Simulation Modeling of Prehospital Trauma Care

Robert L. Wears
University of North Florida
SIMULATION MODELING OF PREHOSPITAL TRAUMA CARE

by

Robert L. Wears, MD

A thesis submitted to the
Department of Computer and Information Sciences
in partial fulfillment of the requirements
for the degree of

Master of Science in Computer and Information Sciences

UNIVERSITY OF NORTH FLORIDA
DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES
April 30, 1993
The thesis "Simulation Modeling of Prehospital Trauma Care" submitted by Robert L. Wears in partial fulfillment of the requirements for the degree of Master of Science in Computer and Information Sciences has been approved by the thesis committee:

Signed by Charles N. Winton
Thesis Adviser and Committee Chairperson

Signed by F. Layne Wallace

Signed by Susan R. Wallace

Accepted for the Department of Computer and Information Sciences:

Signed by Robert F. Roggio
Chairperson of the Department

Accepted for the College of Computing Sciences and Engineering:

Signed by Robert F. Roggio
Dean of the College

Accepted for the University:

Signed by Charles M. Galloway
Dean of Graduate Studies
Acknowledgement/Dedication

I wish to thank my wife and family for their patience, understanding and support during my second postgraduate education.
Table of Contents

Abstract .................................................. xi

Chapter 1 Introduction ................................. 1
  1.1 Statement of the Problem ....................... 1
  1.2 Historical Perspective .......................... 1

Chapter 2 Description of the System ............... 3
  2.1 Definition ......................................... 3
  2.2 System Elements ................................. 3
     2.2.1 Patients .................................... 3
     2.2.2 Vehicles .................................... 4
     2.2.3 Receiving Facilities ....................... 4
     2.2.4 Transportation Network .................... 5
  2.3 System Operation ................................. 5
     2.3.1 Temporal Sequence .......................... 5
     2.3.2 Physiologic Sequence ....................... 7
  2.4 Goals of the Model ............................... 7
  2.5 Potential Enhancements ......................... 8

Chapter 3 Model Design ............................... 9
  3.1 General Design Issues ........................... 9
     3.1.1 Simulation Environment .................... 9
     3.1.2 Verification and Validation ............... 10
     3.1.3 Statistical Issues ......................... 10
  3.2 Specific Design Issues ......................... 12
     3.2.1 Patterns of Injury ......................... 12
6.4 Further Work ........................................... 56
References ................................................. 58
Appendix 1  Program Source Code  ......................... 62
Appendix 2  Data Cross-Reference  ....................... 119
Appendix 3  Data Files ................................... 151
Appendix 4  Sample Output ................................ 160
Appendix 5  Fitting Input Distributions ................... 173
Appendix 6  Log of Model Assumptions .................... 183

6.1 Distribution of Accidents  ............................. 183
6.2 Definitive Care Survival  .............................. 183
6.3 Private Travel ........................................ 183
Appendix 7  Log of Improvements and Enhancements .... 185

7.1 Improvements ........................................ 185
  7.1.1 Transit time ...................................... 185
  7.1.2 Choke points .................................... 185
  7.1.3 Events .......................................... 185
  7.1.4 End-of-run ..................................... 186
  7.1.5 Memory management ............................ 186

7.2 Enhancements ......................................... 186
  7.2.1 Non-trauma patients ............................ 186
  7.2.2 Injury model ................................... 187
  7.2.3 Transfers ....................................... 187
  7.2.4 Data editor ..................................... 187
  7.2.5 Graphical output ............................... 188
  7.2.6 Trace control .................................. 188
  7.2.7 Interruption .................................... 188
  7.2.8 Non-regenerative Simulation .................... 189
Figures

Figure 1. Logical model of the overall transportation network. (See Figure 3 for node acronyms).......................... 19

Figure 2. Logical model of the central region of the transportation network. (See 3 for node acronyms)............... 20

Figure 3. Spatial distribution of trauma incidents in the study area.................. 21

Figure 4. Proportion arrivals in 12 hour periods over one week, corresponding to data in Table 2......................... 45

Figure 5. Histogram of 2500 variates from mygamma.f given arguments 1.5, 1.5. .......................... 46

Figure 6. Probability plot of 2500 variates from mygamma.f given arguments 1.5, 1.5. .......................... 47

Figure 7. Quantile plot comparing the model’s transport times with those provided by Campbell.......................... 51

Figure 8. Distribution given need to secure.......................... 173

Figure 9. Probability plot: exponential.......................... 173

Figure 10. Quantile plot: exponential.......................... 173

Figure 11. Probability plot: Weibull distribution.......................... 174

Figure 12. Quantile plot: Weibull(6.686, 1.412).......................... 174

Figure 13. Probability plot: lognormal.......................... 174

Figure 14. Quantile plot: lognormal.......................... 174

Figure 15. Distribution of time to patient.......................... 175

Figure 16. Probability plot: exponential.......................... 175

Figure 17. Quantile plot: exponential.......................... 175
Figure 18. Distribution of scene treatment time (includes extrication if needed)...

Figure 19. Probability plot: Weibull.

Figure 20. Quantile plot: Weibull(11.023, 1.892).

Figure 21. Probability plot: gamma(3).

Figure 22. Quantile plot: gamma(3).

Figure 23. Weibull (dashed line) and gamma pdf's.

Figure 24. Distribution of time to hospital.

Figure 25. Probability plot: Weibull.

Figure 26. Quantile plot: Weibull(9.974, 2.242).

Figure 27. Probability plot: lognormal.

Figure 28. Quantile plot: lognormal.

Figure 29. Probability plot: gamma(3).

Figure 30. Quantile plot: gamma(3).

Figure 31. Distribution of time to release of patient.

Figure 32. Probability plot: Weibull.

Figure 33. Quantile plot: Weibull(2.718, 1.810).

Figure 34. Probability plot: lognormal.

Figure 35. Quantile plot: lognormal.

Figure 36. Probability plot: gamma.

Figure 37. Quantile plot: gamma(1.5).

Figure 38. Probability plot: exponential.

Figure 39. Quantile plot: exponential.
Tables

Table 1. Random variables used in the model. . . . . . . 42
Table 2. Example data file of mean arrivals by interval for testing the nsp.f routine. 44
Table 3. Characteristics of ISS scores obtained by the model and those reported by Baker. . . . 46
Table 4. Portions of trace output demonstrating the manifestation of specific model design items in the implementation. . . . . . . . 48
Table 5. Comparison of outcome estimates produced by the model with those estimated by Jacksonville Fire/Rescue. . . . . . . 49
Table 6. System performance (mean ± 95% confidence interval) under different trauma center triage criteria. . . . . . . . . . 53
Table 7. Convergence points including at least a full seven day cycle. . . . . . . . . . 54
Table 8. System performance (mean ± 95% confidence interval) under alternate helicopter dispatch criteria. . . . . . . . . . 55
Abstract

Prehospital emergency care systems are complex and do not necessarily respond predictably to changes in management. A combined discrete-continuous simulation model focusing on trauma care was designed and implemented in SIMSCRIPT II.5 to allow prediction of the systems response to policy changes in terms of its effect on the system and on patient survival.

The utility of the completed model was demonstrated by the results of experiments on triage and helicopter dispatching policies. Experiments on current and two alternate triage policies showed that helicopter utilization is significantly increased by more liberal triage to Level 1 trauma centers, which was expected, but that the waiting time for pending accidents tended to decrease, an unexpected consequence. Experiments on helicopter dispatch policy showed that liberalization of the dispatch policy would have much greater consequences than would changing the triage criteria. Again, this result was unexpected and has received little attention from system planners and administrators, especially with respect to the degree of discussion and controversy surrounding triage criteria.
Chapter 1
Introduction

1.1 Statement of the Problem

Prehospital care of the sick and injured has developed into a complex system in the last 30 years. Much of this development has been "bottom-up," driven by technological factors and the availability heuristic (any available tool will eventually be used). This has eventually led to considerable debate in the medical literature over the appropriate role of several treatment modalities routinely employed in many localities. Furthermore, as resource constraints and other external factors have stressed the system, the need for a systematic overview of the system has become apparent. This project will develop a simulation model of a prehospital trauma care system in order to provide a method by which the effect of modifications to the system can be estimated.

1.2 Historical Perspective

Since prehospital care systems form complex networks of interacting entities that are difficult to work with analytically, simulation has frequently been used as an aid in planning and organizing such systems. The majority of
these simulations have concentrated on relatively static aspects of the system, such as the number and location of responders [Fitzsimmons82, Uyeno84], improvements in response or transport time, etc. [Valenzuela90]. This project will focus more on clinical issues which are more easily modified on a dynamic basis by changing clinical and administrative policies.
Chapter 2
Description of the System

2.1 Definition

The system under consideration is the that portion of the pre-hospital emergency medical care system (EMS) which deals with injury in the seven county service area of northeast Florida and southeast Georgia. The EMS system is obviously impacted by non-traumatic illness as well, so the model must include some representation of their effects, but they will not be the focus of the model.

2.2 System Elements

The system can be decomposed into four fundamental elements: patients, vehicles, receiving facilities, and a transportation network over which vehicles move patients from sites of injury to or between receiving facilities.

2.2.1 Patients. Patients suffer injuries in a particular temporal and spatial distribution. Their occurrence is frequently not independent; for example, most automobile accidents involve two cars and therefore at least two patients. In addition, injuries occur in the two broad, nonexclusive categories of blunt and penetrating. Within these categories, patterns of correlated injuries exist; for
example, brain injury is typically isolated in penetrating trauma, but typically associated with chest and abdominal injuries in blunt. Injuries differ in severity, which affects the probability of survival.

2.2.2 Vehicles. Vehicles in the system are helicopter ambulances, ground ambulances, and private conveyances. Helicopter ambulances are typically few and therefore subject to more stringent dispatching criteria than ground ambulances. The receiving facilities have a degree of control over the destination of ambulances, and receive prior notification of incoming ambulance patients, but benefit from neither with respect to patients arriving by private conveyances. Additionally, ambulance personnel may perform a limited number of therapeutic interventions prior to transporting the patient to a receiving facility. Ground ambulances and private conveyance are constrained to use the transportation network; helicopter ambulances generally travel faster and by line of sight, but are constrained by weather conditions and the need for a safe landing zone.

2.2.3 Receiving Facilities. Receiving facilities in the system are hospitals and other acute care facilities such as clinics or physicians' offices. Hospitals may be classified into Level 1, 2, or 3 trauma centers as defined by Florida statute. Alternatively, they may choose not to participate in the trauma center system; their actual capabilities typically do not change by virtue of this decision.
Receiving facilities will perform initial resuscitation and evaluation of incoming patients, and then transfer them out of the system to definitive care.

2.2.4 Transportation Network. The transportation network consists of existing major roads, highways and bridges. A patient’s transport time by ground conveyance is a function of the available path through the transportation network and the time of day. Geographic barriers such as the St. John’s River are reflected in the transportation network. Because ambulances are most commonly directly managed by county governments, political boundaries also may affect transportation decisions. For example, in patients with relatively minor injuries, the target receiving facility may be chosen such that the path to it does not involve crossing a county or state line; these considerations are dropped in the face of severe injury.

2.3 System Operation

System operation consists of a temporal sequence of events running in parallel and interacting with a continuous pattern of physiological changes.

2.3.1 Temporal Sequence. A typical cycle begins with an injury-producing episode which generates one or more patients at a particular location and time with a given pattern and severity of injuries. The prehospital system is then activated and an ambulance dispatched to the location,
typically on a proximity basis. The time from injury to arrival on scene is termed "activation time," and will be noted as $t_a$. Once on scene, EMS personnel may have to locate and/or extricate patients, and may perform some therapeutic services such as starting intravenous fluids, endotracheal intubation, etc. These maneuvers typically will extend the "on scene time" ($t_s$). Their efficacy is a matter of some debate and could be an item of study in the simulation model.

Once extrication, initial assessment, and initial therapy (if any) have been performed, the patient is transported to a receiving facility in "transport time" ($t_t$). The means of choosing a receiving facility (e.g., nearest hospital, nearest hospital of a given level, etc.) has also provoked considerable debate, and will be examined in the simulation.

The receiving facility will perform initial resuscitation and evaluation and will then deliver the patient to definitive care (e.g., the operating room, admitted to the hospital, etc.) after "resuscitation time" ($t_r$) and some additional waiting time ($t_x$). Definitive care is considered to be outside the system. In some cases, the receiving facility may transfer the patient to another facility, repeating the transport and resuscitation stages of the cycle.
2.3.2 Physiologic Sequence. During this process, the patient's physiological state will change depending on his injuries and the therapy received. Some patients will die before being delivered to definitive care; for those that do not, their probability of survival will be estimated from their injuries and their physiological state at the time of exit from the system [Wears90, Champion91]. Based on their major immediate physiologic effects, injuries can be categorized into three large groups: those producing blood loss; those interfering with respiratory exchange; and those affecting the central nervous system. The physiologic state in each of these deteriorates over time without intervention. Indirect evidence of the severity of injury in these categories is combined into a "trauma score" which is used by EMS personnel to make therapeutic and transportation decisions.

2.4 Goals of the Model

Any simulation model should be constructed to answer specific questions, rather than just show that a model can be constructed. This model will be designed to estimate the effects of changes in:

a. Triage criteria that determine the center to which a patient should be routed.

b. Number of trauma centers of specified level.

c. Criteria for helicopter transportation vs ground transportation.
d. Divert policy (the circumstances and length of time during which a hospital may divert incoming cases to another facility).

e. Location of trauma centers.

These effects will be measured from two perspectives: from the point of view of the system (numbers of patients received, percent utilization, etc.) and from the point of view of the patient (length of time until definitive care, change in survival probability).

2.5 Potential Enhancements

While not an immediate goal of this project, the potential for enhancement of the model to handle additional questions will be kept in mind as a secondary goal. Such additional questions might include analysis of the system during periods of drastically increased demand and/or reduced capacity, as might occur during a natural or man-made disaster; extension of the model to handle non-traumatic medical conditions. Another secondary goal will be portability to other geographic areas without re-compilation; thus to the extent it is practical, area-specific information will be represented by data elements read in from a file, perhaps in a pre-computing step, rather than directly embedded in the program code.
Chapter 3
Model Design

3.1 General Design Issues

General design issues for this project are those common to virtually all simulation models: selection of a simulation environment and the appropriate level of detail, verification of the implementation, validation of model, and the design and analysis of appropriate experiments.

3.1.1 Simulation Environment. The model was implemented in SIMSCRIPT II.5 (CACI Products, La Jolla, CA) for several reasons. SIMSCRIPT is available on a large number of computer systems and has wide general acceptance as a simulation language, thereby facilitating the potential portability of the model. The EMS model proper lends itself easily to discrete simulation, while the physiologic model is more naturally thought of as continuous; SIMSCRIPT provides support for simultaneous continuous and discrete simulation, thus facilitating modeling the interaction between these two components. And finally, local expertise and experience with SIMSCRIPT was available.
3.1.2 Verification and Validation. Separate verification runs checking aspects of the model's logic have been performed and compared to specific test cases derived from available Trauma Registry data. Many of these verification runs were initially performed at the module level so that the desired (true) behavior of the model can be more easily predicted. An activity trace is produced by the model to aid in verification and validation.

The model was validated by checking its output against aggregate data on injury types, patterns of transportation and survival using published data and University Medical Center's local trauma registry. It is unfortunately the case that detailed data on the overall operation of the prehospital care system are not maintained; a modified Turing test may assist in further model validation. The current level of validation of the model is not considered sufficiently definitive for the model to be used in establishing policy. Further validation will require explicit collection of data from the system for comparison to model output.

3.1.3 Statistical Issues. Care has been taken to maintain synchronization of the random number streams when considering policy alternatives; this reduces the variance of the difference between policy alternatives, yielding an increase in statistical power and perhaps a reduction in computing time.
The system under study does not possess well-defined starting and ending times. However, it is the case that the system as defined here does empty out from time to time. Therefore, no warm-up period to eliminate the effect of start-up transients was used. Instead, the model is started empty and idle, and the regenerative method will be used to determine run lengths; i.e., a run will be ended when the system returns to the empty and idle state. It should be noted that this method of experimental design might not be desired when the goal is determining system performance under overload (mass casualty) situations; however, only the method of experimentation, not the actual model, would have to be changed.

The primary goal of the model is effect estimation, not hypothesis testing. Statistical testing of the differences between model outputs under differing policies is complicated by the use of the regenerative method, since it cannot be guaranteed that parallel runs will always be directly comparable, even though every random component for each patient is guaranteed to be comparable. For example, individual runs might not necessarily have the same numbers of patients; in general, parallel runs will diverge and reconverge at unpredictable points. A naive direct

---

1 This does not make it a terminating simulation, because even though the system is empty of patients, the ending value of time for the first run is the beginning value of time for the second, and the time until the next accident is dependent on the current time [Law91].
comparison of alternatives as if they were independent will typically overestimate the variance of the difference in effect. To compare the alternatives properly, summary measures must be calculated at a point where the model has reconverged under each alternative.

3.2 Specific Design Issues.

Certain problems peculiar to this project arose in the development of the model, and were dealt with as follows.

3.2.1 Patterns of Injury. The spatial pattern of injury was assumed to be roughly proportional to population density. This has been shown to be the case in at least one major city [Zachariah92]. Zachariah also showed that the distribution of types of accidents (e.g., assault, auto accident, gun-shot wound, etc.) was to a large extent invariant across time and space; therefore these variables were assumed to be constant in the model.

The temporal pattern of injury was modeled by a non-stationary batch Poisson process, using the method of Çinlar [Çinlar75]. Raw data kindly provided by Zachariah (personal communication) was used to estimate the diurnal pattern of injury occurrence. Variation across days of the week was obtained from Baker92, and the two items combined to produce the weekly cycle of injury incidence used in the model.
3.2.2 Transportation Network. The geographic area of interest was represented at a higher level than blocks or map coordinates by modeling the area as a digraph. Nodes in this graph represent certain critical areas, such as: neighborhoods or fire-rescue service areas from which requests for care arise; choke points -- areas such as bridges which transporters must traverse en route to their destination; and receivers, typically hospitals categorized according to Florida's trauma statute. Arcs in the digraph were assigned weights representing transport time across that arc; these weights may vary with time of day. While some information on average transport times is available from the Fire-Rescue system, information about the distribution of transport times is not. However, Campbell [Campbell92] has published detailed summary results of a variety of pre-hospital time intervals, and kindly agreed to provide his raw data for use in the project (personal communication). Therefore, distributions were fit to Campbell's data using quantile and probability plots, or occasionally using the method of moments.

Since there are extensive and highly functional mutual assistance agreements among the political jurisdictions in the study area, political boundaries have not been explicitly represented in this model. It would be possible, if desired, to represent political boundaries by placing an empirical penalty function on the pertinent arcs; such a
penalty function should be greater for minor injuries and zero for major injuries.

3.2.3 Edge Effects. Only a finite area will be simulated, but resources located near the boundary of the simulated area might be called to service events occurring beyond the boundary; similarly, injuries occurring within the boundary might be managed at hospitals outside the boundary. Carter74 handled this problem by simulating those events at a lower level of detail. However, this merely moves the problem further away, although at a smaller cost than simply enlarging the simulated area. In the system under consideration here, the boundaries tend to fall at "watershed" lines, where events are rare, and very little boundary crossing occurs. For example, it is common for ambulances in St. John's county to respond to calls in Duval county or to transport patients into Duval county. It is very uncommon that they do so with respect to Flagler county, because of the population densities and pre-existing referral patterns. Therefore we will neglect edge effects in this model (although this assumption might be subjected to sensitivity analysis), save for judiciously choosing the boundaries of the simulated area to keep such effects to a minimum. In the current model, the Keystone Heights area of Clay County was removed from consideration, since the flow of referral in that area tends to move towards Stark and Gainesville, i.e., away from the center of the study system.
For similar reasons, only the Kings Bay area in Georgia was included in the model.

3.2.4 Ambulance Routing. Average node to node times within the transportation network routes are precomputed and stored prior to a simulation run. These times are used to generate ambulance call lists for each node, and hospital destination lists for each node, and the basis of shortest expected travel time. The call and dispatch lists are saved in a file that can be edited to reflect special circumstances. Helicopter ambulances are assumed to be callable to any locations, and to alternate calls. The choice of helicopter vs ground ambulance is based on Trauma Score and distance by current policy, and will be the subject of experimentation.

3.2.5 Physiologic Model. Each patient will be represented as a distinct entity within the model, as will resources such as ambulances, helicopters, and hospitals. A limited set of physiologic variables will be modeled for each patient; however, since detailed physiological modelling [Mazzoni88] is computationally intensive, this information will be kept to the minimum necessary to assess probability of survival at different times.

The model of hemorrhage developed by Wears and Winton [Wears90] will be adapted for use in this project. This model can be easily extended to accommodate respiratory exchange as well. Direct CNS injury seems to be a distinct
problem [Baxt87], which is synergistic with both hemorrhage and respiratory injury. It will be modeled as a "black box" process, whose main effect is to cause a downward adjustment in the probability of survival.

The three components of the physiologic model will be used to compute the Revised Trauma Score (RTS), [Champion81, Champion91], which, in conjunction with the Injury Severity Score [Baker74] or ISS, has achieved general acceptance in predicting survival. The RTS assigns each component of the physiologic model a value on a 0 to 4 scale. These scores may then be simply summed to form a 0 to 12 scale, but a weighted sum [Champion91] with a maximum total of 7.804 is thought to provide better prediction. A mapping between the hemorrhage component and these scores has already been developed in Wears90.

3.2.6 Injury Pattern. Injuries occur in identifiable patterns which a model should represent in order to achieve face validity. This would ideally require generation of categorical variables having a given correlation pattern. While many simulation models have assumed independence of variables, apparently successfully, there is are several instances [Law91] in which it has been shown that the failure to model correlation between variables substantially affected the results. Devroye [Devroye86] offers several plausible approaches towards the general problem of generating correlated random variates, although he does not
specifically address this particular situation. Alternative approaches have been suggested by Johnson [Johnson87]. Unfortunately, the covariance structure of injury patterns has yet to be described quantitatively. Therefore, it was assumed that blunt and penetrating injuries had the same ISS distribution. Injuries were then modeled by assigning an ISS value (drawn from a scaled beta distribution fit to data from MacKenzie86), partitioning the total ISS among the three major categories of physiologic derangement as suggested in Baxt87 and MacKenzie86, and mapping those components to either direct physiologic variables (e.g., blood pressure) or to RTS components. The Revised Trauma Score values thus computed were validated by comparing their distribution to the distribution of TS reported by Champion81 and Morris86, with good agreement.

3.2.7 Critical Outputs. Certain critical variables were used as the basis of comparison between policy alternatives. These included the dynamic proportion of utilization of trauma centers at each level. Since trauma centers typically must maintain excess capacity, an alternative measure of utilization, the proportion of time the center is at or over capacity, will also be tracked. Other important outcome measures include the total time in the system, the mortality in each phase prior to definitive care, and the overall probability of survival following definitive care.
Chapter 4
Implementation

4.1 Overview

The model's realization in SIMSCRIPT is provided in detail in the Appendices. Appendix 1 contains the program code and Appendix 3 the data files used to instantiate the model. This chapter provides an overview of the entire implementation. The geographic area selected was modelled as a digraph as illustrated in Figure 1 and Figure 2. (Routes in these Figures are shown with single lines for clarity only; inspection of the data files in Appendix 3 will confirm the implementation as a digraph). The spatial distribution of trauma incidents used in the model is illustrated in Figure 3, and roughly corresponds to population density in the target area. The remainder of the implementation can be divided into two major sections; data structures and procedures.

4.2 Data Structures

A variety of SIMSCRIPT data structures were used to represent the various model elements. Two general principles were used in representing entities in the model. First, entities having a potential lifespan in the model greater than a typical run length were be represented as
Figure 1. Logical model of the overall transportation network. (See Appendix 3 for node acronyms).

SIMSCRIPT permanent entities, while entities that potentially might "come and go" during the course of a run were represented as SIMSCRIPT temporary entities. Second, no entity should have greater knowledge about itself or about conditions in the system than its would its real-world analog. Application of these principles to the model entities described in 2.2 produced the following set of data structures (lines 39 - 256 in the preamble, Appendix 1.)

4.2.1 Permanent Entities. The following structures are set up by the initialization code and exist throughout the entire simulation.
Figure 2. Logical model of the central region of the transportation network. (See Appendix 3 for node acronyms).

4.2.2 Nodes (lines 43 - 54). Nodes in the transportation network represent areas in which accidents might arise. They are identified by location (latitude and longitude), and have an edge set of arcs representing paths to and from other nodes. Additionally, nodes can own ambulances or hospitals if one is located in a node's area. Each node has a list of ambulances to call for events occurring in its area, and a list of hospitals to which patients will be transported from accidents occurring in the node. Nodes are
collected in sets so that nodes containing hospitals or ambulance bases may be easily identified.

4.2.2.1 Paths (lines 56 - 63). Every node - node pair is connected by a path in each direction. The path is held in the pair's route set, and consists of the sequence of arcs to be traversed in proceeding from one node to the other. Each path is associated with a mean transit time and a flight time. Unidirectional paths may be implemented by
assigning them an essentially infinite travel time in the reverse direction.

4.2.2.2 Hospitals (lines 65 - 100). Each hospital in the system is assigned a level according to Florida Trauma Center designation standards. Hospitals also are assigned a capacity, based on the maximum number of active resuscitations they can handle, and a number of flags indicated whether they are allowed to divert, whether or not they have diverted in a given period, etc. Hospitals also track the number of patients they are currently resuscitating, and a variety of other statistical counters. Hospital divert status is represented by membership in a green (no divert) and a red (divert) set.

4.2.2.3 Ambulances (lines 102 - 117). Each ambulance has a type, indicating whether it is a ground or an air (helicopter) ambulance, and a base node. It also maintains a pointer to an ambulance run process (if any) representing an actual run, and has storage for its current location, although this attribute is not always guaranteed to be current. Ambulances belong to a variety of sets to track their activity, the most important being the ready.set. Membership in the ready.set indicates the ambulance is available to be called to an accident. This representation was chosen over SIMSCRIPT's built-in 'resource' entity since ambulances are not entirely interchangeable.
4.2.3 Temporary Entities. The following structures may be created and destroyed as needed throughout the simulation. The procedures (if any) associated with temporary entities are discussed in Section 4.3.

4.2.3.1 Arcs (lines 121 - 132). Arcs representing logical (not necessarily physical) routes of travel are implemented as temporary entities under the supposition that they could, at least in theory, come into and disappear from existence during the course of a simulation. Arc's are unidirectional and are identified by their source and sink nodes, and a weight representing the average travel time in minutes from the center of the source node to the center of the sink node. A choke weight is also available to represent the average additional delay that might be experience at a choke point. Cumulative weights are used in the calculation of best routes from node to node, but are not subsequently used during the simulation. Since an arc may not belong to more than one set of a given type in SIMSCRIPT, but a given arc may be part of many different routes, duplicate arcs are created, at some cost in storage space (see Appendix 7.1.5).

4.2.3.2 Dispatch lists (lines 134 - 143). Two dispatch lists are maintained by each node; a list of call.items (pointers to ambulances) to be called for incidents occurring in that node, and a list of go.items (pointers to hospitals) indicating destination hospitals for incidents occurring in a node.
4.2.3.3 Ambulance runs (lines 148 - 165). An ambulance run is represented by a process that is created when the dispatcher assigns an ambulance to an accident, and ends when the ambulance returns to its base and is back in service. Ambulance runs are divided into two kinds, trauma and medical. An ambulance run is always associated with a particular ambulance, and has attributes for identifying the accident it is serving, the node from which it travels, and the node, hospital, and hospital's level to which it is bound. Two flags are maintained: status, to identify when a run ends in a recall, and helo.coming, to indicate when an ambulance should wait for the helicopter's arrival, even though it would otherwise be ready to travel (i.e., its scene.time is over). Ambulances keep track of the patients they manage on a run by filing them in the amb.patient.set.

4.2.3.4 Patients (lines 167 - 213). Patients are represented as processes created by accidents, and are destroyed when they either die or are transferred to definitive care. A patient's condition may be alive or dead; they move through several phases (e.g., awaiting treatment, scene treatment, transport, etc.), with phase changes being triggered by setting the change.flag. Physiologic information about a patient consists of the hemodynamic components of the Lewis model of hemorrhage [Lewis86] as modified by Wears and Winton [Wears90], extended to account for the impact of respiratory
embarrassment on oxygen delivery. The patient's degree of injury is measured by the ISS. From the ISS and systolic blood pressure (sbp), the components of the revised Champions trauma score, and the score itself can be determined. A functional attribute, cts.f, allows the trauma score to be updated periodically to reflect the patient's changing condition. Finally, patient maintains a variety of time intervals of interest for reporting purposes.

4.2.3.5 Accidents (lines 216 - 235). Accidents are represented as processes that are created by the generator and are destroyed when the last patient associated with an accident is removed from the accident site. Accidents are of two kinds, medical and trauma. In the current model, medical accidents are served by ambulances just as trauma accidents are; they create no patients but do constitute a demand on the system. Trauma accidents may create several patients, and may be blunt or not (i.e., penetrating). Both medical and trauma cases may be placed on a pending list if insufficient ambulances are available to meet their needs.

4.2.3.6 Events (lines 240 - 244). Events were used to handle a hospital's going on and coming off of divert status. A hospital that places itself on divert status frequently must reopen to ambulances after a certain period of time, regardless of its status at that time. The event go.off.red is used to schedule this status change. In
addition, many jurisdictions clear all their divert status once a day; the event clear.reds performs this function. Finally, the event resp.support is used to model the effect of therapeutic interventions assisting respiration and ventilation occurring in the course of scene treatment or resuscitation.

4.3 Procedures

The model has a natural structure that can be described as a collection of independent but communicating entities. This suggests that an object-oriented approach would have provided the most natural implementation. Since an object-oriented simulation environment was not available, a monitor process was used to handle interprocess communications. A natural monitor, the dispatcher, exists in the real-world system, so this approach meshed nicely with the target model. Interestingly, the monitor function was more easily provided as a procedure, rather than as a SIMSCRIPT entity. Thus the dispatcher is the only major real-world entity that has only an implicit representation in the model.

In this implementation, procedures can be divided into three classes: initialization procedures, trace and reporting procedures, and the actual modeling procedures (with their supporting utility procedures). The individual procedures are described here; their relationship and the flow of control among the procedures is described in Section 4.3.4.
4.3.1 Initialization Procedures. The initialization procedures get system information from a set of files and create the data structures outlined in Section 4.2. They are not called again and in principle could be physically separated from the simulation code itself.

4.3.1.1 Main (lines 417 - 498). Main is the fundamental control routine in a SIMSCRIPT program. Main opens the output files and then calls the routine initialize (see 4.3.1.2). At this point, main could begin a loop for each arm of an experiment; this has been left for future implementation if desired. Next, important global variables such as time.v, nsp.tprime and nsp.last.time (used by the non-stationary Poisson routine nsp.f) are initialized to zero, and the run loop is entered. This loop resets all the runwise totals and schedules the next time for clearing hospital divert status, and then activates the generator process to start the simulation. Once a run is finished, runwise statistics are calculated and the run.report routine is called, and then the loop is re-executed until the requested number of runs has been obtained. The routine final.report is then called and the program terminates.

4.3.1.2 Initialize (lines 2333 - 2365). Initialize gets data to characterize this instantiation of the model. It calls a series of initialization routines (get.table, get.sim, get.net, get.hosp, get.ems, get.accs) that read information from datafiles into model variables in a
straightforward manner. In addition to simply reading in
data on the transportation network, the routine get.net
creates the digraph of nodes and arcs, and calculates mean
transit times to all nodes in the system by calling the
function best.route (see 4.3.1.5).

The ambulance call list and hospital preference list are
either read from datafiles, if they exist, or are
constructed (and written to datafiles for editing or future
use) using default dispatching rules if they do not. This
process is controlled by the routine get.list (lines 1853 -
1873), which is driven by its subprogram variable
(SIMSCRIPT's term for pointer to function) arguments. The
actual construction of the lists is performed by the
routines build.call.list (see 4.3.1.3) and build.hosp.list
(see 4.3.1.4). Before returning, the initialize calls the
routine print.net to output the data it has read for
verification of the model's initialization.

4.3.1.3 Build.call.list (lines 884 - 972). Build.call.list
uses the internode transit times along the best path to
construct for each node a list of ambulances to be called
for incidents in that node, based on least travel time. The
default rules governing ambulance selection are: helicopter
ambulances will travel to all nodes; every node will have a
minimum of min.amb ambulances on its list; ambulances are
ranked on the list in order of closest travel time; and that
travel times that are within a given proportion of each
other (stored in the global variable atol) are presumed to be sufficiently equal that all such units will be placed on the list, even if the minimum number of ambulances is exceeded. This is necessary to handle ties in travel time, and to prevent the list from being unreasonably limited by small differences in travel time.

4.3.1.4 Build.hosp.list (lines 975 - 1069).
Build.hosp.list constructs for each node a list of hospitals to which victims in that node will be sent. It is functionally similar to build.call.list (above), but handles the additional complication of maintaining a list for each trauma center level. The default rules here are analogous to those for build.call.list, except that every node must have at least one Level 1 center and at least one Level 2 center on its list. Again, as many hospitals whose travel times are with a given proportion (htol) of each other may be put on a list.

4.3.1.5 Best.route (lines 752 - 828). The function best.route is given a source and a destination node as arguments and returns the shortest possible travel time between nodes using Prim’s algorithm. If the path involves an arc identified as a choke point, the path is penalized by adding the choke point’s weight (representing the mean additional travel time) to the regular weight. Once the best route has been determined, the routine build.route is called to file the ordered list of arcs in the route set.
owned by the from.node, to.node compound entity. Finally, since the calculated travel time has been based on inter-node transit times over arterial highways, the function adj.time.f is called to adjust for the time it typically takes to move from secondary and tertiary roads to major highways and back again, and this adjusted time is returned to the caller.

4.3.2 Trace and Reporting Procedures. A collection of procedures named tr.XXX provides the majority of the trace output (lines 3087 - 3344). These procedures are triggered by filing or removing an entity from a set (see lines 383 - 411 in the preamble). They do not provide any service in the model other than the trace, and so in theory could be commented out in a final, validated version. However, since trace output can be redirected to a file (or to the NUL device), and since a trace is extremely helpful in debugging, there should be no reason to remove these functions.

In addition to the trace functions there are three other reporting functions: run.report, final.report, and pt.report. Run.report (lines 3006 - 3084) is called after the completion of a run and writes summary information on that run to two files: a text file called summary.res and a formatted data file named run.res. Summary.res contains easily read summary information for a quick impression of the model's output, while run.res contains detailed
information in a record format where each run is a line, and each field in that line is a numeric data item such as the run number, duration, midpoint, number of accidents, etc. This format is easily loaded by spreadsheets and statistical software, so that more sophisticated analysis can be performed.

4.3.3 Modeling Procedures. The following section explains the flow of control in the model, followed by documentation of the procedures involved in the actual execution of the instantiated model.

4.3.3.1 Generator (lines 1591 - 1625). If this instance of the generator process is for the first run of a series, the process calls the function nsp.f to determine the time until the next event occurs. It waits until that event, and then enters a loop where it creates, initializes and activates an accident process; generates the next inter-event time and waits for that event; and then just prior to starting the next event, checks to see if the run termination criteria have been met. If they have, it cancels any scheduled clear.reds events (because this might be the last run), but saves the time remaining (the time.a attribute of clear.reds) so that a clear.reds event can be scheduled at the proper time in the following run (if any).
If this instance of generator is not the first run in a series, the previous run's value of time.v is retained, and the run will begin by activating an accident process.

4.3.3.2 Accident (lines 501 - 537). The attributes of the process accident will already have been initialized by the procedure init.accident (lines 2204 - 2258). These attributes include the location, type of event (trauma or medical), number of victims, and type of trauma (blunt or penetrating) if traumatic. Init.accident also initializes (by calling init.pt (lines 2261 - 2330) and activates any patients that it has created. Once the accident process is activated, it uses an empiric function based on average severity to compute a probability that the victims will find their own way to the nearest hospital (greater severity implies lower probability of private transport). This function is based on expressions of probability by various domain experts; no data could be found on this topic.

If private travel is chosen, the routine pvt.travel (see 4.3.3.7) is called. If not, the dispatcher is sent an ambulance request after a lag time. This request is updated with more accurate information after another lag. Once the dispatcher has been fully notified, the accident process suspends itself. It has no further work to do, but must remain in existence until all patients have been picked up by an ambulance or have otherwise left the scene, since the only way to keep from losing track of a patient is to keep
it filed in the acc.patient.set until an ambulance assumes responsibility for it.

