# DETERMINATION OF CRITICALITY INDICES IN THE PERT PROBLEM

GUILLERMO PONCE-CAMPOS
CIVIL ENGINEERING DEPARTMENT
THE UNIVERSITY OF MICHIGAN
ANN ARBOR, MICHIGAN

THOMAS J. SCHRIBER
GRADUATE SCHOOL OF BUSINESS
THE UNIVERSITY OF MICHIGAN
ANN ARBOR, MICHIGAN

## ABSTRACT

A GPSS/360 model which estimates criticality indices and project duration distribution for the generalized PERT network is presented. The cost of developing this model and using it oı. a production run basis is compared with the corresponding costs associated with an equivalent FORTRAN IV (Level G) model. Although GPSS/360 facilitates the model-building process, significant overall cost economies can be realized by building the same model in FORTRAN when extended production runs are to be màde.

## 1. INTRODUCTION

PERT (Program Evaluation and Review Technique) has attained considerable recognition as a management tool in program planning. The PERT model is a directed, acyclic network of "activities" and "events" in which the time durations required for activity completion are positive random variables characterized by known density functions. Given the range and mode of each activity duration and invoking simplifying assumptions, the PERT approach is to analyze the project network by viewing the problem as being quasi-deterministic. Working with expected values of activity durations, PERT identifies the most obstructive path (or paths) in a network, termed the Critical Path(s), then uses estimates of the variance of critical activity durations to produce an estimate of the mean project duration and its variance. In essence, then, no use is made of the probabilistic element in the activity time estimates until <u>after</u> the critical path is found.

Unfortunately, because of the essential randomness of the time duration of various activities, almost any path in a project network is capable of becoming most obstructive, or critical. The concept of a unique Critical Path (or of two or more parallel or partially-parallel Critical Paths) is consequently misleading conceptually, and produces a bias in the PERT results. It is more realistic to talk about the probability that any particular activity will be critical, i.e., will lie on the Critical Path. VanSlyke (1) termed this probability the "criticality index" of an activity and proposed Monte Carlo simulation as an approach for estimatnng the criticality indices of the various activities making up an overall project. Such Monte Carlo simulation provides unbiased estimates of the mean project duration and its variance, and also produces other information not available from a conventional PERT analysis, including an estimate of the distribution of project completion time and estimates of the criticality index for each activity. A complete discussion of the PERT assumptions and the assumptions in effect when a Monte Carlo approach is taken can be found in VanSlyke's paper.

## 2. APPLICABILITY OF GPSS TO THE PERT PROBLEM

VanSlyke used FORTRAN in building his model to implement the Monte Carlo sampling process. When such an approach is taken, programmer-supplied logic must be used to guarantee that the various precedence constraints inherent in a project network are recognized and honored in the model. In contrast with these procedure-oriented logical requirements, the problem-oriented language GPSS has built-in features which can be exploited to relieve the model builder of certain of these logical tasks.

In GPSS modeling, analogies are drawn between the various GPSS entities and elements of the system being modeled. For example, consider Figure 1, where a portion of a project network is represented.

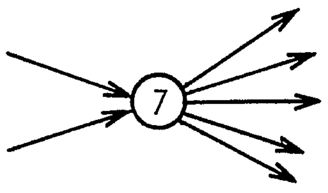

FIGURE 1. PORTION OF A PROJECT NETWORK

Figure 1 shows two activities leading into "event 7" and indicates that, after the event takes place, five more activities can then be initiated. Now suppose that each activity in a project network is represented in GPSS by a "Transaction". Then the situation in Figure 1 can be represented by having two Transactions flow into a common point, after which five Transactions can issue from the point and proceed onward. In PERT terminology, this situation would be described by saying that two activities "merge" into an event, after which five activities "burst" from the event. The GPSS "ASSEMBLE Block" lends itself directly to representation of the merging process, destroying all Transactions which arrive at the Block except the first, which is permitted to move on only after the "Assembly Count" (i.e., "merge count") has been satisfied. After the Assembly Count is satisfied, the first Transaction can then move immediately into a "SPLIT Block", causing creation of as many additional Transactions as may be necessary to represent each of the activities bursting from the just-completed event. This movement on the part of Transactions in the model is automatically accomplished by the GPSS processor, relieving the model builder of the need to provide the necessary underlying timing and testing procedures.

Furthermore, project completion time is easily determined in GPSS. Transactions bursting from the source event in the network are "marked" with the reading of the system's clock at that time. Later, some Transaction finally reaches the project's sink event (terminal event) and simultaneously satisfies the merge count at that point in the model. The "transit time" of this Transaction is then computed. This transit time is the project completion time for the network realization just completed.

Finally, GPSS lends itself to effortless estimation of the probability distribution of the project completion time. This estimation follows directly from use of the GPSS "Table" entity. The model builder needs simply to define the existence of the Table, then provide the one step necessary for tabulation of the project completion time whenever another network raalization has been completed.

Prior to beginning the work being reported here, one of us had already taken advantage of certain GPSS features to build a model for the PERT problem as one of 17 GPSS case studies (2). That model, in addition to using the ASSEMBLE and SPLIT Blocks and the Table entity, also utilized the GPSS MACRO definition capability and the MATCH Block. The model suffered, however, from the limitation that it was tied to a specific project network and did not provide estimates of activity criticality indices. The model being presented now does not have these disadvantages. The generality in the present model was achieved, however, by using an explicit decrement-test-terminate sequence to "simulate" the just-described effect of the ASSEMBLE Block. This was done to avoid what otherwise would have been the necessity of arranging for a variety of "Assembly Sets" in the model.

## 3. DETERMINATION OF THE CRITICAL PATH(S)

The algorithm for determining the Critical Path(s) in a project network, given a set of activity time realizations, is well known and will not be repeated here. A brief, basic discussion of the algorithm and an illustrative numeric example can be found in Meier, Newell, and Pazer (3).

## 4. GENERAL DESCRIPTION OF THE GPSS MODEL

The GPSS/360 model presented here requires 78 "Blocks" to simulate the behavior of real network situations. Model input consists of: 1) the tail and head events for each activity in the network, 2) optimistic, most likely, and pessimistic time estimates for each activity, and 3) the number of realizations to be used in developing the output information. This data is presented via INITIAL cards.
In execution, the model first processes the activity-ordered network description to produce an event-ordered description. This is done by listing the activities that merge into and burst from each event. The event-ordered description then indicates the network's precedence requirements and provides the merge and burst counts needed for the forward and backward passes, respectively.

