats[number_of_ambulances + 1, 2].utilization +
      (temp.utilization/number_of_ambulances) ** 2;

done := true; max_half := -1;
for i := 1 to number_of_zones do
begin
      if zone_data[i,2] >= min_calls then
      begin
            total_zone_stats[i,1] :=
                  total_zone_stats[i,1] + zone_data[i,1]/zone_data[i,2];

            total_zone_stats[i,2] := total_zone_stats[i,2]
                  + (zone_data[i,1]/zone_data[i,2]) ** 2;

            total_zone_stats[i,3] := total_zone_stats[i,3] + 1;
      end;

      if total_zone_stats[i,3] > 1 then
      begin
            ts := t_stat(round(total_zone_stats[i,3]));

            sigma := total_zone_stats[i,2]/total_zone_stats[i,3]
                  - (total_zone_stats[i,1]/total_zone_stats[i,3])**2;

            if ts * sq_root(sigma/total_zone_stats[i,3])
                  > half_length then
                  done := false;

            if ts * sq_root(sigma/total_zone_stats[i,3])
                  > max_half then
                  max_half := ts * sq_root(sigma/total_zone_stats[i,3]);
      end;
end;
```

```
        check_termination_conditions:= done and (runs > 1);
end;                                    { function check_termination_conditions }

begin                                                          {main-routine}
        open(file_variable := error_file, file_name := 'error.dat',
                history := new, access_method := sequential);
        rewrite(error_file);

        read_termination_conditions;

        runs := 0;
        done := runs >= batch_size;