4.3.3.3 Ambulance.run (lines 561 - 728). The ambulance run process is complex, since it must represent two different ambulances and the interaction between them. The process does some initialization and calls the utility routine travel (lines 3367 - 3377) to model travel to the accident site. It may be interrupted by the dispatcher to be recalled during this time. Once at the scene, the process may take one of two branches (lines 602 - 628 and lines 628 - 671) depending on whether the ambulance is an air or ground unit. The branches are largely similar except for their provision for inter-ambulance interaction; an air ambulance takes a patient from a ground ambulance and leaves, while a ground ambulance must consider if it needs an air ambulance, wait for one to arrive if it has been called (even if the ground unit would have been ready to roll), obtains its patient(s) via the get.patient routine, may give its patient to the helicopter ambulance and then check to see if there are more patients needing care before leaving. Ground ambulances also must handle the medical calls, since these rarely require helicopter transportation. Finally, as each ambulance leaves the scene, it calls the routine check.accident which does housekeeping on the accident process; if all the patients belonging to that accident have been taken care of, it reactivates the
accident so that it may end itself. Ambulance.run then uses the travel routine to model traveling to the hospital and returning to base, notifying the dispatcher of its status at appropriate points along the way.

4.3.3.4 Patient (lines 2646 - 2756). The patient process is used to manage the continuous simulation routines. The design is to enter a 'work continuously' statement, using the function bleed to model the patient's physiologic status, and the function done to determine when this particular phase of the patient's experience is over. The update function is used to make the continuous variables visible to allow for reporting if desired.

The logic used to handle a patient's death requires some explanation. The function living is called periodically to determine if a patient meets the criteria for death. Death before ambulance arrival ends the patient process. In any other phase, death is recorded at the time it occurs, but the patient process continues through the end of the initial resuscitation. This reflects the real world system, in which the initiation of field therapy mandates transport to a hospital and resuscitation, even after the loss of vital signs, before pronouncing a patient dead. This complication is handled by using the routine pass.time (lines 2581 - 2643) to control the continuous simulation statements, and to reexecute the 'work continuously' statement if a patient dies during continuous simulation.
Finally, if the patient survives to be delivered to definitive care, his probability of survival is calculated from his ISS and RTS, using logistic regression coefficients from the MTOS [Champion90].

4.3.3.5 Dispatcher (lines 1238 - 1391). The dispatcher routine serves to coordinate the various processes in the simulation. It consists entirely of a multiway branch (select case) statement. Only one branch is executed each time dispatcher is called. The branches correspond to the various notifications that the real world dispatcher receives, such as request for ambulance or helicopter, updates to previous requests, and reports from ambulances on their status (en route to scene, on scene, at hospital, returning to base, and need assistance). The dispatcher also manages requests that can't be satisfied by filing requesting accidents in a pending set. When an ambulance is placed back in service, either when it returns to its base, or when it is recalled from a run, the dispatcher checks the pending set to see if that ambulance can be dispatched to an accident (if any) on the pending list. Finally, the dispatcher receives notification when the last patient is taken from an accident scene; it can then recall any ambulances still en route, and reactivate the accident so it will terminate.
4.3.3.6 Get.patient (lines 1944 - 1986). The get.patient procedure updates every patient's current RTS and ensures that patients are assigned to ambulances in RTS priority. It then chooses a destination level based on the logic previously described, although this is overridden in the case of helicopter ambulances who always transport back to their own base hospitals, unless a higher level of care is required. Get.patient does some housekeeping by moving the patient from the accident to the ambulance's set, setting intravenous fluid starting times and respiratory support times for the patient, and then uses the system's rules for deciding whether to request the helicopter or not. Get.patient only assigns one patient, and so it is called repeatedly to assign multiple patients to an ambulance if their acuity levels are low enough.

4.3.3.7 Pvt.travel (lines 2892 - 2930). The routine pvt.travel handles a patient's transportation to a hospital outside of and unknown to the EMS system. In this setting patients are assumed to go to the nearest hospital, regardless of its ability to handle their injury, and regardless of its divert status. Private transport is also assumed to take longer than EMS transport, although the overall time may be shorter since there is no wait for the ambulance to arrive and no on-scene treatment.
4.3.4 Flow of Control. The flow of control in this implementation, as in the real world system, is complex. The dispatcher function is critical to understanding the implementation, since it serves as a monitor to coordinate communication between the independent entities in the model. Although not specifically implemented as such, entities such as ambulances and patients can be viewed as finite state automata, with the dispatcher functioning to oversee state transitions. A typical sequence is given in the following.

An episode begins when the generator function creates an accident, and initializes it to have some number of patients, which some constellation of injury characteristics. Each patient process is activated and enters a continuous simulation phase that models the physiologic effect of its injuries. After a brief delay, the accident notifies the dispatcher of its existence, location, an approximate number of patients. The dispatcher determines the number of ambulances that should respond, selects particular ambulances from those available, and creates and activates ambulance run processes for each. Finally, the dispatcher files the accident in the pending set if more ambulances are needed, and then exits. The dispatcher function is reinvoked upon arrival of an ambulance at the scene, or after a lag interval, whichever occurs first. At this point, information assumed to be accurate about the number of victims is sent to the
dispatcher, which may in turn recall some of the enroute ambulances, or may dispatch additional units as needed.

At this point, patient and ambulance run processes are executing simultaneously. Once an ambulance run process was worked for its designated travel time, it notifies the dispatcher that it has arrived on scene, and after a brief interval, picks up patients from the accident in order of apparent severity. Once the ambulance run "owns" some patients, it adjusts the patient's characteristics to reflect interventions such as respiratory support and intravenous fluid therapy. It also determines the appropriate destination, based on the level of care needed for the degree of injury. (One might think the dispatcher should do this, but the real world system operates in this manner). The ambulance run may also call for the helicopter ambulance to come to the scene and take over, depending on the severity of injury and distance from the appropriate level of care. The accident process terminates when the last patient is picked up by an ambulance.

After scene care has been rendered, the ambulance run notifies the dispatcher that it is enroute to a hospital. This notification also includes a request for additional help if there is still a disparity between the number of victims remaining and the number of ambulances on scene or enroute to the accident. The dispatcher updates the
ambulance run's set membership appropriately to reflect this new status.

After the hospital transport time has passed, the ambulance run notifies the dispatcher of its arrival at the hospital. The patient(s) are then transferred to the resus.set, and undergo their resuscitative care, after which they are transferred to definitive care and leave the system. After a cleanup time, the dispatcher sends the ambulance back to its base, and upon arrival there, refiles the ambulance in the ready.set and terminates the ambulance.run process. The cycle is now ready to begin anew.

4.4 Selection of Input Distributions

The distributional form of the input random variables was chosen after consideration of both theoretical and practical issues. For example, for those distributions known to be bounded, beta distributions were chosen since they were also bounded, and were then scaled and fit using moment matching or maximum likelihood methods. Similarly, if a distribution was known to be skewed to the right, or nonnegative, candidate distributions were restricted to those having the appropriate general characteristics.

For all distributions for which empirical data was available, the choice among candidate distributions was made by visually assessing probability and quantile plots [Law91], after matching the first two moments (mean and
variance) to the empirical data. For each quantile ordinate, the quantile plot graphs the abscissa of the candidate distribution corresponding to that ordinate against the abscissa of the empiric distribution corresponding to that ordinate. A good fit will thus appear as a straight line, and differences in the tails of the candidate and empirical distributions will show up as deviations from a straight line. In the probability plot, for each abscissa value, the ordinate of the cumulative distribution function of the empiric distribution is plotted against the ordinate of the candidate distribution for the same abscissa value. Again, good fit will appear as a straight line, but differences in the middle of the distributions are magnified and appear as deviations from a straight line. In addition, the slope and intercept of the plots (or the logged plots, for Weibull distributions) can be used to estimate the parameters of the fitted distributions [Wilkinson90]. Several examples of distributions fitted using these techniques are given in Appendix 5.

Finally, although several of the time duration variables were well-approximated by Weibull distributions, the final decision was made to use gamma variates instead, since the parameters of a gamma distribution are simple functions of the mean, which is not the case for the Weibull. This allows more flexibility in using the model, since mean times
can be easily changed in the data file. Inspection of the probability and quantile plots in 5 shows that this is not likely to produce large differences in the output, since gamma distributions also fit the data very well. A listing of all the random variables used in the model, the distributions and arguments chosen to generate them and the random number streams assigned is provided in Table 1.
<table>
<thead>
<tr>
<th>module</th>
<th>variable</th>
<th>distribution</th>
<th>arguments</th>
<th>stream</th>
</tr>
</thead>
<tbody>
<tr>
<td>accident</td>
<td>notification lag</td>
<td>exponential.f</td>
<td>alarm.lag</td>
<td>15</td>
</tr>
<tr>
<td>accident</td>
<td>pvt vs ambulance</td>
<td>random.f</td>
<td>n/a</td>
<td>3</td>
</tr>
<tr>
<td>accident</td>
<td>update lag</td>
<td>exponential.f</td>
<td>info.lag</td>
<td></td>
</tr>
<tr>
<td>ambulance.run</td>
<td>scene time</td>
<td>mygamma.f</td>
<td>t.on.scene,</td>
<td>4</td>
</tr>
<tr>
<td>&quot;</td>
<td>time to patient</td>
<td>exponential.f</td>
<td>t.to.pt</td>
<td>5</td>
</tr>
<tr>
<td>&quot;</td>
<td>cleanup time</td>
<td>exponential.f</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>&quot;</td>
<td>secure time</td>
<td>lognormal.f</td>
<td>m.secure,</td>
<td>18</td>
</tr>
<tr>
<td>&quot;</td>
<td>need secure time?</td>
<td>random.f</td>
<td>n/a</td>
<td>19</td>
</tr>
<tr>
<td>&quot;</td>
<td>deliver time</td>
<td>lognormal.f</td>
<td>m.deliver,</td>
<td>21</td>
</tr>
<tr>
<td>&quot;</td>
<td>choke point delay</td>
<td>exponential.f</td>
<td>choke.pt.wt</td>
<td>23</td>
</tr>
<tr>
<td>dispatcher</td>
<td>pt count error</td>
<td>uniform.f</td>
<td>-1, 1</td>
<td>7</td>
</tr>
<tr>
<td>&quot;</td>
<td>to hosp time</td>
<td>get.travel.time</td>
<td>node dependent</td>
<td>13</td>
</tr>
<tr>
<td>&quot;</td>
<td>to base time</td>
<td>&quot;</td>
<td>&quot;</td>
<td>17</td>
</tr>
<tr>
<td>find.hosp</td>
<td>hosp proportion</td>
<td>random.f</td>
<td>n/a</td>
<td>12</td>
</tr>
<tr>
<td>generator</td>
<td>inter-acc interval</td>
<td>nsp.f</td>
<td>n/a</td>
<td>1</td>
</tr>
<tr>
<td>&quot;</td>
<td>prop trauma</td>
<td>random.f</td>
<td>n/a</td>
<td>22</td>
</tr>
<tr>
<td>get.amb</td>
<td>to acc time</td>
<td>get.travel.time</td>
<td>node dependent</td>
<td>14</td>
</tr>
<tr>
<td>get.loc</td>
<td>location</td>
<td>random.f</td>
<td>n/a</td>
<td>8</td>
</tr>
<tr>
<td>get.travel.time</td>
<td>travel time</td>
<td>mygamma.f</td>
<td>trav time,</td>
<td>3</td>
</tr>
<tr>
<td>&quot;</td>
<td>travel time</td>
<td>&quot;</td>
<td>per caller</td>
<td></td>
</tr>
<tr>
<td>get.patient</td>
<td>iv.rate</td>
<td>log.normal.f</td>
<td>120 - sbp,</td>
<td>27</td>
</tr>
<tr>
<td>&quot;</td>
<td></td>
<td></td>
<td>10% cof.var</td>
<td></td>
</tr>
<tr>
<td>init.accident</td>
<td>no. victims</td>
<td>poisson.f</td>
<td>1.5</td>
<td>9</td>
</tr>
<tr>
<td>&quot;</td>
<td>blunt v pen</td>
<td>random.f</td>
<td>n/a</td>
<td>28</td>
</tr>
<tr>
<td>init.pt</td>
<td>severe brain inj</td>
<td>random.f</td>
<td>n/a</td>
<td>25</td>
</tr>
<tr>
<td>&quot;</td>
<td>hem v resp inj</td>
<td>random.f</td>
<td>n/a</td>
<td>26</td>
</tr>
<tr>
<td>&quot;</td>
<td>iss</td>
<td>mybeta</td>
<td>1.390, 9.632</td>
<td>10</td>
</tr>
<tr>
<td>patient</td>
<td>resus time</td>
<td>mygamma.f</td>
<td>t.resus,</td>
<td>11</td>
</tr>
<tr>
<td>&quot;</td>
<td>transfer time</td>
<td>exponential.f</td>
<td>t.tx</td>
<td>24</td>
</tr>
<tr>
<td>pvt.travel</td>
<td>hosp proportion</td>
<td>random.f</td>
<td>n/a</td>
<td>16</td>
</tr>
<tr>
<td>&quot;</td>
<td>travel time</td>
<td>mygamma.f</td>
<td>1.2*trav time,</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 1. Random variables used in the model.
Chapter 5
Verification and Validation

5.1 Verification

Major components of the implementation were verified against predictable model elements wherever possible. This was done by independent testing of "stub" routines where practical, and by inspection of the simulation trace or outputs elsewhere.

5.1.1 Random variate generators. Two new random number generators were implemented and verified; the non-stationary Poisson distribution routine nsp.f, and mygamma.f, a replacement for SIMSCRIPT's error-prone gamma variate generator.

5.1.1.1 Nsp.f. Çinlar's method [Çinlar75] of generating the interarrival times for a non-stationary Poisson arrival process was implemented in the function nsp.f (lines 2545 - 2578). An example test data set is given in Table 2, approximating the mean cumulative "arrivals" for a week. The nsp.f routine used this data to produce the 5000 arrival times summarized in Figure 3, Figure 4. The variation in generated arrival times closely follows the data in Table 2; a goodness of fit test shows no evidence of bad fit ($\chi^2_{13} = 13.752, P = .392$).
Table 2. Example data file of mean arrivals by interval for testing the nsp.f routine.

This process was repeated for a variety of data sets to establish acceptance of the nsp.f function.

5.1.1.2 Mygamma.f. A new gamma variate generator (lines 2442 - 2512) was implemented from two published algorithms [Bratley87]. For shape parameter greater than one, Tadikamalla's method was used, and for order one or less, Ahrens' method was used. Verification examples were produced over a wide range of arguments including those known to return invalid results for the SIMSCRIPT generator. An example of results for 2500 variates from mygamma.f given arguments 1.5, 1.5 ($\alpha = 1.5, \beta = 1.0$). This distribution has theoretical mean and variance equal to 1.5. The sample mean and variance of the output from mygamma.f was 1.517 and 1.529. Figure 5 presents a histogram of the output; the
Figure 4. Proportion arrivals in 12 hour periods over one week, corresponding to data in Table 2.

Probability plots Figure 6 demonstrate the closeness to the theoretical distributional shape.

5.1.2 Static and Dynamic Analysis. Attributes of entities in the implementation were checked to confirm that they indeed matched the input parameters and distributional form specified in the model. For example, the distribution of observed ISS scores in the model compared reasonably well to that described by Baker [Baker92], which it was designed to match (Table 3). Similarly, the proportion of blunt to penetrating injury, the spatial distribution of injuries, the number of victims per accident, and other elements were confirmed to approximately match their inputs.
Figure 5. Histogram of 2500 variates from mygamma.f given arguments 1.5, 1.5.

<table>
<thead>
<tr>
<th>statistic</th>
<th>model</th>
<th>Baker</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>1082</td>
<td>8791</td>
</tr>
<tr>
<td>mean</td>
<td>9.86</td>
<td>9.46</td>
</tr>
<tr>
<td>median</td>
<td>8.00</td>
<td>10.00</td>
</tr>
<tr>
<td>std deviation</td>
<td>7.30</td>
<td>7.18</td>
</tr>
<tr>
<td>skewness</td>
<td>1.26</td>
<td>1.71</td>
</tr>
</tbody>
</table>

Table 3. Characteristics of ISS scores obtained by the model and those reported by Baker.

The dynamic behavior of the implementation were verified to be compatible with the model by careful inspection of the
trace output and temporal outputs such as blood pressure. Special attention was paid to dispatching rules, such as alternating assignments between helicopter ambulances, or between two ambulances based in the same node. It was possible to confirm from the trace dispatched ambulances that were recalled had indeed not reached the scene. It was also confirmed that ambulances treated patients in order of
Ambulances are dispatched in order of travel time to WJX. The last ambulance that has not yet arrived (D9) is recalled when more information on numbers of patients is available.

Ambulances begin treating sicker patients (lower RTS) first. Both patients meet Level 1 criteria, so they are transported to UMC, not the nearest hospital.

St. Luke's goes on divert at 12:11. All subsequent runs which might routinely go to St. Luke's are diverted to alternate destinations until St. Luke's can began accepting incoming ambulances again at 13:11

severity as manifested by the current value of the RTS. And finally, the trace confirmed that no ambulance was dispatched to the "wrong" node or to the "wrong" hospital, and that no ambulance traveled to a hospital without carrying a patient. This method of verification can never absolutely confirm the reliability of the system, but it does serve to increase confidence that the implementation behaves according to the model's specifications. Table 4
shows a portion of the trace output that demonstrates the appearance in the implementation of several specific model behaviors.

5.2 Validation

Rigorous validation of a system such as this is extremely difficult, primarily because of the inadequacy of existing data sets useful for confirming model performance [McCoy92]. However, it was possible to compare measures of the model’s performance to locally available data elements, to establish at least order of magnitude validity. The following items had sufficient data available to allow such comparisons: number of ambulance runs, number of helicopter runs, proportion of deaths prior to definitive care, etc. The model’s predictions for these variables are compared with

<table>
<thead>
<tr>
<th>Item</th>
<th>Model</th>
<th>Fire/Rescue</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>mean daily ground runs</td>
<td>41.734</td>
<td>44</td>
<td>n/a</td>
</tr>
<tr>
<td>mean daily helo runs</td>
<td>5.07</td>
<td>5</td>
<td>n/a</td>
</tr>
<tr>
<td>prob dead on scene</td>
<td>.018</td>
<td>.01</td>
<td>n/a</td>
</tr>
<tr>
<td>mean transport time (min, Duval Co only)</td>
<td>22.1</td>
<td>20</td>
<td>n/a</td>
</tr>
<tr>
<td>prob death prior to definitive care</td>
<td>.128</td>
<td>.05</td>
<td>.085 (Baker)</td>
</tr>
<tr>
<td>mean SBP at definitive care</td>
<td>93.8</td>
<td>100</td>
<td>95.4 (Baxt)</td>
</tr>
</tbody>
</table>

Table 5. Comparison of outcome estimates produced by the model with those estimated by Jacksonville Fire/Rescue.

convenience sample estimates from Jacksonville Fire Rescue and published data in Table 5. The distribution of transit times was compared with that derived from Campbell’s data. The mean transit times were different, reflecting differing
geography, but quantile plots of the two data sets revealed that they have approximately the same differing only by a scaling factor (see Figure 7).
Figure 7. Quantile plot comparing the model's transport times with those provided by Campbell.
Chapter 6
Demonstrative Experiments

6.1 Triage Policy

To demonstrate the utility of the model in assessing policy choices, three sets of runs were performed using a different cutoff point to determine when a patient should be triaged directly to a Level 1 trauma center, bypassing other (possibly closer) hospitals. Current standard operating procedure calls for all patients with an RTS less than or equal to 90% of the maximum of 7.8408 (this corresponds to a score of approximately 10-11 on the 0-12 RTS scale) to be transported directly to a Level 1 center; this baseline case and several alternative cases were simulated. The minimum run length was set to 24 hours, and a total of 28 runs (approximately one month) were performed. The following outputs were used as measures of system performance under each scenario: trauma center utilization, red time, and reserve; proportion of accidents pended (i.e., no ambulance immediately available), mean waiting time until ambulance is dispatched among pended accidents; helicopter utilization; mean probability of death prior to receiving definitive care; unmet need (patients who met helicopter dispatch criteria but for whom a helicopter was unavailable); and total waiting time until EMS arrival ($t_a$). The classical
approach [Law91] was used to calculate point and interval estimates from the results of the 28 regeneration cycles. For this experiment, and the following one, output from the formatted data files was more useful than that directly reported by the program. An example of the direct output is provided in Appendix 4.

<table>
<thead>
<tr>
<th>RTS cutoff (%) of max</th>
<th>80</th>
<th>90</th>
<th>95</th>
</tr>
</thead>
<tbody>
<tr>
<td>trauma center</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>utilization</td>
<td>.284±.035</td>
<td>.303±.031</td>
<td>.302±.034</td>
</tr>
<tr>
<td>red</td>
<td>.0</td>
<td>.0</td>
<td>.0016±.9E-5</td>
</tr>
<tr>
<td>reserve</td>
<td>.9998±.0003</td>
<td>.9999±.7E-5</td>
<td>.9997±.0006</td>
</tr>
<tr>
<td>pr acc pended</td>
<td>.041±.017</td>
<td>.054±.020</td>
<td>.040±.015</td>
</tr>
<tr>
<td>wait (min)</td>
<td>28.5±11.5</td>
<td>22.0±8.29</td>
<td>21.1±1.53</td>
</tr>
<tr>
<td>helicopter</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>utilization</td>
<td>.111±.015</td>
<td>.136±.020</td>
<td>.149±.022</td>
</tr>
<tr>
<td>unmet need</td>
<td>.359±.056</td>
<td>.319±.045</td>
<td>.319±.045</td>
</tr>
<tr>
<td>pr death*</td>
<td>.120±.021</td>
<td>.128±.024</td>
<td>.121±.018</td>
</tr>
<tr>
<td>t.</td>
<td>21.4±1.32</td>
<td>21.7±1.35</td>
<td>21.1±1.53</td>
</tr>
</tbody>
</table>

Table 6. System performance (mean ± 95% confidence interval) under different trauma center triage criteria.

The results for the baseline case and two alternatives (triage cutoffs of 80% and 95% of maximum) are summarized in Table 6. Compared to the baseline case, the main effects of liberalizing the triage cutoff are an increase in helicopter utilization, and a decrease in the length of time that a pended accident must wait to have an ambulance assigned to it.
The three alternative models converged at six points in the simulation; results at the two convergence points spanning at least a full week cycle are shown in Table 7. After adjusting for multiple comparisons, the results show that helicopter utilization is significantly different under the 80% and 95% triage cutoffs ($P = .013$, paired t test); the 95% confidence interval on the difference in utilization between these two alternatives is $0.036 \pm 0.0014$, or about a 33% increase. Although the mean difference in waiting time for pended accidents is large between the 80% and 95% policies (12.3 minutes), the standard deviation of the difference is also large (7.81 minutes), so the results are not statistically significant. This result could be due to inadequate power since only two point estimates were obtained; further runs would be required to improve the precision of the estimate to determine if a true effect on waiting time should be expected.

<table>
<thead>
<tr>
<th>Triage cutoff</th>
<th>Helo utilization</th>
<th>Waiting time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>80 90 95</td>
<td>80 90 95</td>
</tr>
<tr>
<td>Convergence</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time (days)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13.919</td>
<td>.128 .139 .163</td>
<td>34.2 21.0 16.3</td>
</tr>
<tr>
<td>30.928</td>
<td>.100 .133 .136</td>
<td>28.0 28.3 21.2</td>
</tr>
</tbody>
</table>

Table 7. Convergence points including at least a full seven day cycle.
6.2 Helicopter Dispatch Policy

Currently, helicopter ambulances are dispatched for patients needing a level 1 center whose transport time is over 19 minutes, and patients needing a level 2 center whose transport time is over 39 minutes. The effects of reducing these times by about 50% (to 10 and 20 minutes, respectively) are shown in Table 8; the triage cutoff was kept at 90% of maximum, so these results should be compared to the center column in Table 6. It appears that the effect of liberalizing time and distance transport criteria on helicopter utilization is much greater than that of liberalizing the triage cutpoint, yet the latter has received considerably more attention.

<table>
<thead>
<tr>
<th>variable</th>
<th>mean ± 95% confidence interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>trauma center utilization</td>
<td>.298 ± .037</td>
</tr>
<tr>
<td>red</td>
<td>.000</td>
</tr>
<tr>
<td>reserve</td>
<td>.9999 ± .0001</td>
</tr>
<tr>
<td>pr acc pended</td>
<td>.042 ± .019</td>
</tr>
<tr>
<td>wait (min)</td>
<td>24.4 ± 8.86</td>
</tr>
<tr>
<td>helicopter utilization</td>
<td>.160 ± .021</td>
</tr>
<tr>
<td>unmet need</td>
<td>.293 ± .037</td>
</tr>
<tr>
<td>pr death*</td>
<td>.112 ± .020</td>
</tr>
<tr>
<td>time til arrival</td>
<td>22.2 ± 1.74</td>
</tr>
</tbody>
</table>

* prior to definitive care

Table 8. System performance (mean ± 95% confidence interval) under alternate helicopter dispatch criteria.
6.3 Conclusion

It is interesting to note that the trauma triage cutoff, which has been the subject of vehement debate at times, had little effect on the overall load on the system, while a factor that has received little attention, the retriaging of less severely injured patients to a higher level of care if such a center is reasonably "close" had a much greater impact. This leads to the conclusion that the common knowledge of domain experts may not always be helpful in predicting the response of a complex system to change, and that computer models of such systems may enhance the decision makers accuracy and reliability by adding insight into the possible responses of the system to variables that were not previously thought important.

6.4 Further Work

Concern for the validity of current disaster planning and a demonstration of the potential of this model has led to community-wide interest in using a more fully validated version of the model to assist in planning for several events of importance in northeast Florida. The particular areas of interest are:

a. Loss of a hospital and subsequent evacuation of its patients to other facilities.

b. Loss of a major "choke point" such as a bridge for a period of hours to days.
c. Widespread flooding of low areas eliminating multiple transportation routes and isolating some hospitals and nodes.

d. An area-wide disaster such as a hurricane, which might combine all of the preceding elements.

e. Modification of the physiologic model to use the a more detailed physiologic score such as ASCOT [Champion90], and to estimate the covariance structure of injuries from the American College of Surgeons National Trauma Registry Data (TRACS).
References

[Baker74]

[Baker92]

[Baxt87]

[Bratley87]

[CACI88]

[Campbell92]

[Carter74]

[Champion81]
[Champion86]

[Champion90]

[Champion91]

[Cinlar75]

[Devroye86]

[Fitzsimmons82]

[Johnson87]

[Law91]

[Lewis79]

[Lewis86]
[MacKenzie86]

[May90]

[Mazzoni88]

[McCoy92]

[Morris86]

[Uyeno84]

[Valenzuela90]

[Wears90]
[Wilkinson90]

[Zachariah92]
Appendix 1

Program Source Code

Table of Contents

<table>
<thead>
<tr>
<th>module</th>
<th>line number</th>
</tr>
</thead>
<tbody>
<tr>
<td>preamble</td>
<td>2</td>
</tr>
<tr>
<td>main</td>
<td>418</td>
</tr>
<tr>
<td>accident</td>
<td>502</td>
</tr>
<tr>
<td>adj.time.f</td>
<td>541</td>
</tr>
<tr>
<td>ambulance.run</td>
<td>562</td>
</tr>
<tr>
<td>assign.amb</td>
<td>732</td>
</tr>
<tr>
<td>best.route</td>
<td>757</td>
</tr>
<tr>
<td>bleed</td>
<td>836</td>
</tr>
<tr>
<td>build.call.list</td>
<td>889</td>
</tr>
<tr>
<td>build.hosp.list</td>
<td>980</td>
</tr>
<tr>
<td>build.route</td>
<td>1077</td>
</tr>
<tr>
<td>check.accident</td>
<td>1100</td>
</tr>
<tr>
<td>check.red</td>
<td>1133</td>
</tr>
<tr>
<td>choke.f</td>
<td>1166</td>
</tr>
<tr>
<td>clear.reds</td>
<td>1184</td>
</tr>
<tr>
<td>cts.f</td>
<td>1208</td>
</tr>
<tr>
<td>dispatcher</td>
<td>1238</td>
</tr>
<tr>
<td>done</td>
<td>1395</td>
</tr>
<tr>
<td>final.report</td>
<td>1429</td>
</tr>
<tr>
<td>find.hosp</td>
<td>1482</td>
</tr>
<tr>
<td>find.loc</td>
<td>1538</td>
</tr>
<tr>
<td>ftime.f</td>
<td>1575</td>
</tr>
<tr>
<td>generator</td>
<td>1592</td>
</tr>
<tr>
<td>get.accs</td>
<td>1629</td>
</tr>
<tr>
<td>get.amb</td>
<td>1661</td>
</tr>
<tr>
<td>get.ems</td>
<td>1744</td>
</tr>
<tr>
<td>get.hosp</td>
<td>1783</td>
</tr>
<tr>
<td>get.iv.rate</td>
<td>1815</td>
</tr>
<tr>
<td>get.level</td>
<td>1826</td>
</tr>
<tr>
<td>get.list</td>
<td>1854</td>
</tr>
<tr>
<td>get.net</td>
<td>1877</td>
</tr>
<tr>
<td>get.num.amb</td>
<td>1933</td>
</tr>
<tr>
<td>get.patient</td>
<td>1945</td>
</tr>
<tr>
<td>get.sim</td>
<td>1990</td>
</tr>
<tr>
<td>get.table</td>
<td>2088</td>
</tr>
<tr>
<td>get.travel.time</td>
<td>2116</td>
</tr>
<tr>
<td>go.green</td>
<td>2142</td>
</tr>
<tr>
<td>go.off.red</td>
<td>2160</td>
</tr>
<tr>
<td>go.red</td>
<td>2186</td>
</tr>
<tr>
<td>init.accident</td>
<td>2205</td>
</tr>
<tr>
<td>init.pt</td>
<td>2262</td>
</tr>
<tr>
<td>initialize</td>
<td>2334</td>
</tr>
<tr>
<td>lin.int.f</td>
<td>2389</td>
</tr>
<tr>
<td>living</td>
<td>2414</td>
</tr>
<tr>
<td>mybeta.f</td>
<td>2427</td>
</tr>
<tr>
<td>mygamma.f</td>
<td>2443</td>
</tr>
<tr>
<td>no.pts</td>
<td>2516</td>
</tr>
<tr>
<td>nsp.f</td>
<td>2546</td>
</tr>
<tr>
<td>pass.time</td>
<td>2582</td>
</tr>
</tbody>
</table>
patient
print.net
pt.report
pvt.travel
read.call.list
read.hosp.list
recall
resp.support
run.report
tr.check.in.acc
tr.check.in.amb
tr.check.in.pt
tr.check.out.acc
tr.check.out.pt
tr.deliver.pt
tr.enroute.hosp
tr.go.green
tr.go.red
tr.on.scene
tr.pickup.pt
tr.pvt.tr.pt
tr.resus.pt
tr.send.amb
tr.stack.acc
tr.to.home
tr.unstack.acc
travel
update
write.call.list
write.hosp.list

2647  2760
2869  2893
2934  2955
2977  3014
3027  3108
3126  3138
3149  3167
3179  3192
3212  3223
3234  3254
3274  3286
3300  3320
3338  3350
3381  3405
3426  3443
Source Code

```
preamble " Prehospital Trauma Care Model
 normally, mode is undefined

'' functions and globals
define nsp.f, lin.int.f, adj.time.f, ftime.f, get.iv.rate
as double functions
define laplace.f, mygamma.f, mybeta.f, choke.f as double functions
define get.table, get.accs as pointer functions
define done, get.num.amb, get.level, find.loc, find.hosp, get.amb,
go.red, go.green, living as integer functions
define best.route, get.travel.time as double functions
define cum.events as a double, 2-dim array
define prop.accs as a double, 2-dim array
define min.length, pts.per.acc, alarm.lag, info.lag, start.time,
t.to.pt, t.on.scene, m.secure, p.secure, t.resus, t.tx,
m.deliver, s.deliver, tr.prop, temp.dur, temp.amb, temp.mn.pend
as double variables
define no.runs, run, num.helo, temp.pts, temp.accs, temp.mn.pend,
no.deaths, no.blunt, no.pended as integer variables
define patient.counter, t.patient.counter, m.patient.counter,
tot.counter, acc.counter, med.counter, divert.counter,
override.counter, run.counter, red.limit, green.limit,
max.red.hosp, min.red, max.red, reds.per.day as integer variables
define TCRUISE, atol, htol, airspeed, range, clear.time, est.deaths,
r.est.deaths, ll.time, major.cutoff, minor.cutoff
as double variables
define min.amb, minor.time, major.time as integer variables
define nsp.tprime, nsp.last.time as double variables

'' entities and sets
permanent entities
every node has
 a node.id,
a node.name,
a lat,
a long,
owns an edge.set, a call.list, a hosp.list, a node.amb.set, and
may own a hosp.set, and
may belong to a node.set, an amb.base.set, a hosp.base.set,
a temp.set and an assignment.set
define node.id as an integer variable
define node.name as a text variable
define lat, long as double variables
'' could modify this to node, node, time.period to allow for
'' known changes in travel times by time of day
every node, node has
 a transit.time,
a flight.time, and
owns a route
```

---

1. preamble " Prehospital Trauma Care Model
2. normally, mode is undefined
3. define nsp.f, lin.int.f, adj.time.f, ftime.f, get.iv.rate
4. define laplace.f, mygamma.f, mybeta.f, choke.f
5. define get.table, get.accs
6. define done, get.num.amb, get.level, find.loc, find.hosp, get.amb,
7. go.red, go.green, living
8. define best.route, get.travel.time
9. define cum.events
10. define prop.accs
11. define min.length, pts.per.acc, alarm.lag, info.lag, start.time,
12. t.to.pt, t.on.scene, m.secure, p.secure, t.resus, t.tx,
13. m.deliver, s.deliver, tr.prop, temp.dur, temp.amb, temp.mn.pend
14. define no.runs, run, num.helo, temp.pts, temp.accs, temp.mn.pend,
15. no.deaths, no.blunt, no.pended
16. define patient.counter, t.patient.counter, m.patient.counter,
17. tot.counter, acc.counter, med.counter, divert.counter,
18. override.counter, run.counter, red.limit, green.limit,
19. max.red.hosp, min.red, max.red, reds.per.day
20. define TCRUISE, atol, htol, airspeed, range, clear.time, est.deaths,
21. r.est.deaths, ll.time, major.cutoff, minor.cutoff
22. define min.amb, minor.time, major.time
23. define nsp.tprime, nsp.last.time
24. define node.id
25. define node.name
26. define lat, long
27. define edge.set, call.list, hosp.list, node.amb.set
28. may own a hosp.set
29. may belong to node.set, amb.base.set, hosp.base.set
30. define node.id
31. define node.name
32. define lat, long
33. could modify this
34. known changes in travel times by time of day
35. every node, node
36. transit.time
37. flight.time
38. owns a route
define transit.time, flight.time as double variables

define route as a lifo set

every hospital has
   a hosp.id,
   a hosp.name,
   a no.pts,
   a capacity,
   a full,
   a red,
   a can.divert,
   a red.today,
   a red.start,
   a clear.notice,
   an r.max.pts,
   an s.max.pts,
   a gr.max.pts,
   an update.time,
   an r.accum.pts,
   a g.accum.pts,
   a g.ssq.pts,
   an r.accum.cap,
   a g.accum.cap,
   a g.ssq.cap,
   a hosp.base,
   a hosp.volume,
   "'ann ED visits in 1000s
   a level and
owns a resus.patient.set and
belongs to a hosp.set
may belong to the green.set and the red.set

define hosp.name as a text variable

define hosp.id, hosp.base, capacity, full, level, r.max.pts,
   s.max.pts, gr.max.pts, red, can.divert, red.today
   as integer variables

define hosp.volume, red.start, update.time, r.accum.pts,
   g.accum.pts, r.accum.cap, g.accum.cap, r.ssq.pts, r.ssq.cap,
   g.ssq.pts, g.ssq.cap as double variables

define no.pts as an integer variable monitored on the left

define clear.notice as a pointer variable


every ambulance has
   a type,
   an amb.id,
   an amb.name,
   an amb.run,
   an amb.base,
   a cur.location,
   a travel.time and
may belong to the ready.set, the h.ready.set,
   the out.of.service.set, the to.scene.set, the on.scene.set,
   the to.hosp.set, the at.hosp.set, the to.base.set,
   the at.base.set, the node.amb.set and the amb.set

define amb.name as a text variable

define type, amb.id, amb.base, cur.location as integer variables

define amb.run as a pointer variable

define travel.time as a double variable

temporary entities


every arc has
   a source,
   a sink,
   a weight,
a choke.pt.wt,
a ch.cum.weight,
a cum.weight,
an arc.status, and
may belong to a route, a v.set, an mdt and an edge.set
define source, sink, arc.status as integer variables
define weight, choke.pt.wt, ch.cum.weight, cum.weight as double variables
every call.item has
  an ambulance.id,
and belongs to a call.list
define ambulance.id as an integer variable
every go.item has
  a hospital.id,
a hosp.level
and belongs to a hosp.list
define hospital.id, hosp.level as integer variables
processes include generator
every ambulance.run has
  a kind,
an ambulance.id,
a run.id,
a acc,
a src,
a destination,
  "destination is a node
a hosp,
a dest.level,
a helo.coming,
  "flag
a status,
a scene.time and
owns an amb.patient.set
and may belong to an h.waiting.set
define kind, run.id, src, destination, dest.level, hosp, status,
  helo.coming as integer variables
define acc as a pointer variable
define scene.time as a double variable
every patient has
  an pt.id,
a condition,
a phase,
a change.flag,
a transp.mode,
a acc,
a hosp,
an injury.loc,
an injury.time,
a wait.time,
a scene.time,
an iv.start.time,
a transp.time,
a resus.time,
a trf.start.time,
a trf.rate,
a resp.time,
a tx.time,
a tod,
a pvt.tr.arrive,
a rescue.bp, a hosp.bp, a def.bp,
a r.cts,  " Champion trauma score components
a h.cts,
a n.cts,
a cts.f function,
a cts,  " revise tr score
a iss,  " injury severity score
a prob.surv,
a bleeding.rate, a br.0,
a blood.volume, an sbp, an O2.sat,
a rbc.mass, a hct, an O2.delivery,
a iv.start.time, an iv.rate,
a blunt,  " blunt v penetrating flag
a need.helo, a got.helo ' flags
and
belongs to the patient.set and
may belong to the pvt.tr.set, an acc.patient.set,
a resus.patient.set, and an amb.patient.set
define pt.id, condition, phase, transp.mode, r.cts, h.cts, n.cts,
iss, change.flag, blunt, injury.loc, need.helo,
got.helo as integer variables
define pvt.tr.arrive, injury.time, wait.time, resus.time, tx.time,
transp.time, br.0, iv.start.time, prob.surv, iv.rate,
trf.start.time, trf.rate, rescue.bp, hosp.bp, def.bp, tod,
resp.time, O2.sat, cts as double variables
define cts.f as a double function
define blood.volume, rbc.mass, sbp, bleeding.rate, hct, O2.delivery
as continuous double variables