For each network realization, Monte Carlo sampling is then used to establish the time required for completion of each activity in the network. In particular, each activity's time is determined by a random draw from a triangular distribution. Points on the triangle are determined by the optimistic, most likely, and pessimistic estimates of the times required for the activity in question.

Next, forward and backward passes through the network are performed. In the forward (backward) pass, a single Transaction at the source (sink) event SPLITs to create additional Transactions according to the number of activities bursting from (merging into) that event. Each of these Transactions experiences a delay corresponding to the time required for the activity it represents. Then it decrements a merge (burst) counter associated with the corresponding activity's head (tail) event and destroys itself unless the post-decrement counter value is zero. The one Transaction that decrements a given counter to zero in effect represents completion of the associated event. Rather than destroying itself, then, it records the event's early (late) time, then SPLITs to create additional Transactions according to the number of activities bursting from (merging into) the just-completed event. The process continues until the just-completed event is found to be the sink (source) event. (For the case of the sink event, the early time is tabled as the project completion time.) When the forward and backward passes are complete, early and late event times are used to compute total slack for each activity. Those activities having zero slack then have their "criticality counter" incremented by one. Then the next network realization is begun.

After the total number of realizations requested by the user have been accomplished, criticality indices are computed for each activity by dividing the various criticality counters by the number of realizations. Information on the distribution of project completion times is directly available via the GPSS Table output. If desired, the distribution can be displayed in histogram form with use of the GPSS Output Editor.

The model accommodates itself to any general network satisfying these several restrictions: 1) Not more than 5 activities may merge into an event; 2) Not more than 5 activities may burst from an event; 3) There may not be more than 95 activities in the network; and 6) There may not be more than 95 events in the network. These restrictions can easily be relaxed.

Specific model details are available in Figure 2 and Table 1, where the annotated GPSS/360 Block Diagram of the model and a supporting Table of Definitions are presented, respectively. The reader familiar with the PERT algorithm and with GPSS can come to a complete understanding of the model by careful study of these details.

## 5. MODEL VERIFICATION

The GPSS model presented here and the corresponding FORTRAN model mentioned in the Abstract were each tested on five sample networks.* The models were verified by specifying that all three time estimates for any given activity were identical. The PERT algorithm was carried out by hand for these circumstances, and the resulting project duration and critical activities were compared with the corresponding model output. As expected, activities on the critical path each had a criticality index of 1.0, and criticality indices were zero elsewhere. Project duration computed by hand also matched that produced by the computer models.

Further verification took the form of providing random activity durations for a variety of selected single activities in each network, with other activity times taken to be deterministic. The probability that the one activity would be critical was then computed from the corresponding triangular distribution. This result was compared with the model-produced criticality index for that activity. As the number of realizations became large, the simulated result tended toward the theoretically expected value.

## 6. STATISTICAL CONSIDERATIONS

For our purposes in this work, it was not necessary to determine the sample size required to attain a stated level of precision in the statistics being observed. We are now contemplating modifying the model for that purpose along lines suggested by Fishman (4).

*The FORTRAN model is available on request from the authors.

