1981 Winter Simulation Conference Proceedings
T.I. Ören, C.M. Delfosse, C.M. Shub (Eds.)

ADAPTATION OF THE TL EVENT LIST ALGORITHM TO THE GASP IV SIMULATION LANGUAGE

S. Hurtubise, T. Gavin, A. Girard
INRS - Télécommunications
3, Place du commerce
Ile des Soeurs
Verdun, Montreal

Simulation models for large systems have earned the reputation of eating up hours of computer time. This problem can be attributed in part to the fact that a significant amount of bookkeeping time is required to keep future events in their proper sequence. Many methods have been proposed to improve the efficiency of this operation but few results have been published to indicate how they perform in a real simulation environment. This paper shows how simulation time, using a particular simulation language (GASP IV), can be substantially reduced by replacing the existing filing system with a slightly modified version of a recently proposed event scheduling algorithm called the TL algorithm. Guidelines are given for the selection of the parameters of the algorithm, and also some experimental comparisons between the current GASP IV filing system and the new TL/GASP implementation.

The management of the list of future events constitutes a fair portion of the computation required by many simulation systems and a considerable amount of work has been devoted to the design of efficient list management algorithms [5-10]. These various algorithms are generally compared on the basis of the computation time required to perform a standard operation as a function of the length of the list; and the results, although reproducible are still producing some controversy [11].

For the practitioner of simulation, however, these studies constitute only the first step, since the list-management algorithm is only one component of a simulation system where many elements may affect the global efficiency of the program. The purpose of this paper is to report on the various steps that had to be taken to go from one of these theoretical algorithms to a program that could be used in a production environment and the total gains that were achieved.

The purpose of the exercise being to obtain a usable and efficient simulation program, we had to decide a priori on a specific list-management algorithm; we selected somewhat arbitrarily the TL algorithm [7], mostly because of its insensitivity to the length of the list. Our second arbitrary choice was to decide whether to write our own simulation program or to use a ready-made package; here again, we decided that the GASP-IV [3] language would be used for the following reasons; it is widely distributed, written in FORTRAN, and thus can easily be modified, is event-oriented, and finally could give us a point of reference in comparing the results we obtained with some kind of "standard".

Having made these arbitrary choices, a number of questions had to be resolved: how to integrate Tl to GASP, what value to assign to the various TL parameters, and how much better was TL/GASP over the standard version, when used both in isolation and within the context of a real simulation. The answer to these three questions will be found in the three following sections.

81CH1709-5/81/0000-0599$00.75

1. MODIFICATIONS TO THE TL ALGORITHM FOR ADAPTATION TO THE GASP IV SIMULATION LANGUAGE

This section provides technical details of the implementation of the TL algorithm. The parameters are defined below:

DU = number of intervals

DT = width of intervals

n = number of events in the list

NLIM = maximum number of events per sublist

KEYCUR= points to the primary key whose associated sublist contains current event

MFE = points to the current event at the head of the list

μ = exponential distribution parameter used to generate events during testing

$\bar{t}$ = average holding time of an event in the list

TNOW = current simulation time

In GASP, new events are inserted in the future events list by beginning the search for the desired location at the bottom of the list (corresponding to the most future event, and pointed to by a variable MLEX) and proceeding event by event toward the top. The purpose of the TL/GASP implementation proposed here is to ensure that the starting point for the search (we continue to use MLEX for the corresponding pointer) is in fact close to the desired location. The search is thus restricted to the sublist pointed to by the appropriate key. We now describe the modifications made to the TL algorithm for use with GASP IV.

The main difference between the TL algorithm described in [7], which we denote by TL/F, and our version of the TL algorithm, (TL/GASP), lies in the elimination of the INDEX table. If the DU (=number of intervals) first keys are reserved for use as primary keys, then the calculation of the index I, which previously led in the INDEX table to the pointer of the desired primary key, now yields directly the index of this key. The variable KEYCUR, which points to the primary key containing the current event, is kept, but the variable ICUR is dropped.

Another difference is the elimination of the dummy primary key at the head of the list. The reason for this is simple: adding a dummy event having a value of t = 0, to the head of the list is incompatible with the GASP subroutine (RMOVE) which removes events from the list. Instead of a dummy key technique, in TL/GASP the key at the head of the list will be recognized using the same technique as for the events. The right pointer of the key at the head of the list will be equal to zero. (See fig. 1b)

The INTERVAL and CURRENT registers have also been eliminated. CURRENT is the equivalent of MFE in GASP IV, the pointer which indicates the event at the head of the list. INTERVAL is no longer necessary and is therefore eliminated. Only the subroutines SET, FILEM, RMOVE and PRNTQ were modified for setting up the TL/GASP algorithm. Figure 1a shows the TL/GASP list structure, and figure 1b the original TL structure for comparison.