' note that accident is used for medical calls as well
every accident has
a kind,
an acc.id,
a site,
a no.victims,
a needed,
a sent,
an updated,  " flag
a pended,  " flag
an acc.start.time,
an acc.arrive.time and
an acc.end.time,
a blunt and
owns an acc.patient.set, an h.waiting.set and an amb.set, and
may belong to the pending.set and
may belong to the active.set
define acc.id, site, no.victims, needed, sent, updated, pended
as integer variables
define acc.start.time, acc.arrive.time, acc.end.time
as double variables
define acc.patient.set as a set ranked by low cts

' events

event notices include clear.reds
every go.off.red has a gor.hsp
define gor.hsp as an integer variable
every resp.support has an r.pt
define r.pt as a pointer variable
the system owns
the amb.base.set, the hosp.base.set, the temp.set,
the assignment.set,
the node.set, the v.set, the mdt,
the patient.set, the pvt.tr.set,
the ready.set, the h.ready.set, the out.of.service.set,
the to.scene.set, the on.scene.set, the to.hosp.set,
the at.hosp.set, the to.base.set, the at.base.set,
the pending.set, the active.set,
the green.set and the red.set
define h.ready.set as a fifo set "helos alternate response

' convenient defines

define TRUE to mean 1
define FALSE to mean 0
define NULL to mean 0
define FOREVER to mean until 1 = 2
define calls.pending to mean n.pending.set > 0
define FREE to mean 0
define IN.USE to mean 1
define A.DUPLICATE to mean 2
define NMPD to mean 60 "nautical miles / deg latitude
define DEBUG to mean 1 = 1 "change to eliminate some output

' process status for sta.a

define WRK to mean 1
define SSPND to mean 2
define NTRPT to mean 3
define ground to mean 1
define air to mean 2
define private to mean 3
define trauma to mean 1
define medical to mean 2
define alive to mean 0
define dead to mean 1
define sbp.0 to mean 120.0
define bv.0 to mean 4900.0
define hct.0 to mean 42.0
define 02.sat.0 to mean 1.0
define croak to mean sbp.0 * hct.0 / 02.sat.0 / 4
define OPS to mean 1.6 "Lewis constants
define PSTF to mean .625
define TCONST to mean 25.0
define is.to.iv to mean 2.3 "interstitial:intravascular ratio
define MAXRTS to mean 7.8408

' messages

define req.amb to mean 1
define req.car to mean 2 "private conveyance
define to.scene to mean 3
define at.scene to mean 4
define to.hosp to mean 5
define at.hosp to mean 6
define to.base to mean 7
define at.base to mean 8
define req.help to mean 10
define new.info to mean 11
define no.pts.left to mean 12
define req.helo to mean 13
' ambulance run status
define working to mean 1
define not.working to mean 2

' pt phase
define waiting to mean 1
define scene.rx to mean 2
define en.route to mean 3
define resus to mean 4
define pvt.trav to mean 5
define done.resus to mean 6

' hospital capabilities
define level1 to mean 1
define level2 to mean 2
define level2a to mean 3 'level 2 hosp that doesn't req tc stat
define level3 to mean 4

' statistical counters

' run duration
tally g.mean.dur as the mean,  
g.var.dur as the variance, and  
g.max.dur as the maximum of tmp.dur

' pts and accidents

' only trauma pts are tracked
tally r.no.t.pts as the runwise number of t.patient.counter  
tally r.no.m.pts as the runwise number of m.patient.counter  
tally r.no.accs as the runwise number of acc.counter  
tally r.no.meds as the runwise number of med.counter  
tally r.no.deaths as the runwise number of no.deaths  
tally r.no.blunt as the runwise number of no.blunt

tally g.mean.pts as the mean,  
g.var.pts as the variance, and  
g.max.pts as the maximum of tmp.pts

tally g.mean.accs as the mean,  
g.var.accs as the variance, and  
g.max.accs as the maximum of tmp.accs

' ambulance utilization and capacity
accumulate r.idle.amb as the runwise mean of n.ready.set  
accumulate r.idle.helo as the runwise mean of n.h.ready.set

tally g.mean.amb as the mean, and  
g.var.amb as the variance of tmp.ambs

' queue for ambulance service
accumulate r.mean.pending as the runwise mean and  
r.max.pending as the runwise maximum of n.pending.set

tally g.mean.pend as the mean, and  
g.var.pend as the variance of tmp.mn.pend

tally g.mean.maxpend as the mean,  
g.max.maxpend as the maximum, and
g.var.maxpend as the variance of tmp.mx.pend

'' hospitals
accumulate r.mean.red as the runwise mean of red
tally r.no.divert as the runwise number of divert.counter
tally r.no.override as the runwise number of override.counter
tally r.no.pended as the runwise number of no.pended

'' trace routines

'' ambulances
after filing in the ready.set, call tr.check.in.amb
after filing in the h.ready.set, call tr.check.in.amb
before filing in the to.scene.set, call tr.send.amb
before filing in the on.scene.set, call tr.on.scene
before filing in the to.hosp.set, call tr.enroute.hosp
before filing in the at.hosp.set, call tr.deliver.pt
before filing in the to.base.set, call tr.to.home

'' accidents
before filing in the active.set, call tr.check.in.acc
before filing in the pending.set, call tr.stack.acc
after removing from the pending.set, call tr.unstack.acc
after removing from the active.set, call tr.check.out.acc

'' patients
before filing in the acc.patient.set, call tr.check.in.pt
after removing from the patient.set, call tr.check.out.pt
before filing in the amb.patient.set, call tr.pickup.pt
before filing in the pvt.tr.set, call tr.pvt.tr.pt
before filing in the resus.patient.set, call tr.resus.pt

'' hospitals
after filing in the red.set, call tr.go.red
after filing in the green.set, call tr.go.green

end'' of preamble

main '' <f>
'' <s> main

define mn as a double variable
open 4 for output, name is "summary.res"
use 4 for output
lines.v = 0

'' output data for statistical analysis
open 7 for output, name is "run.res"
use 7 for output
lines.v = 0
open 8 for output, name is "pt.res"
use 8 for output
lines.v = 0
use 6 for output
lines.v = 0

call initialize
'' outer loop to set up each arm of an experiment goes here
'' restore random number seeds

time.v = 0
nsp.tprime = 0
nsp.last.time = 0

for run = 1 to no.runs
do
reset runwise totals of t.patient.counter, m.patient.counter,
   acc.counter, med.counter, divert.counter, override.counter,
   n.ready.set, n.h.ready.set, n.pending.set, no.blunt,
   no.deaths and no.pended
r.est.deaths = 0.
for each hospital
do
reset runwise totals of red(hospital)
r.accum.pts(hospital) = 0
r.accum.cap(hospital) = 0
r.max.pts(hospital) = 0
loop
start.time = time.v
schedule a clear.reds in clear.time hours
activate a generator now
start simulation

'' record runwise means here to ensure independence
tmp.dur = time.v - start.time
tmp.pts = r.no.t.pts
tmp.accs = r.no.accs
tmp.ambs = 1.0 - r.idle.amb / n.ambulance 
   'utilization
tmp.mn.pend = r.mean.pending
tmp.mx.pend = r.max.pending
for each hospital
do
mn = r.accum.pts(hospital) / tmp.dur
add mn to g.accum.pts(hospital)
add mn * mn to g.ssq.pts(hospital)
mn = r.accum.cap(hospital) / tmp.dur
add mn to g.accum.cap(hospital)
add mn * mn to g.ssq.cap(hospital)
add r.max.pts(hospital) to s.max.pts(hospital)
gr.max.pts(hospital) = max.f(gr.max.pts(hospital),
   r.max.pts(hospital))
loop
call run.report

loop
call final.report '' for this arm

'' end of outer loop

close unit 4
close unit 7
close unit 8

end '' of main

process accident '' <f>18
'' <s> accident
503 ' ' create patients, assign their characteristics, and activate the ems system
504 define self, pt as pointer variables
505 define avg.cts, limit as double variables
506 self = accident
507 ' ' check pts to see if pvt tsp -- send them on
508 for each pt in acc.patient.set,
509 compute avg.cts as the avg of cts.f(pt)
510 ' ' empiric function for private transport
511 limit = .29 * ((avg.cts/7.8408)**6) + .01
512 wait 1 + exponential.f(alarm.lag - 1, 15) minutes
513 if random.f(2) >= limit or kind(self) = medical
514 call dispatcher giving NULL, req.amb, self
515 ' ' wait til accurate info is sent -- if ambulance arrives 1st,
516 ' ' it will force accurate info to be sent then
517 wait 1 + exponential.f(info.lag - 1, 3) minutes
518 call dispatcher giving NULL, new.info, self
519 updated(self) = TRUE
520 ' ' wait until all patients have been picked up
521 suspend
522 else
523 call pvt.travel(self) ' ' go by pvt transport
524 endif
525 remove self from the active.set
526 acc.end.time(self) = time.v
527 end ' ' of accident
528
529 function adj.time.f(t) ' ' <f>12
530 ' ' transit times for nodes assume cruising speed. This function adjusts
531 ' ' total travel time after Carter74 (taken from Kolesar and Walker 1974).
532 ' ' Their model assumes that a units speed gradually increases as it moves
533 ' ' progressively onto larger and larger roads, and then decreases as it
534 ' ' moves off thoroughfares to the accident scene, or the hospital.
535 define t as a double variable
536 if t <= 2 * TCRUISE ' ' never made it to cruising speed
537 t = 2 * sqrt.f(t * TCRUISE)
538 else
539 t = t + TCRUISE
540 endif
541 return with t
542 end ' ' of adj.time.f
543
544 process ambulance.run ' ' <f>
545 ' ' <s> ambulance.run
546 define this.ambulance as an integer variable
547 define self, this.accident, victim, other.run as pointer variables
define secure.time, null.time, cleanup.time, temp
    as double variables
self = ambulance.run
this.ambulance = ambulance.id(self)
this.accident = acc(self)
status(self) = working
src(self) = cur.location(this.ambulance)
'' assign random variates here to preserve synchronization
if type(this.ambulance) = air
    temp = t.on.scene / 2.0
else
    temp = t.on.scene
endif
'' critical times adjusted to keep them above minimum step size in the
'' integrator routine -- shape parameter adjusted to make final returned
'' value approx a gamma(3)
scene.time(self) = 1 + mygamma.f(temp - 1, 3 * ((temp - 1) / temp)**2, 4)
null.time = exponential.f(t.to.pt, 5)  '' Campbells time-to-pt
cleanup.time = exponential.f(3, 6)
secure.time = log.normal.f(m.secure, s.secure, 18)
'' travel to accident
call travel(cur.location(this.ambulance),
    destination(self), this.ambulance)
if status(self) = working  ''check if recalled
call dispatcher giving this.ambulance, at.scene, this.accident
'' check if scene is secure
if kind(this.accident) = trauma
    if random.f(19) < p.secure
        work secure.time minutes
    endif
endif
'' work on scene
if acc.patient.set(this.accident) is not empty or
    type(this.ambulance) = air or kind(this.accident) = medical
if type(this.ambulance) = air
    remove the first other.run from
    the h.waiting.set(this.accident)
if sta.a(other.run) = WRK
    interrupt the ambulance.run called other.run
    add time.a(other.run) to scene.time(self)
    let time.a(other.run) = 0
endif
for each victim in amb.patient.set(other.run)
do
    remove this victim from amb.patient.set(other.run)
    iv.rate(victim) = get.iv.rate(victim)
    transp.mode(victim) = type(this.ambulance)
    file this victim in amb.patient.set(self)
    dest.level(self) = min.f(dest.level(self),
        get.level(victim))
loop
    dest.level(other.run) = level3
    helo.coming(other.run) = FALSE
    resume the ambulance.run called other.run
    work scene.time(self) minutes
else  ''ground

- 73 -
if kind(this.accident) = medical
    dest.level(self) = level3 ''no categories for med pts
    work scene.time(self) minutes
else
    work null.time minutes
FOREVER
    do
        if acc.patient.set(this.accident) is not empty
            call get.patient(this.accident, self)
            if helo.coming = FALSE and
                cts(f.amb.patient.set(self)) > minor.cutoff * MAXRTS and
                acc.patient.set(this.accident) is not empty
                call get.patient(this.accident, self)
        endif
        if helo.coming(self) = FALSE
            work scene.time(self) minutes
            leave '' compiler chokes id *endif-if* is changes to *els
        endif
        '' if still wait suspend
        '' after helo arrival, its pt is gone so
        '' it loops around for another
        after helo arrival, its pt is gone so
        if helo.coming(self)
            work scene.time(self) minutes
            if helo.coming(self) = TRUE
                suspend
            endif
        endif
        '' need second if statement because acc.patient.set
        '' might have changed by now
        if acc.patient.set(this.accident) is empty
            leave '' no more pts
        endif
    loop
    endif
    call check.accident(this.accident, self)
    '' be sure you still have a pt, helo might have taken the last
    if amb.patient.set(self) is not empty or kind(self) = medical
      call dispatcher giving this.ambulance, to.hosp, this.accident
    endif
    '' travel to hosp
    if kind(self) = trauma
      for every victim in amb.patient.set(self)
        do
          phase(victim) = en.route
          hosp(victim) = hosp(self)
          iv.rate(victim) = get.iv.rate(victim)
          change.flag(victim) = TRUE
        loop
      endif
      call travel(cur.location(this.ambulance),
                  destination(self), this.ambulance)
call dispatcher giving this.ambulance, at.hosp, this.accident

' at hospital

work log.normal.f(m.deliver, s.deliver, 21) minutes

if kind(self) = trauma,
for each victim in amb.patient.set(self)
do
    phase(victim) = resus
    iv.rate(victim) = get.iv.rate(victim)
    remove this victim from amb.patient.set(self)
    file this victim in the resus.patient.set(hosp(victim))
    change.flag(victim) = TRUE
loop
endif

' cleanup
work cleanup.time minutes
remove this.ambulance from the at.hosp.set
endif
else work null.time minutes 'look around to be sure
    call check.accident(this.accident, self)
endif
endif

' after all that, return to base
call dispatcher giving this.ambulance, to.base, NULL
call travel(cur.location(this.ambulance),
    destination(self), this.ambulance)

' back at base
call dispatcher giving this.ambulance, at.base, NULL
end ' of ambulance.run

routine assign.amb(amb, accd) ' <f>12
' <s> assign.amb
    define amb as an integer variable
    define accd as a pointer variable
create an ambulance.run
    kind(ambulance.run) = kind(accd)
    ambulance.id(ambulance.run) = amb
    amb.run(amb) = ambulance.run
    acc(ambulance.run) = accd
    destination(ambulance.run) = site(accd)
    if type(amb) = air
        dest.level(ambulance.run) = level2
    else
        dest.level(ambulance.run) = level3 'any hospital
    endif
    add 1 to run.counter
    file amb in amb.set(accd)
    remove amb from the at.base.set
    file amb in the to.scene.set
    activate this ambulance.run now
end ' of assign.amb
function best.route given from.node, to.node    " <f>18
'" best.route
'" Uses Prim's algorithm to find shortest route from 'from' to 'to'.
'" uses choke pt weights as a penalty in deciding best route
756   define from.node, to.node as integer variables
757   define arc, best.arc, current.arc, check as pointer variables
758
'" drain working sets
759   for each arc in mdt
760     remove arc from mdt
761   for each node in node.set
762     remove node from node.set
763   for each arc in v.set
764     remove arc from v.set
765
'" initialize v.set
766   for each arc in edge.set(from.node)
767     do
768       file arc in v.set
769       cum.weight(arc) = weight(arc)
770       ch.cum.weight(arc) = weight(arc) + choke.pt.wt(arc)
771     loop
772
773   file from.node in node.set
774
'" process the graph to produce the minimal distance spanning tree
775 FOREVER
776   do
777     for each arc in v.set compute best.arc as the min(arc) of ch.cum.weight(arc)
778     file best.arc in mdt
779     if sink(best.arc) = to.node
780       leave
781     endif
782
783     file sink(best.arc) in node.set
784     remove best.arc from v.set
785     for each arc in edge.set(sink(best.arc)),
786       when sink(arc) is not in node.set do
787         cum.weight(arc) = weight(arc) + cum.weight(best.arc)
788         ch.cum.weight(arc) = weight(arc) + choke.pt.wt(arc) +
789           ch.cum.weight(best.arc)
790       for each check in v.set,
791         '" check of new candidates for v.set
792         with sink(check) = sink(arc)
793         find the first case       '" at most 1 case
794       if found       '" then sink(arc) is already a v.set destination
795         if ch.cum.weight(arc) < ch.cum.weight(check)
796           remove check from v.set       '" arc is better than check
797           file arc in v.set       '" for this destination
798         endif
799       else       '" arc gives a destination not already in v.set
800         file arc in v.set
801       endif
802     loop
803   loop
804
805
806
807
808
809
810
811
812
813
814
815
816
817
- 76 -
current.arc = best.arc  
until source(current.arc) = from.node  
do  
call build.route(current.arc, from.node, to.node)  
for each arc in mdt,  
   with sink(arc) = source(current.arc)  
      find the first case  
current.arc = arc  
loop  
call build.route(current.arc, from.node, to.node)  
return with adj.time.f(cum.weight(best.arc))  
end ' ' of best.route  

routine bleed(patient)  ' ' <f>12  
' ' <s> bleed  
define patient as a pointer variable  
define x, n, d, rel.loss as double variables  
if condition(patient) = dead  ' 'mark time  
d.blood.volume(patient) = 0  
d.sbp(patient) = 0  
d.bleeding.rate(patient) = 0  
d.rbc.mass(patient) = 0  
d.hct(patient) = 0  
d.02.delivery(patient) = 0  
else  
x = 2**OPS  
if iv.start.time(patient) > time.v  
d.blood.volume(patient) = -bleeding.rate(patient) +  
iv.rate(patient) *  
   (1 / (1 + is.to.iv) * (1.0 + is.to.iv *  
exp.f(-(time.v - iv.start.time(patient)) * hours.v * minutes.v /  
   TCONST)))  
else  
d.blood.volume(patient) = -bleeding.rate(patient)  
endif  
' ' derivative of the Lewis equation  
rel.loss = 1 - blood.volume(patient) / bv.O  
n = (x * abs.f(rel.loss)**0.6)  
if rel.loss >= .5  
   list pt.id(patient), blood.volume(patient),  
d.blood.volume(patient), bleeding.rate(patient),  
d.bleeding.rate(patient), sbp(patient), d.sbp(patient)  
   trace  
endif  
d = (1 - sign.f(rel.loss) * x * abs.f(rel.loss)**1.6) ** .375  
d.sbp(patient) = (sbp.O / bv.O) * (n / d) * d.blood.volume(patient)  
d.bleeding.rate(patient) = br.O / sbp.O * d.sbp(patient)  
d.rbc.mass(patient) = -bleeding.rate(patient) * .01 * hct(patient)  
d.hct(patient) = 100 * (blood.volume(patient) * d.rbc.mass(patient) -  
rbc.mass(patient) * d.blood.volume(patient)) /  
   (blood.volume(patient)**2)  
d.02.delivery(patient) = 02.sat(patient) *
(sbp(patient) * d.hct(patient) + hct(patient) * d.sbp(patient))

endif
end ' of bleed

routine build.call.list ' ' <f>12
' <s> build.call.list
' builds a default call list from the network structure based on
' closest travel times -- each call list will contain at least min.amb
' entries, and may contain more if they are within atol travel time of
' the shortest travel time
' assumes all helos go to all nodes
define i, there, here, count, ncount, amb as integer variables
define limit, tt as double variables
for each node called here
do
count = 0
for each node in the amb.base.set
' copy the amb.base.set
file this node in the temp.set
FOREVER
do
for each node called i, with i in the temp.set
do
compute there as the minimum (i) of transit.time(here, i)
loop
remove there from the temp.set
file there in the assignment.set
.tt = transit.time(here, there)
' get all other nodes same distance away
for each node called i, with i in the temp.set and
transit.time(here, i) = tt
do
remove i from the temp.set
file i in the assignment.set
loop
ncount = 0
if count = 0
limit = (1 + atol) * tt
endif
for each node called i, with i in the assignment.set
do
for each amb in the node.amb.set(i) with
  type(amb) = ground
  do
  add 1 to count
  add 1 to ncount
  loop
loop
if tt > limit
  if count - min.amb >= ncount
  leave
endif
endif
for each node called i, with i in the assignment.set
do
remove i from the assignment.set
for each amb in node.amb.set(i) with type(amb) = ground

- 78 -
do
create a call.item
    ambulance.id(call.item) = amb
    '' be sure local ambulance always called first
    if here = i
        file call.item first in the call.list(here)
    else
        file call.item last in the call.list(here)
    endif
loop
loop
if n.temp.set = 0
    leave
endif
for each node in the assignment.set
    remove this node from the assignment.set
loop
for each node in the temp.set
    remove this node from the temp.set
for each node in the assignment.set
    remove this node from the assignment.set

'' add helo to every call list (default)
for each amb in the h.ready.set
    do
        create a call.item
        ambulance.id(call.item) = amb
        file call.item in the call.list(here)
    loop
end'' of build.call.list

routine build.hosp.list   '' <f>12
    '' <s> build.hosp.list
    '' builds a default hospital list from the network structure based on
    '' closest travel times -- each call list will contain at least one entry
    '' for each level of care, and may contain multiple entries at the same level
    '' if they are within atol travel time of the shortest travel time for that
    '' level
    define i, j, lvl, there, here, hosp, tot as integer variables
    define count, first, finished as integer 1-dim arrays
    define t as a double 1-dim array
    define tt as a double variable
    define NUMLEVELS to mean 3
    reserve count, t, first, finished as NUMLEVELS
for each node called here
    do
        for i = 1 to NUMLEVELS
            do
                count(i) = 0
                t(i) = 0.0
                first(i) = TRUE
                finished(i) = FALSE
            loop
        for each node in the hosp.base.set
            ''copy the amb.base.set
            file this node in the temp.set
FOREVER
do for each node called j, with j in the temp.set
do compute there as the minimum (j) of transit.time(here, j)
loop remove there from the temp.set
if first(3) = TRUE
first(3) = FALSE
'' set max time to go to any level3
 t(3) = transit.time(here, there) * (1 + htol)
 finished(3) = TRUE ''don't care if we get any level3s
endif
for each hosp in the hosp.set(there)
do tt = transit.time(here, there)
if level(hosp) >= level2a ''treat 2a like level2
 lvl=level(hosp) - 1
else
 lvl=level(hosp)
endif
if first(lvl) = TRUE
 first(lvl) = FALSE
 t(lvl) = tt * (1 + htol)
add 1 to count(lvl)
create a go.item
hospital.id(go.item) = hosp
hospital.level(go.item) = level(hosp)
if here = there
 file go.item first in the hosp.list(here)
else
 file go.item last in the hosp.list(here)
endif
else
if tt > t(lvl)
 finished(lvl) = TRUE
else
 add 1 to count(lvl)
create a go.item
hospital.id(go.item) = hosp
hospital.level(go.item) = level(hosp)
if here = there
 file go.item first in the hosp.list(here)
else
 file go.item last in the hosp.list(here)
endif
endif
else
 if tt > t(lvl)
 finished(lvl) = TRUE
else
 add 1 to count(lvl)
create a go.item
hospital.id(go.item) = hosp
hospital.level(go.item) = level(hosp)
if here = there
 file go.item first in the hosp.list(here)
else
 file go.item last in the hosp.list(here)
endif
endif
loop
for i = 1 to NUMLEVELS
compute tot as the sum of finished(i)
if tot = NUMLEVELS
 leave '' found at least 1 hosp for level 1 & 2
endif
if n.temp.set = 0 '' no more candidate hospitals
 leave
endif
loop
for each node in the temp.set
remove this node from the temp.set
loop
1070 release first, finished, t, count
1071 end ' ' of build.hosp.list
1072
1073 routine build.route given arc.id, from.node, to.node   '<f>18
1074 ' ' <s> build.route
1075 ' ' an arc can appear in at most 1 set named route
1076 define arc.id, from.node, to.node as integer variables
1077
1078 if arc.status(arc.id) = FREE   ' ' if arc.id is already in
1079   arc.status(arc.id) = IN USE   ' ' a route, a duplicate
1080   file arc.id in route(from.node, to.node)   ' ' is created
1081 else
1082   create an arc
1083     source(arc) = source(arc.id)
1084     sink(arc) = sink(arc.id)
1085     weight(arc) = weight(arc.id)
1086     choke.pt.wt(arc) = choke.pt.wt(arc.id)
1087     arc.status(arc) = A.DUPLICATE    ' ' marks arc as a duplicate
1088     file arc in route(from.node, to.node)
1089 endif
1090
1091 ' ' do we need to mark duplicates anymore?
1092 end ' ' of build.route
1093
1094 routine check.accident(this.acc, this.run)   '<f>12
1095 ' ' <s> check.accident
1096 ' ' this function ensures that only the last ambulance to leave the
1097   scene destroys the accident process
1098 define this.acc as a pointer variable
1099 define this.run as a pointer variable
1100 define amb as an integer variable
1101 amb = ambulance.id(this.run)
1102 remove amb from the on.scene.set
1103 remove amb from the amb.set(this.acc)
1104 if acc.patient.set(this.acc) is empty
1105   if h.waiting.set(this.acc) is empty
1106     acc(this.run) = NULL
1107     call dispatcher giving ambulance.id(this.run), no.pts.left,
1108     this.acc)
1109 endif
1110 else   ' ' more pts left
1111 if n.amb.set(this.acc) = 0
1112   ' ' if no one else is coming, call for help
1113     call dispatcher giving ambulance.id(this.run), req.help, this.acc
1114   ' ' this unit leaves now -- unmodelled engine co., police, etc
1115   ' ' remain on scene
1116 endif
1117 end
1118
1119 end ' ' of check.accident
1120
1121 routine check.red given hsp and n   '<f>15
1133  ' ' check.red
1134  ' ' logic to place or remove hsp on divert status
1135  define hsp, n as integer variables
1136  define cn as a pointer variable
1137
1138  if hsp is in the green.set
1139    if go.red(hsp, n) = TRUE
1140      remove hsp from the green.set
1141      file hsp in the red.set
1142      red(hsp) = TRUE
1143      add 1 to red.today(hsp)
1144      red.start(hsp) = time.v
1145      create a go.off.red called cn
1146      clear.notice(hsp) = cn
1147      schedule the go.off.red called cn giving hsp in min.red hours
1148    endif
1149  else
1150    if (time.v - red.start(hsp)) * hours.v > min.red
1151      if go.green(hsp, n) = TRUE
1152        cancel the go.off.red called clear.notice(hsp)
1153        clear.notice(hsp) = NULL
1154        remove hsp from the red.set
1155        file hsp in the green.set
1156        red(hsp) = FALSE
1157    endif
1158  endif
1159  endif
1160
1161  end ' ' of check.red
1162
1163  function choke.f(here, there)  ' ' <f>12
1164  ' ' <s> choke.f
1165
1166  define here, there as integer variables
1167  define result as a double variable
1168  define a as a pointer variable
1169  result = 0.0
1170  for each a in route(here, there), with choke.pt.wt(a) > 0.0
1171    add exponential.f(choke.pt.wt(a), 23) to result
1172  return with result
1173  end ' ' of choke.f
1174
1175  event clear.reds  ' ' <f>12
1176  ' ' <s> clear.reds
1177  ' ' clears all red status hospitals unconditionally
1178  define i as an integer variable
1179  for i = 1 to n.hospital
1180    red.today(i) = FALSE
1181  for each i in the red.set
1182    remove i from the red.set
1183    file i in the green.set
1184    red(i) = FALSE
1185  end
if clear.notice(i) <> NULL
    cancel the go.off.red called clear.notice(i)
clear.notice(i) = NULL
endif

loop
    schedule a clear.reds in 24.0 hours
return
end  

function cts.f(pt)  '  '<f>12
  '  '<s> cts.f
define pt as a pointer variable
if sbp(pt) > 89
    h.cts(pt) = 4
else
    if sbp(pt) > 75
        h.cts(pt) = 3
    else
        if sbp(pt) > 49
            h.cts(pt) = 2
        else
            if sbp(pt) > 30
                h.cts(pt) = 1
            else
                h.cts(pt) = 0
            endif
        endif
    endif
endif
endif

'  ' coeff from Champion HR, et al. J Trauma 89; 29:623
cts(pt) = 0.2908 * r.cts(pt) + .7326 * h.cts(pt) + .9368 * n.cts(pt)
return with cts(pt)
end  

routine dispatcher given caller, message, this.accident  '  '<f>
  '  '<s> dispatcher
  '  ' serves as a monitor in this program
define res, caller and message as integer variables
define this.accident, pt as pointer variables
define hsp, num as integer variables
select case message
case req.amb  '  ' caller is not an ambulance
    if kind(this.accident) = medical
        needed(this.accident) = 1
    else
        num = int.f(no.victims(this.accident) + uniform.f(-1, 1, 7) + .5) +
    needed(this.accident) = get.num.amb(num)
endf
disp
'disp
'  ' find nearest ambulance(s)
res = get.amb(this.accident, needed(this.accident), ground)
case req.help  " caller is ambulance requesting more units
    if DEBUG
        write amb.name(caller), acc.id(this.accident),
        hour.f(time.v), minute.f(time.v) as
        " \t 3, " req assist at acc ", i 5, " at ",
        i 2, ":", i 2, /
    endif
    needed(this.accident) =
    get.num.amb(n.acc.patient.set(this.accident))
    res = get.amb(this.accident, needed(this.accident), ground)

case req.helo
    if DEBUG
        write amb.name(caller), acc.id(this.accident),
        hour.f(time.v), minute.f(time.v) as
        " \t 3, " req helo at acc ", i 5, " at ",
        i 2, ":", i 2, /
    endif
    res = get.amb(this.accident, 1, air)
    if res = air
        helo.coming(amb.run(caller)) = TRUE
        for each pt in amb.patient.set(amb.run(caller))
            got.helo(pt) = TRUE
            file amb.run(caller) in the h.waiting.set(this.accident)
            " do not check to see if can recall a ground unit now
            " 1st unit will be tied up til helo arrives, and addl units
            " may be used to remove lower level pts
        else
            " mark these pts as wanting a helo but cant get one
            for each pt in amb.patient.set(amb.run(caller))
                got.helo(pt) = FALSE
        endif
    endif

case new.info  " 1st responder is updating w/ accurate info
    if DEBUG
        write acc.id(this.accident), hour.f(time.v), minute.f(time.v)
        as
        "new info fr event ", i 5, " at ", i 2, ":", i 2, /
    endif
    if kind(this.accident) = medical
        needed(this.accident) = 1
    else
        needed(this.accident) =
        get.num.amb(n.acc.patient.set(this.accident))
    endif

    " if too many ambulances have been sent
    if needed(this.accident) < sent(this.accident)
        call recall(sent(this.accident) - needed(this.accident),
        this.accident)
    endif

    " didn't send enough to begin with
    if needed(this.accident) > sent(this.accident)
        res = get.amb(this.accident,
        needed(this.accident) - sent(this.accident), ground)
    endif