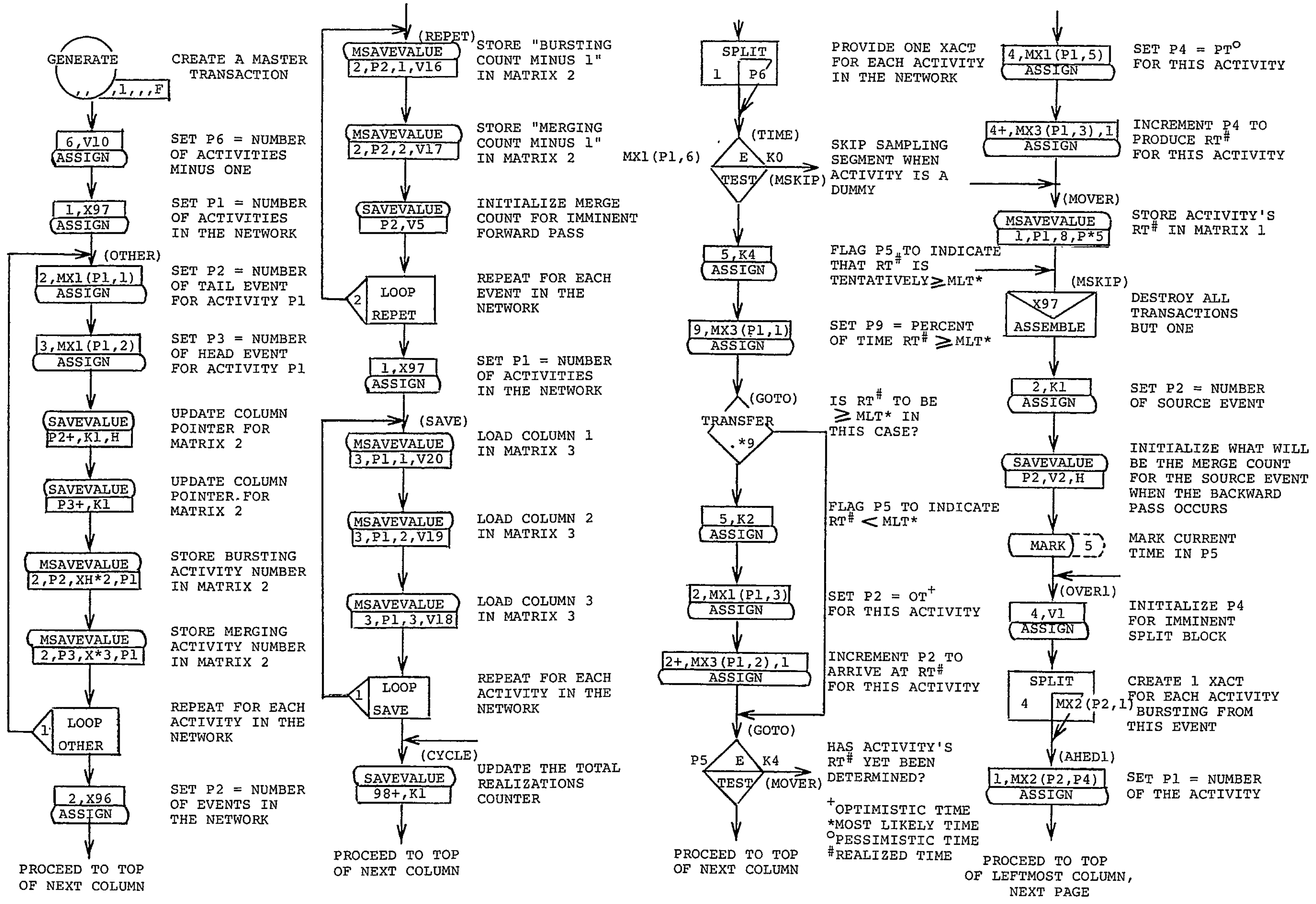
GENERATE
,,,1,,,F
CREATE A MASTER TRANSACTION
6,V10
ASSIGN
SET P6 = NUMBER OF ACTIVITIES MINUS ONE
1,X97
ASSIGN
SET P1 = NUMBER OF ACTIVITIES IN THE NETWORK
(OTHER)
2,MX1(P1,1)
ASSIGN
SET P2 = NUMBER OF TAIL EVENT FOR ACTIVITY P1
3,MX1(P1,2)
ASSIGN
SET P3 = NUMBER OF HEAD EVENT FOR ACTIVITY P1
SAVEVALUE
P2+,K1,H
UPDATE COLUMN POINTER FOR MATRIX 2
SAVEVALUE
P3+,K1
UPDATE COLUMN POINTER FOR MATRIX 2
MSAVEVALUE
2,P2,XH*2,P1
STORE BURSTING ACTIVITY NUMBER IN MATRIX 2
MSAVEVALUE
2,P3,X*3,P1
STORE MERGING ACTIVITY NUMBER IN MATRIX 2
1
LOOP
OTHER
REPEAT FOR EACH ACTIVITY IN THE NETWORK
2,X96
ASSIGN
SET P2 = NUMBER OF EVENTS IN THE NETWORK
PROCEED TO TOP OF NEXT COLUMN
(REPET)
MSAVEVALUE
2,P2,1,V16
STORE "BURSTING COUNT MINUS 1" IN MATRIX 2
MSAVEVALUE
2,P2,2,V17
STORE "MERGING COUNT MINUS 1" IN MATRIX 2
SAVEVALUE
P2,V5
INITIALIZE MERGE COUNT FOR IMMINENT FORWARD PASS
2
LOOP
REPET
REPEAT FOR EACH EVENT IN THE NETWORK
1,X97
ASSIGN
SET P1 = NUMBER OF ACTIVITIES IN THE NETWORK
(SAVE)
MSAVEVALUE
3,P1,1,V20
LOAD COLUMN 1 IN MATRIX 3
MSAVEVALUE
3,P1,2,V19
LOAD COLUMN 2 IN MATRIX 3
MSAVEVALUE
3,P1,3,V18
LOAD COLUMN 3 IN MATRIX 3
1
LOOP
SAVE
REPEAT FOR EACH ACTIVITY IN THE NETWORK
(CYCLE)
SAVEVALUE
98+,K1
UPDATE THE TOTAL REALIZATIONS COUNTER
PROCEED TO TOP OF NEXT COLUMN
SPLIT
1
P6
PROVIDE ONE XACT FOR EACH ACTIVITY IN THE NETWORK
(TIME)
MX1(P1,6)
E
K0
TEST
(MSKIP)
SKIP SAMPLING SEGMENT WHEN ACTIVITY IS A DUMMY
5,K4
ASSIGN
FLAG P5 TO INDICATE THAT RT# IS TENTATIVELY ≥ MLT*
9,MX3(P1,1)
ASSIGN
SET P9 = PERCENT OF TIME RT# ≥ MLT*
(GOTO)
TRANSFER
.*9
IS RT# TO BE ≥ MLT* IN THIS CASE?
5,K2
ASSIGN
FLAG P5 TO INDICATE RT# < MLT*
2,MX1(P1,3)
ASSIGN
SET P2 = OT+ FOR THIS ACTIVITY
2+,MX3(P1,2),1
ASSIGN
INCREMENT P2 TO ARRIVE AT RT# FOR THIS ACTIVITY
(GOTO)
P5
E
K4
TEST
(MOVER)
HAS ACTIVITY'S RT# YET BEEN DETERMINED?
PROCEED TO TOP OF NEXT COLUMN
+OPTIMISTIC TIME
*MOST LIKELY TIME
°PESSIMISTIC TIME
#REALIZED TIME
4,MX1(P1,5)
ASSIGN
SET P4 = PT° FOR THIS ACTIVITY
4+,MX3(P1,3),1
ASSIGN
INCREMENT P4 TO PRODUCE RT# FOR THIS ACTIVITY
(MOVER)
MSAVEVALUE
1,P1,8,P*5
STORE ACTIVITY'S RT# IN MATRIX 1
(MSKIP)
X97
ASSEMBLE
DESTROY ALL TRANSACTIONS BUT ONE
2,K1
ASSIGN
SET P2 = NUMBER OF SOURCE EVENT
SAVEVALUE
P2,V2,H
INITIALIZE WHAT WILL BE THE MERGE COUNT FOR THE SOURCE EVENT WHEN THE BACKWARD PASS OCCURS
MARK
5
MARK CURRENT TIME IN P5
(OVER1)
4,V1
ASSIGN
INITIALIZE P4 FOR IMMINENT SPLIT BLOCK
SPLIT
4
MX2(P2,1)
CREATE 1 XACT FOR EACH ACTIVITY BURSTING FROM THIS EVENT
(AHED1)
1,MX2(P2,P4)
ASSIGN
SET P1 = NUMBER OF THE ACTIVITY
PROCEED TO TOP OF LEFTMOST COLUMN, NEXT PAGE

FIGURE 2. BLOCK DIAGRAM FOR GPSS/360 MODEL (CONTINUED ON NEXT PAGE)