2. THE DESIGN FORMULAS

The equations given in [7] for calculating TL algorithm parameters apply even for very irregular distributions of events. An example would be a distribution where all events occur over a short period of time. The efficiency of the algorithm can be increased if we are willing to sacrifice its robustness. The equations given below are well adapted to fairly regular distributions, but could prove to be less efficient in certain other cases. As each parameter is discussed, its applications are indicated.

a) DU - number of intervals in the list

Based on experiment, the authors of [5] and [7] recommend that the number of intervals be between 20 and 50 and suggest a value of 30. For our pilot runs using the TL/GASP algorithm and a DU value of 30, we found that, as was the case with the TL/F algorithm, the execution time was virtually unaffected by variations in DU.

b) NLIM - maximum number of events between two secondary keys

Tests showed this parameter to be the crucial one. We would like to choose a value for the NLIM parameter such that the same effort would be required for a search at the key level as for a search at the event level. The equation given by Franta for calculating NLIM is:

$$NLIM=\max(8,\sqrt{n}) \qquad (1)$$

To arrive at this equation, we consider the extreme case where all events fall into the same block. In order that the search on both levels be well balanced, there must be as many events associated with one key as there are keys; thus NLIM $=\sqrt{n}$. The lower limit is taken from experimental results and is explained by the fact that the time taken by key rearrangement must have a lower bound.

Equation (1) ensures that even for worst-case distributions, inserting an event into the list requires at most a search of the order of $\sqrt{n}$. On the other hand, tests have shown that if NLIM were varied to find its optimum value, then as long as the value of DT (width of intervals) was well chosen and all events did not occur in the same block, the execution time could be reduced to half that resulting from the use of (1).

Let us now take an optimistic view. Suppose the events are evenly distributed in all the blocks. There would then be n/DU elements per block. If we incorporate this into (1), we arrive at the following expression for NLIM:

$$NLIM = MAX\ (8,\sqrt{n/DU}) \qquad (2)$$

Experimentally, the optimum value for NLIM was 10% to 20% above this optimistic value, but often two or three times lower than the pessimistic value.

We could go even further and assume a different value of NLIM for each block. What we are really trying to do is spread the search evenly over the two levels of a block. We would then have to count the number of elements in each block and define NLIM for each block. This case was not taken into consideration in our implementation of TL/GASP.

If NLIM is to be calculated according to equation (2), then the random distribution of events should be such that the events are fairly evenly divided among the various intervals of the list. In simulations, and particularly for communication network simulators, this restriction is usually met.

c) DT - width of the list intervals

At first thought, it would seem that the value chosen for DT should be of major importance. If DT is too large, the events will have a tendancy to end up in the same interval of the list, and the results will resemble those obtained using the double pointer method. On the other hand, if DT is too small, there will be very few events in each interval and most will fall into the last block to the right (the semi-infinite block). In addition, so much time will be spend rearranging the primary keys that the algorithm will be as inefficient as when DT is too large. It was found experimentally that as long as DT is within certain limits, the value chosen is not so important. This result will be discussed subsequently.

The following equation for the calculation of DT is proposed in [7]

$$DT = [(8/(n-1) + .07] * \bar{t} \qquad (3)$$

Generally, $\bar{t}$ and n are not easily determined. On the other hand, we may attempt to simplify the equation for use in a network simulator when the distribution of the majority of the exogenous events in the list is known. For example, in a telephone network simulator the arrival of calls follows a Poisson distribution and the duration of the calls follows a negative exponential distribution and the duration of the calls follows a negative exponential distribution with parameter μ. Only the next arrival of an exogenous call (for each origin-destination pair) is inserted into the event list. Due to the behavior of the network, a very small percentage of the events could be exogenous call arrivals and the remainder calls in progress. In this case, since the majority of future events consists of events already in the system, we can assume $\bar{t}=\mu$. Under this hypothesis we determined experimentally that

$$DT = 2.1 * \mu/DU \qquad (4)$$

3. EXPERIMENTAL RESULTS

The HOLD operation was used as a control operation. It removes the first event from the list, randomly generates a new event time, and replaces the event in the list. This operation was chosen primarily because it closely represents the operation most often executed in simulation. Since an event is immediately reinserted into the list after having been removed, the number of events in the list is constant. HOLD also has the advantage of combining the insertion and removal operations. It is an accepted operation frequently used in the evaluation of event list simulation algorithms.