    " if just right, be sure acc isn't in pending set
    if needed(this.accident) = sent(this.accident) and
    this.accident is in the pending.set
        remove this.accident from the pending.set
case at.scene
    remove caller from the to.scene.set
    file caller in the on.scene.set
    '' if this ambulance is the first to arrive
    if acc.arrive.time(this.accident) = 0
        acc.arrive.time(this.accident) = time.v
        '' if update hasn't been sent yet, force it to be
        '' sent now
        if updated(this.accident) = FALSE
            interrupt the accident called this.accident
            time.a(this.accident) = 0
            resume the accident called this.accident
    endif
endif

case no.pts.left
    '' reactivate the accident, allowing it to end
    '' a DESTROY may not free allocated memory, per CACI
    if this.accident is in the pending.set
        remove this.accident from the pending.set
    endif
    if amb.set(this.accident) is not empty
        call recall(n.amb.set(this.accident), this.accident)
    endif
    '' cant use an else here, because the recall function might
    '' cause the amb.set to become empty
    if amb.set(this.accident) is empty
        resume the accident called this.accident
    endif

case to.hosp
    '' find receiving hosp
    hsp = find.hosp(caller)
    travel.time(caller) = get.travel.time(cur.location(caller),
                         hosp.base(hsp), type(caller), 13)
    hosp(amb.run(caller)) = hsp
    file caller in the to.hosp.set

case at.hosp
    remove caller from the to.hosp.set
    file caller in the at.hosp.set

case to.base
    file caller in the to.base.set
    destination(amb.run(caller)) = amb.base(caller)
    travel.time(caller) = get.travel.time(cur.location(caller),
                         amb.base(caller), type(caller), 17)

case at.base
    remove caller from the to.base.set
    file caller in the at.base.set
    if type(caller) = ground
        file caller in the ready.set
        if calls.pending
            remove the first accident from the pending.set
            res = get.amb(accident, 1, ground) '' should be only 1 in
        endif
    else
        file caller in the h.ready.set
    endif
    endif
default
  print 1 line with message thus
  Invalid message ***
endselect
end ' ' of dispatcher

function done(patient)  ' ' <f>12
  ' ' <s> done
  define patient as a pointer variable
  if condition(patient) = alive  ' ' if just now died
    if living(patient) = FALSE
      return with TRUE
    endif
  endif
  if phase(patient) = pvt.trav
    if time.v > pvt.tr.arrive
      return with TRUE
    endif
  endif
  if phase(patient) = resus
    if time.v >= injury.time(patient) + (wait.time(patient) +
      transp.time(patient) + scene.time(patient) +
      resus.time(patient)) / hours.v / minutes.v
      return with TRUE
    endif
  endif
  if change.flag(patient) = TRUE
    return with TRUE
  endif
  return with FALSE
end ' ' of done

routine final.report  ' ' <f>18
  ' ' <s> final.report
  define correction as a double variable
  use 4 for output
  correction = no.runs / (no.runs - 1)
  print 8 lines with no.runs, min.length,
  g.mean.dur, sqrt.f(g.var.dur * correction), g.max.dur,
  g.mean.accts, sqrt.f(g.var.accts * correction), g.max.accts,
  100 * no.blunt / (no.runs * g.mean.accts),
  g.mean.pts, sqrt.f(g.var.pts * correction), g.max.pts,
  no.deaths / no.runs and est.deaths / no.runs
  thus
  Results after ** runs of at least **.** days per run
  average  sd (of runwise means)  max
  duration  ***.***  ***.***  ***.***
  no. accidents  ***.***  ***.***  ***
  % blunt  ***.***
no. tr. patients    ***.*    ***.*    ***
no. deaths         ***.*
no. est deaths     ***.*

print 4 lines with g.mean.amb, sqrt.f(g.var.amb * correction),
g.mean.pend, sqrt.f(g.var.pend * correction), g.mean.maxpend,
g.max.maxpend thus

amb. util          *.***    *.***
amb. queue         *.***    *.***    mean ***.*
glob ***

print 3 lines thus
hospital utilization
name    load   max    reserve
avg    sd    avg    global    avg    sd
for each hospital
do
print 1 line with hosp.name(hospital),
g.accum.pts(hospital) / no.runs,
sqrt.f(g.ssq.pts(hospital) - (g.accum.pts(hospital) *
g.accum.pts(hospital) / no.runs)) / (no.runs - 1),
s.max.pts(hospital) / no.runs, gr.max.pts(hospital),
1 - g.accum.cap(hospital) / no.runs,
sqrt.f(g.ssq.cap(hospital) - (g.accum.cap(hospital) *
g.accum.cap(hospital) / no.runs)) / (no.runs - 1))
thus
***    **.*    **.*    ***    ***    ***    *.***    *.****
loop
use 6 for output
end ' ' of final.report

function find.hosp(caller)    '' <f>12
'',    ''<s> find.hosp
'' chooses a destination hospital for the ambulance caller
define caller as an integer variable
declare item as a pointer variable
declare limit as a double variable
declare hsp, got.it, cumtot, rtot, first.choice as integer variables
define limit; random.f(12)
got.it = FALSE
first.choice = NULL
for each item in hosp.list(cur.location(caller)),
with hosp.level(item) <= dest.level(amb.run(caller)) and
hospital.id(item) in the green.set
compute cumtot as the sum of hosp.volume(hospital.id(item))
for each item in hosp.list(cur.location(caller)),
with hosp.level(item) <= dest.level(amb.run(caller))
do
if first.choice = NULL
first.choice = hospital.id(item)
endif
if hospital.id(item) is in the red.set
add 1 to divert.counter
if DEBUG
write amb.name(caller), hosp.name(hospital.id(item)),
hour.f(time.v), minute.f(time.v) as
1511 
1512     t 3, " diverted from ", t 3, " at ", t 2, " ", t 2, /
1513     endif
1514     cycle      'look for another
1515     hsp = hospital.id(item)
1516     add hosp.volume(hsp) to rtot
1517     if rtot / cumtot >= limit
1518         got.it = TRUE
1519         leave
1520     endif
1521     loop
1522     if got.it = FALSE      'app hsp is red & no alternate, so override
1523         add 1 to override.counter
1524         hsp = first.choice
1525         if DEBUG
1526             write amb.name(caller), hour.f(time.v), minute.f(time.v) as
1527                 t 3, " overrides divert at ", t 2, " ", t 2, /
1528         endif
1529     endif
1530     return with hsp
1531     end ' of find.hosp
1532
1533 function find.loc(amb)      '<f>12
1534     define amb as an integer variable
1535     define prop, dist, x, y as double variables
1536     define link as a pointer variable
1537     define ans as an integer variable
1538     prop = 1.0 - time.a(amb.run(amb)) / travel.time(amb)
1539     dist = 0.0
1540     x = 0.0
1541     for every link in route(src(amb.run(amb)),
1542         destination(amb.run(amb)))
1543         add weight(link) to dist
1544     let dist = dist * prop     'distance traveled so far
1545     for every link in route(src(amb.run(amb)),
1546         destination(amb.run(amb)))
1547         do
1548             y = weight(link) / 2.0
1549             add y to x
1550             if x > dist
1551                 ans = src(amb.run(amb))
1552                 leave
1553             endif
1554             add y to x
1555             if x > dist
1556                 ans = destination(amb.run(amb))
1557                 leave
1558             endif
1559         loop
1560     return with ans
1561     end ' of find.loc
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573

- 88 -
function ftime.f(f, t)
   "<f>12
   "<s> ftime.f
   define f, t as integer variables
   define temp1, temp2, av.lat as double variables
   temp1 = (lat(f) - lat(t)) * NMPD
   av.lat = (lat(f) + lat(f)) / 2
   temp2 = (long(f) - long(t)) * cos.f(av.lat) * NMPD
   return with ll.time +
      sqrt.f(temp1*temp1 + temp2*temp2) / airspeed * minutes.v)
end " of ftime.f

process generator "<f>15
   "<s> generator
   " system starts empty and idle, runs min.length days and stops when it
   " is next empty and idle
   " note use of weekday.f instead of day.f -- if desire to model
   " seasonality, etc, must call origin.r with a starting date, and
   " then use day.f throughout
   write run, weekday.f(time.v), hour.f(time.v), minute.f(time.v) as
      "run ", i 5 , " starts at day ", i 1 , " at ", i 2 , " : ", i 2 , /
   if run = 1
      wait nsp.f(cum.events(*, *), 1) hours
   endif
   FOREVER
      do if (time.v - start.time) > min.length and
         r.no.accs > 0 and
         the patient.set is empty and
         n.ambulance = n.ready.set + n.h.ready.set and
         the red.set is empty
            leave
      endif
      create an accident
      add 1 to tot.counter
      acc.id(accident) = tot.counter
      call init.accident(accident)
      activate accident now
      wait nsp.f(cum.events(*, *), 1) hours
      loop
         clear.time = (time.a(clear.reds) - time.v) * hours.v
         cancel clear.reds
      end " of generator
   "<f>12
   "<s> get.accs
   " creates table representing the proportion of
   " accidents occuring at each node
   define table as a double, 2-dim array
   define num, cum as double variables
   define i as an integer variable
   reserve table(*, *) as n.node by 2
open 3 for input, name is "space.dat"
use 3 for input
cum = 0.0
for i = 1 to n.node
  do
    read num                '' number of events in that node
    add num to cum          '' per some unit of time
    table(i, 2) = cum       '' as a cumulative distribution
  loop
close unit 3
for i = 1 to n.node          '' normalize the table
  do
    table(i, 2) = table(i, 2) / cum
  loop
return with table(*, *)
end '' of get.accs
function get.amb given acc, num, and knd              '' <f>18
  '' <s> get.amb
  '' finds nearest ambulance(s)
  '' if can't dispatch no. of ground ambulances requested, the accident is
  '' placed on the pending list.
  '' returns type of ambulance sent for notification of caller
  define acc as a pointer variable
  define num, knd, amb, amb1 as integer variables
  define item, iteml as pointer variables
  define result, target, b, found.count as integer variables
  target = site(acc)
  for each item in call.list(target), with
    (ambulance.id(item) in the ready.set or
    ambulance.id(item) in the h.ready.set)
    and type(ambulance.id(item)) = knd
  do
    amb = ambulance.id(item)
    select case knd
    case ground
      ''handle the rare case of two ambs based at one node
      if n.node.amb.set > 1
        b = amb.base(amb)
        for each ambl in the ready.set,
          with amb.base(ambl) = b
            and type(ambl) = knd  ''guaranteed to have 1
          do
            ''first amb in the ready set that is in the call list
            ''is the next up for a run
            for each iteml in call.list(target), with
              ambulance.id(iteml) = ambl
            find the first case
            if found
              leave
            endif
          loop
        item = iteml
      endif
amb = amb1
endif
remove amb from the ready.set

case air
  'helos alternate call, but are available to all nodes
  if ambulance.id(item) = f.h.ready.set
    remove ambulance.id(item) from the h.ready.set
  else
    cycle
  endif
endselct
tavel.time(amb) = get.travel.time(target, cur.location(amb),
type(amb), 14)
call assign.amb(amb, acc)
add 1 to sent(acc)
add 1 to found.count
if found.count = num
  leave
  endif
endif
loop
if found.count < num
  result = NULL
else
  result = kn0
endif
if kn0 = ground
  if result = NULL and acc is not in the pending.set
    file acc in the pending.set
    if pended(acc) = FALSE
      pended(acc) = TRUE
      add 1 to no.pended
    endif
  endif
endif
return with result
end
" of get.amb

routine get.ems
  ' <f>12
  get.ems
  ' ambulance capability ignored -- no explicit modeling of
  ' first responders -- amb's are assumed to be ACLS-paramedic level
  define i as an integer variable
  open 3 for input, name is "amb.dat"
  use 3 for input
  read n.ambulance
  start new record
  create each ambulance
  i = 1
  for each ambulance
    do
      amb.id(ambulance) = i
      read type(ambulance), amb.base(ambulance), amb.name(ambulance)
      start new record
      cur.location(ambulance) = amb.base(ambulance)
      file this ambulance in the node.amb.set(amb.base(ambulance))
if amb.base(ambulance) is not in the amb.base.set
    file amb.base(ambulance) in the amb.base.set
endif
if type(ambulance) = ground
    file this ambulance in the ready.set
endif
if type(ambulance) = air
    file this ambulance in the h.ready.set
    add 1 to num.helo
endif
file this ambulance in the at.base.set
add 1 to i
loop
end ' of get.ems

close unit 3
end ' of get.ems

routine get.hosp ' '<f>12
    ' '<s> get.hosp
    define i as an integer variable
    open 3 for input, name is "hosp.dat"
    use 3 for input
    read n.hospital
    start new record
    create each hospital
    i = 1
    for each hospital
do
    hosp.id(hospital) = i
    read hosp.name(hospital), hosp.base(hospital), hosp.volume(hospital),
    level(hospital), capacity(hospital), can.divert(hospital)
    start new record
    file this hospital in the green.set
    file this hospital in hosp.set(hosp.base(hospital))
    if hosp.base(hospital) is not in the hosp.base.set
        file hosp.base(hospital) in the hosp.base.set
    endif
    full(hospital) = FALSE
    red(hospital) = FALSE
    red.today(hospital) = FALSE
    add 1 to i
loop
    close unit 3
end ' of get.hosp

function get.iv.rate(pt) ' '<f>12
    ' '<s> get.iv.rate
    define pt as a pointer variable
    return with (10 + log.normal.f(2.0 * max.f(.5, sbp.0 - sbp(pt)),
    .2 * max.f(.5, sbp.0 - sbp(pt)), 27)) * hours.v * minutes.v
end ' of get.iv.rate

function get.level(pt) ' '<f>12
Logic to assign patient to a level of care based SOLELY on pt characteristics. Dispatcher may choose to modify based on travel time, load, availability, etc.

Define pt as a pointer variable
Define lvl as an integer variable

This logic may (should) be modified later, and should be modifiable at runtime w/o recompilation.

If cts(pt) ≤ major.cutoff
lvl = level1
Else
  If cts(pt) > minor.cutoff
    lvl = level3
  Else
    lvl = level2a
  Endif
Endif

Return with lvl

End " of get.level

Routine get.list given fname, readprog, buildprog, writeprog

If no call list is provided from a file, builds a default call list for ground ambulance based on transit times, and writes results to a file for subsequent editing as needed.

Define fname as a text variable
Define readprog, buildprog, writeprog as subprogram variables

If call list file exists, read it in
Open unit 3 for input, name is fname, noerror
Use 3 for input
If roopenerr.v eq FALSE
  Call readprog
Else 'Else build default call list of at least min.amb
  Close unit 3
  Call buildprog
  Call writeprog
Endif
End if
End " of get.call.list

Routine get.net

Define from.node, to.node as integer variables
Define temp as a double variable
Open 3 for input, name is "net.dat" 'transportation network
Use 3 for input
Read n.node
Start new record
Create each node
For each node
do
    read node.id(node), node.name(node), lat(node), long(node)
    start new record
    read airspeed, range
    start new record
    while data is not ended
        do
            create an arc
            read source(arc), sink(arc), weight(arc), choke.pt.wt(arc)
            file arc in the edge.set(source(arc))
        loop
    end
    close unit 3
    '' calculate transit times
    for each node called from.node
        do
            for each node called to.node
                do
                    if from.node = to.node '' calc intranode transit time
                        temp = 0.0
                        for each arc in edge.set(from.node)
                            do
                                add weight(arc) to temp
                            loop
                        temp = 0.5 * temp / n.edge.set(from.node)
                        transit.time(from.node, from.node) = adj.time.f(temp)
                        flight.time(from.node, from.node) = ll.time +
                            (temp / airspeed * minutes.v)
                    else
                        transit.time(from.node, to.node) =
                            best.route(from.node, to.node)
                        flight.time(from.node, to.node) = ftime.f(from.node, to.node)
                    endif
                loop
            end
        end
    end '' of get.net

function get.num.amb(n)  '' <f>12
    '' <= get.num.amb
    '' logic to determine the no. of ambulances to be dispatched
    '' based on est no. of patients, their severity, number dispatched already
    define n as an integer variable
    return with (max.f(l, int.f(n / 2 + .5)))
end '' of get.num.amb

routine get.patient(acc, amb.run)  '' <f>12
    '' <= get.patient
    define acc, amb.run as pointer variables
    define v, victim as a pointer variable
    define temp, poss.dest as integer variables
    define t as a double variable
for each v in the acc.patient.set(acc)
do
    compute victim as the min(v) of cts.f(v)
    " this should update every pts cts
loop
remove victim from acc.patient.set(acc)
phase(victim) = scene.rx
transp.mode(victim) = type(ambulance.id(amb.run))
file victim in amb.patient.set(amb.run)
temp = get.level(victim)
dest.level(amb.run) = min.f(dest.level(amb.run), temp)
iv.start.time(victim) = time.v + (scene.time(amb.run) / 
    hours.v / minutes.v) / 2 resp.time(victim) = scene.time(amb.run) / 3 iv.rate(victim) = get.iv.rate(victim)
"' should helo be requested? if so, pt is still assigned to
' amb for scene rx, but flag is set to make him wait for helo arrival
poss.dest = find.hosp(ambulance.id(amb.run))
if temp < level3
    if temp = level1
        t = major.time
    else
        t = minor.time
    endif
    if transit.time(site(acc), hosp.base(poss.dest)) > t
        need.helo(victim) = TRUE
        call dispatcher giving ambulance.id(amb.run), req.helo, acc
    endif
    change.flag(victim) = TRUE
end
"' of get.patient

routine get.sim  " <f>12
' " <s> get.sim

define i, no.streams as integer variables
open 3 for input, name is "ems.dat"
use 3 for input
read no.runs
start new record
read min.length  'min run length in days
start new record
read tr.prop  'trauma proportion
start new record
read pts.per.acc
start new record
read alarm.lag  'initial alarm latency in minutes
start new record
read info.lag  'delay til first responder reports
start new record
read m.secure, s.secure 'mean & sd secure time
start new record
read p.secure  'prob of needing secure time
start new record
read t.to.pt  'time til patient
start new record

- 95 -
read t.on.scene "scene treatment
start new record
read m.deliver, s.deliver "meand & sd deliver time
start new record
read t.resus "mean resus time in minutes
start new record
read t.tx "tx to def care time
start new record
read TCRUISE "time in minutes til reach cruising speed
start new record
read min.amb "minimum # of amb's on a call list
start new record
read atol "proportional tolerance in dist to ambs
start new record
read htol "proportional tolerance in dist to hospitals
start new record
read minor.cutoff
minor.cutoff = MAXRTS
start new record
read major.cutoff
major.cutoff = MAXRTS
start new record
read minor.time "+> minor.time, call helo
start new record
read major.time "+> major.time, call helo
start new record
read ll.time "mean launch + land time for helo
start new record
read red.limit, green.limit, max.red.hosp
start new record
read reds.per.day 'min & max red.time, no. of reds/day
start new record
read clear.time "time of day for clearing red.status
start new record
open 3 for input, name is "cont.dat" 'continuous simulation control
use 3 for input
read max.step.v
start new record
read min.step.v
start new record
read abs.err.v
start new record
read rel.err.v
start new record
open 3 for input, name is "seed.dat" 'entries to seed.v
use 3 for input
release seed.v(*)
read no.streams
start new record
reserve seed.v(*) as no.streams
for i = 1 to no.streams
   do
read seed.v(i)
start new record
loop
close unit 3
end ' of get.sim

function get.table   '' <f>18
    '' reads table of average arrivals in interval into an array
    '' of cumulative arrivals by interval
    define table as a double, 2-dim array
    define temp, cum as double variables
    define i, n as integer variables
    open unit 3 for input, name is "wkrate.dat"
    use 3 for input
    read n
    reserve table(*,*) as n by 2
    for i = 1 to n
do
    start new record
    read table(i, 1), temp
    add temp to cum
    table(i, 2) = cum
loop
close unit 3
return with table(*,*)
end ' get.table

function get.travel.time(here, there, type, stream)   '' <f>12
    '' returns travel time from here to there, constrained to be > 1 minute,
    '' with shape parameter adjusted to make final result approx gamma(3)
    define here, there, type, stream as integer variables
    define t, temp as double variables
    select case type
    case ground
        temp = transit.time(here, there)
t = 1 + mygamma.f(temp - 1, 3 * ((temp - 1) / temp)**2, stream) + choke.f(here, there)
    case air
        temp = flight.time(here, there)
t = 1 + mygamma.f(temp - 1, 3 * ((temp - 1) / temp)**2, stream)
endselect
return with t
end ' of get.travel.time
function go.green(h, n)  '' <f>12
' '<s> go.green

    define h, n as integer variables
    define result as an integer variable
    if n <= capacity(h) - green.limit
        result = TRUE
    else
        result = FALSE
    endif
    return with result
end ' ' of go.green

event go.off.red given hsp  '' <f>12
' '<s> go.off.red

    define hsp as an integer variable
    define cn as a pointer variable
    if (time.v - red.start(hsp)) * hours.v <= max.red
        if go.green(hsp, no.pts(hsp)) = FALSE
            ' ' schedule another check
            create a go.off.red called cn
            clear.notice(hsp) = cn
            schedule the go.off.red called cn giving hsp in min.red hours
            return
        endif
    endif
    ' 'max red time up, or load has decreased now
    clear.notice(hsp) = NULL
    red(hsp) = FALSE
    remove hsp from the red.set
    file hsp in the green.set
    return
end ' ' of go.off.red

function go.red(h, n)  '' <f>12
' '<s> go.red

    define h, n as integer variables
    define result as an integer variable
    if can.divert(h) = TRUE and n.red.set < max.red.hosp and
        red.today(h) < reds.per.day and n >= capacity(h) + red.limit
        result = TRUE
    else
        result = FALSE
    endif
    return with result
end ' ' of go.red
routine init.accident(accd)  ' ' <f>18
  '< ' init.accident
  define accd as a pointer variable
  define this.patient as a pointer variable
  define i as an integer variables
  define x as a double variable
  acc.start.time(accd) = time.v
  pended(accd) = FALSE
  if random.f(22) <= tr.prop
    add 1 to acc.counter
    kind(accd) = trauma
  else
    add 1 to med.counter
    kind(accd) = medical
  endif
  if random.f(28) > .3
    blunt(accd) = TRUE
    if kind(accd) = trauma
      add 1 to no.blunt
    endif
  else
    blunt(accd) = FALSE
  endif
  x = random.f(8)
  for each node with prop.accs(node, 2) > x
    find the first case
    site(accd) = node
    file accd in the active.set
  '' create victims for this accident
  if kind(accd) = medical  '' dont model medical patients
    add 1 to m.patient.counter
    add 1 to patient.counter
    no.victims(accd) = 1
  else
    no.victims(accd) = poisson.f(1.5, 9)
    for i = 1 to no.victims(accd)
      do
        add 1 to patient.counter
        add 1 to t.patient.counter
        create a patient called this.patient
        pt.id(this.patient) = patient.counter
        acc(this.patient) = accd
        call init.pt giving this.patient
        file this.patient in the patient.set
        file this.patient in the acc.patient.set
        activate the patient called this.patient now
      loop
  endif
  end ' ' of init.accident
routine init.pt given pt  ' ' <f>12
  '< ' init.pt
  define pt as a pointer variable
  define rem.iss, rem.iss.l, tl, t2 as double variables
phase(pt) = waiting
blunt(pt) = blunt(acc(pt))
injury.loc(pt) = site(acc(pt))
t1 = random.f(25)
t2 = random.f(26)
'' beta args from fit to Bakers data w/ percentile matching checked
'' on Baker and on Mackenzie
r.cts(pt) = 4
h.cts(pt) = 4
n.cts(pt) = 4
iss(pt) = int.f(.5 + 75.0 * mybeta.f(1.390, 9.632, 10))
rem.iss = iss(pt)
if iss(pt) >= 10
  'severely injured
  calculate CNS component -- data from Baxt
  if t1 < .40
    '40% of these have CNS injury
    add -2 to r.cts(pt)
    if iss(pt) < 15
      add -2 to n.cts(pt)
      rem.iss = iss(pt) / 2
    else
      if iss(pt) < 32
        add -3 to n.cts(pt)
        rem.iss = iss(pt) / 3
      else
        add -4 to n.cts(pt)
        rem.iss = iss(pt) / 5
      endif
  endif
else
  'not severely injured
  if t1 < .20
    'only 20% have CNS component
    add -1 to n.cts(pt)
    add -1 to r.cts(pt)
    rem.iss = iss(pt) / 2
  endif
endif
else
  'now partition hemorrhage and respiratory injury
  if t2 < .90
    'almost all injuries have some bleeding
    if t2 < .09
      '10% will have some resp injury also
      rem.iss.1 = .9 * rem.iss
    else
      rem.iss.1 = rem.iss
    endif
    ' scale bleeding rate from 4 to 225 ml / min, w/ 50% < 34
    br.0(pt) = 4 + 3 * rem.iss.1
    br.0(pt) = br.0(pt) * hours.v * minutes.v
    rem.iss = rem.iss - rem.iss.1
  endif
else
  'what remains is respiratory component
  r.cts(pt) = max.f(0, r.cts(pt) - int.f(.5 + sqrt.f(rem.iss)))
  02.sat(pt) = .80 + .05 * r.cts(pt)
endif
cts(pt) = cts.f(pt)
end ' of init.pt

routine initialize ' <f>18 <s> initialize ' init.pt
  ' get time series of runs
cum.events(*, *) = get.table
  ' get general simulation data
call get.sim
  ' get transportation network
call get.net
  ' get hospitals
call get.hosp
  ' get ambulances
call get.ems
  ' get ambulance call list
call get.list("call.dat", 'read.call.list', 'build.call.list',
                'write.call.list')
  ' get hospital preference list
call get.list("go.dat", 'read.hosp.list', 'build.hosp.list',
                'write.hosp.list')
  ' output net, call list, etc for confirmation
call print.net
  ' initialize distribution of accidents across nodes
prop.accs(*, *) = get.accs
end ' of initialize

function laplace.f(location, scale, stream)
  ' returns a laplace variate -- for use in the gamma function
  define location, scale as double variables
  define stream as an integer variable
  define y, u as double variables
  let u = random.f(stream)
  if u < .5
    let y = log.e.f(2 * u) * scale
  else
    let y = -log.e.f(2.0 - 2 * u) * scale
  endif
  return with location + y
end 'laplace.f

function lin.int.f(index, table) ' <f>18
  ' performs linear interpolation as follows:
given a value b=B and a table(a, b), where b is a cumulative value,
returns an interpolated value for a
define index as a double variable
define table as a double, 2-dim array
define x, slope, intercept as double variables
define i, maxi as integer variables
maxi = dim.f(table(*, *))
for i = 1 to maxi
  with table(i, 2) >= index
  find the first case
  x = table(i, 2)
slope = (x - table(i - 1, 2)) / (table(i, 1) - table(i - 1, 1))
intercept = x - table(i, 1) * slope
return ((index - intercept) / slope)
end ' lin.int.f

function living(pt) ' ' <f>12
  ' <s> living
  define pt as a pointer variable
  if 02.delivery(pt) le croak
    return with FALSE
  endif
  return with TRUE
end ' of living

function mybeta.f(a, b, stream) ' ' <f>12
  ' <s> mybeta.f
  define a, b as double variables
  define stream as an integer variable
  define x as a double variable
  let x = mygamma.f(a, a, stream)
  let x = x / (x + mygamma.f(b, b, stream))
  return with x
end 'mybeta.f

function mygamma.f(mean, a, stream) ' ' <f>12
  ' <s> mygamma.f
  ' function to replace error prone gamma.f function
  ' for order > 1
  ' taken from Tadikamalla as reported in Bratley B, Fox BL, Schrage LE:
  ' for order <= 1, taken from Ahrens, reported in same source
  define mean, a as double variables
  define stream as an integer variable

- 102 -
define b, temp, y, u, g, scale, u1, u2, p, q as double variables
define .terminated. to mean 1 = 2

' ' check for bad arg's
if mean <= 0.0
  let err.f = 145
endif
if a <= 0.0
  let err.f = 146
endif

let b = mean / a
let temp = a - 1.0

' ' first generate gamma variate for order a, mean a (b=1)
if a <= 1.0
  until .terminated.
    do
      let u1 = random.f(stream)
      let g = (exp.c + a) / exp.c
      let p = u1 * g
      let u2 = random.f(stream)
      if p > 1.0
        y = -log.e.f((g - p) / a)
        q = temp * log.e.f(y)
      else
        y = exp.f(log.e.f(p) / a)
        q = - y
      endif
      if log.e.f(u2) <= q
        leave
      endif
    do
  until .terminated.
  do
    until .terminated.
    do
      let scale = 1.0 + sqrt.f(4.0 * a - 3.0) / 2.0
      let y = laplace.f(temp, scale, stream)
      if y >= 0.0
        leave ' ' inner loop
      endif
    loop
  loop
else ' ' a > 1.0
  until .terminated.
  do
    do
      until .terminated.
      do
        let scale = 1.0 + sqrt.f(4.0 * a - 3.0) / 2.0
        let y = laplace.f(temp, scale, stream)
        if y >= 0.0
          leave ' ' inner loop
        endif
      loop
    loop
  loop
else ' ' now y is gamma a=a, b=1, so scale to a=a, b=b
  return with y * b
endif

end 'mygamma.f

left routine no pts given hospital ' ' <f>12
' ' <s> no pts ' ' used to ACCUMULATE pt stats by hospital -- could have been done by compiler, but now used to check red status dynamically
define hospital as an integer variable
define n as an integer variable
define inc as a double variable
enter with n 'new value for no.pts
inc = time.v - update.time(hospital)
add (no.pts(hospital) * inc) to r.accum.pts(hospital)
add (full(hospital) * inc) to r.accum.cap(hospital)
if n >= capacity(hospital)
  full(hospital) = TRUE
else
  full(hospital) = FALSE
endif
call check.red giving hospital and n
r.max.pts(hospital) = max.f(r.max.pts(hospital), n)
update.time(hospital) = time.v
move from n
return
end ' of no.pts

function nsp.f(table, stream) '' <f>24
  's> nsp.f
  '' returns time til the next event for a non-stationary Poisson arrival
  '' process, given a table of times and cumulative mean arrivals
  '' The arrival times are assumed to wrap around when the end of the
  '' table is reached. Caller is responsible for ensuring that start of
  '' simulation is at the time represented by the first entry in the table.
  '' Algorithm from Cinlar, E. "Introduction to Stochastic Processes,"
define table as a double, 2-dim array
define stream as an integer variable
define maxi, x as integer variables
define index, max.arr, y, z as double variables
define tprime, last.time as saved. double variables
these made global to allow full reset
maxi = dim.f(table(*,*))
max.arr = table(maxi, 2)
nsp.tprime = nsp.tprime - log.e.f(random.f(stream))
x = trunc.f(nsp.tprime / max.arr) 'number of wraps around end of table
index = mod.f(nsp.tprime, max.arr) 'offset into table
y = lin.int.f(index, table(*,*)) + x * table(maxi, 1) 'time of next event
z = y - nsp.last.time
nsp.last.time = y
return (z)
end '' nsp.f
routine pass.time given pt '' <f>12
''<s> pass.time
'' models passage of time for alive patients, and for pts who have
'' become terminal but will still receive an attempt at resuscitation

define pt as a pointer variable
define next.phase as an integer variable

select case phase(pt)
  case pvt.trav
    next.phase = resus
  case scene.rx
    next.phase = en.route
  case en.route
    next.phase = resus
  case resus
    next.phase = done.resus
endselect

until phase(pt) = next.phase
do
  work continuously evaluating 'bleed', testing 'done',
    updating 'update'

  '' if pt dies in the work continuously statement, record
  '' appropriate info and continue until the next phase change
  if condition(pt) = alive
    ''if just now died
    if living(pt) = FALSE
      ''if DEBUG
      write pt.id(pt), hour.f(time.v), minute.f(time.v),
        phase(pt) as
          "pt ", i 4, " arrests at ", i 2, ":", i 2,
          " in phase", i 2, /
        endif
    tod(pt) = (time.v - injury.time(pt)) * hours.v * minutes.v
    add 1 to no.deaths
    condition(pt) = dead
  endif
  endif
  if phase(pt) = pvt.trav
    '' NB -- this condition must agree with termination condition in
    '' the done function
    if time.v > pvt.tr.arrive
      phase(pt) = next.phase
    endif
  endif
  if phase(pt) = resus
    '' same type problem as above
    if time.v > injury.time(pt) + (wait.time(pt) + scene.time(pt) +
      transp.time(pt) + resus.time(pt)) / hours.v / minutes.v
      phase(pt) = next.phase
  endif
  endif
  cts(pt) = cts.f(pt)
  change.flag(pt) = FALSE
loop

end'' of pass.time

- 105 -
process patient

' <f>18
' <s> patient

define self as a pointer variable

define t as a double variable

self = patient
change.flag(self) = FALSE
injury.time(self) = time.v
sbp(self) = sbp.0
blood.volume(self) = bv.0
hct(self) = hct.0
rbc.mass(self) = .01 * hct(self) * blood.volume(self)
02.delivery(self) = sbp(self) * hct(self) * 02.sat(self)

bleeding.rate(self) = br.0(self)
condition(self) = alive

resus.time(self) = 1 + mygamma.f(t.resus - 1,
3 * ((t.resus - 1) / t.resus)**2, 11)
tx.time(self) = exponential.f(t.tx, 24)
t = time.v

' bleed till arrival
work continuously evaluating 'bleed', testing 'done', updating 'update'
change.flag(self) = FALSE
cts(self) = cts.f(self)
if living(self) = FALSE
  condition(self) = dead
tod(self) = (time.v - injury.time(self)) * hours.v * minutes.v
  call pt.report giving self
  remove self from acc.patient.set(acc(self))
  remove self from the patient.set
  add 1 to no.deaths
  if DEBUG
    write pt.id(self), hour.f(time.v), minute.f(time.v), phase(self) as
    "pt ", i 4, " arrests at ", i 2, ":", i 2, " in phase", i 2, /
  endif
  return
endif
wait.time(self) = (time.v - t) * hours.v * minutes.v
t = time.v

if phase(self) = pvt.trav
  call pass.time giving self
  remove self from the pvt.tr.set
  file self in resus.patient.set(hosp(self))
  phase(self) = resus
  transp.time(self) = (time.v - t) * hours.v * minutes.v
  t = time.v
else
  ' ambulance arrival
  rescue.bp(self) = sbp(self)
  call pass.time giving self
  scene.time(self) = (time.v - t) * hours.v * minutes.v
  t = time.v

  ' enroute
  call pass.time giving self
  transp.time(self) = (time.v - t) * hours.v * minutes.v
t = time.v
endif

' hospital arrival
hosp.bp(self) = sbp(self)
trf.start.time(self) = time.v + resus.time(self) / hours.v / minutes.v / 2
schedule a resp.support(self) in resus.time(self) / 3 minutes
trf.rate(self) = .5 * iv.rate(self)
no pts(hosp(self)) = no pts(hosp(self)) + 1
call pass.time giving self

if condition(self) = alive '' if didn't die earlier
  if living(self) = FALSE
    condition(self) = dead
    add 1 to no.deaths
    if DEBUG
      write pt.id(self), hour.f(time.v), minute.f(time.v) as "pt ", i 4, " arrests at ", i 2, ",:", i 2, " after resus", /
    endif
  else
    '' transfer to definitive care
    work tx.time(self) minutes
    def.bp(self) = sbp(self)
    cts(self) = cts.f(self)
  endif
endif
endif

' done with this patient, for purposes of this model
remove self from the patient.set
remove self from the resus.patient.set(hosp(self))
add -1 to no pts(hosp(self))

' coefficients from MTOS, assuming age < 55
if blunt(self) = TRUE
  prob.surv(self) = 1 /
  (1 + exp.f(-(-1.247 + .9544 * cts(self) -.0768 * iss(self)))
endif
else
  prob.surv(self) = 1 /
  (1 + exp.f(-(-.6029 + 1.1430 * cts(self) -.1516 * iss(self)))
endif
add (1 - prob.surv) to est.deaths
add (1 - prob.surv) to r.est.deaths
call pt.report giving self
delete 

routine print.net '' <f>12
'' <s> print.net
'' prints system information as defined by the data files for confirmation
define i as an integer variable
define lvl as a text variable
define item as a pointer variable
define temp as a double variable
use 4 for output
skip 1 line
print 2 lines thus
network structure
weights represent arterial route travel time between node centers
for i = 1 to n.node
    do
        print 1 line with i, node.name(i), n.edge.set(i) thus
        node *** (**) has outdegree ***
        for each arc in edge.set(i)
            do
                print 1 line with sink(arc), node.name(sink(arc)), weight(arc) thus
                arc to *** (**) of weight ****,**
            loop
        loop
    skip 1 line
print 1 line thus
hospital locations
for i = 1 to n.hospital
    do
        select case level{i)
            case level1
                lvl = "level 1"
            case level2
                lvl = "level 2"
            case level2a
                lvl = "level 2a"
            case level3
                lvl = "level 3"
        endselect
        print 1 line with hosp.name{i), lvl, capacity{i),
        node.name(hosp.base{i}) thus
        ***, ********, *** beds, in ***
    loop
    skip 1 line
print 1 line thus
ambulance call list
for each node
    do
        print 1 line with node.name(node) thus
        *** will request these ambulances
        for each item in call.list(node)
            do
                if type(ambulance.id(item)) = ground
                    temp = transit.time(amb.base(ambulance.id(item)), node)
                else
                    temp = flight.time(amb.base(ambulance.id(item)), node)
                endif
                print 1 line with amb.name(ambulance.id(item)),
                node.name(amb.base(ambulance.id(item))), temp
                thus
                **** from *** with mean travel time ****,** min
            loop
        loop
print 1 line thus
hospital dispatch list
for each node
    do
        print 1 line with node.name(node) thus
        *** victims will go to these hospitals
        for each item in hosp.list(node)
            do
        loop
select case level(hospital.id(item))
  case level1
    lvl = "level 1"
  case level2
    lvl = "level 2"
  case level2a
    lvl = "level 2a"
  case level3
    lvl = "level 3"
endselect

print 1 line with hosp.name(hospital.id(item)),
  lvl, node.name(hosp.base(hospital.id(item))),
  transit.time(node, hosp.base(hospital.id(item)))
thus
  ***, ********, in ***, with mean travel time ****.** min

loop
  loop
  skip 1 line

print 7 lines with major.cutoff, minor.cutoff, htol*100, atol*100 thus
Triage rule:
  Champ TS <= *.* goes to level 1, > *.* may go to level 3
Travel times exceeding minimum time by less than tolerance included in
routine dispatch lists
hospital choice tolerance **%
ambulance choice tolerance **%
skip 1 line
use 6 for output

end ' ' of print.net

routine pt.report given pt    ' ' <f>12
  ' ' <s> pt.report

  define pt as a pointer variable

  use 8 for output

  write run, pt.id(pt), blunt(pt), transp.mode(pt), iss(pt),
    injury.time(pt), wait.time(pt), scene.time(pt), transp.time(pt),
    resus.time(pt), tx.time(pt), tod(pt), rescue.bp(pt),
    hosp.bp(pt), def.bp(pt) as
      i 3, s 1, i 5, 2 i 3, i 5, d(22, 15), 9 d(10, 3), +
  write
    prob.surv(pt), br.0(pt) / hours.v / minutes.v, cts(pt),
    condition(pt), injury.loc(pt), hosp(pt), need.helo(pt),
    got.helo(pt) as
      3 d(10, 3), 5 i 4, /

  use 6 for output

  end ' ' of pt.report

routine pvt.travel(acc)    ' ' <f>12
  ' ' <s> pvt.travel

  define acc as a pointer variable
define pt, item as pointer variables
define temp, t, limit as double variables
define hsp, rtot, cumtot as integer variables