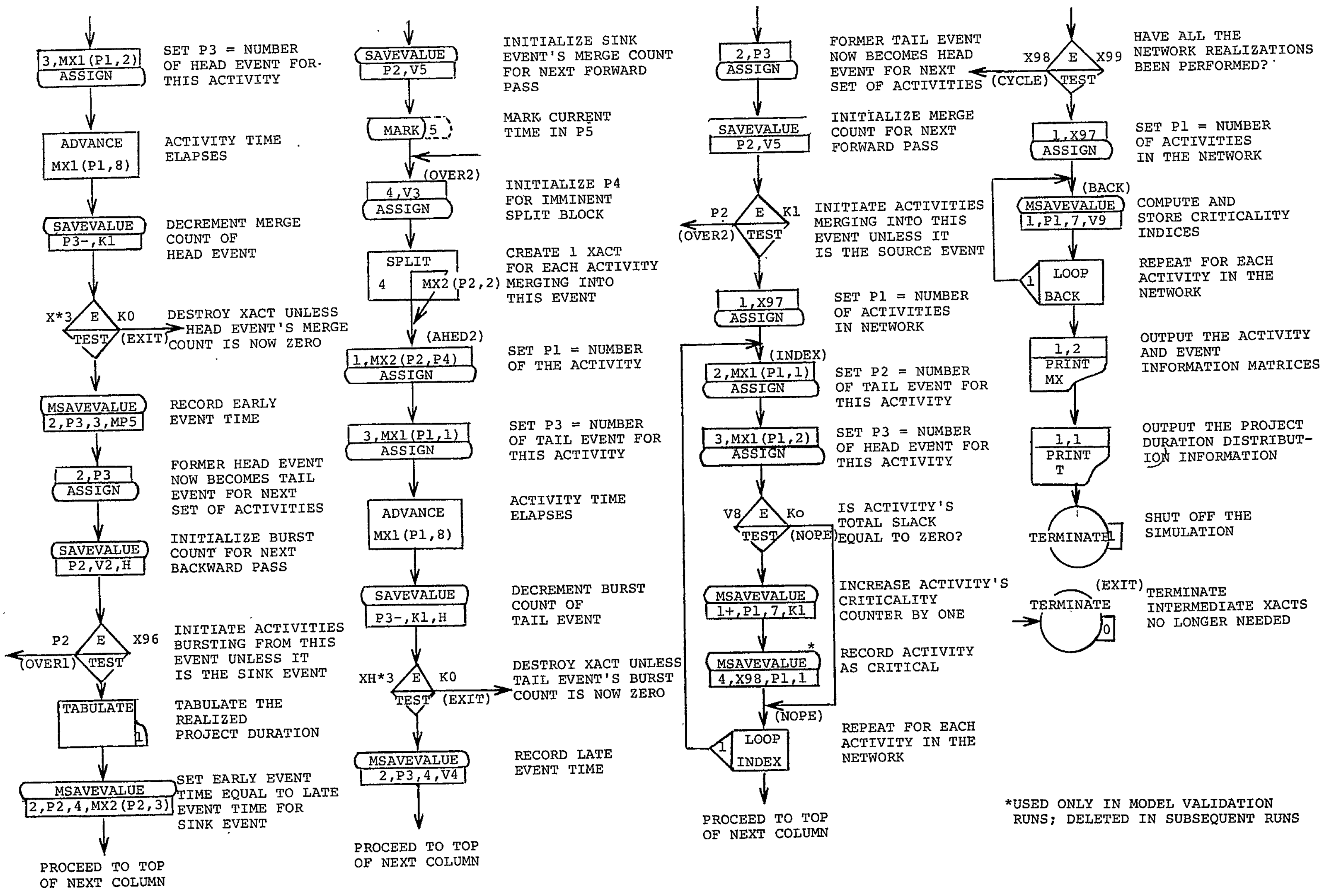
3,MX1(P1,2)
ASSIGN
SET P3 = NUMBER OF HEAD EVENT FOR THIS ACTIVITY
ADVANCE
MX1(P1,8)
ACTIVITY TIME ELAPSES
SAVEVALUE
P3-,K1
DECREMENT MERGE COUNT OF HEAD EVENT
X*3
E
K0
TEST
(EXIT)
DESTROY XACT UNLESS HEAD EVENT'S MERGE COUNT IS NOW ZERO
MSAVEVALUE
2,P3,3,MP5
RECORD EARLY EVENT TIME
2,P3
ASSIGN
FORMER HEAD EVENT NOW BECOMES TAIL EVENT FOR NEXT SET OF ACTIVITIES
SAVEVALUE
P2,V2,H
INITIALIZE BURST COUNT FOR NEXT BACKWARD PASS
P2
E
X96
(OVER1)
TEST
INITIATE ACTIVITIES BURSTING FROM THIS EVENT UNLESS IT IS THE SINK EVENT
TABULATE
1
TABULATE THE REALIZED PROJECT DURATION
MSAVEVALUE
2,P2,4,MX2(P2,3)
SET EARLY EVENT TIME EQUAL TO LATE EVENT TIME FOR SINK EVENT
PROCEED TO TOP OF NEXT COLUMN
SAVEVALUE
P2,V5
INITIALIZE SINK EVENT'S MERGE COUNT FOR NEXT FORWARD PASS
MARK
5
MARK CURRENT TIME IN P5
(OVER2)
4,V3
ASSIGN
INITIALIZE P4 FOR IMMINENT SPLIT BLOCK
SPLIT
4
MX2(P2,2)
CREATE 1 XACT FOR EACH ACTIVITY MERGING INTO THIS EVENT
(AHED2)
1,MX2(P2,P4)
ASSIGN
SET P1 = NUMBER OF THE ACTIVITY
3,MX1(P1,1)
ASSIGN
SET P3 = NUMBER OF TAIL EVENT FOR THIS ACTIVITY
ADVANCE
MX1(P1,8)
ACTIVITY TIME ELAPSES
SAVEVALUE
P3-,K1,H
DECREMENT BURST COUNT OF TAIL EVENT
XH*3
E
K0
TEST
(EXIT)
DESTROY XACT UNLESS TAIL EVENT'S BURST COUNT IS NOW ZERO
MSAVEVALUE
2,P3,4,V4
RECORD LATE EVENT TIME
PROCEED TO TOP OF NEXT COLUMN
2,P3
ASSIGN
FORMER TAIL EVENT NOW BECOMES HEAD EVENT FOR NEXT SET OF ACTIVITIES
SAVEVALUE
P2,V5
INITIALIZE MERGE COUNT FOR NEXT FORWARD PASS
P2
E
K1
(OVER2)
TEST
INITIATE ACTIVITIES MERGING INTO THIS EVENT UNLESS IT IS THE SOURCE EVENT
1,X97
ASSIGN
SET P1 = NUMBER OF ACTIVITIES IN NETWORK
(INDEX)
2,MX1(P1,1)
ASSIGN
SET P2 = NUMBER OF TAIL EVENT FOR THIS ACTIVITY
3,MX1(P1,2)
ASSIGN
SET P3 = NUMBER OF HEAD EVENT FOR THIS ACTIVITY
V8
E
Ko
TEST
(NOPE)
IS ACTIVITY'S TOTAL SLACK EQUAL TO ZERO?
MSAVEVALUE
1+,P1,7,K1
INCREASE ACTIVITY'S CRITICALITY COUNTER BY ONE
*
MSAVEVALUE
4,X98,P1,1
RECORD ACTIVITY AS CRITICAL
(NOPE)
1
LOOP
INDEX
REPEAT FOR EACH ACTIVITY IN THE NETWORK
PROCEED TO TOP OF NEXT COLUMN
X98
E
X99
(CYCLE)
TEST
HAVE ALL THE NETWORK REALIZATIONS BEEN PERFORMED?
1,X97
ASSIGN
SET P1 = NUMBER OF ACTIVITIES IN THE NETWORK
(BACK)
MSAVEVALUE
1,P1,7,V9
COMPUTE AND STORE CRITICALITY INDICES
1
LOOP
BACK
REPEAT FOR EACH ACTIVITY IN THE NETWORK
1,2
PRINT
MX
OUTPUT THE ACTIVITY AND EVENT INFORMATION MATRICES
1,1
PRINT
T
OUTPUT THE PROJECT DURATION DISTRIBUTION INFORMATION
TERMINATE
1
SHUT OFF THE SIMULATION
(EXIT)
TERMINATE
0
TERMINATE INTERMEDIATE XACTS NO LONGER NEEDED
*USED ONLY IN MODEL VALIDATION RUNS; DELETED IN SUBSEQUENT RUNS