For the results given below, the events were generated using the GASP IV random number generator (DRAND) with the Poisson parameter μ equal to 1.

a) Comparison of execution times

In Figure (2), we compare the event list simulation algorithm presently used in GASP IV with the proposed TL/GASP algorithm. This figure was obtained by executing the HOLD operation 1000 times (on an IBM 3033 computer with a FORTRAN H OPTIMIZATION 3 compiler) using the following parameters:

DU = 30

DT = .1

NLIM = 15

μ = 1.0

It appears that the TL/GASP algorithm is more efficient once there are 25 or more events in the list.

b) Sensitivity study of the parameter NLIM (# of elements in a block) Figure (3) shows a curve representing the execution time in microseconds of one HOLD operation as a function of the value chosen for NLIM. It should be noted that, according to equation (1) proposed by Franta, the value for NLIM should have been 70 since the list has 5000 elements.

According to equation (2) NLIM should be equal to 13. The optimum value for NLIM is around 20. If the optimum value for NLIM is used rather than the value suggested by Franta, execution time will be reduced by nearly 35%. A sensitivity study has shown that if NLIM varies from the optimum by plus or minus 25%, the execution time will vary by approximately 15%. The oscillations in this curve would merit further study.

c) Sensitivity study of the parameter DT (width of intervals)

For the distribution studied, equation (4) yields DT = .07, which is very close to the optimum shown on figure (4). If DT is varied from the optimum by plus or minus 33%, the execution time will vary in the order of 1.2%. This shows that as long as the value of DT falls within certain limits, it will have almost no influence on the efficiency of the event list algorithm.

Notice that according to the equation given by Franta, DT would also be .07. It does, however, require knowledge of the average holding time and the number of events in the list.

d) Performance in network simulation.

The TL/Gasp algorithm was developed in the course of a study of routing algorithms for telephone networks. These models gave us an opportunity to compare our program with the standard GASP in the context of a realistic simulation, and to evaluate the overall benefits that could be gained by TL/GASP.

In the first three cases reported in Table I, the total number of calls was kept constant, and only the rate of production of external events (call arrivals) was increased; thus, the same number of events was processed by the simulator, but with an increasingly large number of elements in the future events list. The results clearly show that TL/GASP has effectively removed the list length as a factor in running time.

A fourth case is also given of a larger network showing the substantial reduction in running time (here by a factor of 25) possible for practical simulations; these values are by no means an upper limit, since the more future events are present in the model, the larger will be the reduction. All times given here are in seconds of CPU on an IBM 3033.

	Run times (CPU Seconds)	
	TL/GASP	Standard GASP
1	5.58	10.79
2	5.46	15.19
3	5.69	20.58
4	6.19	154.4

Table I

Comparison of TL/GASP and Standard GASP
on network simulations.

4. CONCLUSION

The main results are these:

(1) Given a few simple modifications, the TL list management algorithm can easily be incorporated to the GASP-IV simulation system.

(2) The overhead associated with the management of the future events list in the TL/GASP implementation is essentially independent of the size of the list. The simulation time is simply a function of the number of events selected to obtain statistics, and of the efficiency with which the latter are processed.

(3) The calculation of the parameter DT has been simplified. In most cases we do not need an estimate of the number of events in the list, but just some knowledge of the distribution of the events.

(4) We have shown that an optimization of the NLIM parameter can diminish execution time by as much as 35% in comparison with the value suggested by Franta's equation.

(5) The increased efficiencies observed for isolated tests using the "HOLD" operation are also present in the simulation of a "real" system; reductions by a factor of 25 have been measured in the simulations of telephone networks.

The compatibility of the GASP IV subroutines with this new TL/GASP algorithm was checked for discrete simulations. For those used in continous simulations, no compatibility problems are expected, but this was not verified.

5. REFERENCES

[1] Schriber, T.J., A GPSS Primer. John Wiley, New York 1972.

[2] Wyman F.P., Simulation Modeling: A guide to using SIMSCRIPT, John Wiley, New York, 1970.

[3] Pritsker, A.A.B., The GASP IV simulation language, John Wiley & Sons 1974, 451 p.