' find nearest hospital, regardless of status
limit = random.f(16)
rtot = 0
for each item in hosp.list(site(acc))
  compute cumtot as the sum of hosp.volume(hospital.id(item))
for each item in hosp.list(site(acc))
do
  add hosp.volume(hospital.id(item)) to rtot
  if rtot / cumtot >= limit
    hsp = hospital.id(item)
    leave
endif
loop
until acc.patient.set(acc) is empty
do
  remove first pt from acc.patient.set(acc)
t = 1.2 * transit.time(site(acc), hosp.base(hsp))
temp = 1 + mygamma.f(t - 1, 3 * ((t - 1) / t)**2, 20) +
  choke.f(site(acc), hosp.base(hsp))
' pvt travel time 20% more than ambulance
pvt.tr.arrive(pt) = time.v + temp / minutes.v / hours.v
phase(pt) = pvt.trav
change.flag(pt) = TRUE
hosp(pt) = hsp
transp.mode(pt) = private
file pt in the pvt.tr.set
loop
end '' of pvt.travel

routine read.call.list '' <f>18
'' <s> read.call.list
'' unit 3 is already open on entry
define here as an integer variable
while data is not ended
do
  read here
  while card is not new
do
    create a call.item
    read ambulance.id(call.item)
    file this call.item in call.list(here)
  loop
loop
end '' of read.call.list

routine read.hosp.list '' <f>12
'' <s> read.hosp.list
'' unit 3 is already open on entry
define here as an integer variable
while data is not ended
do  
  read here
  while card is not new
    do
      create a go.item
        read hospital.id(go.item)
        read hosp.level(go.item)
        file this go.item in hosp.list(here)
    loop
  loop
end '' of read.call.list

routine recall given num, acc "<f>18
  '" recall
  '" recall num ambulances starting from the end of the
  amb.set (farthest from acc)
  define num as an integer variable
  define acc as a pointer variable
  define i, amb as integer variables
  define arun as a pointer variable
  i = 0
  for each amb in the amb.set(acc) in reverse order, with type(amb)=ground
    do
      if amb is in the to.scene.set
        add 1 to i
        remove amb from the amb.set(acc)
        remove amb from the to.scene.set
        arun = amb.run(amb)
        interrupt the ambulance.run called arun
        status(arun) = not.working
        cur.location(amb) = find.loc(amb)
        time.a(arun) = 0
        resume the ambulance.run called arun
      if DEBUG
        write amb.name(amb), hour.f(time.v), minute.f(time.v) as
        t 3, " recalled at ", i 2, ":", i 2, /
      endif
    endif
    if i = num
      leave
    endif
  loop
end '' of recall

event resp.support given pt "<f>12
  '" resp.support
  '" event to represent provision of O2, intubations, chest decompression, etc
  define pt as a pointer variable
  r.cts(pt) = 4.0 - r.cts(pt) / 2
  O2.sat(pt) = .80 + .05 * r.cts(pt)
  cts(pt) = cts.f(pt)
end ' of resp.support

routine run.report ' <f>12
  ' <s> run.report

  define p as a double variable
  if r.no.pended > 0
    p = r.mean.pending * tmp.dur * hours.v * minutes.v / r.no.pended
  else
    p = 0
  endif

  use 4 for output
  print 10 lines with run, tmp.dur * hours.v,
    r.no.accs, r.no.t.pts, 100 * r.no.blunt / r.no.accs,
    1.0 - r.idle.amb / (n.ambulance - num.helo),
    1.0 - r.idle.helo / num.helo,
    r.no.divert, r.no.override, r.no.deaths,
    r.est.deaths thus

  run **
    duration ***.*** hours no.accs *** no. tr. pts ***
    % blunt **.**
    amb. utilization ***.***
    helo utilization ***.***
    diverts ***
    overrides ***
    deaths ***
    est deaths **.**

  print 4 lines with r.mean.pending, r.max.pending, r.no.pended, p
  thus
  queue for amb mean ***.*** max ***
  no.accs pended: ****
  av pending time: ***.*** minutes

  print 2 lines thus
  hospital utilization
  name avg tr load max tr load reserve % red time
  for each hospital
  do
    print 1 line with hosp.name(hospital), r.accum.pts(hospital) / tmp.dur,
      r.max.pts(hospital), 1 - r.accum.cap(hospital) / tmp.dur,
      r.mean.red(hospital) * 100 thus
      *** ***.*** *** *.* ***istrore
    loop
    skip 1 line

  use 7 for output
  run, duration, midpoint, no.accs, no.blunt, no.pts, amb utilization, hosp
  utilization, reserve, diverts, overrides, deaths, est.deaths
  write run, tmp.dur, time.v - tmp.dur / 2.0, r.no.accs, r.no.meds,
  r.no.blunt, r.no.t.pts, 1.0 - r.idle.amb / (n.ambulance - num.helo),
  1.0 - r.idle.helo / num.helo, r.no.divert, r.no.override, r.no.deaths
  as
  i 5, 2 d(12, 5), 4 i 7, 2 d(10, 5), 3 i 5, +
  write r.est.deaths, r.no.pended, p as
  d(8, 3), i 7, d(8, 3), +

  ' repeat the loop so variables are grouped (easier to read in SYSTAT)
  for each hospital
do
  write r.accum.pts(hospital) / tmp.dur as d(12, 5), +
loop
for each hospital
do
  write 0 - r.accum.cap(hospital) / tmp.dur as d(10, 5), +
loop
for each hospital
do
  write r.mean.red(hospital) as d(10, 5), +
loop
write as /
use 6 for output
end
" run.report

routine tr.check.in.acc given acc  
" <f>12
" <> tr.check.in.acc

  define acc as a pointer variable
  define t as a text variable
  if kind(acc) = trauma
    t = "acc"
  else
    t = "med"
  endif
  write acc.id(acc), t, node.name(site(acc)),
     hour.f(time.v), minute.f(time.v) as "event ", t, " (", t, ") occurs in ", t, " at ", i, ":", i, /
end  // of tr.check.in.acc

routine tr.check.in.amb given amb  
" <f>12
" <> tr.check.in.amb

  define amb as an integer variable
  write amb.name(amb), node.name(amb.base(amb)), hour.f(time.v),
     minute.f(time.v) as " ", t, " at base ", t, ") at ", i, ":", i, /
end  // of tr.check.in.amb

routine tr.check.in.pt given pt and acc  
" <f>12
" <> tr.check.in.pt

  define pt, acc as pointer variables
  write acc.id(acc), pt.id(pt) as "acc ", t, " has pt ", i, /
end  // tr.check.in.pt

routine tr.check.out.acc given acc  
" <f>12
```
3149  '  <s>  tr.check.out.acc
3150  define  acc  as  a  pointer  variable
3151  define  t  as  a  text  variable
3152  if  kind(acc)  =  trauma
3153    t  =  "acc"
3154  else
3155    t  =  "med"
3156  endif
3157  write  acc.id(acc),  t,  hour.f(time.v),  minute.f(time.v)  as
3158    "event",  i 5,  "("  t  3,  ")"  done  at",  i 2,  ":",  i 2,  /
3159  end''  of  tr.check.out.acc
3160  routine  tr.check.out.pt  given  pt  '  <f>12
3161  '  <s>  tr.check.out.pt
3162  define  pt  as  a  pointer  variable
3163  write  pt.id(pt),  hour.f(time.v),  minute.f(time.v),  n.patient.set  as
3164    "pt",  i 5,  "finished  at",  i 2,  ":",  i 2,
3165    "leaving",  i 3,  "in  system",  /
3166  end''  of  tr.check.out.pt
3167  routine  tr.deliver.pt  given  amb  '  <f>12
3168  '  <s>  tr.deliver.pt
3169  define  amb  as  an  integer  variable
3170  write  amb.name(amb),  hosp.name(hosp(amb.run(amb))),  hour.f(time.v),  minute.f(time.v)  as
3171    "at",  i 2,  ":",  i 2,  /
3172  end''  of  tr.deliver.pt
3173  routine  tr.enroute.hosp(amb)  '  <f>12
3174  '  <s>  tr.enroute.hosp
3175  define  amb  as  an  integer  variable
3176  define  n  as  an  integer  variable
3177  if  kind(amb.run(amb))  =  trauma
3178    n  =  n.amb.patient.set(amb.run(amb))
3179  else
3180    n  =  1
3181  endif
3182  write  amb.name(amb),  hosp.name(hosp(amb.run(amb))),  hour.f(time.v),  minute.f(time.v),  n  as
3183    "enroute  to",  i 2,  ":",  i 2,  "pts",  /
3184  end''  of  tr.enroute.hosp
3185  routine  tr.go.green(hsp)  '  <f>12
```
define hsp as an integer variable
write hosp.name(hsp), hour.f(time.v), minute.f(time.v) as t 3, " went green at ", t 3, ":", t 3, /
end

define hsp as an integer variable
write hosp.name(hsp), hour.f(time.v), minute.f(time.v) as t 3, " went red at ", t 3, ":", t 3, /
end

define amb as an integer variable
define t as a text variable
if kind(amb.run(amb» = trauma
  t = "acc"
else
  t = "med"
endif
write amb.name(amb), t, acc.id(acc(amb.run(amb»), node.name(cur.location(amb», hour.f(time.v), minute.f(time.v) as in ", t 3, " on scene at ", t 3, s 1, i 5,
  " at ", t 3, ":", t 3, /
end

define pt, amb.run as pointer variables
if kind(amb.run) = trauma
  write amb.name(ambulance.id(amb.run», pt.id(pt), cts(pt), hour.f(time.v), minute.f(time.v) as " ", t 3, " treating pt ", i 5, " cts = ", d(3, 1),
  " at ", t 3, ":", t 3, /
else
  write amb.name(ambulance.id(amb.run»), hour.f(time.v), minute.f(time.v) as " ", t 3, " treating med pt at ",
  i 2, ":", i 2, /
end
end

define pt as a pointer variable
if kind(amb.run) = trauma
  write amb.name(ambulance.id(amb.run», pt.id(pt), cts(pt), hour.f(time.v), minute.f(time.v) as " ", t 3, " treating pt ", i 5, " cts = ", d(3, 1),
  " at ", t 3, ":", t 3, /
else
  write amb.name(ambulance.id(amb.run»), hour.f(time.v), minute.f(time.v) as " ", t 3, " treating med pt at ",
  i 2, ":", i 2, /
end
end

- 115 -
define pt as a pointer variable
write pt.id(pt), hosp.name(hosp(pt)),
    hour.f(time.v), minute.f(time.v) as
    " pt", i 5, " heading to ", t 3, " at ", i 2, ":", i 2, /
end ' ' of tr.pvt.tr.pt

routine tr.resus.pt given pt and hosp ' ' <f>12
    '<s> tr.resus.pt
    define pt as a pointer variable
    define hosp as an integer variable
    write pt.id(pt), hosp.name(hosp), weekday.f(time.v), hour.f(time.v) and
        minute.f(time.v) as " pt", i 5, " at ", t 3, " on day", i 2, " at ",
        i 2, ":", i 2, /
end ' ' of tr.resus.pt

routine tr.send.amb given amb ' ' <f>12
    '<s> tr.send.amb
    define amb as an integer variable
    define t as a text variable
    if kind(amb.run(amb)) = trauma
        t = "acc"
    else
        t = "med"
    endif
    write amb.name(amb), t, aec.id(aec(amb.run(amb))),
        hour.f(time.v), minute.f(time.v)
        as " t 3, " enroute to ", t 3, s 1, i 5, " at ",
        i 2, ":", i 2, /
end ' ' of tr.send.amb

routine tr.stack.acc given acc ' ' <f>12
    '<s> tr.stack.acc
    define acc as a pointer variable
    define t as a text variable
    if kind(acc) = trauma
        t = "acc"
    else
        t = "med"
    endif
    write t, acc.id(acc) as
t 3, s 1, i 5, " awaiting service", /
end ' ' tr.stack.acc

routine tr.to.home given amb ' ' <f>12
define amb as an integer variable
write amb.name(amb), hour.f(time.v), minute.f(time.v)
as "", i 2, " heading home at ", 
    i 2, 
end ' ' of tr.to.home

routine tr.unstack.acc given acc     ' ' <f>12
' ' <s> tr.unstack.acc
define acc as a pointer variable
define t as a text variable
if kind(acc) = trauma
t = "acc"
else
t = "med"
endif
write t, acc.id(acc), hour.f(time.v), minute.f(time.v)
as "pending", t 3, s 1, i 5, " served at ", i 2, 
end ' ' of tr.unstack.acc

routine travel(from, to, amb)     ' ' <f>12
' ' <s> travel
define from, to, amb as integer variables
remove amb from node.amb.set(from)
work travel.time(amb) minutes
file amb in node.amb.set(to)
cur.location(amb) = to
end ' ' of travel

routine update(patient)     ' ' <f>12
' ' <s> update'
' these statements are required for changes in continuous simulation
' variables to be visible outside the integrator routine
define patient as a pointer variable
blood.volume(patient) = blood.volume(patient)
d.blood.volume(patient) = d.blood.volume(patient)
rbc.mass(patient) = rbc.mass(patient)
d.rbc.mass(patient) = d.rbc.mass(patient)
sbp(patient) = sbp(patient)
d.sbp(patient) = d.sbp(patient)
bleeding.rate(patient) = bleeding.rate(patient)
d.bleeding.rate(patient) = d.bleeding.rate(patient)
hct(patient) = hct(patient)
d.hct(patient) = d.hct(patient)
02.delivery(patient) = 02.delivery(patient)
d.02.delivery(patient) = d.02.delivery(patient)
end ' of update

routine write.call.list '' <f>12
'' <s> write.call.list

define item as a pointer variable
open unit 3 for output, name is "call.dat"
use 3 for output
for each node
do
write node as i 3, s 2, +
for each item in call.list(node)
do
write ambulance.id(item) as i 3, s 2, +
loop
write as /
loop
close unit 3
end ' of write.call.list

routine write.hosp.list '' <f>12
'' <s> write.hosp.list

define item as a pointer variable
open unit 3 for output, name is "go.dat"
use 3 for output
for each node
do
write node as i 3, s 2, +
for each item in hosp.list(node)
do
write hospital.id(item), hosp.level(item) as i 3, s 2, i 3, s 2, +
loop
write as /
loop
close unit 3
close unit 3
end ' of write.call.list
## Appendix 2

### Data Cross-Reference

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>MODE (+ distinct usage)</th>
<th>ROUTINE</th>
<th>REFERENCES (= assigned)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.DUPLICATE</td>
<td>Define to mean</td>
<td>PREAMBLE</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>BUILD.ROUTE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABS.ERR.V</td>
<td>Permanent attribute</td>
<td>GET.SIM</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Real</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABS.F</td>
<td>Library routine</td>
<td>BLEED</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MYGAMMA.F</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ACC</td>
<td>Temporary attribute</td>
<td>PREAMBLE</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pointer</td>
<td>AMBULANCE.RUN</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASSIGN.AMB</td>
<td>1</td>
<td>=</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHECK.ACCIDENT</td>
<td>1</td>
<td>=</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INIT.ACCIDENT</td>
<td>1</td>
<td>=</td>
</tr>
<tr>
<td></td>
<td></td>
<td>INIT.PT</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PATIENT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TR.ON.SCENE</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TR.SEND.AMB</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ACC.ARRIVE.TIME</td>
<td>Temporary attribute</td>
<td>PREAMBLE</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Double</td>
<td>DISPATCHER</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>ACC.COUNTER</td>
<td>Global variable</td>
<td>PREAMBLE</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Integer</td>
<td>MAIN</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INIT.ACCIDENT</td>
<td>1</td>
<td>=</td>
</tr>
<tr>
<td>ACC.END.TIME</td>
<td>Temporary attribute</td>
<td>PREAMBLE</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Double</td>
<td>ACCIDENT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ACC.ID</td>
<td>Temporary attribute</td>
<td>PREAMBLE</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Integer</td>
<td>DISPATCHER</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GENERATOR</td>
<td>1</td>
<td>=</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TR.CHECK.IN.ACC</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TR.CHECK.IN.PT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TR.CHECK.OUT.ACC</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TR.ON.SCENE</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TR.SEND.AMB</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TR.STACK.ACC</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>TR.UNSTACK.ACC</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ACC.PATIENT.SET</td>
<td>Set</td>
<td>PREAMBLE</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACCIDENT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AMBULANCE.RUN</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHECK.ACCIDENT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET.PATIENT</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INIT.ACCIDENT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PATIENT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PVT.TRAVEL</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Value</td>
<td>Lines</td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------------</td>
<td>---------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>ACC.START.TIME</td>
<td>Temporary attribute</td>
<td>Double</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>ACCIDENT</td>
<td>Process notice</td>
<td>PREAMBLE</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>+ Global variable</td>
<td>Pointer</td>
<td>INIT.ACCIDENT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>+ Implied subscript</td>
<td></td>
<td>ACCIDENT</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ACTIVE.SET</td>
<td>Set</td>
<td>PREAMBLE</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>ADJ.TIME.F</td>
<td>Routine</td>
<td>Double</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>AIR</td>
<td>Define to mean</td>
<td>PREAMBLE</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>AIRSPEED</td>
<td>Global variable</td>
<td>Double</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ALARM.LAG</td>
<td>Global variable</td>
<td>Double</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ALIVE</td>
<td>Define to mean</td>
<td>PREAMBLE</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>AMB.BASE</td>
<td>Permanent attribute</td>
<td>Integer</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>AMB.BASE.SET</td>
<td>Set</td>
<td>PREAMBLE</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>AMB.ID</td>
<td>Permanent attribute</td>
<td>Integer</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>AMB.NAME</td>
<td>Permanent attribute</td>
<td>Text</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>AMB.PATIENT.SET</td>
<td>Set</td>
<td>PREAMBLE</td>
<td>AMBULANCE.RUN</td>
<td>DISPATCHER</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----</td>
<td>----------</td>
<td>--------------</td>
<td>------------</td>
</tr>
<tr>
<td>AMB.RUN</td>
<td>Permanent attribute, Pointer</td>
<td>PREAMBLE</td>
<td>ASSIGN.AMB</td>
<td>DISPATCHER</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DISP.CHECK.ACCIDENT</td>
<td>FIND.LOC</td>
</tr>
<tr>
<td>AMB.SET</td>
<td>Set</td>
<td>PREAMBLE</td>
<td>ASSIGN.AMB</td>
<td>CHECK.ACCIDENT</td>
</tr>
<tr>
<td>AMBULANCE</td>
<td>Permanent entity</td>
<td>PREAMBLE</td>
<td>GET.EMS</td>
<td>GET.EMS</td>
</tr>
<tr>
<td>AMBULANCE.ID</td>
<td>Temporary attribute, Integer</td>
<td>PREAMBLE</td>
<td>AMBULANCE.RUN</td>
<td>ASSIGN.AMB</td>
</tr>
<tr>
<td>AMBULANCE.RUN</td>
<td>Process notice</td>
<td>PREAMBLE</td>
<td>AMBULANCE.RUN</td>
<td>ASSIGN.AMB</td>
</tr>
<tr>
<td>ARC</td>
<td>Temporary entity</td>
<td>PREAMBLE</td>
<td>BUILD.ROUTE</td>
<td>GET.NET</td>
</tr>
<tr>
<td>ARC.STATUS</td>
<td>Temporary attribute, Integer</td>
<td>PREAMBLE</td>
<td>BUILD.ROUTE</td>
<td>GET.NET</td>
</tr>
</tbody>
</table>

- 121 -
<p>| ASSIGN.AMB | Routine | ASSIGN.AMB | 1 |
| ASSIGNMENT.SET | Set | PREAMBLE | 2 |
| AT.BASE | Define to mean | AMBULANCE.RUN | 1 |
| AT.BASE.SET | Set | AMBULANCE.RUN | 1 |
| AT.HOSP | Define to mean | AMBULANCE.RUN | 1 |
| AT.HOSP.SET | Set | AMBULANCE.RUN | 1 |
| AT.SCENE | Define to mean | AMBULANCE.RUN | 1 |
| ATOL | Global variable | Double | 1 |
| BEST_ROUTE | Routine | Double | 1 |
| BLEED | Routine | BLEED | 1 |
| BLEEDING.RATE | Temporary attribute | Double | 2 |
| BLOOD.VOLUME | Temporary attribute | Double | 2 |
| BLUNT | Temporary attribute | Integer | 3 |
| BR.O | Temporary attribute | Double | 2 |
| BUILD.CALL.LIST | Routine | BUILD.CALL.LIST | 1 |</p>
<table>
<thead>
<tr>
<th><strong>BUILD.HOSP.LIST</strong></th>
<th><strong>Routine</strong></th>
<th><strong>BUILD.HOSP.LIST</strong></th>
<th><strong>1</strong></th>
<th><strong>INITIALIZE</strong></th>
<th><strong>1</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BUILD.ROUTE</strong></td>
<td><strong>Routine</strong></td>
<td><strong>BEST.ROUTE</strong></td>
<td><strong>2</strong></td>
<td><strong>BUILD.ROUTE</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td><strong>BV.O</strong></td>
<td><strong>Define to mean</strong></td>
<td><strong>PREAMBLE</strong></td>
<td><strong>1</strong></td>
<td><strong>BLEED</strong></td>
<td><strong>2</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>PATIENT</strong></td>
<td><strong>1</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CALL.ITEM</strong></td>
<td><strong>Temporary entity</strong></td>
<td><strong>PREAMBLE</strong></td>
<td><strong>1</strong></td>
<td><strong>BUILD.CALL.LIST</strong></td>
<td><strong>2</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>READ.CALL.LIST</strong></td>
<td><strong>1</strong></td>
<td><strong>BUILD.CALL.LIST</strong></td>
<td><strong>7</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>READ.CALL.LIST</strong></td>
<td><strong>3</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CALL.LIST</strong></td>
<td><strong>Set</strong></td>
<td><strong>PREAMBLE</strong></td>
<td><strong>2</strong></td>
<td><strong>BUILD.CALL.LIST</strong></td>
<td><strong>3</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>GET.AMB</strong></td>
<td><strong>2</strong></td>
<td><strong>PRINT.NET</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>READ.CALL.LIST</strong></td>
<td><strong>1</strong></td>
<td><strong>WRITE.CALL.LIST</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td><strong>CALLS.PENDING</strong></td>
<td><strong>Define to mean</strong></td>
<td><strong>PREAMBLE</strong></td>
<td><strong>1</strong></td>
<td><strong>DISPATCHER</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td><strong>CAN.DIVERT</strong></td>
<td><strong>Permanent attribute Integer</strong></td>
<td><strong>PREAMBLE</strong></td>
<td><strong>2</strong></td>
<td><strong>GET.HOSP</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>GO.RED</strong></td>
<td><strong>1</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CAPACITY</strong></td>
<td><strong>Permanent attribute Integer</strong></td>
<td><strong>PREAMBLE</strong></td>
<td><strong>2</strong></td>
<td><strong>GET.HOSP</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>GO.GREEN</strong></td>
<td><strong>1</strong></td>
<td><strong>GO.RED</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>LEFT NO.PTS</strong></td>
<td><strong>1</strong></td>
<td><strong>PRINT.NET</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td><strong>CH.CUM.WEIGHT</strong></td>
<td><strong>Temporary attribute Double</strong></td>
<td><strong>PREAMBLE</strong></td>
<td><strong>2</strong></td>
<td><strong>BEST.ROUTE</strong></td>
<td><strong>5</strong></td>
</tr>
<tr>
<td><strong>CHANGE.FLAG</strong></td>
<td><strong>Temporary attribute Integer</strong></td>
<td><strong>PREAMBLE</strong></td>
<td><strong>2</strong></td>
<td><strong>AMBULANCE.RUN</strong></td>
<td><strong>2</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>DONE</strong></td>
<td><strong>1</strong></td>
<td><strong>GET.PATIENT</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>PASS.TIME</strong></td>
<td><strong>1</strong></td>
<td><strong>PATIENT</strong></td>
<td><strong>2</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>PVT.TRAVEL</strong></td>
<td><strong>1</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>CHECK.ACCIDENT</strong></td>
<td><strong>Routine</strong></td>
<td><strong>AMBULANCE.RUN</strong></td>
<td><strong>2</strong></td>
<td><strong>CHECK.ACCIDENT</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td><strong>CHECK.RED</strong></td>
<td><strong>Routine</strong></td>
<td><strong>CHECK.RED</strong></td>
<td><strong>1</strong></td>
<td><strong>LEFT NO.PTS</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td><strong>CHOOSE.F</strong></td>
<td><strong>Routine</strong></td>
<td><strong>Double</strong></td>
<td><strong>1</strong></td>
<td><strong>CHOOSE.F</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>GET.TRAVEL.TIME</strong></td>
<td><strong>1</strong></td>
<td><strong>PVT.TRAVEL</strong></td>
<td><strong>1</strong></td>
</tr>
<tr>
<td><strong>CHOOSE.PT.WT</strong></td>
<td><strong>Temporary attribute Double</strong></td>
<td><strong>PREAMBLE</strong></td>
<td><strong>2</strong></td>
<td><strong>BEST.ROUTE</strong></td>
<td><strong>2</strong></td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Attributes</td>
<td>Example Values</td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------</td>
<td>------------</td>
<td>----------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLEAR.NOTICE</td>
<td>Permanent attribute</td>
<td>Pointer</td>
<td>PREAMBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLEAR.REDS</td>
<td>Event notice</td>
<td></td>
<td>PREAMBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>+ Global variable</td>
<td>Pointer</td>
<td>PREAMBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLEAR.TIME</td>
<td>Global variable</td>
<td>Double</td>
<td>PREAMBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONDITION</td>
<td>Temporary attribute</td>
<td>Integer</td>
<td>PREAMBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BLEED</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DONE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PASS.TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PATIENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PT.REPORT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>COS.F</td>
<td>Library routine</td>
<td></td>
<td>FTIME.F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CROAK</td>
<td>Define to mean</td>
<td></td>
<td>PREAMBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>LIVING</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTS</td>
<td>Temporary attribute</td>
<td>Double</td>
<td>PREAMBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AMBULANCE.RUN</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CTS.F</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GET.LEVEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INIT.PT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PASS.TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PATIENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PT.REPORT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESP.SUPPORT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>TR.PICKUP.PT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CTS.F</td>
<td>Routine</td>
<td>Double</td>
<td>PREAMBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>ACCIDENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>CTS.F</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GET.PATIENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INIT.PT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PASS.TIME</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PATIENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>RESP.SUPPORT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CUM.EVENTS</td>
<td>Global variable</td>
<td>Double</td>
<td>PREAMBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>GENERATOR</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>INITIALIZE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CUM.WEIGHT</td>
<td>Temporary attribute</td>
<td>Double</td>
<td>PREAMBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>BEST.ROUTE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CUR.LOCATION</td>
<td>Permanent attribute</td>
<td>Integer</td>
<td>PREAMBLE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>AMBULANCE.RUN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
<td>Type</td>
<td>Action 1</td>
<td>Action 2</td>
<td></td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------</td>
<td>---------------</td>
<td>----------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>D.BLEEDING RATE</td>
<td>Temporary attribute Double</td>
<td>BLEED</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPDATE</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>D.BLOOD VOLUME</td>
<td>Temporary attribute Double</td>
<td>BLEED</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPDATE</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>D.HCT</td>
<td>Temporary attribute Double</td>
<td>BLEED</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPDATE</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>D.O2.DELIVERY</td>
<td>Temporary attribute Double</td>
<td>BLEED</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPDATE</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>D.RBC.MASS</td>
<td>Temporary attribute Double</td>
<td>BLEED</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPDATE</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>D.SBP</td>
<td>Temporary attribute Double</td>
<td>BLEED</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>UPDATE</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DEAD</td>
<td>Define to mean</td>
<td>PREAMBLE</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>BLEED</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PASS.TIME</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PATIENT</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>DEBUG</td>
<td>Define to mean</td>
<td>PREAMBLE</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DIS派出</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FIND.HOSP</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PASS.TIME</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PATIENT</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>RECALL</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DEF.BP</td>
<td>Temporary attribute Double</td>
<td>PREAMBLE</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PATIENT</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PT.REPORT</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DEST.LEVEL</td>
<td>Temporary attribute Integer</td>
<td>PREAMBLE</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AMBULANCE.RUN</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASSIGN.AMB</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FIND.HOSP</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET.PATIENT</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DESTINATION</td>
<td>Temporary attribute Integer</td>
<td>PREAMBLE</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AMBULANCE.RUN</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASSIGN.AMB</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISPATCHER</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FIND.LOC</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>DIM.F</td>
<td>Library routine</td>
<td>LIN.INT.F</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NSP.F</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DISPATCHER</td>
<td>Routine</td>
<td>ACCIDENT</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>AMBULANCE.RUN</td>
<td>5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHECK.ACCIDENT</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DISPATCHER</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET.PATIENT</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DIVERT.COUNTER</td>
<td>Global variable</td>
<td>Integer</td>
<td>PREAMBLE</td>
<td>MAIN</td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>----------------</td>
<td>---------</td>
<td>----------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>DONE</td>
<td>Routine</td>
<td>Integer</td>
<td>PREAMBLE</td>
<td>DONE</td>
<td></td>
</tr>
<tr>
<td>DONE.RESUS</td>
<td>Define to mean</td>
<td>PREAMBLE</td>
<td>DONE</td>
<td>PASS.</td>
<td></td>
</tr>
<tr>
<td>EDGE.SET</td>
<td>Set</td>
<td>PREAMBLE</td>
<td>BEST.ROUTE</td>
<td>GET.NET</td>
<td></td>
</tr>
<tr>
<td>EN_ROUTE</td>
<td>Define to mean</td>
<td>PREAMBLE</td>
<td>AMBULANCE.RUN</td>
<td>PASS.</td>
<td></td>
</tr>
<tr>
<td>ERR.F</td>
<td>Library routine</td>
<td>MYGAMMA.F</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EST.DEATHS</td>
<td>Global variable</td>
<td>Double</td>
<td>PREAMBLE</td>
<td>FINAL.REPORT</td>
<td></td>
</tr>
<tr>
<td>EXP.C</td>
<td>Permanent attribute</td>
<td>Double</td>
<td>MYGAMMA.F</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>EXP.F</td>
<td>Library routine</td>
<td>BLEED</td>
<td>MYGAMMA.F</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>EXPONENTIAL.F</td>
<td>Library routine</td>
<td>ACCIDENT</td>
<td>AMBULANCE.RUN</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>F.AMB.PATIENT.SET</td>
<td>Temporary attribute</td>
<td>Pointer</td>
<td>AMBULANCE.RUN</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>F.H.READY.SET</td>
<td>Permanent attribute</td>
<td>Integer</td>
<td>GET.AMB</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>FALSE</td>
<td>Define to mean</td>
<td>PREAMBLE</td>
<td>AMBULANCE.RUN</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>FINAL.REPORT</td>
<td>Routine</td>
<td>MAIN</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Routine/Attribute</td>
<td>Type</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------------</td>
<td>-------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIND.HOSP</td>
<td>Routine</td>
<td>Integer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIND.LOC</td>
<td>Routine</td>
<td>Integer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLIGHT.TIME</td>
<td>Permanent attribute</td>
<td>Double</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FOREVER</td>
<td>Define to mean</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FREE</td>
<td>Define to mean</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FTIME.F</td>
<td>Routine</td>
<td>Double</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FULL</td>
<td>Permanent attribute</td>
<td>Integer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.ACCUM.CAP</td>
<td>Permanent attribute</td>
<td>Double</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.ACCUM.PTS</td>
<td>Permanent attribute</td>
<td>Double</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.MAX.ACCS</td>
<td>Global variable</td>
<td>Double</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.MAX.DUR</td>
<td>Global variable</td>
<td>Double</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.MAX.MAXPEND</td>
<td>Global variable</td>
<td>Double</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.MAX.PTS</td>
<td>Global variable</td>
<td>Double</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.MEAN.ACCS</td>
<td>Routine</td>
<td>Double</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.MEAN.AMB</td>
<td>Routine</td>
<td>Double</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G.MEAN.DUR</td>
<td>Routine</td>
<td>Double</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 127 -
<table>
<thead>
<tr>
<th>Routine</th>
<th>Type</th>
<th>Final Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.MEAN.MAXPEND</td>
<td>Double</td>
<td>1</td>
</tr>
<tr>
<td>G.MEAN.PEND</td>
<td>Double</td>
<td>1</td>
</tr>
<tr>
<td>G.MEAN.PTS</td>
<td>Double</td>
<td>1</td>
</tr>
<tr>
<td>G.SSQ.CAP</td>
<td>Double</td>
<td>2</td>
</tr>
<tr>
<td>G.SSQ.PTS</td>
<td>Double</td>
<td>2</td>
</tr>
<tr>
<td>G.VAR.ACCS</td>
<td>Double</td>
<td>1</td>
</tr>
<tr>
<td>G.VAR.AMB</td>
<td>Double</td>
<td>1</td>
</tr>
<tr>
<td>G.VAR.DUR</td>
<td>Double</td>
<td>1</td>
</tr>
<tr>
<td>G.VAR.MAXPEND</td>
<td>Double</td>
<td>1</td>
</tr>
<tr>
<td>G.VAR.PEND</td>
<td>Double</td>
<td>1</td>
</tr>
<tr>
<td>G.VAR.PTS</td>
<td>Double</td>
<td>1</td>
</tr>
<tr>
<td>GENERATOR</td>
<td>Process</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>GET.ACCS</td>
<td>Pointer</td>
<td>1</td>
</tr>
<tr>
<td>GET.AMB</td>
<td>Integer</td>
<td>1</td>
</tr>
<tr>
<td>GET.EMS</td>
<td>Pointer</td>
<td>1</td>
</tr>
<tr>
<td>GET.HOSP</td>
<td>Pointer</td>
<td>1</td>
</tr>
<tr>
<td>GET.IV.RATE</td>
<td>Double</td>
<td>1</td>
</tr>
<tr>
<td>GET.LEVEL</td>
<td>Integer</td>
<td>1</td>
</tr>
<tr>
<td>Routine/Entity</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>GET.LIST</td>
<td>Routine</td>
<td>GET.LIST 1 INITIALIZE 2</td>
</tr>
<tr>
<td>GET.NET</td>
<td>Routine</td>
<td>GET.NET 1 INITIALIZE 1</td>
</tr>
<tr>
<td>GET.NUM.AMB</td>
<td>Integer</td>
<td>PREAMBLE DISPATCHER 3 GET.NUM.AMB 1</td>
</tr>
<tr>
<td>GET.PATIENT</td>
<td>Routine</td>
<td>AMBULANCE.RUN 2 GET.PATIENT 1</td>
</tr>
<tr>
<td>GET.SIM</td>
<td>Routine</td>
<td>GET.SIM 1 INITIALIZE 1</td>
</tr>
<tr>
<td>GET.TABLE</td>
<td>Pointer</td>
<td>GET.TABLE 1 INITIALIZE 1</td>
</tr>
<tr>
<td>GET.TRAVEL.TIME</td>
<td>Routine</td>
<td>Double PREAMBLE DISPATCHER 2 GET.AMB 1 GET.TRAVEL.TIME 1</td>
</tr>
<tr>
<td>GO.BEACON</td>
<td>Routine</td>
<td>PREAMBLE CHECK.BEACON 1</td>
</tr>
<tr>
<td>GO.COLOR</td>
<td>Temporary entity</td>
<td>BUILD.COLOR 1 READ.COLOR 1</td>
</tr>
<tr>
<td>+ Global variable</td>
<td>Pointer</td>
<td>PREAMBLE READ.COLOR 10 = READ.COLOR 4 =</td>
</tr>
<tr>
<td>GO.ITEM</td>
<td>Temporary attribute</td>
<td>PREAMBLE CHECK.RED 1 GO.BEACON 1 GO.OFF.RED 1</td>
</tr>
<tr>
<td>GO.OFF.RED</td>
<td>Event notice</td>
<td>PREAMBLE CHECK.RED 3 CLEAR.REDS 1 GO.OFF.RED 3</td>
</tr>
<tr>
<td>+ Global variable</td>
<td>Pointer</td>
<td>PREAMBLE GO.OFF.RED 3 = GO.OFF.RED 1</td>
</tr>
<tr>
<td>GO.RED</td>
<td>Routine</td>
<td>PREAMBLE CHECK.RED 1 GO.RED 1</td>
</tr>
<tr>
<td>GOR.HSP</td>
<td>Temporary attribute</td>
<td>PREAMBLE CHECK.RED 1 = GO.OFF.RED 1 =</td>
</tr>
<tr>
<td>GOT.HELO</td>
<td>Temporary attribute</td>
<td>PREAMBLE DISPATCHER 2 PT.REPORT 1</td>
</tr>
<tr>
<td>GR.MAX.PTS</td>
<td>Permanent attribute</td>
<td>PREAMBLE MAIN 1 FINAL.REPORT 1</td>
</tr>
<tr>
<td>GREEN.LIMIT</td>
<td>Global variable</td>
<td>PREAMBLE GET.SIM 1 = GO.BEACON 1</td>
</tr>
<tr>
<td>GREEN.SET</td>
<td>Set</td>
<td>PREAMBLE 3</td>
</tr>
</tbody>
</table>
CHECK.RED 3
CLEAR.REDS 1
FIND.HOSP 1
GET.HOSP 1
GO.OFF.RED 1