        while not done do
        begin
                runs := runs + 1;
                run_simulation(clock);
                done := check_termination_conditions(clock)
                        or (runs >= batch_size);
        end;
        report;
        close(error_file);
end.                                    { program ambulance–main routine }
```

## APPENDIX B

Goldberg's analytic model (denoted A-Model) is:

$$Maximize \sum_{c\epsilon R}\sum_{i} d_{ic}\left[\sum_{k}\sum_{j}\left[(P_{ij}x_{ijk}(1-\rho_j))(\prod_{l=1}^{k-1}\sum_{r} x_{irl}\rho_r)\right]\right] \quad [A1]$$

$$Subject\ to: \sum_{j} x_{ijk} = 1,\ for\ each\ (i,k)\ pair \quad [A2]$$

$$\sum_{k} x_{ijk} \leq 1,\ for\ each\ (i,j)\ pair \quad [A3]$$

$$\sum_{j} x_j \leq f \quad [A4]$$

$$\sum_{i}\sum_{k} x_{ijk} \leq M x_j,\ for\ each\ j \quad [A5]$$

$$x_r \sum_{k} k x_{ijk} \leq \sum_{k} k x_{irk},$$

$$for\ each\ base\ pair\ (j,r),\ where\ j\ is\ closer\ than\ r \quad [A6]$$

$$\rho_j = \frac{\sum_i \sum_c \left[d_{ic}(t_{ij}+S_{ic})(1-\rho_j)(\sum_k x_{ijk}\prod_{l=1}^{k-1}\sum_r x_{irl}\rho_r)\right]}{TT},\ for\ each\ j \quad [A7]$$

$$x_j \in [0,1], \quad x_{ijk} \in [0,1], \quad 0 \leq \rho_j \leq 1 \quad [A8]$$

in which

$$x_j = \begin{cases} 1, & \text{if base j is open,} \\ 0, & \text{else,} \end{cases}$$

$$x_{ijk} = \begin{cases} 1, & \text{if base } j \text{ is the } k\text{th closest open base to zone } i, \\ 0, & \text{else,} \end{cases}$$

$\rho_j =$ the utilization of an ambulance located at base $j$,

$d_{ic} =$ the total number of expected calls of type $c$ for zone $i$,

$P_{ij} =$ the probability that an ambulance at base $j$ can reach a call at $i$ within 8 minutes,

$f =$ the total number of facilities to open in the area,

$S_{ic} =$ the mean service time per call of type $c$ in zone $i$ excluding travel from a base to $i$,

$TT =$ the total time available for an ambulance in the study, and

$M =$ a large integer.

## APPENDIX C

The steps of the pairwise interchange heuristic are as follows. For detailed descriptions, see Goldberg et. al. ([15], 1988).

Step 0 : Find an initial set of $f$ locations. Denote this set as the incumbent solution. Set the incumbent objective value to the objective function value generated by the initial solution (using steps 3, 4, and 5 below). Set the improvement flag to 1 and go to step 1.

Step 1 : If the improvement flag equals to 0, stop, the incumbent solution is the heuristic solution. Else (improvement on last iteration), using the incumbent solution, set up two lists. List $O$ represents the open bases while list $C$ represents the closed bases. Set the improvement flag to 0. Set the counters of the lists, $a$ and $b$, to 1, go to step 2.

Step 2 : Create the current solution. Start with the open list $O$, delete $O(a)$ and add $C(b)$. Go to step 3.

Step 3 : Rank order the current solution for each zone according to the travel time expected values $t_{ij}$. This ranking represents the $x_{ijk}$ values. Go to step 4.

Step 4 : Using the rank order, solve the non-linear equations [A7] (see Appendix B) for the $\rho_j$. Go to step 5.

Step 5 : Using the $\rho_j$ values and the rank order, evaluate the objective function for the current solution. If the objective is larger than the incumbent objective value, replace the incumbent objective with the current objective, replace the incumbent solution with the current solution, and set the improvement flag to 1. Add one to $b$. If $b$ is greater than $J - f$ then add one to $a$, and set $b$ to 1. If $a$ is greater than $f$ then go to step 1, else go to step 2.

## REFERENCES


[1] R. Benveniste, "Solving the combined zoning and location probelm for several emergency units," *Journal of the Operational Research Society*, Vol 36 (5), 1985, pp. 433-450.

[2] G. N. Berlin and J. C. Liebman, "Mathematical analysis of emergency ambulance location," *Socio-Economic Planning Sciences*, Vol 8, 1974, pp. 323-328.

[3] G. N. Berlin, C. ReVelle, and D. J. Elzinga, "Determining ambulance-hospital locations for on-scene and hospital services," *Environment and Planning A*, Vol 8 (5), 1976, pp. 553-561.

[4] O. Berman, R. C.Larson, and S. S. Chiu, "Optimal Server Location on a Network Operating as an M/G/1 Queue," *Operations Research*, Vol 33, July-August 1985, pp. 746-771.

[5] J. M. Chaiken and R. C. Larson, "Methods for allocating urban emergency units: A survey," *Management Science*, Vol 19 (4), Part II, 1972, pp. 110-130.

[6] S. S. Chiu, O. Berman, and R. C. Larson, "Locating a Mobile Server Queueing Facility on a Tree Network," *Management Science*, Vol 31 (6), June 1985, pp. 764- 772.

[7] R. Church and C. ReVelle, "The maximal covering location problem," *Papers of the Regional Science Association*, Vol 32, 1974, pp. 101-118.

[8] M. S. Daskin, "A maximum expected covering location model: Formulation, properties and heuristic solution," *Transportation Science*, Vol 17 (1), Feb. 1983, pp. 48-70.

[9] M. S. Daskin, "A hierarchical objective set covering model for emergency medical service vehicle deployment," *Transportation Science*, Vol 15 (2), May 1981, pp. 137-152.

[10] M. S. Daskin and E. H. Stern, "A framework for EMS crew scheduling," *Modeling and Simulating*, Vol 11 (4), 1980, pp. 1443-1448.

[11] D. J. Eaton, M. S. Daskin, D. Simmons, B. Bulloch, and G. Jansma, "Location techniques for emergency medical service vehicles," *Policy Research Report*, Vol I-IV (34), Lyndon B. Johnson School of Public Affairs, University of Texas at Austin, Texas, 1979.

[12] D. J. Eaton, M. S. Daskin, D. Simmons, B. Bulloch, and G. Jansma, "Determining Emergency Medical Service Vehicle Deployment in Austin, Texas," *Interfaces*, Vol 15, Jan.-Feb. 1985, pp. 96-108.

[13] J. A. Fitzsimmons, "A methodology for emergency ambulance deployment," *Management Science*, Vol 19 (6), 1973, pp. 627-636.

[14] J. Goldberg, R. Dietrich, J. M. Chen, M. Mitwasi, T. Valenzuela, and E. Criss, "Using discrete event simulation to locate emergency medical units," *Technical Report*, Department of Systems and Industrial Engineering, University of Arizona, 1988.

[15] J. Goldberg, R. Dietrich, J. M. Chen, M. Mitwasi, T. Valenzuela, and E. Criss, "Locating emergency medical units in an urban area," *Working Paper #88-010*, Department of Systems and Industrial Engineering, University of Arizona, 1988.

[16] L. Green, "A Multiple Dispatch Queueing Model of Police Patrol Operations," *Management Science*, Vol 30 (6), June 1984, pp. 653-664.

[17] L. Green and P. Kolesar, "A Comparison of the Multiple Dispatch and M/M/c Priority Queueing Models of Police Patrol," *Management Science*, Vol 30 (6), June 1984, pp.665-670.

[18] J. P. Jarvis, "Models for the location and dispatch of emergency medical vehicles," Operations Research Center, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1975, pp. 83-99.

[19] P. Kolesar and E. H. Blum, "Square root laws for fire engine response distances," *Management Science*, Vol 19 (12), 1973, pp. 1368- 1378.

[20] R. C. Larson, "A hypercube queueing model for facility location and redistricting in urban emergency services," *Computing and Operations Research*, Vol 1 (1), 1974, PP.67-95.

[21] R. C. Larson, "Approximating the performance of urban emergency service systems," *Operations Research*, Vol 23 (5), Sept.-Oct. 1975, pp. 845-868.

[22] A. M. Law and W. D. Kelton, *Simulation Modeling and Analysis*, McGraw-Hill, New York, 1982.

[23] L. C. Santone and G. N. Berlin, "A computer model for the evaluation of fire station location," *National Bureau of Standard Report*, U.S Department of Commerce, Washington D.C., 1969.

[24] E. S. Savas, "Simulation and cost-effectiveness of New York's emergency ambulance service," *Management Science*, Vol 15 (12), August 1969, pp. B608-B627.

[25] C. Saydam and M. Mcknew, "A separable programming approach to expected coverage: An application to ambulance location," *Decision Sciences*, Vol 16, 1985, pp. 381-197.

[26] D. Schilling, D. J. Elzinga, J. Cohon, and C. ReVelle, "Design and analysis of location and relocation alternatives in a fire protection system," *Pittsburg Conference on Modeling and Simulation Proceedings*, Vol 11 (4), May 1980, pp. 1455-1460.

[27] R. Serfozo, "Allocation of Servers for Stochastic Service Stations with one Overflow Station," *Management Science*, Vol 31 (8), August 1985.

[28] C. Toregas, R. Swain, C. ReVelle, and L. Bergman, "The location of emergency service facilities," *Operations Research*, Vol 19 (6), 1971, pp. 1363-1373.

[29] R. A. Volz, "Optimum ambulance location in semi-rural areas," *Transportation Science*, Vol 5 (2), 1971, pp. 193-203.

[30] J. R. Weaver and R. L. Church, "A multi-criteria approach to ambulance location," *Pittsburg Conference on Modeling and Simulation Proceedings*, Vol 11 (4), May 1980, pp. 1461-1466.