[4] Knuth, D.E., The art of computer programming, Vol. 3, Addison-Wesley, Reading, Mass., 1973, ch.5.

[5] Vaucher, J.G. and Duval, P., A comparison of simulation event list algorithms, Communications of the ACM 18, 4 (April 1975), 223-230.

[6] Wyman, F.P., Improved event-scanning mechanisms for discrete event simulation, Communications of the ACM 18, 6 (June 1975), 350-353.

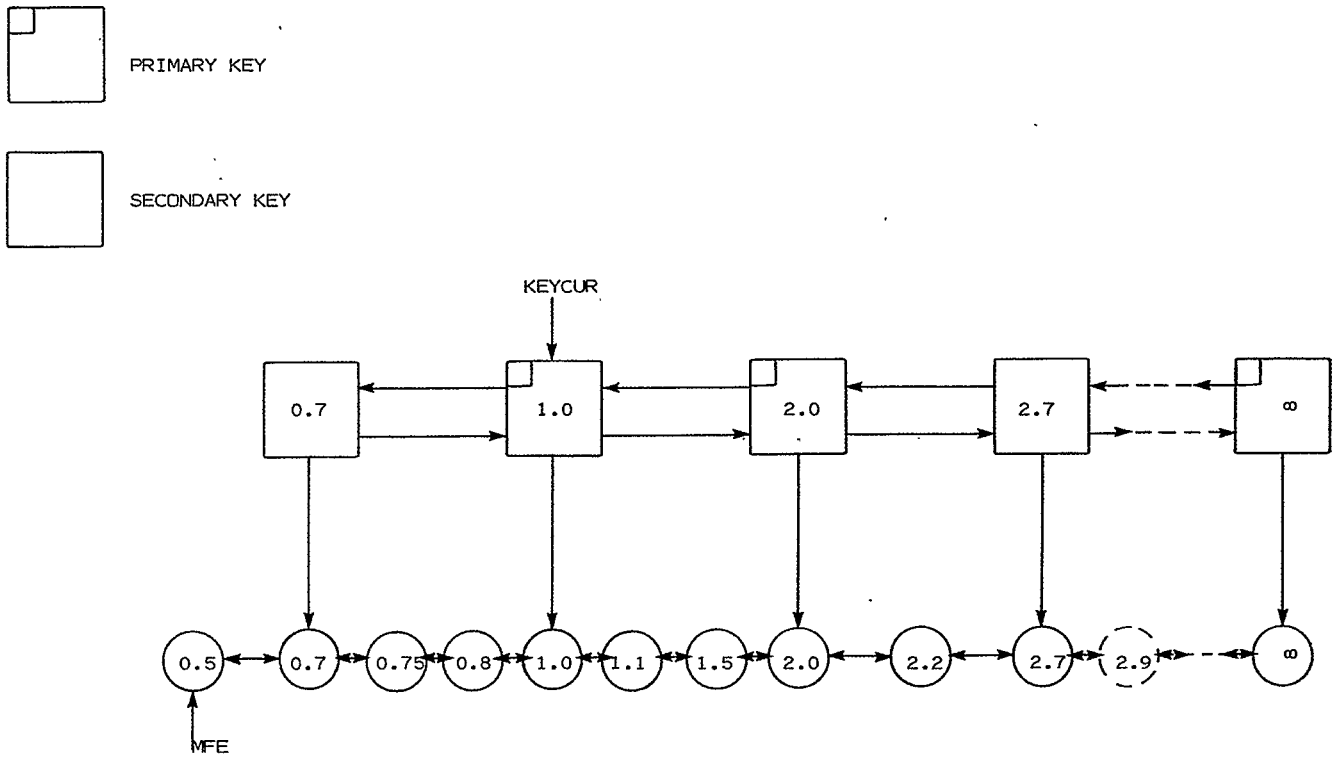
[7] Franta, W.R. and Maly, K., An efficient data structure for the simulation event set, Communications of the ACM 20, 8 (August 1977), 596-602.

[8] Franta, W.R. and Maly, K., An event scanning algorithm of nearly constant complexity, Tech. Rep. 75-18. Univ. of Minnesota, Minneapolis, Minn., Nov. 1975.

[9] Ulrich, E.G., "Event Manipulation for Discrete Simulations Requiring Large Number of Events". CACM, 21, 9, (Sept. 78) pp. 777-785.