GROUND Define to mean
CHECK.RED 1
BUILD.CALL.LIST 2
DISPATCHER 5
GET.AMB 2
GET.EMS 1
GET.TRAVEL.TIME 1
PRINT.NET 1
RECALL 1

H.CTS Temporary attribute Integer
PREAMBLE 2
CTS.F 6 =
INIT.PT 1 =

H.READY.SET Set
PREAMBLE 4
BUILD.CALL.LIST 1
DISPATCHER 1
GET.AMB 2
GET.EMS 1

H.WAITING.SET Set
PREAMBLE 2
AMBULANCE.RUN 1
CHECK.ACCIDENT 1
DISPATCHER 1

HCT Temporary attribute Double
PREAMBLE 2
BLEED 2
PATIENT 3 =
UPDATE 1 =

HCT.O Define to mean
PATIENT 1

HELO.COMING Temporary attribute Integer
PREAMBLE 2
AMBULANCE.RUN 5 =
DISPATCHER 1 =

HOSP Temporary attribute Integer
PREAMBLE 3
AMBULANCE.RUN 2 =
DISPATCHER 1 =
PATIENT 4
PT.REPORT 1
PVT.TRAVEL 1 =
TR.DELIVER.PT 1
TR.ENROUTE.HOSP 1
TR.PVT.TR.PT 1

HOSP.BASE Permanent attribute Integer
PREAMBLE 2
DISPATCHER 1
GET.HOSP 4 =
GET.PATIENT 1
PRINT.NET 2
PVT.TRAVEL 2

HOSP.BASE.SET Set
PREAMBLE 2
BUILD.HOSP.LIST 1
GET.HOSP 2

- 130 -
<table>
<thead>
<tr>
<th>Identifier</th>
<th>Attribute Type</th>
<th>Data Type</th>
<th>PREAMBLE</th>
<th>PATIENT</th>
<th>PT.REPORT</th>
<th>GET.HOSP</th>
<th>FIND.HOSP</th>
<th>READ.HOSP.LIST</th>
<th>WRITE.HOSP.LIST</th>
<th>BUILD.HOSP.LIST</th>
<th>FIND.HOSP</th>
<th>PRINT.NET</th>
<th>PVT.TRAVEL</th>
<th>READ.HOSP.LIST</th>
<th>WRITE.HOSP.LIST</th>
<th>HOUR.F</th>
<th>DISPACHTER</th>
<th>FIND.HOSP</th>
<th>GENERATOR</th>
<th>PASS.TIME</th>
<th>PATIENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOSP.BP</td>
<td>Temporary</td>
<td>Double</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>HOSP.ID</td>
<td>Permanent</td>
<td>Integer</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>HOSP.LEVEL</td>
<td>Temporary</td>
<td>Integer</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>HOSP.NAME</td>
<td>Permanent</td>
<td>Text</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>HOSP.SET</td>
<td>Set</td>
<td></td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>HOSP.VOLUME</td>
<td>Permanent</td>
<td>Double</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>HOSPITAL</td>
<td>Permanent entity</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>HOSPITAL.ID</td>
<td>Temporary</td>
<td>Integer</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>HOUR.F</td>
<td>Library routine</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
RECALL
TR.CHECK.IN.ACC
TR.CHECK.IN.AMB
TR.CHECK.OUT.ACC
TR.CHECK.OUT.PT
TR.DELIVER.PT
TR.ENROUTE.HOSP
TR.GO.GREEN
TR.GO.RED
TR.ON.SCENE
TR.PICKUP.PT
TR.PVT.TR.PT
TR.RESUS.PT
TR.SEND.AMB
TR.TO.HOME
TR.UNSTACK.ACC

HOURS.V Permanent attribute Double BLEED 1
CHECK.RED 1
DONE 1
GENERATOR 1
GET.IV.RATE 1
GET.PATIENT 1
GO.OFF.RED 1
INIT.PT 1
PASS.TIME 2
PATIENT 6
PT.REPORT 1
PVT.TRAVEL 1
RUN.REPORT 2

HTOL Global variable Double PREAMBLE 1
BUILD.HOSP.LIST 2
GET.SIM 1 =
PRINT.NET 1

IN.USE Define to mean PREAMBLE 1
BUILD.ROUTE 1

INFO.LAG Global variable Double PREAMBLE 1
ACCIDENT 1
GET.SIM 1 =

INIT.ACCIDENT Routine GENERATOR 1
INIT.ACCIDENT 1

INIT.PT Routine INIT.ACCIDENT 1
INIT.PT 1

INITIALIZE Routine MAIN 1
INITIALIZE 1

INJURY.LOC Temporary attribute Integer PREAMBLE 2
INIT.PT 1 =
PT.REPORT 1

INJURY.TIME Temporary attribute Double PREAMBLE 2
DONE 1
PASS.TIME 2
PATIENT 2 =
PT.REPORT 1

INT.F Library routine DISPATCHER 1
**INTEGRATOR.V**  
Permanent attribute Integer  
GET.NUM.AMB 1  
INIT.PT 2

**IS.TO.IV**  
Define to mean  
PREAMBLE 1  
BLEED 1

**ISS**  
Temporary attribute Integer  
PREAMBLE 2  
INIT.PT 9 =  
PATIENT 2  
PT.REPORT 1

**IV.RATE**  
Temporary attribute Double  
PREAMBLE 3  
AMBULANCE.RUN 3 =  
BLEED 1  
GET.PATIENT 1 =  
PATIENT 1

**IV.START.TIME**  
Temporary attribute Double  
PREAMBLE 3  
BLEED 2  
GET.PATIENT 1 =

**KIND**  
Temporary attribute Integer  
PREAMBLE 3  
ACCIDENT 1  
AMBULANCE.RUN 6  
ASSIGN.AMB 1 =  
DISPATCHER 2  
INIT.ACCIDENT 4 =  
TR.CHECK.IN.ACC 1  
TR.CHECK.OUT.ACC 1  
TR.ENROUTE.HOSP 1  
TR.ON.SCENE 1  
TR.PICKUP.PT 1  
TR.SEND.AMB 1  
TR.STACK.ACC 1  
TR.UNSTACK.ACC 1

**LAPLACE.F**  
Routine Double  
PREAMBLE 1  
LAPLACE.F 1  
MYGAMMA.F 1

**LAT**  
Permanent attribute Double  
PREAMBLE 2  
FTIME.F 2  
GET.NET 1 =

**LEVEL**  
Permanent attribute Integer  
PREAMBLE 2  
BUILD.HOSP.LIST 5  
GET.HOSP 1 =  
PRINT.NET 2

**LEVEL1**  
Define to mean  
PREAMBLE 1  
GET.LEVEL 1  
GET.PATIENT 1  
PRINT.NET 2

**LEVEL2**  
Define to mean  
PREAMBLE 1  
ASSIGN.AMB 1  
PRINT.NET 2

**LEVEL2A**  
Define to mean  
PREAMBLE 1  
BUILD.HOSP.LIST 1  
GET.LEVEL 1
<table>
<thead>
<tr>
<th>Variable/Function</th>
<th>Type/Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEVEL3</td>
<td>Define to mean</td>
<td>PRINT.NET 2</td>
</tr>
<tr>
<td>LIN.INT.F</td>
<td>Routine</td>
<td>PREAMBLE 1, AMBULANCE.RUN 2, ASSIGN.AMB 1, GET.LEVEL 1, GET.PATIENT 1, PRINT.NET 2</td>
</tr>
<tr>
<td>LINES.V</td>
<td>Temporary attribute Integer</td>
<td>MAIN 4 =</td>
</tr>
<tr>
<td>LIVING</td>
<td>Routine</td>
<td>PREAMBLE 1, DONE 1, LIVING 1, PASS.TIME 1, PATIENT 2</td>
</tr>
<tr>
<td>LL.TIME</td>
<td>Global variable</td>
<td>PREAMBLE 1, TIIME.F 1, GET.NET 1, GET.SIM 1 =</td>
</tr>
<tr>
<td>LOG.E.F</td>
<td>Library routine</td>
<td>LAPLACE.F 2, MYGAMMA.F 4, NSP.F 1</td>
</tr>
<tr>
<td>LOG.NORMAL.F</td>
<td>Library routine</td>
<td>AMBULANCE.RUN 2, GET.IV.RATE 1</td>
</tr>
<tr>
<td>LONG</td>
<td>Permanent attribute</td>
<td>PREAMBLE 1, TFIME.F 1, GET.NET 1 =</td>
</tr>
<tr>
<td>M.DELIVER</td>
<td>Global variable</td>
<td>PREAMBLE 1, AMBULANCE.RUN 1, GET.SIM 1 =</td>
</tr>
<tr>
<td>M.PATIENT.COUNTER</td>
<td>Global variable</td>
<td>PREAMBLE 2, MAIN 1, INIT.ACCIDENT 1 =</td>
</tr>
<tr>
<td>M.SECURE</td>
<td>Global variable</td>
<td>PREAMBLE 1, AMBULANCE.RUN 1, GET.SIM 1 =</td>
</tr>
<tr>
<td>MAJOR.CUTOFF</td>
<td>Global variable</td>
<td>PREAMBLE 1, GET.LEVEL 1, GET.SIM 2 =</td>
</tr>
<tr>
<td>MAJOR.TIME</td>
<td>Global variable</td>
<td>PREAMBLE 1, GET.PATIENT 1, GET.SIM 1 =</td>
</tr>
<tr>
<td>MAX.F</td>
<td>Library routine</td>
<td>MAIN 1, GET.IV.RATE 2, GET.NUM.AMB 1, INIT.PT 1, LEFT NO.PTS 1</td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>--------------------</td>
<td>------------------------------------</td>
</tr>
<tr>
<td>MAX.RED</td>
<td>Global variable</td>
<td>Integer</td>
</tr>
<tr>
<td>MAX.RED.HOSP</td>
<td>Global variable</td>
<td>Integer</td>
</tr>
<tr>
<td>MAX.STEP.V</td>
<td>Permanent attribute</td>
<td>Real</td>
</tr>
<tr>
<td>MAXRTS</td>
<td>Define to mean</td>
<td></td>
</tr>
<tr>
<td>MDT</td>
<td>Set</td>
<td></td>
</tr>
<tr>
<td>MED.COUNTER</td>
<td>Global variable</td>
<td>Integer</td>
</tr>
<tr>
<td>MEDICAL</td>
<td>Define to mean</td>
<td></td>
</tr>
<tr>
<td>MIN.AMB</td>
<td>Global variable</td>
<td>Integer</td>
</tr>
<tr>
<td>MIN.F</td>
<td>Library routine</td>
<td></td>
</tr>
<tr>
<td>MIN.LENGTH</td>
<td>Global variable</td>
<td>Double</td>
</tr>
<tr>
<td>MIN.RED</td>
<td>Global variable</td>
<td>Integer</td>
</tr>
<tr>
<td>MIN.STEP.V</td>
<td>Permanent attribute</td>
<td>Real</td>
</tr>
<tr>
<td>MINOR.CUTOFF</td>
<td>Global variable</td>
<td>Double</td>
</tr>
<tr>
<td>MINOR.TIME</td>
<td>Global variable</td>
<td>Integer</td>
</tr>
<tr>
<td>MINUTE.F</td>
<td>Library routine</td>
<td></td>
</tr>
</tbody>
</table>

- 135 -
TR.CHECK.IN.AMB 1
TR.CHECK.OUT.ACC 1
TR.CHECK.OUT.PT 1
TR.DELIVER.PT 1
TR.ENROUTE.HOSP 1
TR.GO.GREEN 1
TR.GO.RED 1
TR.ON.SCENE 1
TR.PICKUP.PT 2
TR.PVT.TR.PT 1
TR.RESUS.PT 1
TR.SEND.AMB 1
TR.TO.HOME 1
TR.UNSTACK.ACC 1

MINUTES.V Permanent attribute Double BLEED 1
DONE 1
FTIME.F 1
GET.IV.RATE 1
GET.NET 1
GET.PATIENT 1
INIT.PT 1
PASS.TIME 2
PATIENT 6
PT.REPORT 1
PVT.TRAVEL 1
RUN.REPORT 1

MOD.F Library routine NSP.F 1
MYBETA.F Routine Double PREAMBLE 1
INIT.PT 1
MYBETA.F 1
MYGAMMA.F Routine Double PREAMBLE 1
AMBULANCE.RUN 1
GET.TRAVEL.TIME 2
MYBETA.F 2
MYGAMMA.F 1
PATIENT 1
PVT.TRAVEL 1

N.ACC.PATIENT.SET Temporary attribute Integer2 DISPATCHER 2
N.AMB.PATIENT.SET Temporary attribute Integer2 TR.ENROUTE.HOSP 1
N.AMB.SET Temporary attribute Integer2 CHECK.ACCIDENT DISPATCHER 1
N.AMBULANCE Global variable Integer2 MAIN 1
gENERATOR 1
N.CTS Temporary attribute Integer PREAMBLE 2
CTS.F 1
INIT.PT 5

N.EDGE.SET Permanent attribute Integer2 GET.NET 1
N.H.READY.SET Permanent attribute Integer2 PREAMBLE MAIN 1
gENERATOR 1
N.NODE.AMB.SET Permanent attribute Integer2 GET.AMB 1
<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N.PENDING.SET</td>
<td>Permanent attribute</td>
<td>Integer 2</td>
</tr>
<tr>
<td>N.READY.SET</td>
<td>Permanent attribute</td>
<td>Integer 2</td>
</tr>
<tr>
<td>N.RED.SET</td>
<td>Permanent attribute</td>
<td>Integer 2, GO.RED</td>
</tr>
<tr>
<td>N.TEMP.SET</td>
<td>Permanent attribute</td>
<td>Integer 2, BUILD.CALL.LIST, BUILD.HOSP.LIST</td>
</tr>
<tr>
<td>NEED.HELO</td>
<td>Temporary attribute</td>
<td>Integer, PREAMBLE, GET.PATIENT, PT.REPORT</td>
</tr>
<tr>
<td>NEEDED</td>
<td>Temporary attribute</td>
<td>Integer, PREAMBLE, DISPATCHER</td>
</tr>
<tr>
<td>NEW.INFO</td>
<td>Define to mean</td>
<td>PREAMBLE, ACCIDENT, DISPATCHER</td>
</tr>
<tr>
<td>NMPD</td>
<td>Define to mean</td>
<td>PREAMBLE, FTIME.F</td>
</tr>
<tr>
<td>NO.BLUNT</td>
<td>Global variable</td>
<td>Integer, PREAMBLE, MAIN, FINAL.REPORT, INIT.ACCIDENT</td>
</tr>
<tr>
<td>NO.DEATHS</td>
<td>Global variable</td>
<td>Integer, PREAMBLE, MAIN, FINAL.REPORT, PASS.TIME, PATIENT</td>
</tr>
<tr>
<td>NO.PENDED</td>
<td>Global variable</td>
<td>Integer, PREAMBLE, MAIN, GET.AMB</td>
</tr>
<tr>
<td>NO.PTS</td>
<td>Permanent attribute</td>
<td>Integer, PREAMBLE, GO.OFF.RED, LEFT NO.PTS, LEFT NO.PTS, PATIENT</td>
</tr>
<tr>
<td>NO.PTS.LEFT</td>
<td>Define to mean</td>
<td>PREAMBLE, CHECK.ACCIDENT, DISPATCHER</td>
</tr>
<tr>
<td>NO.RUNS</td>
<td>Global variable</td>
<td>Integer, PREAMBLE, MAIN, FINAL.REPORT, GET.SIM</td>
</tr>
<tr>
<td>NO.VICTIMS</td>
<td>Temporary attribute</td>
<td>Integer, PREAMBLE, DISPATCHER, INIT.ACCIDENT</td>
</tr>
<tr>
<td>NODE</td>
<td>Permanent entity</td>
<td>PREAMBLE, BUILD.CALL.LIST</td>
</tr>
</tbody>
</table>

- 137 -
+ Global variable Integer2

BUILD.HOSP.LIST 2
GET.NET 3
BEST.ROUTE 2 =
BUILD.CALL.LIST 8 =
BUILD.HOSP.LIST 4 =
GET.NET 2 =
INIT.ACCIDENT 2 =
PRINT.NET 9 =
WRITE.CALL.LIST 3 =
WRITE.HOSP.LIST 3 =
GET.AMB 1 =

+ Implied subscript

NODE.AMB.SET Set

PREAMBLE 2
BUILD.CALL.LIST 2
GET.EMS 1
TRAVEL 2

NODE.ID Permanent attribute Integer

PREAMBLE 2
GET.NET 1 =

NODE.NAME Permanent attribute Text

PREAMBLE 2
GET.NET 1 =
PRINT.NET 7
TR.CHECK.IN.ACC 1
TR.CHECK.IN.AMB 1
TR.ON.SCENE 1

NODE.SET Set

PREAMBLE 2
BEST.ROUTE 5

NOT.WORKING Define to mean

PREAMBLE 1
RECALL 1

NSP.F Routine Double

PREAMBLE 1
GENERATOR 2
NSP.F 1

NSP.LAST.TIME Global variable Double

PREAMBLE 1
MAIN 1 =
NSP.F 2 =

NSP.TPRIME Global variable Double

PREAMBLE 1
MAIN 1 =
NSP.F 3 =

NTRPT Define to mean

PREAMBLE 1

NULL Define to mean

PREAMBLE 1
ACCIDENT 2
AMBULANCE.RUN 2
CHECK.ACCIDENT 1
CHECK.RED 1
CLEAR.REDS 2
FIN.HOSP 2
GET.AMB 2
GO.OFF.RED 1

NUM.HELO Global variable Integer

PREAMBLE 1
GET.EMS 1 =
RUN.REPORT 2

O2.DELIVERY Temporary attribute Double

PREAMBLE 2
LIVING 1
<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O2.SAT</td>
<td></td>
<td>Temporary attribute Double</td>
</tr>
<tr>
<td>O2.SAT.O</td>
<td></td>
<td>Define to mean</td>
</tr>
<tr>
<td>ON.SCENE.SET</td>
<td></td>
<td>Set</td>
</tr>
<tr>
<td>OPS</td>
<td></td>
<td>Define to mean</td>
</tr>
<tr>
<td>OUT.OF.SERVICE.SET</td>
<td></td>
<td>Set</td>
</tr>
<tr>
<td>OVERRIDE.COUNTER</td>
<td>Global variable Integer</td>
<td></td>
</tr>
<tr>
<td>P.SECURE</td>
<td>Global variable Double</td>
<td></td>
</tr>
<tr>
<td>PASS.TIME</td>
<td>Routine</td>
<td></td>
</tr>
<tr>
<td>PATIENT</td>
<td>Process notice</td>
<td></td>
</tr>
<tr>
<td>PATIENT.COUNTER</td>
<td>Global variable Integer</td>
<td></td>
</tr>
<tr>
<td>PENDED</td>
<td>Temporary attribute Integer</td>
<td></td>
</tr>
<tr>
<td>PENDING.SET</td>
<td>Set</td>
<td></td>
</tr>
<tr>
<td>PHASE</td>
<td>Temporary attribute Integer</td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Definition</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>PVT.TRAVEL</td>
<td>Routine</td>
<td>INIT.ACCIDENT</td>
</tr>
<tr>
<td>PRINT.NET</td>
<td>Routine</td>
<td>INITIALIZE, PRINT.NET</td>
</tr>
<tr>
<td>PRIVATE</td>
<td>Define to mean</td>
<td>PREAMBLE, PVT.TRAVEL</td>
</tr>
<tr>
<td>PROB.SURV</td>
<td>Temporary attribute</td>
<td>Double PREAMBLE, PATIENT, PT.REPORT</td>
</tr>
<tr>
<td>PROP.ACCE</td>
<td>Global variable</td>
<td>Double PREAMBLE, INIT.ACCIDENT, INITIALIZE</td>
</tr>
<tr>
<td>PSTF</td>
<td>Define to mean</td>
<td>PREAMBLE</td>
</tr>
<tr>
<td>PT.ID</td>
<td>Temporary attribute</td>
<td>Integer PREAMBLE, BLEED, INIT.ACCIDENT, PASS.TIME, PATIENT, PT.REPORT, TR.CHECK.IN.PT, TR.CHECK.OUT.PT, TR.PICKUP.PT, TR.PVT.TR.PT, TR.RESUS.PT</td>
</tr>
<tr>
<td>PT.REPORT</td>
<td>Routine</td>
<td>PATIENT, PT.REPORT</td>
</tr>
<tr>
<td>PTS.PER.ACC</td>
<td>Global variable</td>
<td>Double PREAMBLE, GET.SIM</td>
</tr>
<tr>
<td>PVT.TR.ARRIVE</td>
<td>Temporary attribute</td>
<td>Double PREAMBLE, DONE, PASS.TIME, PVT.TRAVEL</td>
</tr>
<tr>
<td>PVT.TR.SET</td>
<td>Set</td>
<td>PREAMBLE, PATIENT, PVT.TRAVEL</td>
</tr>
<tr>
<td>PVT.TRAV</td>
<td>Define to mean</td>
<td>PREAMBLE, DONE, PASS.TIME, PATIENT, PVT.TRAVEL</td>
</tr>
<tr>
<td>PVT.TRAVEL</td>
<td>Routine</td>
<td>ACCIDENT, PVT.TRAVEL</td>
</tr>
<tr>
<td>R.ACCUM.CAP</td>
<td>Permanent attribute</td>
<td>Double PREAMBLE, MAIN, LEFT NO.PTS, RUN.REPORT</td>
</tr>
<tr>
<td>R.ACCUM.PTS</td>
<td>Permanent attribute</td>
<td>Double PREAMBLE, MAIN</td>
</tr>
<tr>
<td>Name</td>
<td>Type</td>
<td>Data Type</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>R.CTS</td>
<td>Temporary attribute</td>
<td>Integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R.EST.DEATHS</td>
<td>Global variable</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R.IDLE.AMB</td>
<td>Routine</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R.IDLE.HELO</td>
<td>Routine</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R.MAX.PENDING</td>
<td>Global variable</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R.MAX.PTS</td>
<td>Permanent attribute</td>
<td>Integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R.MEAN.PENDING</td>
<td>Routine</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R.MEAN.RED</td>
<td>Routine</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R.NO.ACCS</td>
<td>Global variable</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R.NO.BLUNT</td>
<td>Global variable</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R.NO.DEATHS</td>
<td>Global variable</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R.NO.DIVERT</td>
<td>Global variable</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R.NO.M.PTS</td>
<td>Global variable</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R.NO.MEDS</td>
<td>Global variable</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R.NO.OVERRIDE</td>
<td>Global variable</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R.NO.PENDED</td>
<td>Global variable</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R.NO.T.PTS</td>
<td>Global variable</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identifier</td>
<td>Type</td>
<td>Value</td>
</tr>
<tr>
<td>------------</td>
<td>----------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>R.PT</td>
<td>Temporary attribute</td>
<td>Pointer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R.SSQ.CAP</td>
<td>Global variable</td>
<td>Double</td>
</tr>
<tr>
<td>R.SSQ.PTS</td>
<td>Global variable</td>
<td>Double</td>
</tr>
<tr>
<td>RANDOM.F</td>
<td>Library routine</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RANGE</td>
<td>Global variable</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RBC.MASS</td>
<td>Temporary attribute</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>READ.CALL.LIST</td>
<td>Routine</td>
<td></td>
</tr>
<tr>
<td>READ.HOSP.LIST</td>
<td>Routine</td>
<td></td>
</tr>
<tr>
<td>READY.SET</td>
<td>Set</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RECALL</td>
<td>Routine</td>
<td></td>
</tr>
<tr>
<td>RED</td>
<td>Permanent attribute</td>
<td>Integer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RED.LIMIT</td>
<td>Global variable</td>
<td>Integer</td>
</tr>
<tr>
<td>RED.SET</td>
<td>Set</td>
<td></td>
</tr>
<tr>
<td>RED.START</td>
<td>Permanent attribute</td>
<td>Double</td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Attribute Type</td>
</tr>
<tr>
<td>-------------------------</td>
<td>---------------</td>
<td>----------------</td>
</tr>
<tr>
<td>RED.TODAY</td>
<td>Permanent</td>
<td>Integer</td>
</tr>
<tr>
<td></td>
<td>attribute</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REDS.PER.DAY</td>
<td>Global</td>
<td>Integer</td>
</tr>
<tr>
<td></td>
<td>variable</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REL.ERR.V</td>
<td>Permanent</td>
<td>Real</td>
</tr>
<tr>
<td></td>
<td>attribute</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REQ.AMB</td>
<td>Define to mean</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REQ.CAR</td>
<td>Define to mean</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REQ.HELO</td>
<td>Define to mean</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REQ.HELP</td>
<td>Define to mean</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESCUE.BP</td>
<td>Temporary</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td>attribute</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESP.SUPPORT</td>
<td>Event notice</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>+ Global</td>
<td>Pointer</td>
</tr>
<tr>
<td></td>
<td>variable</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESP.TIME</td>
<td>Temporary</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td>attribute</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESUS</td>
<td>Define to mean</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESUS.PATIENT.SET</td>
<td>Set</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESUS.TIME</td>
<td>Temporary</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td>attribute</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROPENERR.V</td>
<td>Temporary</td>
<td>Integer</td>
</tr>
<tr>
<td></td>
<td>attribute</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROUTE</td>
<td>Set</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RUN</td>
<td>Global variable Integer</td>
<td>PREAMBLE MAIN = 1, GENERATOR = 2, PT.REPORT = 1, RUN.REPORT = 2</td>
</tr>
<tr>
<td>RUN.COUNTER</td>
<td>Global variable Integer</td>
<td>PREAMBLE ASSIGN.AMB = 1</td>
</tr>
<tr>
<td>RUN.ID</td>
<td>Temporary attribute Integer</td>
<td>PREAMBLE MAIN = 1, RUN.REPORT = 1</td>
</tr>
<tr>
<td>RUN.REPORT</td>
<td>Routine</td>
<td>PREAMBLE MAIN, RUN.REPORT = 1</td>
</tr>
<tr>
<td>RUNGE.KUTTA.R</td>
<td>Library routine</td>
<td>GET.SIM = 1</td>
</tr>
<tr>
<td>S.DELIVER</td>
<td>Global variable Double</td>
<td>PREAMBLE AMBULANCE.RUN = 1, GET.SIM = 1</td>
</tr>
<tr>
<td>S.MAX.PTS</td>
<td>Permanent attribute Integer</td>
<td>PREAMBLE MAIN = 1, FINAL.REPORT = 1</td>
</tr>
<tr>
<td>S.SECURE</td>
<td>Global variable Double</td>
<td>PREAMBLE AMBULANCE.RUN = 1, GET.SIM = 1</td>
</tr>
<tr>
<td>SBP</td>
<td>Temporary attribute Double</td>
<td>PREAMBLE BLEED = 2, CTS.F = 4, GET.IV.RATE = 2, UPDATE = 1</td>
</tr>
<tr>
<td>SBP.O</td>
<td>Define to mean</td>
<td>PREAMBLE BLEED = 2, GET.IV.RATE = 2, PATIENT = 1</td>
</tr>
<tr>
<td>SCENE.RX</td>
<td>Define to mean</td>
<td>PREAMBLE GET.PATIENT = 1, PASS.TIME = 1</td>
</tr>
<tr>
<td>SCENE.TIME</td>
<td>Temporary attribute Double</td>
<td>PREAMBLE AMBULANCE.RUN = 6, DONE = 1, GET.PATIENT = 2, PASS.TIME = 1, PATIENT = 1, PT.REPORT = 1</td>
</tr>
<tr>
<td>SEED.V</td>
<td>Permanent attribute Integer</td>
<td>GET.SIM = 3</td>
</tr>
<tr>
<td>SENT</td>
<td>Temporary attribute Integer</td>
<td>PREAMBLE DISPATCHER = 5, GET.AMB = 1</td>
</tr>
<tr>
<td>SIGN.F</td>
<td>Library routine</td>
<td>BLEED = 1</td>
</tr>
<tr>
<td>SINK</td>
<td>Temporary attribute Integer</td>
<td>PREAMBLE BEST.ROUTE = 6, BUILD.ROUTE = 1</td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SITE</td>
<td>Temporary attribute</td>
<td>Integer PREAMBLE 2 = GET.NET 1 = PRINT.NET 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASSIGN.AMB 1</td>
</tr>
<tr>
<td>SOURCE</td>
<td>Temporary attribute</td>
<td>Integer PREAMBLE 2 = GET.NET 2 = PRINT.NET 2 = BEST.ROUTE 2 = BUILD.ROUTE 1 = GET.NET 1</td>
</tr>
<tr>
<td>SQRT.F</td>
<td>Library routine</td>
<td>ADJ.TIME.F 1</td>
</tr>
<tr>
<td>SRC</td>
<td>Temporary attribute</td>
<td>Integer PREAMBLE 2 = AMBULANCE.RUN 1 = FIND.LOC 3</td>
</tr>
<tr>
<td>SSPND</td>
<td>Define to mean</td>
<td>PREAMBLE 1</td>
</tr>
<tr>
<td>STA.A</td>
<td>Temporary attribute</td>
<td>Integer PREAMBLE 2 = AMBULANCE.RUN 1</td>
</tr>
<tr>
<td>START.TIME</td>
<td>Global variable</td>
<td>Double PREAMBLE 1 = MAIN 2 = GENERATOR 1</td>
</tr>
<tr>
<td>STATUS</td>
<td>Temporary attribute</td>
<td>Integer PREAMBLE 2 = AMBULANCE.RUN 2 = RECALL 1 =</td>
</tr>
<tr>
<td>T.ON.SCENE</td>
<td>Global variable</td>
<td>Double PREAMBLE 1 = AMBULANCE.RUN 2 = GET.SIM 1 =</td>
</tr>
<tr>
<td>T.PATIENT.COUNTER</td>
<td>Global variable</td>
<td>Integer PREAMBLE 2 = MAIN 1 = INIT.ACCIDENT 1 =</td>
</tr>
<tr>
<td>T.RESUS</td>
<td>Global variable</td>
<td>Double PREAMBLE 1 = GET.SIM 1 = PATIENT 2</td>
</tr>
<tr>
<td>T.TO.PT</td>
<td>Global variable</td>
<td>Double PREAMBLE 1 = AMBULANCE.RUN 1 = GET.SIM 1 =</td>
</tr>
<tr>
<td>T.TX</td>
<td>Global variable</td>
<td>Double PREAMBLE 1 = GET.SIM 1 = PATIENT 1</td>
</tr>
<tr>
<td>TCONST</td>
<td>Define to mean</td>
<td>PREAMBLE 1</td>
</tr>
<tr>
<td>TCRUISE</td>
<td>Global variable</td>
<td>Double PREAMBLE 1</td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>TEMP.SET</td>
<td>Set</td>
<td>PREAMBLE</td>
</tr>
<tr>
<td>TIME.A</td>
<td>Temporary attribute</td>
<td>Double</td>
</tr>
<tr>
<td>TIME.V</td>
<td>Permanent attribute</td>
<td>Double</td>
</tr>
<tr>
<td>TMP.ACDS</td>
<td>Global variable</td>
<td>Integer</td>
</tr>
<tr>
<td>TMP.AMBS</td>
<td>Global variable</td>
<td>Double</td>
</tr>
<tr>
<td>TMP.DUR</td>
<td>Global variable</td>
<td>Double</td>
</tr>
<tr>
<td>TMP.MN.PEND</td>
<td>Global variable</td>
<td>Double</td>
</tr>
<tr>
<td>TMP.MX.PEND</td>
<td>Global variable</td>
<td>Integer</td>
</tr>
<tr>
<td>Variable</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>TMP.PTS</td>
<td>Global variable</td>
<td>Integer</td>
</tr>
<tr>
<td>TO.BASE</td>
<td>Define to mean</td>
<td>PREAMBLE MAIN 1</td>
</tr>
<tr>
<td>TO.BASE.SET</td>
<td>Set</td>
<td>PREAMBLE DISPATCHER 2</td>
</tr>
<tr>
<td>TO.HOSP</td>
<td>Define to mean</td>
<td>PREAMBLE AMBULANCE.RUN 1</td>
</tr>
<tr>
<td>TO.HOSP.SET</td>
<td>Set</td>
<td>PREAMBLE DISPATCHER 2</td>
</tr>
<tr>
<td>TO.SCENE</td>
<td>Define to mean</td>
<td>PREAMBLE AMBULANCE.RUN 1</td>
</tr>
<tr>
<td>TO.SCENE.SET</td>
<td>Set</td>
<td>PREAMBLE DISPATCHER 2</td>
</tr>
<tr>
<td>TOD</td>
<td>Temporary attribute</td>
<td>Double</td>
</tr>
<tr>
<td>TOT.COUNTER</td>
<td>Global variable</td>
<td>Integer</td>
</tr>
<tr>
<td>TR.CHECK.IN.ACC</td>
<td>Routine</td>
<td>PREAMBLE INIT.ACCIDENT 1</td>
</tr>
<tr>
<td>TR.CHECK.IN.AMB</td>
<td>Routine</td>
<td>PREAMBLE DISPATCHER 2</td>
</tr>
<tr>
<td>TR.CHECK.IN.PT</td>
<td>Routine</td>
<td>PREAMBLE INIT.ACCIDENT 1</td>
</tr>
<tr>
<td>TR.CHECK.OUT.ACC</td>
<td>Routine</td>
<td>PREAMBLE TR.CHECK.IN.AMB 1</td>
</tr>
<tr>
<td>TR.CHECK.OUT.PT</td>
<td>Routine</td>
<td>PREAMBLE TR.CHECK.IN.PT 1</td>
</tr>
<tr>
<td>TR.DELIVER.PT</td>
<td>Routine</td>
<td>PREAMBLE TR.DELIVER.PT 1</td>
</tr>
<tr>
<td>TR.ENROUTE.HOSP</td>
<td>Routine</td>
<td>PREAMBLE TR.ENROUTE.HOSP 1</td>
</tr>
<tr>
<td>TR.GO.GREEN</td>
<td>Routine</td>
<td>PREAMBLE CHECK.RED 1</td>
</tr>
</tbody>
</table>

- 147 -
CLEAR. REDS 1
GET. HOSP 1
GO. OFF. RED 1
TR. GO. GREEN 1

TR. GO. RED Routine
PREAMBLE 1
CHECK. RED 1
TR. GO. RED 1

TR. ON. SCENE Routine
PREAMBLE 1
DISPATCHER 1
TR. ON. SCENE 1

TR. PICKUP. PT Routine
PREAMBLE 1
AMBULANCE. RUN 1
GET. PATIENT 1
TR. PICKUP. PT 1

TR. PROP Global variable Double
PREAMBLE 1
GET. SIM = INIT. ACCIDENT 1

TR. PVT. TR. PT Routine
PREAMBLE 1
PVT. TRAVEL 1
TR. PVT. TR. PT 1

TR. RESUS. PT Routine
PREAMBLE 1
AMBULANCE. RUN 1
PATIENT 1
TR. RESUS. PT 1

TR. SEND. AMB Routine
PREAMBLE 1
ASSIGN. AMB 1
TR. SEND. AMB 1

TR. STACK. ACC Routine
PREAMBLE 1
GET. AMB 1
TR. STACK. ACC 1

TR. TO. HOME Routine
PREAMBLE 1
DISPATCHER 1
TR. TO. HOME 1

TR. UNSTACK. ACC Routine
PREAMBLE 1
DISPATCHER 3
TR. UNSTACK. ACC 1

TRANSIT. TIME Permanent attribute Double
PREAMBLE 2
BUILD. CALL. LIST 3
BUILD. HOSP. LIST 3
GET. NET 2 =
GET. PATIENT 1
GET. TRAVEL. TIME 1
PRINT. NET 2
PVT. TRAVEL 1