FIGURE 2. BLOCK DIAGRAM FOR GPSS/360 MODEL (CONTINUED FROM PRECEDING PAGE)

| GPSS Entity | Interpretation |
|---|---|
| Transaction | "Project Supervisor"<br>P1: Looping Parameter<br>P2: Looping Parameter<br>"Activity Foreman"<br>P1: Number of the activity involved<br>P2: Activity tail (head) event number<br>P3: Activity head (tail) event number<br>P4: Holds bursting (merging) activity number as stored in Columns 5-10 (11-16) in Matrix 2<br>P5: Time of activity inception<br>P6: Number of activities minus one<br>P9: Used on a temporary basis when sampling from the activity duration distribution |
| Function 1 | The inverse cumulative distribution function corresponding to the density function: $f(x) = 2x,\ 0 \leq x \leq 1$ |
| Matrix 1 (Fullword) | Activity Information Matrix<br>Rows 1,2,3,...,X97 carry information about activities 1,2,3,...,X97, resp. |

| Column | Information |
|---|---|
| 1 | Activity tail event |
| 2 | Activity head event |
| 3 | Optimistic time estimate |
| 4 | Most likely time estimate |
| 5 | pessimistic time estimate |
| 6 | Activity code; value is 1 if the activity is a dummy; value is 0 otherwise |
| 7 | Criticality index counter |
| 8 | Realized activity time |

| GPSS Entity | Interpretation |
|---|---|
| Matrix 2 (Fullword) | Event Information Matrix<br>Rows 1,2,3,...,X96 carry information about events 1,2,3,...,X96, resp. |

| Column | Information |
|---|---|
| 1 | Number of activities bursting from the event, minus one |
| 2 | Number of activities merging into the event, minus one |
| 3 | Early event time |
| 4 | Late event time |
| 5-10 | Numbers of activities bursting from the event |
| 11-16 | Numbers of activities merging into the event |

| GPSS Entity | Interpretation |
|---|---|
| Matrix 3 (Fullword | Activity Sampling Information Matrix<br>Rows 1,2,3,...,X97 carry information for activities 1,2,3,...,X97, resp. |

| Column | Information |
|---|---|
| 1 | Fraction of time (parts per thousand) actual activity time exceeds the most likely estimate |
| 2 | Most likely time minus optimistic time |
| 3 | Most likely time minus pessimistic time |

| GPSS Entity | Interpretation |
|---|---|
| Savevalue j (Fullword)<br>j=1,2,3,...,X96 | Points to one of columns 11-16 in Matrix 2; later used to record the number of activities merging into event j, j=1,2,3,...,X96 |
| Savevalue j (Halfword)<br>j=1,2,3,...,X96 | Points to one of columns 5-10 in Matrix 2; later used to record the number of activities bursting from event j, j=1,2,3,...,X96 |
| Savevalue 96 (Fullword) | Total number of events in the network |
| Savevalue 97 (Fullword) | Total number of activities in the network |
| Savevalue 98 (Fullword) | Number of project completions to date |
| Savevalue 99 (Fullword) | Number of project completions requested |
| Savevalue 100 (Fullword) | Conversion factor used to express outputted information in terms of the original implicit time unit |
| Table 1 | Table of project completion times; Table argument is V15 |
| Variable 1 | Assumes 4 as its value if more than one activity bursts from an event; assumes 5 as its value otherwise (V1=K5-MX2(P2,1)/MX2(P2,1)) |
| Variable 2 | Number of activities bursting from an event (V2=MX2(P2,1)+K1) |
| Variable 3 | Assumes 10 as its value if more than one activity merges into an event; assumes 11 as its value otherwise (V3=K11-MX2(P2,2)/MX2(P2,2)) |
| Variable 4 | Assumes late event time as its value (V4=MX2(X96,4)-MP5) |
| Variable 5 | Number of activities merging into an event (V5=MX2(P2,2)+K1) |
| Variable 8 | Activity's total slack, defined as late time of head event, minus early time of tail event, minus activity's duration (V8=MX2(P3,4)-MX2(P2,3)-MX1(P1,8)) |
| Variable 9 | Used in computation of criticality indices (V9=K10000*MX1(P1,7)/X98) |
| Variable 10 | Number of network activities minus one (V10=X97-K1) |
| Variable 15 | Project completion time (V15=MP5/X100) |
| Variable 16 | Number of activities bursting from an event, minus one (V16=XH*2-K5) |
| Variable 17 | Number of activities merging into an event, minus one (V17=X*2-K11) |
| Variable 18 | Most likely time minus pessimistic time (V18=MX1(P1,4)-MX1(P1,5) |
| Variable 19 | Most likely time minus optimistic time (V19=MX1(P1,4)-MX1(P1,3) |
| Variable 20 | Fraction of time (parts per thousand) that actual activity time exceeds most likely estimate (V20=K1000*(MX1(P1,5)-MX1(P1,4))/(MX1(P1,5)-MX1(P1,3)) |

TABLE 1. TABLE OF DEFINITIONS FOR GPSS/360 MODEL

## 7. RESULTS PRODUCED BY THE GPSS AND FORTRAN MODELS

Although experimentation was performed with five networks, only the results associated with the network in Figure 3 will be discussed here. This network, also used by VanSlyke (1), displays the various time estimates for each activity as a three-component vector located above the arrow representing the activity. The Figure 3 network was studied for the cases of 200, 400, 600, 800, and 1,000 realizations with both models. In addition, it was further studied with the FORTRAN model for runs involving 5,000 and 10,000 realizations. Figures 4 and 5 show the various model-generated approximations to mean project duration and the associated standard deviation, respectively. These model-generated estimates rapidly approach stable values which, as is to be expected, are independent of the model being used. Also displayed in Figures 4 and 5 are the project mean time and standard deviation as calculated from the PERT algorithm. The lack of correspondence between sampled results and PERT results is to be expected in general because of the simplifying assumptions made in the PERT analysis.

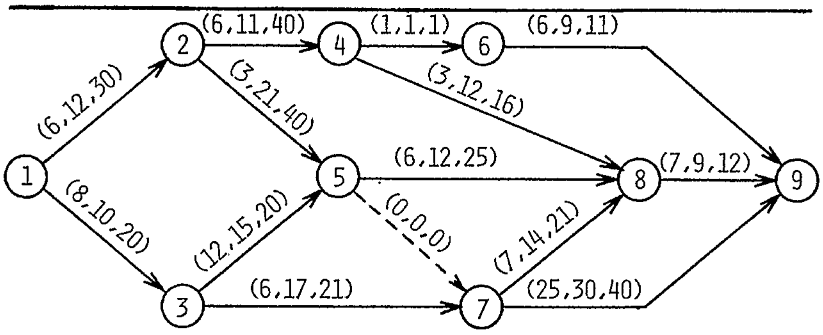

FIGURE 3. PROJECT NETWORK FOR WHICH RESULTS ARE REPORTED HERE

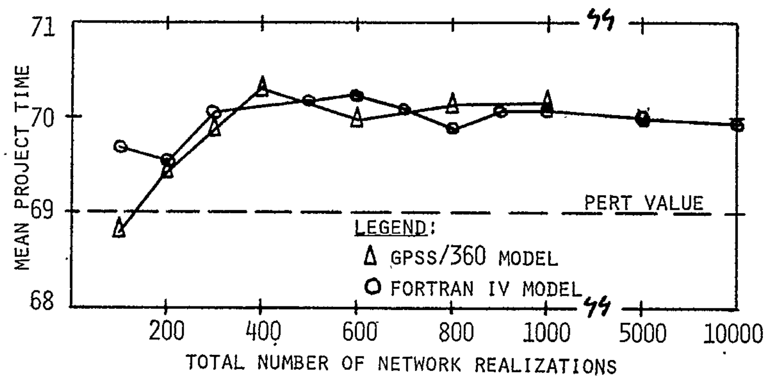

FIGURE 4. ESTIMATED PROJECT COMPLETION TIME AS A FUNCTION OF SAMPLE SIZE

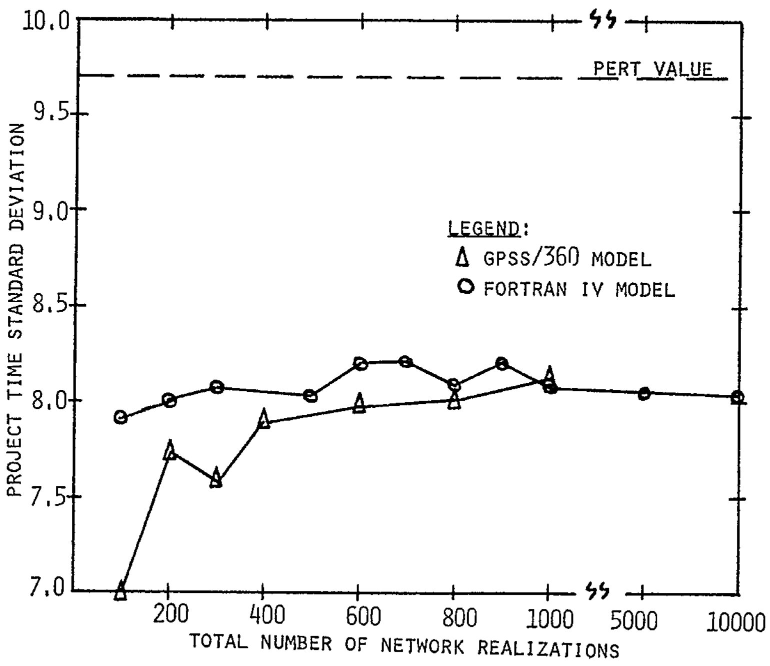

FIGURE 5. STANDARD DEVIATION OF PROJECT COMPLETION TIME AS A FUNCTION OF SAMPLE SIZE

Figures 6 and 7 display the estimates of the project time's density function and cumulative distribution, resp., as developed from use of each of the two models. These plots use information corresponding to runs of 1,000 realizations. Results produced by the two models are fully consistent.

Table 2 shows the simulated estimates of criticality indices as a function both of the model being used and the number of realizations involved. Inspection of Table 2 reveals that the criticality indices are a more sensitive function of the number of realizations than is overall project duration and its associated standard deviation.