TRANS. MODE Temporary attribute Integer
PREAMBLE 2
AMBULANCE. RUN 1 =
GET. PATIENT 1 =
PT. REPORT 1
PVT. TRAVEL 1 =

TRANS. TIME Temporary attribute Double
PREAMBLE 2
<table>
<thead>
<tr>
<th>Variable</th>
<th>Definition</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRAUMA</td>
<td>Define to mean trauma</td>
<td>PREAMBLE, AMBULANCE.RUN, INIT.ACCIDENT, TR.CHECK.IN.ACC, TR.CHECK.OUT.ACC, TR.ENROUTE.HOSP, TR.ON.SCE, TR.PICKUP.PT, TR.SEND.AMB, TR.STACK.ACC, TR.UNSTACK.ACC</td>
</tr>
<tr>
<td>TRAVEL</td>
<td>Routine</td>
<td>AMBULANCE.RUN, TRAVEL</td>
</tr>
<tr>
<td>TRAVEL.TIME</td>
<td>Permanent attribute Double</td>
<td>PREAMBLE, TRAVEL</td>
</tr>
<tr>
<td>TRF.RATE</td>
<td>Temporary attribute Double</td>
<td>PREAMBLE, PATIENT</td>
</tr>
<tr>
<td>TRF.START.TIME</td>
<td>Temporary attribute Double</td>
<td>PREAMBLE, PATIENT</td>
</tr>
<tr>
<td>TRUE</td>
<td>Define to mean true</td>
<td>PREAMBLE, ACCIDENT, AMBULANCE.RUN, BUILD.HOSP.LIST, CHECK.RED, DISPATCHER, DONE, FIND.HOSP, GET.AMB, GET.PATIENT, GO.GREEN, GO.RED, INIT.ACCIDENT, LIVING, LEFT.NO.PTS, PATIENT, PVT.TRAVEL</td>
</tr>
<tr>
<td>TRUNC.F</td>
<td>Library routine</td>
<td>NSP.F</td>
</tr>
<tr>
<td>TX.TIME</td>
<td>Temporary attribute Double</td>
<td>PREAMBLE, PATIENT, PT.REPORT</td>
</tr>
<tr>
<td>TYPE</td>
<td>Permanent attribute Integer</td>
<td>PREAMBLE, AMBULANCE.RUN, ASSIGN.AMB, BUILD.CALL.LIST, DISPATCHER, GET.AMB, GET.EMS</td>
</tr>
<tr>
<td>Category</td>
<td>Description</td>
<td>Example</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>UIB.R</td>
<td>Implied subscript</td>
<td>GET.PATIENT</td>
</tr>
<tr>
<td>UIB.W</td>
<td>Implied subscript</td>
<td>PRINT.NET</td>
</tr>
<tr>
<td>UNIFORM.F</td>
<td>Library routine</td>
<td>RECALL</td>
</tr>
<tr>
<td>UPDATE</td>
<td>Routine</td>
<td>MAIN</td>
</tr>
<tr>
<td>UPDATE.TIME</td>
<td>Permanent attribute Double</td>
<td>PASS.TIME</td>
</tr>
<tr>
<td>UPDATED</td>
<td>Temporary attribute Integer</td>
<td>PATIENT</td>
</tr>
<tr>
<td>V.SET</td>
<td>Set</td>
<td>UPDATE</td>
</tr>
<tr>
<td>WAIT.TIME</td>
<td>Temporary attribute Double</td>
<td>TIME</td>
</tr>
<tr>
<td>WAITING</td>
<td>Define to mean</td>
<td>WAITING</td>
</tr>
<tr>
<td>WEEKDAY.F</td>
<td>Library routine</td>
<td>WEEKDAY.F</td>
</tr>
<tr>
<td>WEIGHT</td>
<td>Temporary attribute Double</td>
<td>WEEKDAY.F</td>
</tr>
<tr>
<td>WORKING</td>
<td>Define to mean</td>
<td>WORKING</td>
</tr>
<tr>
<td>WRITE.CALL.LIST</td>
<td>Routine</td>
<td>WRITE.CALL.LIST</td>
</tr>
<tr>
<td>WRITE.HOSP.LIST</td>
<td>Routine</td>
<td>WRITE.HOSP.LIST</td>
</tr>
<tr>
<td>WRK</td>
<td>Define to mean</td>
<td>WRK</td>
</tr>
</tbody>
</table>
Appendix 3

Data Files

cems.dat -- general simulation control information

4 ;number of runs -- must be at least 2 for valid summary stats
.25 ;min number of days in a run
.5 ;trauma proportion
.5 ;average no. pts / accident
3.0 ;minutes before alarm turned in
3.0 ;minutes til 1st responder calls disp w/ accurate info
6.15 6.82 ;mean & sd secure time
.097 ;prob of needing secure time
2.22 ;minutes til pt found
10.4 ;minutes on scene
2.60 1.95 ;mean & sd deliver time
15.0 ;resus time
5.0 ;transfer time
5 ;TCRUISE -- time in minutes to attain cruising speed
3 ;min.amb -- minimum no. of ambulance's on a call list
.35 ;atol -- proportional diff in times deemed negligible
.30 ;htol -- same
.95 ;% of cts cutoff for minor trauma--note rts range of 0 to 7.8408
.90 ;cts <= this value goes to level 1 center
39 ;minor.time
19 ;major.time
4.0 ;mean launch+land time for helo
1 1 2 ;red.limit, green.limit, max.red.hosp
1 4 1 ;min and max red time in hours, no.reds per day
8.00 ; time of day to clear red status (assume sim starts at MN)

cont.dat -- continuous simulation control
.0003472 ;max.step ~0.5 minute
.00001157 ;min.step ~1.0 second
.1 ;abs.error
.01 ;rel.error

amb.dat -- ambulances
41 ;no. ambulances
1 4 D1 ;type (1=ground, 2=air), node of base, name
1 1 D4 ;letter = county, # = unit ID
1 1 D9
1 3 D7
1 6 D10
1 2 D13
1 10 D19
1 14 D20
1 19 D22
1 5 D23
1 8 D24
1 18 D25
1 7 D26
1 15 D28
1 11 D30
1 20 D32
1 9 D35

- 151 -
hosp.dat -- hospital information
12 ;no. hospitals -- must agree w/ no. of lines following!
UMC 1 95 1 5 1 ;University name, base, vol in 1000's, level, tcenter
BAP 2 35 2 3 1 ;Baptist capacity, can.divert (1=yes, 0=no)
MMC 14 30 3 2 1 ;Memorial watch coding! 4 => level 3!
MTH 1 15 4 1 1 ;Methodist
STL 15 15 3 1 1 ;St Luke's
BCH 13 15 3 1 1 ;Beaches
STV 6 35 3 2 1 ;St Vincent's
RVS 6 12 4 1 1 ;Riverside
HUM 28 18 3 1 0 ;Humana (Orange Park Community)
NGH 26 15 4 1 0 ;Nassau General
FLG 34 18 4 1 0 ;Flagler
PCH 31 18 4 1 0 ;Putnam Community

net.dat -- nodes and arcs
37 ; n.node -- centered around Rescue stations
1 SPF 30.347 81.639 ;Springfield node, name, lat, long
2 SBK 30.317 81.623 ;Southbank
3 WLX 30.347 81.665 ;West Jacksonville
4 EJX 30.347 81.623 ;East Jacksonville
5 AVD 30.319 81.672 ;Avondale
6 RVS 30.308 81.663 ;Riverside
7 PKV 30.364 81.718 ;Pickettville
8 LMT 30.415 81.67 ;Lem Turner
9 OWY 30.466 81.619 ;Oceanway
10 ARL 30.338 81.585 ;Arlington
11 RGY 30.323 81.551 ;Regency
12 ATB 30.33 81.408 ;Atlantic Beach
13 JXB 30.289 81.405 ;Jacksonville Beach
14 SSD 30.29 81.579 ;Southside
15 JTB 30.25 81.575 ;J Turner Butler
16 SPL 30.289 81.449 ;San Pablo
17 MND 30.17 81.609 ;Mandarin
18 TMQ 30.25 81.704 ;Timuquana
19 WSD 30.25 81.734 ;Westside
20 MWH 30.321 81.773 ;Marietta-Whitehouse
21 MCL 30.289 82.148 ;McClenney
<table>
<thead>
<tr>
<th></th>
<th>City</th>
<th>Speed (mph)</th>
<th>Range (mi)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Bryceville</td>
<td>30.398</td>
<td>81.985</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Callahan</td>
<td>30.566</td>
<td>81.889</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Hilliard</td>
<td>30.687</td>
<td>81.966</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Yulee</td>
<td>30.628</td>
<td>81.569</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Fernandina Beach</td>
<td>30.67</td>
<td>81.443</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Orange Park East</td>
<td>30.163</td>
<td>81.685</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Orange Park West</td>
<td>30.178</td>
<td>81.7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Green Cove Springs</td>
<td>30</td>
<td>81.587</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Middleburg</td>
<td>30.064</td>
<td>81.875</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Palatka</td>
<td>29.648</td>
<td>81.636</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Ponte Vedra Beach</td>
<td>30.238</td>
<td>81.398</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>Switzerland</td>
<td>30.102</td>
<td>81.5</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>St Augustine</td>
<td>29.867</td>
<td>81.347</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>Crescent Beach</td>
<td>29.83</td>
<td>81.219</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Hastings</td>
<td>29.796</td>
<td>81.5</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>King's Bay</td>
<td>30.796</td>
<td>81.545</td>
<td></td>
</tr>
</tbody>
</table>

Airspeed, range

1. 2 10 5.0
2. 3 5 0.0
3. 4 5 0.0
4. 5 10 0.0
5. 6 10 0.0
6. 8 10 0.0
7. 9 10 0.0
8. 1 10 5.0
9. 4 10 5.0
10. 2 10 10 0.0
11. 2 14 15 0.0
12. 2 15 10 0.0
13. 2 17 20 0.0
14. 2 5 10 5.0
15. 3 1 5 0.0
16. 3 19 20 0.0
17. 3 20 20 0.0
18. 3 7 5 0.0
19. 3 23 20 0.0
20. 3 8 10 0.0
21. 4 1 5 0.0
22. 4 10 5 5.0
23. 4 1 5 5.0
24. 2 2 10 5.0
25. 5 1 10 0.0
26. 5 2 10 0.0
27. 5 6 5 0.0
28. 5 18 10 0.0
29. 5 19 10 0.0
30. 5 20 10 0.0
31. 6 1 10 0.0
32. 6 5 5 0.0
33. 7 3 5 0.0
34. 7 19 15 0.0
35. 7 20 15 0.0
36. 7 8 10 0.0
37. 8 1 10 0.0
38. 8 3 10 0.0
39. 8 7 10 0.0
40. 8 23 20 0.0
41. 8 9 10 0.0
42. 9 1 10 0.0
43. 9 8 10 0.0
44. 9 25 20 0.0
45. 9 11 15 0.0
46. 10 4 5 5.0
47. 10 11 5 0.0
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>14</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>5</td>
<td>0.0</td>
</tr>
<tr>
<td>11</td>
<td>9</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>11</td>
<td>15</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>11</td>
<td>14</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>12</td>
<td>16</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>13</td>
<td>12</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>13</td>
<td>16</td>
<td>5</td>
<td>0.0</td>
</tr>
<tr>
<td>13</td>
<td>32</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>13</td>
<td>15</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>14</td>
<td>2</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>14</td>
<td>11</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>14</td>
<td>16</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>7.5</td>
<td>0.0</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>7.5</td>
<td>0.0</td>
</tr>
<tr>
<td>15</td>
<td>11</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>15</td>
<td>17</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>15</td>
<td>13</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>16</td>
<td>14</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>16</td>
<td>12</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>16</td>
<td>13</td>
<td>5</td>
<td>0.0</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>17</td>
<td>15</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>17</td>
<td>33</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>17</td>
<td>27</td>
<td>20</td>
<td>5.0</td>
</tr>
<tr>
<td>18</td>
<td>19</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>18</td>
<td>5</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>18</td>
<td>27</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>19</td>
<td>18</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>19</td>
<td>7</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>19</td>
<td>3</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>19</td>
<td>5</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>19</td>
<td>18</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>20</td>
<td>19</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>20</td>
<td>3</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>20</td>
<td>7</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>20</td>
<td>22</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>21</td>
<td>20</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>22</td>
<td>20</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>22</td>
<td>23</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>23</td>
<td>22</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>23</td>
<td>24</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>23</td>
<td>25</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>23</td>
<td>8</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>23</td>
<td>3</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>23</td>
<td>24</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>25</td>
<td>23</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>25</td>
<td>9</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>25</td>
<td>26</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>25</td>
<td>37</td>
<td>25</td>
<td>0.0</td>
</tr>
<tr>
<td>26</td>
<td>25</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>27</td>
<td>14</td>
<td>20</td>
<td>5.0</td>
</tr>
<tr>
<td>27</td>
<td>18</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>27</td>
<td>29</td>
<td>15</td>
<td>0.0</td>
</tr>
</tbody>
</table>

- 154 -
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>28</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>28</td>
<td>19</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>28</td>
<td>27</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>28</td>
<td>30</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>29</td>
<td>27</td>
<td>15</td>
<td>0.0</td>
</tr>
<tr>
<td>29</td>
<td>30</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>29</td>
<td>31</td>
<td>30</td>
<td>0.0</td>
</tr>
<tr>
<td>29</td>
<td>33</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>30</td>
<td>28</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>30</td>
<td>29</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>31</td>
<td>29</td>
<td>30</td>
<td>0.0</td>
</tr>
<tr>
<td>31</td>
<td>36</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>32</td>
<td>13</td>
<td>10</td>
<td>0.0</td>
</tr>
<tr>
<td>32</td>
<td>34</td>
<td>35</td>
<td>0.0</td>
</tr>
<tr>
<td>33</td>
<td>17</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>33</td>
<td>29</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>33</td>
<td>34</td>
<td>35</td>
<td>0.0</td>
</tr>
<tr>
<td>33</td>
<td>36</td>
<td>30</td>
<td>0.0</td>
</tr>
<tr>
<td>34</td>
<td>33</td>
<td>35</td>
<td>0.0</td>
</tr>
<tr>
<td>34</td>
<td>32</td>
<td>35</td>
<td>0.0</td>
</tr>
<tr>
<td>34</td>
<td>35</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>34</td>
<td>36</td>
<td>25</td>
<td>0.0</td>
</tr>
<tr>
<td>35</td>
<td>34</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>35</td>
<td>36</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>36</td>
<td>35</td>
<td>20</td>
<td>0.0</td>
</tr>
<tr>
<td>36</td>
<td>33</td>
<td>30</td>
<td>0.0</td>
</tr>
<tr>
<td>37</td>
<td>25</td>
<td>25</td>
<td>0.0</td>
</tr>
</tbody>
</table>

space.dat -- relative incidents per node in node order

5
5
5
4
2
2
4
2
3
2
2
4
2
4
2
2
3
3
4
2
2
1
2
1
1
4
4
4
3
2
4
3
1
4
wkrate.dat -- time distribution of incidents
84  :number of entries
  0  0  :first number is time,
  2  4.77  :second is number of events
  4  3.81  
  6  1.14  ; in that time period
  8  2.07  ; assumed to "wrap around" after reaching the last
 10  2.55  
 12  3.18  ; time period
 14  3.86  
 16  4.55  
 18  5.66  
 20  5.34  
 22  5.28  
 24  5.78  
 26  3.88  
 28  3.09  
 30  .93  
 32  1.68  
 34  2.07  
 36  2.59  
 38  3.14  
 40  3.69  
 42  4.60  
 44  4.34  
 46  4.29  
 48  4.70  
 50  3.88  
 52  3.09  
 54  .93  
 56  1.68  
 58  2.07  
 60  2.59  
 62  3.14  
 64  3.69  
 66  4.60  
 68  4.34  
 70  4.29  
 72  4.70  
 74  4.03  
 76  3.21  
 78  .96  
 80  1.75  
 82  2.15  
 84  2.68  
 86  3.26  
 88  3.83  
 90  4.78  
 92  4.51  
 94  4.46  
 96  4.88  
 98  4.03  
100  3.21  
102  .96  
104  1.75  
106  2.15  
108  2.68  
110  3.26  
112  3.83  

- 156 -
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>114</td>
<td>4.78</td>
</tr>
<tr>
<td>116</td>
<td>4.51</td>
</tr>
<tr>
<td>118</td>
<td>4.46</td>
</tr>
<tr>
<td>120</td>
<td>4.88</td>
</tr>
<tr>
<td>122</td>
<td>4.17</td>
</tr>
<tr>
<td>124</td>
<td>3.33</td>
</tr>
<tr>
<td>126</td>
<td>1.00</td>
</tr>
<tr>
<td>128</td>
<td>1.81</td>
</tr>
<tr>
<td>130</td>
<td>2.23</td>
</tr>
<tr>
<td>132</td>
<td>2.78</td>
</tr>
<tr>
<td>134</td>
<td>3.38</td>
</tr>
<tr>
<td>136</td>
<td>3.08</td>
</tr>
<tr>
<td>138</td>
<td>4.96</td>
</tr>
<tr>
<td>140</td>
<td>4.67</td>
</tr>
<tr>
<td>142</td>
<td>4.82</td>
</tr>
<tr>
<td>144</td>
<td>5.01</td>
</tr>
<tr>
<td>146</td>
<td>5.07</td>
</tr>
<tr>
<td>148</td>
<td>4.04</td>
</tr>
<tr>
<td>150</td>
<td>1.21</td>
</tr>
<tr>
<td>152</td>
<td>2.20</td>
</tr>
<tr>
<td>154</td>
<td>2.71</td>
</tr>
<tr>
<td>156</td>
<td>3.38</td>
</tr>
<tr>
<td>158</td>
<td>4.11</td>
</tr>
<tr>
<td>160</td>
<td>4.83</td>
</tr>
<tr>
<td>162</td>
<td>6.07</td>
</tr>
<tr>
<td>164</td>
<td>5.68</td>
</tr>
<tr>
<td>166</td>
<td>5.61</td>
</tr>
<tr>
<td>168</td>
<td>6.14</td>
</tr>
</tbody>
</table>

seed.dat -- random number stream seeds
28 ;no.streams -- must agree with no. lines -- these are 1M apart

683743814
604901985
72646604
622401386
1645973084
1901633463
67784357
2026948561
1545929719
547070247
1110948479
1400311458
1471803249
1232207518
195239450
281826375
416426318
380841429
1055454678
711617330
1416275180
780018608
1357689651
2130853749
152149214
550317865
32645035
871378447
The following two files are produced by the best.route routine

call.dat -- node, ambulances to be called in order
1 3 2 4 1 40 41
2 6 2 3 1 10 7 14 40 41
3 4 2 3 13 40 41
4 1 2 3 7 40 41
5 10 5 2 3 6 12 9 16 40 41
6 5 10 2 3 40 41
7 13 4 2 3 11 40 41
8 11 2 3 4 13 17 40 41
9 17 2 3 11 40 41
10 7 1 15 40 41
11 15 7 1 8 40 41
12 18 21 20 40 41
13 21 20 18 36 40 41
14 8 14 7 15 40 41
15 14 8 6 19 40 41
16 20 21 18 40 41
17 19 14 8 40 41
18 12 10 9 40 41
19 9 10 12 40 41
20 16 10 5 13 9 27 40 41
21 28 16 10 40 41
22 27 16 26 40 41
23 26 25 4 11 27 24 40 41
24 25 26 4 11 27 24 40 41
25 24 17 26 22 23 40 41
26 23 22 24 40 41
27 30 32 12 29 40 41
28 32 30 9 40 41
29 29 30 31 37 40 41
30 31 32 29 40 41
31 33 38 29 40 41
32 36 21 20 40 41
33 37 19 29 40 41
34 35 34 38 40 41
35 34 35 38 40 41
36 38 37 34 35 40 41
37 39 24 17 26 22 23 40 41

go.dat -- node, hospital id, hosp level, . . .
1 4 4 1 1 2 2 7 3
2 2 2 1 1 4 4 5 3
3 1 1 4 4 2 2 7 3
4 1 1 4 4 2 2
5 7 3 8 4 1 1
6 8 4 7 3 1 1
7 1 1 4 4 2 2 7 3
8 1 1 4 4 2 2 7 3
9 1 1 4 4 2 2 7 3 3 3
10 1 1 4 4 2 2 3 3
11 3 3 1 1
12 6 3 1 1
13 6 3 1 1
14 3 3 5 3 1 1
15 5 3 3 1 1
16 5 3 1 1
17 5 3 1 1
18 7 3 8 4 1 1 4 4 2 2
19 7 3 8 4 9 3 1 1 4 4 2 2
20 7 3 8 4 1 1 4 4 2 2
21 7 3 8 4 1 1 4 4 2 2

- 158 -
<p>| | | | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>7</td>
<td>3</td>
<td>8</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td>8</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>25</td>
<td>10</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>26</td>
<td>10</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>27</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>9</td>
<td>3</td>
<td>12</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>12</td>
<td>4</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>5</td>
<td>3</td>
<td>11</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>11</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>11</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>11</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>37</td>
<td>10</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

- 159 -
network structure
weights represent arterial route travel time between node centers

node 1 (SPF) has outdegree 7
arc to 2 (SBK) of weight 10.00
arc to 3 (WJX) of weight 5.00
arc to 4 (EJX) of weight 5.00
arc to 5 (AVD) of weight 10.00
arc to 6 (RVS) of weight 10.00
arc to 8 (LMT) of weight 10.00
arc to 9 (OWY) of weight 10.00

node 2 (SBK) has outdegree 7
arc to 1 (SPF) of weight 10.00
arc to 4 (EJX) of weight 10.00
arc to 10 (ARL) of weight 10.00
arc to 14 (SSD) of weight 15.00
arc to 15 (JTB) of weight 10.00
arc to 17 (MND) of weight 20.00
arc to 5 (AVD) of weight 10.00

node 3 (WJX) has outdegree 6
arc to 1 (SPF) of weight 5.00
arc to 19 (WSD) of weight 20.00
arc to 20 (MWH) of weight 20.00
arc to 7 (PKV) of weight 5.00
arc to 23 (CAL) of weight 20.00
arc to 8 (LMT) of weight 10.00

node 4 (EJX) has outdegree 3
arc to 1 (SPF) of weight 5.00
arc to 10 (ARL) of weight 5.00
arc to 2 (SBK) of weight 10.00

node 5 (AVD) has outdegree 6
arc to 1 (SPF) of weight 10.00
arc to 2 (SBK) of weight 10.00
arc to 6 (RVS) of weight 5.00
arc to 18 (TMQ) of weight 10.00
arc to 19 (WSD) of weight 10.00
arc to 20 (MWH) of weight 10.00

node 6 (RVS) has outdegree 2
arc to 1 (SPF) of weight 10.00
arc to 5 (AVD) of weight 5.00

node 7 (PKV) has outdegree 4
arc to 3 (WJX) of weight 5.00
arc to 19 (WSD) of weight 15.00
arc to 20 (MWH) of weight 15.00
arc to 8 (LMT) of weight 10.00

node 8 (LMT) has outdegree 5
arc to 1 (SPF) of weight 10.00
arc to 3 (WJX) of weight 10.00
arc to 7 (PKV) of weight 10.00
arc to 23 (CAL) of weight 20.00
arc to 9 (OWY) of weight 10.00

node 9 (OWY) has outdegree 4
arc to 1 (SPF) of weight 10.00
arc to 8 (LMT) of weight 10.00
arc to 25 (YUL) of weight 20.00
arc to 11 (RGY) of weight 15.00

node 10 (ARL) has outdegree 4
arc to 4 (EJX) of weight 5.00
arc to 11 (RGY) of weight 5.00
arc to 14 (SSD) of weight 10.00
arc to 2 (SBK) of weight 10.00

node 11 (RGY) has outdegree 5
arc to 10 (ARL) of weight 5.00
arc to 9 (OWY) of weight 15.00
arc to 12 (ATB) of weight 20.00
arc to 15 (JTB) of weight 15.00
arc to 14 (SSD) of weight 10.00

node 12 (ATB) has outdegree 3
arc to 11 (RGY) of weight 20.00
arc to 16 (SPL) of weight 10.00
arc to 13 (JXB) of weight 10.00

node 13 (JXB) has outdegree 4
arc to 12 (ATB) of weight 10.00
arc to 16 (SPL) of weight 5.00
arc to 32 (PVB) of weight 10.00
arc to 15 (JTB) of weight 15.00

node 14 (SSD) has outdegree 5
arc to 2 (SBK) of weight 15.00
arc to 10 (ARL) of weight 10.00
arc to 11 (RGY) of weight 10.00
arc to 16 (SPL) of weight 15.00
arc to 15 (JTB) of weight 7.50

node 15 (JTB) has outdegree 5
arc to 2 (SBK) of weight 10.00
arc to 14 (SSD) of weight 7.50
arc to 11 (RGY) of weight 15.00
arc to 17 (MND) of weight 10.00
arc to 13 (JXB) of weight 15.00

node 16 (SPL) has outdegree 3
arc to 14 (SSD) of weight 15.00
arc to 12 (ATB) of weight 10.00
arc to 13 (JXB) of weight 5.00

node 17 (MND) has outdegree 4
arc to 2 (SBK) of weight 20.00
arc to 15 (JTB) of weight 10.00
arc to 33 (SNZ) of weight 20.00
arc to 27 (OPE) of weight 20.00

node 18 (TMQ) has outdegree 3
arc to 19 (WSD) of weight 10.00
arc to 5 (AVD) of weight 10.00
arc to 27 (OPE) of weight 15.00

node 19 (WSD) has outdegree 6
arc to 28 (OPW) of weight 15.00
arc to 20 (MWH) of weight 15.00
arc to 7 (PKV) of weight 15.00
arc to 3 (WJX) of weight 20.00
arc to 5 (AVD) of weight 10.00
arc to 18 (TMQ) of weight 10.00

node 20 (MWH) has outdegree 6
arc to 19 (WSD) of weight 15.00
arc to 5 (AVD) of weight 10.00
arc to 3 (WJX) of weight 20.00
arc to 7 (PKV) of weight 15.00
arc to 22 (BRY) of weight 15.00
arc to 21 (MCL) of weight 20.00

node 21 (MCL) has outdegree 1
arc to 20 (MWH) of weight 20.00
node 22 (BRY) has outdegree 2
  arc to 20 (MWH) of weight 15.00
  arc to 23 (CAL) of weight 20.00
node 23 (CAL) has outdegree 5
  arc to 22 (BRY) of weight 20.00
  arc to 24 (HIL) of weight 15.00
  arc to 25 (YUL) of weight 20.00
  arc to 8 (LMT) of weight 20.00
  arc to 3 (WJX) of weight 20.00
node 24 (HIL) has outdegree 1
  arc to 23 (CAL) of weight 20.00
node 25 (YUL) has outdegree 4
  arc to 23 (CAL) of weight 20.00
  arc to 9 (OWY) of weight 20.00
  arc to 26 (FDB) of weight 20.00
  arc to 37 (KGB) of weight 25.00
node 26 (FDB) has outdegree 1
  arc to 25 (YUL) of weight 20.00
node 27 (OPE) has outdegree 4
  arc to 17 (MND) of weight 20.00
  arc to 18 (TMQ) of weight 15.00
  arc to 29 (GCS) of weight 15.00
  arc to 28 (OPW) of weight 10.00
node 28 (OPW) has outdegree 3
  arc to 19 (WSD) of weight 15.00
  arc to 27 (OPE) of weight 10.00
  arc to 30 (MBG) of weight 20.00
node 29 (GCS) has outdegree 4
  arc to 27 (OPE) of weight 15.00
  arc to 30 (MBG) of weight 20.00
  arc to 31 (PLK) of weight 30.00
  arc to 33 (SWZ) of weight 20.00
node 30 (MBG) has outdegree 2
  arc to 28 (OPW) of weight 20.00
  arc to 29 (GCS) of weight 20.00
node 31 (PLK) has outdegree 2
  arc to 29 (GCS) of weight 30.00
  arc to 36 (HST) of weight 20.00
node 32 (PVB) has outdegree 2
  arc to 13 (JXB) of weight 10.00
  arc to 34 (STA) of weight 35.00
node 33 (SWZ) has outdegree 4
  arc to 17 (MND) of weight 20.00
  arc to 29 (GCS) of weight 20.00
  arc to 34 (STA) of weight 35.00
  arc to 36 (HST) of weight 30.00
node 34 (STA) has outdegree 4
  arc to 33 (SWZ) of weight 35.00
  arc to 32 (PVB) of weight 35.00
  arc to 35 (CRB) of weight 20.00
  arc to 36 (HST) of weight 25.00
node 35 (CRB) has outdegree 2
  arc to 34 (STA) of weight 20.00
  arc to 36 (HST) of weight 20.00
node 36 (HST) has outdegree 2
  arc to 35 (CRB) of weight 20.00
  arc to 33 (SWZ) of weight 30.00
node 37 (KGB) has outdegree 1
  arc to 25 (YUL) of weight 25.00

hospital locations
  UMC, level 1, 5 beds, in SPF

- 162 -
BAP, level 2, 3 beds, in SBK
MMC, level 2a, 2 beds, in SSD
MTH, level 3, 1 beds, in SPF
STL, level 2a, 1 beds, in JTB
BCH, level 2a, 1 beds, in JXB
STV, level 2a, 2 beds, in RVS
RVS, level 3, 1 beds, in RVS
HUM, level 2a, 1 beds, in OPW
NGH, level 3, 1 beds, in FDB
FLG, level 3, 1 beds, in STA
PCH, level 3, 1 beds, in PLK

ambulance call list

SPF will request these ambulances
D9 from SPF with mean travel time 9.26 min
D4 from SPF with mean travel time 9.26 min
D7 from WJX with mean travel time 10.00 min
D1 from EJX with mean travel time 10.00 min
H1 from SPF with mean travel time 5.71 min
H2 from SBK with mean travel time 4.74 min

SBK will request these ambulances
D13 from SBK with mean travel time 11.02 min
D4 from SPF with mean travel time 14.14 min
D9 from SPF with mean travel time 14.14 min
D1 from EJX with mean travel time 10.00 min
D23 from AVD with mean travel time 14.14 min
D19 from ARL with mean travel time 14.14 min
D29 from JTB with mean travel time 14.14 min
H1 from SPF with mean travel time 4.74 min
H2 from SBK with mean travel time 6.43 min

WJX will request these ambulances
D7 from WJX with mean travel time 11.55 min
D4 from SPF with mean travel time 10.00 min
D9 from SPF with mean travel time 14.14 min
D26 from PKV with mean travel time 10.00 min
H1 from SPF with mean travel time 4.30 min
H2 from SBK with mean travel time 4.85 min

EJX will request these ambulances
D1 from EJX with mean travel time 8.16 min
D4 from SPF with mean travel time 14.14 min
D9 from SPF with mean travel time 14.14 min
D19 from ARL with mean travel time 10.00 min
H1 from SPF with mean travel time 4.18 min
H2 from SBK with mean travel time 4.72 min

AVD will request these ambulances
D23 from AVD with mean travel time 9.57 min
D10 from RVS with mean travel time 10.00 min
D4 from SPF with mean travel time 14.14 min
D9 from SPF with mean travel time 14.14 min
D13 from SBK with mean travel time 14.14 min
D25 from TMQ with mean travel time 14.14 min
D22 from WSD with mean travel time 14.14 min
D32 from MWH with mean travel time 14.14 min
H1 from SPF with mean travel time 4.77 min
H2 from SBK with mean travel time 4.54 min

RVS will request these ambulances
D10 from RVS with mean travel time 8.66 min
D23 from AVD with mean travel time 14.14 min
D4 from SPF with mean travel time 14.14 min
D9 from SPF with mean travel time 14.14 min
H1 from SPF with mean travel time 4.98 min
H2 from SBK with mean travel time 4.49 min

PKV will request these ambulances
<table>
<thead>
<tr>
<th>Requesting Agency</th>
<th>Ambulance ID</th>
<th>From Location</th>
<th>Mean Travel Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>LMT</td>
<td>D26</td>
<td>PKV</td>
<td>10.61 min</td>
</tr>
<tr>
<td></td>
<td>D7</td>
<td>WJX</td>
<td>10.00 min</td>
</tr>
<tr>
<td></td>
<td>D4</td>
<td>SPF</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>D9</td>
<td>SPF</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>D24</td>
<td>LMT</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>H1</td>
<td>SPF</td>
<td>5.00 min</td>
</tr>
<tr>
<td></td>
<td>H2</td>
<td>SBK</td>
<td>5.53 min</td>
</tr>
<tr>
<td></td>
<td>LMT</td>
<td>request these ambulances</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D24</td>
<td>LMT</td>
<td>10.95 min</td>
</tr>
<tr>
<td></td>
<td>D4</td>
<td>SPF</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>D9</td>
<td>SPF</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>D7</td>
<td>WJX</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>D26</td>
<td>PKV</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>D35</td>
<td>OWY</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>H1</td>
<td>SPF</td>
<td>5.67 min</td>
</tr>
<tr>
<td></td>
<td>H2</td>
<td>SBK</td>
<td>6.41 min</td>
</tr>
<tr>
<td></td>
<td>OWY</td>
<td>request these ambulances</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D35</td>
<td>OWY</td>
<td>11.73 min</td>
</tr>
<tr>
<td></td>
<td>D4</td>
<td>SPF</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>D9</td>
<td>SPF</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>D24</td>
<td>LMT</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>H1</td>
<td>SPF</td>
<td>6.87 min</td>
</tr>
<tr>
<td></td>
<td>H2</td>
<td>SBK</td>
<td>7.58 min</td>
</tr>
<tr>
<td></td>
<td>ARL</td>
<td>request these ambulances</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D19</td>
<td>ARL</td>
<td>8.66 min</td>
</tr>
<tr>
<td></td>
<td>D1</td>
<td>EJX</td>
<td>10.00 min</td>
</tr>
<tr>
<td></td>
<td>D30</td>
<td>RGY</td>
<td>10.00 min</td>
</tr>
<tr>
<td></td>
<td>H1</td>
<td>SPF</td>
<td>4.66 min</td>
</tr>
<tr>
<td></td>
<td>H2</td>
<td>SBK</td>
<td>4.65 min</td>
</tr>
<tr>
<td></td>
<td>RGY</td>
<td>request these ambulances</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D30</td>
<td>RGY</td>
<td>11.40 min</td>
</tr>
<tr>
<td></td>
<td>D19</td>
<td>ARL</td>
<td>10.00 min</td>
</tr>
<tr>
<td></td>
<td>D1</td>
<td>EJX</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>D1</td>
<td>EJX</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>D20</td>
<td>SSD</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>H1</td>
<td>SPF</td>
<td>5.17 min</td>
</tr>
<tr>
<td></td>
<td>H2</td>
<td>SBK</td>
<td>4.80 min</td>
</tr>
<tr>
<td></td>
<td>ATB</td>
<td>request these ambulances</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D41</td>
<td>ATB</td>
<td>11.55 min</td>
</tr>
<tr>
<td></td>
<td>D71</td>
<td>JXB</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>D50</td>
<td>SPL</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>H1</td>
<td>SPF</td>
<td>6.70 min</td>
</tr>
<tr>
<td></td>
<td>H2</td>
<td>SBK</td>
<td>6.37 min</td>
</tr>
<tr>
<td></td>
<td>JXB</td>
<td>request these ambulances</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D71</td>
<td>JXB</td>
<td>10.00 min</td>
</tr>
<tr>
<td></td>
<td>D50</td>
<td>SPL</td>
<td>10.00 min</td>
</tr>
<tr>
<td></td>
<td>D41</td>
<td>ATB</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>S81</td>
<td>PVB</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>H1</td>
<td>SPF</td>
<td>7.04 min</td>
</tr>
<tr>
<td></td>
<td>H2</td>
<td>SBK</td>
<td>6.47 min</td>
</tr>
<tr>
<td></td>
<td>SSD</td>
<td>request these ambulances</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D20</td>
<td>SSD</td>
<td>10.72 min</td>
</tr>
<tr>
<td></td>
<td>D28</td>
<td>JTB</td>
<td>12.25 min</td>
</tr>
<tr>
<td></td>
<td>D19</td>
<td>ARL</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>D30</td>
<td>RGY</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>H1</td>
<td>SPF</td>
<td>5.53 min</td>
</tr>
<tr>
<td></td>
<td>H2</td>
<td>SBK</td>
<td>4.81 min</td>
</tr>
<tr>
<td></td>
<td>JTB</td>
<td>request these ambulances</td>
<td></td>
</tr>
<tr>
<td></td>
<td>D28</td>
<td>JTB</td>
<td>10.72 min</td>
</tr>
<tr>
<td></td>
<td>D20</td>
<td>SSD</td>
<td>12.25 min</td>
</tr>
<tr>
<td></td>
<td>D13</td>
<td>SBK</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>D42</td>
<td>MND</td>
<td>14.14 min</td>
</tr>
<tr>
<td></td>
<td>H1</td>
<td>SPF</td>
<td>6.44 min</td>
</tr>
<tr>
<td></td>
<td>H2</td>
<td>SBK</td>
<td>5.69 min</td>
</tr>
</tbody>
</table>
SPL will request these ambulances
D50 from SPL with mean travel time 10.00 min
D71 from JXB with mean travel time 10.00 min
D41 from ATB with mean travel time 14.14 min
H1 from SPF with mean travel time 6.60 min
H2 from SBK with mean travel time 6.01 min