Two published papers were used for purposes of testing the criticality indices produced by the present models. VanSlyke's results for the Figure 3 network and our results based on 10,000 realizations vary by as much as 20%. Given the slow rate of stabilization of these indices as suggested in Table 2, such a large difference is not improbable. A further check on the indices produced by our models was made

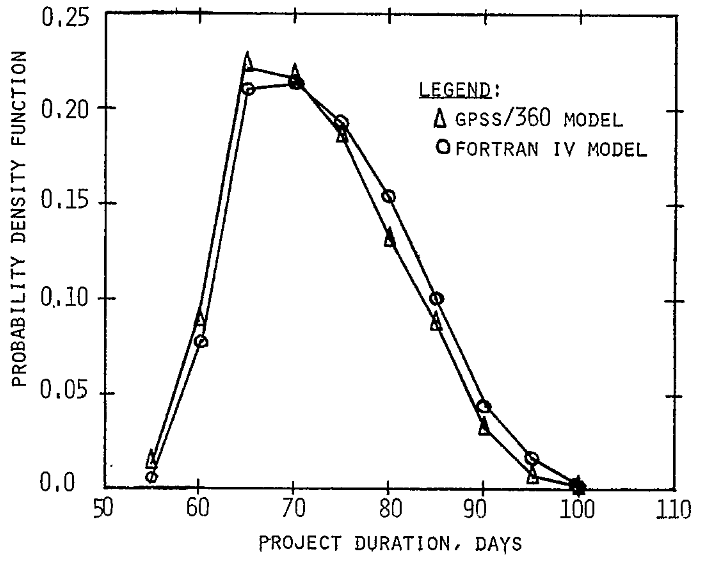

FIGURE 6. PROJECT TIME DENSITY FUNCTION

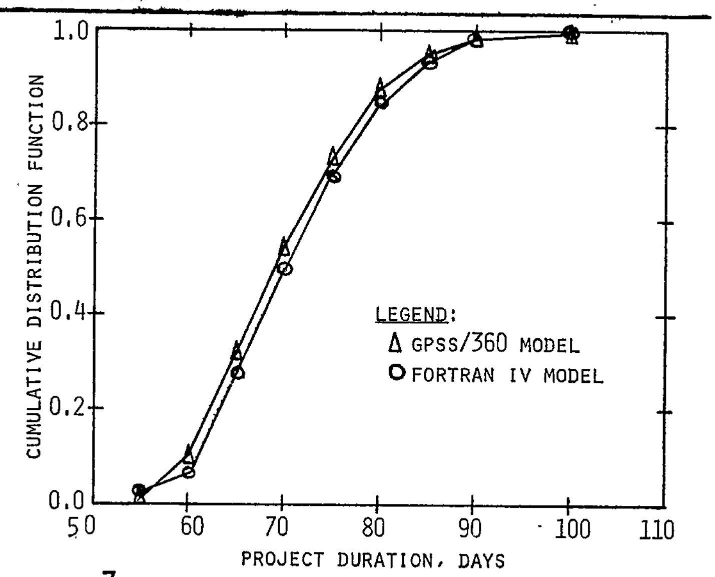

FIGURE 7. PROJECT TIME CUMULATIVE DISTRIBUTION

TOTAL NUMBER OF NETWORK REALIZATIONS

| ACTIVITY | 200 | | 400 | | 600 | | 800 | | 1000 | | 5000 | | 10000 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MODEL: | FORTRAN | GPSS | FORTRAN | GPSS | FORTRAN | GPSS | FORTRAN | GPSS | FORTRAN | GPSS | FORTRAN | GPSS | FORTRAN | GPSS |
| 1-2 | .825 | .820 | .825 | .848 | .810 | .822 | .820 | .830 | .822 | .827 | .835 | | .829 | |
| 1-3 | .175 | .180 | .175 | .155 | .190 | .180 | .180 | .171 | .180 | .174 | .169 | | .175 | |
| 2-4 | .090 | .090 | .080 | .098 | .077 | .072 | .070 | .079 | .072 | .082 | .085 | | .081 | |
| 2-5 | .735 | .730 | .745 | .750 | .733 | .751 | .750 | .751 | .751 | .745 | .750 | | .748 | |
| 3-5 | .105 | .095 | .100 | .083 | .107 | .100 | .104 | .094 | .103 | .099 | .097 | | .098 | |
| 3-7 | .070 | .085 | .075 | .075 | .083 | .078 | .076 | .081 | .078 | .078 | .073 | | .079 | |
| 4-8 | .090 | .090 | .080 | .097 | .077 | .088 | .070 | .088 | .072 | .082 | .085 | | .081 | |
| 5-8 | .095 | .035 | .080 | .040 | .073 | .040 | .068 | .043 | .064 | .045 | .057 | | .055 | |
| 5-7 | .750 | .790 | .768 | .792 | .768 | .798 | .787 | .804 | .791 | .802 | .788 | | .789 | |
| 7-8 | .035 | .015 | .033 | .018 | .030 | .020 | .024 | .020 | .020 | .021 | .014 | | .014 | |
| 7-9 | .785 | .860 | .813 | .848 | .823 | .855 | .841 | .863 | .847 | .857 | .847 | | .853 | |
| 8-9 | .220 | .140 | .193 | .152 | .180 | .147 | .161 | .140 | .156 | .147 | .156 | | .151 | |

TABLE 2. ESTIMATES OF CRITICALITY INDICES FOR PROJECT ACTIVITIES

by using both the GPSS and FORTRAN models with a network discussed by Gray and Reiman (5). In this case, disagreement of criticality indices did not exceed 10%. (It is not clear in (5) how many realizations were used to produce the results presented there. It was inferred that 2,500 realizations were involved, and that number was used for comparative purposes.)

## 8. COST COMPARISONS BETWEEN THE GPSS/360 AND FORTRAN IV MODELS

Figure 8 shows the computer billing charge plotted against the number of network realizations as a function of the model being used.* A marked difference in the cost of using the two models is evident. An object deck was cut and used for the FORTRAN model. This has the effect of lowering the intercept of the FORTRAN cost line slightly but does not change the slope of the line. Relative inefficiencies of the GPSS/360 processor when used for production runs are brought into sharp focus in Figure 8. On the other hand, the relative ease and convenience of model-building using GPSS cannot be overlooked. In fact, this project was approached by doing all the model building initially in GPSS. Various alternatives were tested in a search for that approach which would be optimal in the sense of minimizing the GPSS billing charge. (Billing charge proved to be quite insensitive to the various approaches used in model construction.) The model was implemented in FORTRAN only after considerable insight had been gained through the GPSS modeling process. As a result, the FORTRAN model was constructed with relative ease. In fact,

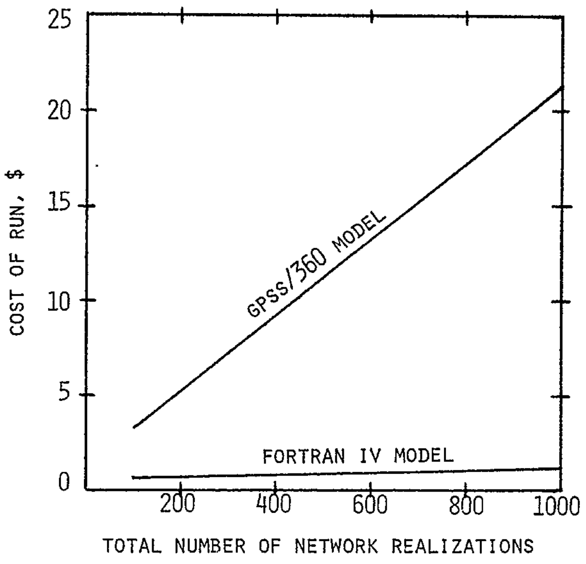

FIGURE 8. COST OF PRODUCTION RUNS FOR THE PROJECT NETWORK IN FIGURE 3.

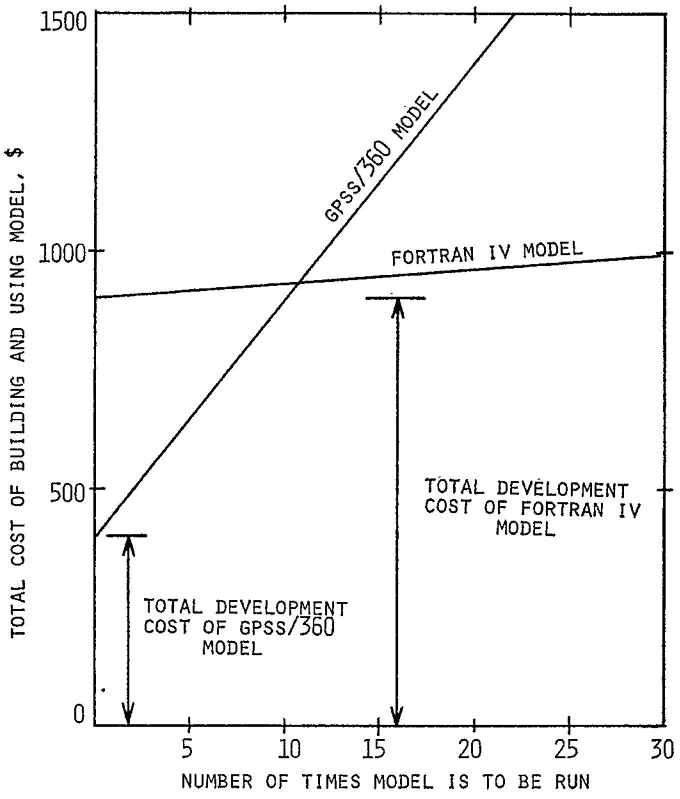

FIGURE 9. TOTAL COST OF BUILDING AND USING THE GPSS AND FORTRAN MODELS FOR THE FIGURE 3 NETWORK

*All runs were made on The University of Michigan's 360/67 multiprocessing system. In this system, two Model 67's are used in multiprocessing mode under supervision of the Michigan Terminal System.

it was found that a high degree of the insight gained by modeling in GPSS carried over into the FORTRAN implementation.

The cost tradeoff between GPSS and FORTRAN modeling of the problem at hand can be further discussed by considering the cost components associated with program development and testing as well as the cost of production runs. We estimate that the FORTRAN model took three times longer to build than did the GPSS model; also, the FORTRAN model required about twice as much time (human) to "debug". On an absolute basis, it is estimated that a total of 80 hours of programming time was required in development of the two final models. Suppose we assume a programmer pay rate of $15 per hour (which, in light of recent industry developments, may be unduly conservative). Assume further that each production run in a given application, or family of applications, costs $3 and $50 for the FORTRAN and GPSS models, respectively (these relative costs are obtained from extrapolating the information available in Figure 8.) Then total cost of building and using the model can be plotted against the number of times the program is to be used, as shown in Figure 9. The Figure 9 information suggests that if a given model is to be used repeatedly, significant cost savings may result if the model is eventually implemented in FORTRAN as opposed to GPSS.

## 9. SUMMARY

GPSS/360 lends itself readily to modeling the fully-generalized PERT network for estimation of activity criticality indices via Monte Carlo sampling. The GPSS model documented here can be duplicated and used without further creative effort being required, and can be studied to the point of understanding if so desired. GPSS is a convenient vehicle for experimenting with alternative approaches to the model-building process. If extended production runs are to be made, significant cost savings can be realized by eventually building the finalized model in FORTRAN. Data are presented to show production run cost differences on a quantitative basis.

## 10. BIOGRAPHIES

Guillermo Ponce-Campos is a Ph.D. candidate in Construction Management in the Civil Engineering Department at The University of Michigan. A native of Peru, he is doing his doctoral dissertation in the area of multi-relationship network analysis. He will tentatively graduate in December, 1970.

Thomas J. Schriber is an Associate Professor of Statistics and Management Science in the Graduate School of Business at The University of Michigan. A 1968-1969 ACM National Lecturer, his research interests include computer simulation of discrete stochastic systems and deterministic applications of the computer in operations research. He is the author of Fundamentals of Flowcharting (Wiley, 1969).

## 11. REFERENCES

(1) VanSlyke, R.M., "Monte Carlo Methods and the PERT Problem", Operations Research, Vol. 11, pp 339-369 (1963)

(2) Schriber, Thomas J., General Purpose Simulation System/360: Introductory Concepts and Case Studies, pp 172-181, (Ulrich's Bookstore, Ann Arbor, Michigan, 1968)

(3) Meier, R.C., W.T. Newell, and H.L. Pazer, Simulation in Business and Economics, pp 37-43 (Prentice-Hall, 1969)

(4) Fishman, George S., Digital Computer Simulation: Estimating Sample Size, The RAND Corporation, Memorandum RM-5866-PR ( August, 1969)

(5) Gray, C.F., and R.E. Reiman, "PERT Simulation: A Dynamic Approach to the PERT Technique", Journal of Systems Management, pp 18-23 (1969)

(6) Fulkerson, D.R., "Expected Critical Path Lengths in PERT Networks", Operations Research, Vol. 10, pp 808-817 (1962)

(7) Hartley, H.O., and A.W. Worham, "A Statistical Theory for PERT Critical Path Analysis", Management Science, Vol. 12, pp 469-481 (1966)

(8) MacCrimmon, K.R., and C.A. Ryavec, "An Analytic Study of the PERT Assumptions", OR, Vol. 12, pp 16-37 (1964)