MND will request these ambulances
D42 from MND with mean travel time 13.23 min
D28 from JTB with mean travel time 14.14 min
D20 from SSD with mean travel time 22.50 min
H1 from SPF with mean travel time 8.26 min
H2 from SBK with mean travel time 7.53 min

TMQ will request these ambulances
D25 from TMQ with mean travel time 10.80 min
D23 from AVD with mean travel time 14.14 min
D22 from WSD with mean travel time 14.14 min
H1 from SPF with mean travel time 6.45 min
H2 from SBK with mean travel time 5.83 min

MWH will request these ambulances
D32 from MWH with mean travel time 12.58 min
D23 from AVD with mean travel time 14.14 min
D10 from RVS with mean travel time 20.00 min
D26 from PKV with mean travel time 20.00 min
D22 from WSD with mean travel time 20.00 min
N6 from BRY with mean travel time 20.00 min
H1 from SPF with mean travel time 5.67 min
H2 from SBK with mean travel time 5.64 min

MCL will request these ambulances
B32 from MCL with mean travel time 14.14 min
D32 from MWH with mean travel time 25.00 min
D23 from AVD with mean travel time 35.00 min
H1 from SPF with mean travel time 10.04 min
H2 from SBK with mean travel time 9.77 min

BRY will request these ambulances
N6 from BRY with mean travel time 13.23 min
D32 from MWH with mean travel time 20.00 min
N5 from CAL with mean travel time 25.00 min
H1 from SPF with mean travel time 8.18 min
H2 from SBK with mean travel time 8.40 min

CAL will request these ambulances
N5 from CAL with mean travel time 13.78 min
N4 from HIL with mean travel time 20.00 min
D7 from WJX with mean travel time 25.00 min
D24 from LMT with mean travel time 25.00 min
N6 from BRY with mean travel time 25.00 min
N3 from YUL with mean travel time 25.00 min
H1 from SPF with mean travel time 9.87 min
H2 from SBK with mean travel time 10.51 min

HIL will request these ambulances
N4 from HIL with mean travel time 12.25 min
N5 from CAL with mean travel time 20.00 min
D7 from WJX with mean travel time 40.00 min
D24 from LMT with mean travel time 40.00 min
N6 from BRY with mean travel time 40.00 min
N3 from YUL with mean travel time 40.00 min
H1 from SPF with mean travel time 12.99 min
H2 from SBK with mean travel time 13.64 min
YUL will request these ambulances
N3 from YUL with mean travel time 15.62 min
D35 from OWY with mean travel time 25.00 min
H5 from CAL with mean travel time 25.00 min
N1 from FDB with mean travel time 25.00 min
N2 from FDB with mean travel time 25.00 min
H1 from SPF with mean travel time 10.79 min
H2 from SBK with mean travel time 11.49 min

FDB will request these ambulances
N2 from FDB with mean travel time 14.14 min
N1 from FDB with mean travel time 14.14 min
N3 from YUL with mean travel time 25.00 min
H1 from SPF with mean travel time 12.08 min
H2 from SBK with mean travel time 12.70 min

OPE will request these ambulances
C72 from OPE with mean travel time 12.25 min
C76 from OPW with mean travel time 14.14 min
D25 from TMQ with mean travel time 20.00 min
C70 from GCS with mean travel time 20.00 min
H1 from SPF with mean travel time 8.45 min
H2 from SBK with mean travel time 7.76 min

OPW will request these ambulances
C76 from OPW with mean travel time 12.25 min
C72 from OPE with mean travel time 14.14 min
D22 from WSD with mean travel time 20.00 min
H1 from SPF with mean travel time 8.12 min
H2 from SBK with mean travel time 7.44 min

GCS will request these ambulances
C70 from GCS with mean travel time 15.62 min
C72 from OPE with mean travel time 20.00 min
C74 from MBG with mean travel time 25.00 min
S82 from SWZ with mean travel time 25.00 min
H1 from SPF with mean travel time 12.35 min
H2 from SBK with mean travel time 11.62 min

MBG will request these ambulances
C74 from MBG with mean travel time 14.14 min
C76 from OPW with mean travel time 25.00 min
C70 from GCS with mean travel time 25.00 min
H1 from SPF with mean travel time 11.32 min
H2 from SBK with mean travel time 10.67 min

PLK will request these ambulances
P1 from PLK with mean travel time 17.50 min
S88 from HST with mean travel time 85.00 min
C70 from GCS with mean travel time 35.00 min
H1 from SPF with mean travel time 20.78 min
H2 from SBK with mean travel time 20.06 min

PVB will request these ambulances
S81 from PVB with mean travel time 16.25 min
D71 from JXB with mean travel time 14.14 min
D50 from SPL with mean travel time 20.00 min
H1 from SPF with mean travel time 7.82 min
H2 from SBK with mean travel time 7.10 min

SWZ will request these ambulances
S82 from SWZ with mean travel time 18.12 min
D42 from MND with mean travel time 25.00 min
C70 from GCS with mean travel time 25.00 min
H1 from SPF with mean travel time 10.10 min
H2 from SBK with mean travel time 9.33 min

STA will request these ambulances
S83 from STA with mean travel time 19.37 min
S81 from STA with mean travel time 19.37 min
S88 from HST with mean travel time 45.00 min
H1 from SPF with mean travel time 16.00 min
H2 from SBK with mean travel time 15.21 min
CRB will request these ambulances
SB8 from STA with mean travel time 25.00 min
SB3 from STA with mean travel time 25.00 min
H1 from SPA with mean travel time 17.32 min
H2 from SBK with mean travel time 16.49 min
HST will request these ambulances
SB8 from HST with mean travel time 25.00 min
SB2 from SWZ with mean travel time 30.00 min
SB2 from STA with mean travel time 30.00 min
H1 from SPA with mean travel time 17.46 min
H2 from SBK with mean travel time 16.72 min
KGB will request these ambulances
G1 from KGB with mean travel time 17.50 min
N3 from YUL with mean travel time 30.00 min
D35 from OWY with mean travel time 50.00 min
N4 from CAL with mean travel time 50.00 min
N1 from FDB with mean travel time 50.00 min
N2 from FDB with mean travel time 50.00 min
H1 from SPA with mean travel time 14.69 min
H2 from SBK with mean travel time 15.38 min
hospital dispatch list
SPF victims will go to these hospitals
MTH, level 3, in SPF, with mean travel time 9.26 min
UMC, level 1, in SPF, with mean travel time 9.26 min
BAP, level 2, in SBK, with mean travel time 14.14 min
STV, level 2a, in RVS, with mean travel time 14.14 min
SBK victims will go to these hospitals
BAP, level 2, in SBK, with mean travel time 11.02 min
UMC, level 1, in SPF, with mean travel time 14.14 min
MTH, level 3, in SPF, with mean travel time 14.14 min
STL, level 2a, in JTB, with mean travel time 14.14 min
WJX victims will go to these hospitals
UMC, level 1, in SPF, with mean travel time 10.00 min
MTH, level 3, in SPF, with mean travel time 10.00 min
BAP, level 2, in SBK, with mean travel time 20.00 min
STV, level 2a, in RVS, with mean travel time 20.00 min
EJX victims will go to these hospitals
UMC, level 1, in SPF, with mean travel time 10.00 min
MTH, level 3, in SPF, with mean travel time 10.00 min
BAP, level 2, in SBK, with mean travel time 14.14 min
AVD victims will go to these hospitals
STV, level 2a, in RVS, with mean travel time 10.00 min
RVS, level 3, in RVS, with mean travel time 10.00 min
UMC, level 1, in SPF, with mean travel time 14.14 min
RVS victims will go to these hospitals
RVS, level 3, in RVS, with mean travel time 8.66 min
STV, level 2a, in RVS, with mean travel time 8.66 min
UMC, level 1, in SPF, with mean travel time 14.14 min
PKV victims will go to these hospitals
UMC, level 1, in SPF, with mean travel time 14.14 min
MTH, level 3, in SPF, with mean travel time 14.14 min
BAP, level 2, in SBK, with mean travel time 25.00 min
STV, level 2a, in RVS, with mean travel time 25.00 min
LMT victims will go to these hospitals
UMC, level 1, in SPF, with mean travel time 14.14 min
MTH, level 3, in SPF, with mean travel time 14.14 min
BAP, level 2, in SBK, with mean travel time 25.00 min
STV, level 2a, in RVS, with mean travel time 25.00 min
OWY victims will go to these hospitals
ARL victims will go to these hospitals
UMC, level 1, in SPF, with mean travel time 14.14 min
MTH, level 3, in SPF, with mean travel time 14.14 min
BAP, level 2, in SBK, with mean travel time 25.00 min
STV, level 2a, in RVS, with mean travel time 25.00 min
MMC, level 2a, in SSD, with mean travel time 30.00 min

MND victims will go to these hospitals
UMC, level 1, in SPF, with mean travel time 14.14 min
MTH, level 3, in SPF, with mean travel time 14.14 min
BAP, level 2, in SBK, with mean travel time 14.14 min
MMC, level 2a, in SSD, with mean travel time 14.14 min

ATB victims will go to these hospitals
UMC, level 1, in SPF, with mean travel time 20.00 min

JTB victims will go to these hospitals
BCH, level 2a, in JXB, with mean travel time 14.14 min
UMC, level 1, in SPF, with mean travel time 40.00 min

SPL victims will go to these hospitals
BCH, level 2a, in JXB, with mean travel time 14.14 min
UMC, level 1, in SPF, with mean travel time 40.00 min

MND victims will go to these hospitals
STL, level 2a, in JTB, with mean travel time 12.25 min
UMC, level 1, in SPF, with mean travel time 25.00 min

MWW victims will go to these hospitals
STV, level 2a, in RVS, with mean travel time 20.00 min
RVS, level 3, in RVS, with mean travel time 20.00 min
UMC, level 1, in SPF, with mean travel time 25.00 min
MTH, level 3, in SPF, with mean travel time 25.00 min
BAP, level 2, in SBK, with mean travel time 25.00 min

BRY victims will go to these hospitals
STV, level 2a, in RVS, with mean travel time 35.00 min
RVS, level 3, in RVS, with mean travel time 35.00 min
UMC, level 1, in SPF, with mean travel time 40.00 min
MTH, level 3, in SPF, with mean travel time 40.00 min
BAP, level 2, in SBK, with mean travel time 40.00 min

CAL victims will go to these hospitals
UNC, level 1, in SPF, with mean travel time 30.00 min
MTH, level 3, in SPF, with mean travel time 30.00 min
BAP, level 2, in SBK, with mean travel time 40.00 min
STV, level 2a, in RVS, with mean travel time 40.00 min
MMC, level 2a, in SSD, with mean travel time 50.00 min
STL, level 2a, in JTB, with mean travel time 50.00 min

HIL victims will go to these hospitals
UNC, level 1, in SPF, with mean travel time 45.00 min
MTH, level 3, in SPF, with mean travel time 45.00 min
BAP, level 2, in SBK, with mean travel time 55.00 min
STV, level 2a, in RVS, with mean travel time 55.00 min
RVS, level 3, in RVS, with mean travel time 55.00 min
MMC, level 2a, in SSD, with mean travel time 65.00 min
STL, level 2a, in JTB, with mean travel time 65.00 min

YUL victims will go to these hospitals
NGH, level 3, in FOB, with mean travel time 25.00 min
UNC, level 1, in SPF, with mean travel time 35.00 min
BAP, level 2, in SBK, with mean travel time 45.00 min
STV, level 2a, in RVS, with mean travel time 45.00 min
MMC, level 2a, in SSD, with mean travel time 50.00 min
STL, level 2a, in JTB, with mean travel time 50.00 min

FOB victims will go to these hospitals
NGH, level 3, in FOB, with mean travel time 14.14 min
UNC, level 1, in SPF, with mean travel time 40.00 min

OPW victims will go to these hospitals
HUM, level 2a, in OPW, with mean travel time 14.14 min
UNC, level 1, in SPF, with mean travel time 40.00 min

GCS victims will go to these hospitals
HUM, level 2a, in OPW, with mean travel time 30.00 min
PCH, level 3, in PLK, with mean travel time 35.00 min
UNC, level 1, in SPF, with mean travel time 55.00 min

MBG victims will go to these hospitals
HUM, level 2a, in OPW, with mean travel time 25.00 min
UNC, level 1, in SPF, with mean travel time 60.00 min

PLK victims will go to these hospitals
PCH, level 3, in PLK, with mean travel time 17.50 min
HUM, level 2a, in OPW, with mean travel time 60.00 min
UNC, level 1, in SPF, with mean travel time 85.00 min

PVB victims will go to these hospitals
BCH, level 2a, in JXB, with mean travel time 14.14 min
UNC, level 1, in SPF, with mean travel time 50.00 min

SWZ victims will go to these hospitals
STL, level 2a, in JTB, with mean travel time 35.00 min
FLG, level 3, in STA, with mean travel time 40.00 min
MMC, level 2a, in SSD, with mean travel time 42.50 min
BAP, level 2, in SBK, with mean travel time 45.00 min
UNC, level 1, in SPF, with mean travel time 55.00 min

STA victims will go to these hospitals
FLG, level 3, in STA, with mean travel time 19.37 min
BCH, level 2a, in JXB, with mean travel time 50.00 min
STL, level 2a, in JTB, with mean travel time 65.00 min
UNC, level 1, in SPF, with mean travel time 85.00 min

CRB victims will go to these hospitals
FLG, level 3, in STA, with mean travel time 25.00 min
BCH, level 2a, in JXB, with mean travel time 70.00 min
STL, level 2a, in JTB, with mean travel time 85.00 min
MMC, level 2a, in SSD, with mean travel time 90.00 min
UMC, level 1, in SPF, with mean travel time 105.00 min

HST victims will go to these hospitals
FLG, level 3, in STA, with mean travel time 45.00 min
STL, level 2a, in JTB, with mean travel time 65.00 min
MMC, level 2a, in SSD, with mean travel time 72.50 min
BAP, level 2, in SBK, with mean travel time 75.00 min
BCH, level 2a, in JXB, with mean travel time 80.00 min
HUM, level 2a, in OPW, with mean travel time 80.00 min
UMC, level 1, in SPF, with mean travel time 85.00 min

KGB victims will go to these hospitals
NGH, level 3, in FDB, with mean travel time 50.00 min
UMC, level 1, in SPF, with mean travel time 60.00 min
MTH, level 3, in SPF, with mean travel time 60.00 min
BAP, level 2, in SBK, with mean travel time 70.00 min
STV, level 2a, in RVS, with mean travel time 75.00 min
STL, level 2a, in JTB, with mean travel time 80.00 min

Triage rule:
Champ TS <= 7.1 goes to level 1, > 7.4 may go to level 3

Travel times exceeding minimum time by less than tolerance included in routine dispatch lists
hospital choice tolerance 30%
ambulance choice tolerance 35%

Run 1

<table>
<thead>
<tr>
<th>duration</th>
<th>12.20 hours</th>
<th>no.accs</th>
<th>9</th>
<th>no. tr. pts</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>% blunt</td>
<td>55.56</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>amb. utilization</td>
<td>.041</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>helo utilization</td>
<td>.045</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>diverts</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>overrides</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>deaths</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>est deaths</td>
<td>.59</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>queue for amb</td>
<td>mean 0.0, max 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>no.accs pended</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>av pending time</td>
<td>0.0 minutes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>avg tr load</th>
<th>max tr load</th>
<th>reserve</th>
<th>% red time</th>
</tr>
</thead>
<tbody>
<tr>
<td>UMC</td>
<td>.258</td>
<td>3</td>
<td>1.000</td>
<td>0.0</td>
</tr>
<tr>
<td>BAP</td>
<td>0.0</td>
<td>0</td>
<td>1.000</td>
<td>0.0</td>
</tr>
<tr>
<td>MMC</td>
<td>0.0</td>
<td>0</td>
<td>1.000</td>
<td>0.0</td>
</tr>
<tr>
<td>MTH</td>
<td>.023</td>
<td>1</td>
<td>.977</td>
<td>0.0</td>
</tr>
<tr>
<td>STL</td>
<td>0.0</td>
<td>0</td>
<td>1.000</td>
<td>0.0</td>
</tr>
<tr>
<td>BCH</td>
<td>0.0</td>
<td>0</td>
<td>1.000</td>
<td>0.0</td>
</tr>
<tr>
<td>STV</td>
<td>0.0</td>
<td>0</td>
<td>1.000</td>
<td>0.0</td>
</tr>
<tr>
<td>RVS</td>
<td>0.0</td>
<td>0</td>
<td>1.000</td>
<td>0.0</td>
</tr>
<tr>
<td>HUM</td>
<td>.025</td>
<td>1</td>
<td>.975</td>
<td>0.0</td>
</tr>
<tr>
<td>NGH</td>
<td>0.0</td>
<td>0</td>
<td>1.000</td>
<td>0.0</td>
</tr>
<tr>
<td>FLG</td>
<td>0.0</td>
<td>0</td>
<td>1.000</td>
<td>0.0</td>
</tr>
<tr>
<td>PCH</td>
<td>0.0</td>
<td>0</td>
<td>1.000</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Run 2

<table>
<thead>
<tr>
<th>duration</th>
<th>18.58 hours</th>
<th>no.accs</th>
<th>21</th>
<th>no. tr. pts</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>% blunt</td>
<td>52.38</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>amb. utilization</td>
<td>.090</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 170 -
helo utilization  \(0.156\)
diverts 0
overrides 0
deaths 6
est deaths 2.15
queue for amb mean 0. max 0
no.accs pended: 0
av pending time: 0 minutes

### Hospital Utilization

<table>
<thead>
<tr>
<th>Name</th>
<th>Avg Tr Load</th>
<th>Max Tr Load</th>
<th>Reserve</th>
<th>% Red Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>UMC</td>
<td>0.355</td>
<td>4</td>
<td>1.000</td>
<td>0.</td>
</tr>
<tr>
<td>BAP</td>
<td>0.038</td>
<td>1</td>
<td>1.000</td>
<td>0.</td>
</tr>
<tr>
<td>MMC</td>
<td>0.038</td>
<td>2</td>
<td>0.989</td>
<td>0.</td>
</tr>
<tr>
<td>MTH</td>
<td>0.025</td>
<td>1</td>
<td>0.975</td>
<td>0.</td>
</tr>
<tr>
<td>STL</td>
<td>0.031</td>
<td>1</td>
<td>0.969</td>
<td>0.</td>
</tr>
<tr>
<td>BCH</td>
<td>0.023</td>
<td>1</td>
<td>0.977</td>
<td>0.</td>
</tr>
<tr>
<td>STV</td>
<td>0.038</td>
<td>2</td>
<td>0.989</td>
<td>0.</td>
</tr>
<tr>
<td>RVS</td>
<td>0.024</td>
<td>1</td>
<td>0.976</td>
<td>0.</td>
</tr>
<tr>
<td>HUM</td>
<td>0.038</td>
<td>1</td>
<td>1.000</td>
<td>0.</td>
</tr>
<tr>
<td>NGH</td>
<td>0.024</td>
<td>1</td>
<td>0.976</td>
<td>0.</td>
</tr>
<tr>
<td>FLG</td>
<td>0.024</td>
<td>1</td>
<td>0.976</td>
<td>0.</td>
</tr>
</tbody>
</table>

### Run 3

duration 13.50 hours no.accs 12 no.tr. pts 20
% blunt 83.33
amb. utilization 0.067
helo utilization 0.087
diverts 0
overrides 0
deaths 4
est deaths 1.20
queue for amb mean 0.079 max 1
no.accs pended: 1
av pending time: 64.258 minutes

### Hospital Utilization

<table>
<thead>
<tr>
<th>Name</th>
<th>Avg Tr Load</th>
<th>Max Tr Load</th>
<th>Reserve</th>
<th>% Red Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>UMC</td>
<td>0.355</td>
<td>4</td>
<td>1.000</td>
<td>0.</td>
</tr>
<tr>
<td>BAP</td>
<td>0.038</td>
<td>1</td>
<td>1.000</td>
<td>0.</td>
</tr>
<tr>
<td>MMC</td>
<td>0.038</td>
<td>2</td>
<td>0.989</td>
<td>0.</td>
</tr>
<tr>
<td>MTH</td>
<td>0.024</td>
<td>1</td>
<td>0.976</td>
<td>0.</td>
</tr>
<tr>
<td>STL</td>
<td>0.031</td>
<td>1</td>
<td>0.969</td>
<td>0.</td>
</tr>
<tr>
<td>BCH</td>
<td>0.023</td>
<td>1</td>
<td>0.977</td>
<td>0.</td>
</tr>
<tr>
<td>STV</td>
<td>0.038</td>
<td>2</td>
<td>0.989</td>
<td>0.</td>
</tr>
<tr>
<td>RVS</td>
<td>0.025</td>
<td>1</td>
<td>0.975</td>
<td>0.</td>
</tr>
<tr>
<td>HUM</td>
<td>0.038</td>
<td>1</td>
<td>1.000</td>
<td>0.</td>
</tr>
<tr>
<td>NGH</td>
<td>0.024</td>
<td>1</td>
<td>0.976</td>
<td>0.</td>
</tr>
<tr>
<td>FLG</td>
<td>0.024</td>
<td>1</td>
<td>0.976</td>
<td>0.</td>
</tr>
<tr>
<td>PCH</td>
<td>0.024</td>
<td>1</td>
<td>0.976</td>
<td>0.</td>
</tr>
</tbody>
</table>

### Run 4

duration 9.81 hours no.accs 4 no.tr. pts 3
% blunt 50.00
amb. utilization 0.018
helo utilization 0.
diverts 0
overrides 0
deaths 0
est deaths 0.01
queue for amb mean 0. max 0
no.accs pended: 0
av pending time: 0. minutes

hospital utilization
name avg tr load max tr load reserve % red time
UMC .091 2 1.000 0.
BAP 0. 0 1.000 0.
MMC 0. 0 1.000 0.
MTH 0. 0 1.000 0.
STL 0. 0 1.000 0.
BCH 0. 0 1.000 0.
STV 0. 0 1.000 0.
RVS 0. 0 1.000 0.
HUM 0. 0 1.000 0.
NGH 0. 0 1.000 0.
FLG 0. 0 1.000 0.
PCH 0. 0 1.000 0.

Results after 4 runs of at least .25 days per run
average sd (of runwise means) max
duration .56 .154 .77
no. accidents 11.5 7.14 21
% blunt 60.9
no. tr. patients 16.5 11.90 31
no. deaths 3.0
no. est deaths 1.0
amb. util .10 .030
amb. queue .020 .0397 mean .3
glob 1

hospital utilization
name load max reserve
UMC .24 .110 3 4 1.000 0.
BAP .01 .019 0 1 1.000 0.
MMC 0. 0. 0 0 1.000 0.
MTH .01 .014 0 1 .988 .0135
STL .03 .037 1 2 .980 .0241
BCH .01 .011 0 1 .994 .0114
STV .03 .041 1 2 .997 .0054
RVS .01 .012 0 1 .994 .0123
HUM .01 .013 0 1 .994 .0127
NGH 0. 0. 0 0 1.000 0.
FLG .01 .012 0 1 .994 .0122
PCH 0. 0. 0 0 1.000 0.
Appendix 5

Fitting Input Distributions

Time to secure

20 of 215 needed securing:
model by beta(21, 196)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N OF CASE</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MINIMUM</td>
<td>0.830</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>32.770</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RANGE</td>
<td>31.940</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEAN</td>
<td>6.150</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VARIANCE</td>
<td>46.505</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STANDARD DEV</td>
<td>6.819</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STD. ERROR</td>
<td>1.525</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SKEWNESS(G1)</td>
<td>3.141</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>KURTOSIS(G2)</td>
<td>9.933</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUM</td>
<td>123.000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C.V.</td>
<td>1.109</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MEDIAN</td>
<td>4.230</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 8. Distribution given need to secure.

Figure 9. Probability plot: exponential.

Figure 10. Quantile plot: exponential.
Figure 11. Probability plot: Weibull distribution.

Figure 12. Quantile plot: Weibull(6.686, 1.412).

Figure 13. Probability plot: lognormal.

Figure 14. Quantile plot: lognormal.
Time to patient found

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>215</td>
</tr>
<tr>
<td>MINIMUM</td>
<td>0.050</td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>12.920</td>
</tr>
<tr>
<td>RANGE</td>
<td>12.870</td>
</tr>
<tr>
<td>MEAN</td>
<td>2.220</td>
</tr>
<tr>
<td>VARIANCE</td>
<td>5.818</td>
</tr>
<tr>
<td>STANDARD DEV</td>
<td>2.412</td>
</tr>
<tr>
<td>STD. ERROR</td>
<td>0.165</td>
</tr>
<tr>
<td>SKEWNESS(G1)</td>
<td>1.696</td>
</tr>
<tr>
<td>KURTOSIS(G2)</td>
<td>2.697</td>
</tr>
<tr>
<td>SUM</td>
<td>477.380</td>
</tr>
<tr>
<td>C.V.</td>
<td>1.086</td>
</tr>
<tr>
<td>MEDIAN</td>
<td>1.120</td>
</tr>
</tbody>
</table>

Figure 15. Distribution of time to patient.

Figure 16. Probability plot: exponential.

Exponential fits very well.

Figure 17. Quantile plot: exponential.
Time on scene

Total scene time for those needing extrication and those not were not significantly different (Mann-Whitney $P = .15$, Komolgorov-Smirnov $P = .11$).

<table>
<thead>
<tr>
<th>N</th>
<th>185</th>
</tr>
</thead>
<tbody>
<tr>
<td>MINIMUM</td>
<td>1.000</td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>30.830</td>
</tr>
<tr>
<td>RANGE</td>
<td>29.830</td>
</tr>
<tr>
<td>MEAN</td>
<td>10.404</td>
</tr>
<tr>
<td>VARIANCE</td>
<td>34.368</td>
</tr>
<tr>
<td>STANDARD DEV</td>
<td>5.862</td>
</tr>
<tr>
<td>STD. ERROR</td>
<td>0.431</td>
</tr>
<tr>
<td>SKEWNESS(G1)</td>
<td>1.149</td>
</tr>
<tr>
<td>KURTOSIS(G2)</td>
<td>1.394</td>
</tr>
<tr>
<td>SUM</td>
<td>1924.780</td>
</tr>
<tr>
<td>C.V.</td>
<td>0.563</td>
</tr>
<tr>
<td>MEDIAN</td>
<td>9.070</td>
</tr>
</tbody>
</table>

Figure 18. Distribution of scene treatment time (includes extrication if needed).

Figure 19. Probability plot: Weibull.

Figure 20. Quantile plot: Weibull(11.023, 1.892).
Figure 21. Probability plot: gamma(3).

Figure 22. Quantile plot: gamma(3).

Figure 23. Weibull (dashed line) and gamma pdf's.
Time to hospital

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>184</td>
</tr>
<tr>
<td>MINIMUM</td>
<td>1.370</td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>37.230</td>
</tr>
<tr>
<td>RANGE</td>
<td>35.860</td>
</tr>
<tr>
<td>MEAN</td>
<td>10.295</td>
</tr>
<tr>
<td>VARIANCE</td>
<td>29.925</td>
</tr>
<tr>
<td>STANDARD DEV</td>
<td>5.470</td>
</tr>
<tr>
<td>STD. ERROR</td>
<td>0.403</td>
</tr>
<tr>
<td>SKEWNESS(G1)</td>
<td>1.072</td>
</tr>
<tr>
<td>KURTOSIS(G2)</td>
<td>2.268</td>
</tr>
<tr>
<td>SUM</td>
<td>1894.250</td>
</tr>
<tr>
<td>C.V.</td>
<td>0.531</td>
</tr>
<tr>
<td>MEDIAN</td>
<td>9.775</td>
</tr>
</tbody>
</table>

Figure 24. Distribution of time to hospital.

Figure 25. Probability plot: Weibull.

Figure 26. Quantile plot: Weibull(9.974, 2.242).
Figure 27. Probability plot: lognormal.

Figure 28. Quantile plot: lognormal.

Figure 29. Probability plot: gamma(3).

Figure 30. Quantile plot: gamma(3).
Time to release of pt

N 177
MINIMUM 0.270
MAXIMUM 12.000
RANGE 11.730
MEAN 2.598
VARIANCE 3.788
STANDARD DEV 1.946
STD. ERROR 0.146
SKEWNESS(G1) 2.366
KURTOSIS(G2) 7.257
SUM 459.880
C.V. 0.749
MEDIAN 2.070

Figure 31. Distribution of time to release of patient.

Figure 32. Probability plot: Weibull.

Figure 33. Quantile plot: Weibull(2.718, 1.810).
Figure 34. Probability plot: lognormal.

Figure 35. Quantile plot: lognormal.

Figure 36. Probability plot: gamma.

Figure 37. Quantile plot: gamma(1.5).
Figure 38. Probability plot: exponential.

Figure 39. Quantile plot: exponential.
Appendix 6
Log of Model Assumptions

6.1 Distribution of Accidents

It is assumed that the average number of accidents per unit time is unchanging; i.e., there is no long term growth or decline in the level of demand. Only the number of injuring incidents is assumed to vary with time and location; severity of injury and type of injury are assumed to be independent. Similarly, it is assumed that there is no interaction or synergism between time and location.

6.2 Definitive Care Survival

There is no assumption that patients receiving definitive care in a Level 1 center have increased survival compared to those in other centers (after adjusting for severity of injury). There are published reports that this may not be true (that Level 1 patients do better), so this assumption may need to be examined critically.

6.3 Private Travel

The function expressing the probability of private travel as inversely proportional to severity and between .30 and .01 is entirely empiric, based on experience, plausibility, and scant data suggesting that overall about one-fifth of non-
trivial trauma patients find their own way to medical care. In addition, the model currently assumes that all patients involved in an accident make the same decision. This clearly plays a role in the real world model but is not completely valid. Other considerations, including proximity to a hospital and the actual activity producing the injury are also involved in as yet unspecified ways.
Appendix 7
Log of Improvements and Enhancements

7.1 Improvements

The following items (in no particular order) will enable the existing model to run more efficiently, more realistically, or be more easily maintained, but do not add additional function.

7.1.1 Transit time. The permanent entity ambulance currently maintains the attribute of transit time. This is more realistically an attribute of an ambulance run process than it is of an ambulance, and in keeping with the second principle of Section 4.2 should be moved to the process.

7.1.2 Choke points. Routines for building call and dispatch lists should take better account of the choke point variable than they do presently.

7.1.3 Events. Coding the respiratory support therapeutic interventions as events, rather than inserting them directly into either (or both) of the patient or ambulance run process routines allowed produced much cleaner, more easily maintained code. Other critical therapeutic maneuvers such as IV starting or blood transfusion should be recoded into this form.
7.1.4 End-of-run. The process of cancelling and then rescheduling the clear.reds event if another run follows the current should be changed to leave the event scheduled unless no runs are to follow.

7.1.5 Memory management. The duplicate representation of arcs in the current implementation requires large amounts of memory, leading to disk swapping and poor performance in large models. Since much of this information is redundant, and since it is infrequently referenced once the actual simulation begins, a more efficient representation should produce disproportionate benefits in run times.

7.2 Enhancements

The following items are additional (new) capabilities that would increase the utility of the model, but will be deferred at this time as they are not critical to the proof of concept.

7.2.1 Non-trauma patients. Information on medical patients handled by the system should be added to the model. In particular, some method of representing the "walk-in" medical load on emergency departments should be added as this is a major reason for a hospital’s going on divert. For example, this could be done quite simply as a random external event, without having to explicitly model large numbers of "walk-in" patients.
7.2.2 Injury model. Better and more detailed models of injury such as ASCOT [Champion90] are now available. Modification of the model to use ASCOT or a similar measure of injury severity might allow better prediction of outcome and identification of subsets of patients for whom special policies may be beneficial. The use of ASCOT, however, would require far more detailed epidemiologic information about injury patterns than is currently available, although it is being collected in the national Trauma Registry program of the American College of Surgeons (TRACS).

7.2.3 Transfers. While they are only a small fraction of the total volume of trauma patients, inter-hospital transfers are frequently a considerable source of contention. It would be desirable to model the transfer of patients among hospitals as representing an important aspect of the system, but it has proven extremely difficult to obtain reliable information, and what data is available is highly suspect as misleading at best.

7.2.4 Data editor. There is currently no support for creating the data files used to drive the simulation, nor is there any error checking for illogical or impossible conditions, e.g., a node with efferent but no afferent arcs. For complex models, creating the data files with a text can be tedious and prone to error; modification may be even more difficult. A data editor, particularly if it were able to graphically represent the transportation network, would make
the model much easier to use. Error checking could be provided along with the editing function to ensure that the files finally submitted to the model did not contain logical errors.

7.2.5 Graphical output. Although it may restrict portability somewhat, a graphical display of system activity may be useful in establishing face validity of the model.

7.2.6 Trace control. Currently, the trace output is "all or nothing," making it unwieldy when the region of interest lies deep in a long run or series of runs. The ability to turn the trace on or off from within the program would enhance the usefulness of the trace.

7.2.7 Interruption. Direct (paired) comparison of regenerative method simulation data must be done at comparable points in each experimental arm. The number and location of such points is unpredictable, and they become fewer and are spread farther apart as the number of arms in an experiment increases. It would therefore be advantageous to make model runs interruptible, so that if there were not sufficient convergence points for analysis after some period of simulated time, the model could be restarted at that point and run further forward. Since the simulation mechanism is regenerative, this can in principle be done manually simply by preserving the random number seeds (or choosing different random number generators altogether) and
starting another series of runs with the RNG's reset and
time.v set to its value at the end of the previous
corresponding run.

7.2.8 Non-regenerative Simulation. The difficulties of
statistical analysis of the regenerative method data suggest
that it might be advantageous to abandon this method in
favor of traditional non-terminating simulation analysis of
steady-state cycle parameters. This would require analysis
of the startup transients which has not been performed for
this model. It is not clear whether this would increase or
decrease the required length of the simulation.
Appendix 8
Log of Program Bugs

8.1 Discrete-continuous Interaction

A variety of hard-to-diagnose problems sometimes appear when the discrete simulation portions of the program interact with the continuous simulation portions. For example, if the minimum step size for the integrator routine in the continuous simulation modules is too large, it is at least theoretically possible for the two sections to become unsynchronized. While it is felt that most of the areas at risk have been protected by a combination of local code and by a policy of keeping the maximum step size small, confidence in the model's reliability would be significantly enhanced by systematically eliminating potential areas of interaction.

8.2 Pended Accidents

The dispatcher currently only checks the first accident in the pending set when a new ambulance becomes available; the entire set should be checked. Since, under the conditions modeled for northeast Florida, the pending set virtually never contains more than one accident, this error was not initially apparent.
Vita

Robert L. Wears has BA and MD degrees from the Johns Hopkins University (1969 and 1973, respectively). He is currently Associate Professor in the Department of Surgery, Division of Emergency Medicine at the University of Florida Health Science Center in Jacksonville, and expects to receive an MS in Computer and Information Sciences from the University of North Florida on April 30, 1993.

Robert has extensive interests in the application of computer-based modeling and analysis techniques to clinical and health policy problems, and has published several examples of their use in the medical literature with Dr. Charles N. Winton, his thesis advisor. His work includes statistical and connectionist approaches to computer modeling, with extensive experience in C, SYSTAT, and SIMSCRIPT. He has developed a fellowship in medical informatics for graduate physicians, installed the first LAN at UFHSC-Jax, and is active in promoting the application of modern information methods in the clinical setting.

Dr. Wears has been married to his first wife for 21 years, and has two teenage children, a dog, a fish, and is a raving windsurfer.