[10] Phillips, D.N. and Tellier, J.G., Efficient Event Manipulation - the Key to Large-Scale Simulation. Proc. Semiconductor Test Conference, Nov. 1978, pp. 266-273.

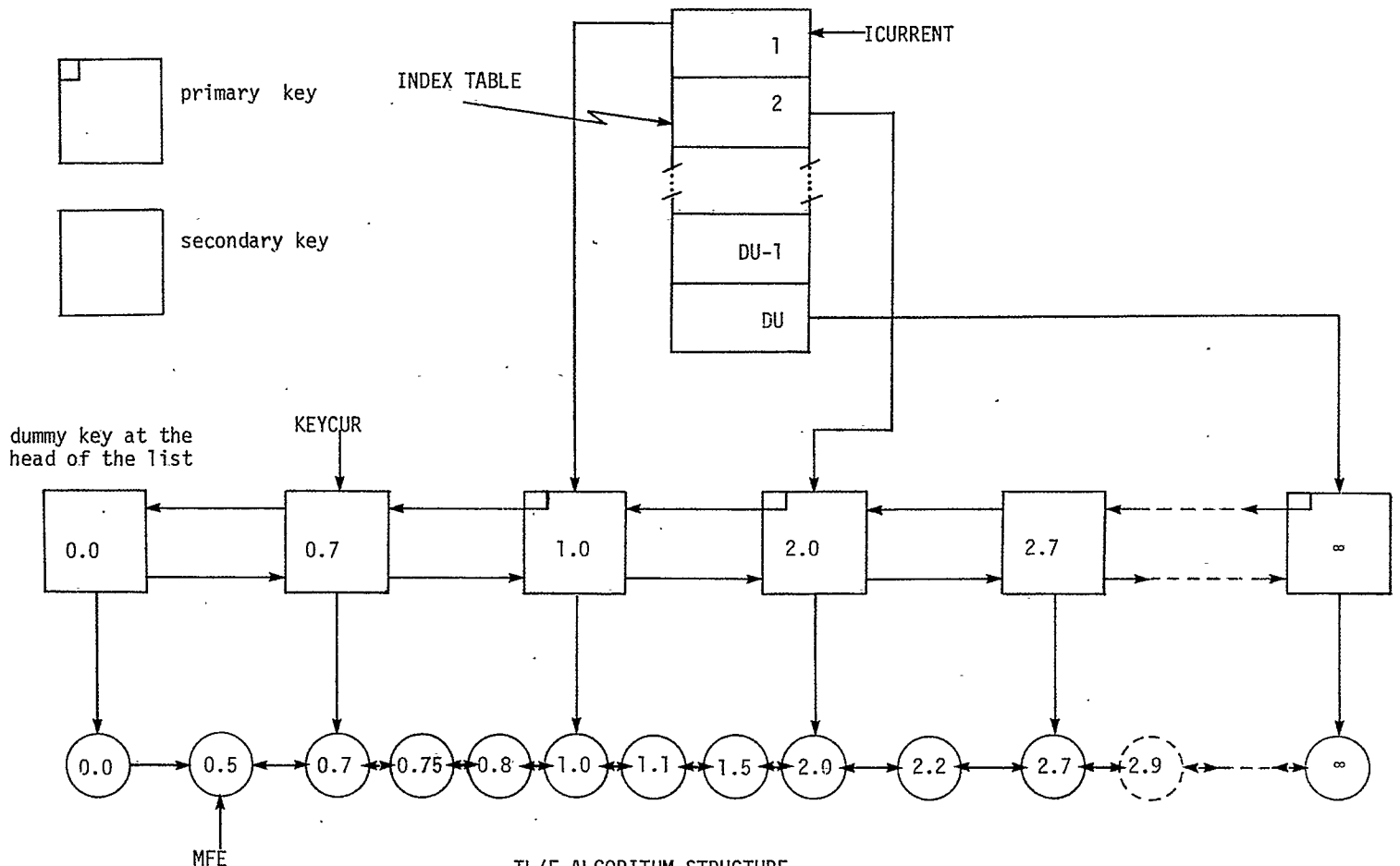
[11] Technical Correspondence, CACM, March 1980, pp. 180-181.



TL/GASP ALGORITHM STRUCTURE

(DT: 1.0, TNOW: 0.5, NLIM: 3)

FIGURE #1A



TL/F ALGORITHM STRUCTURE

(DT:1.0, TNOW:0.5, NLIM:3)

FIGURE #1b

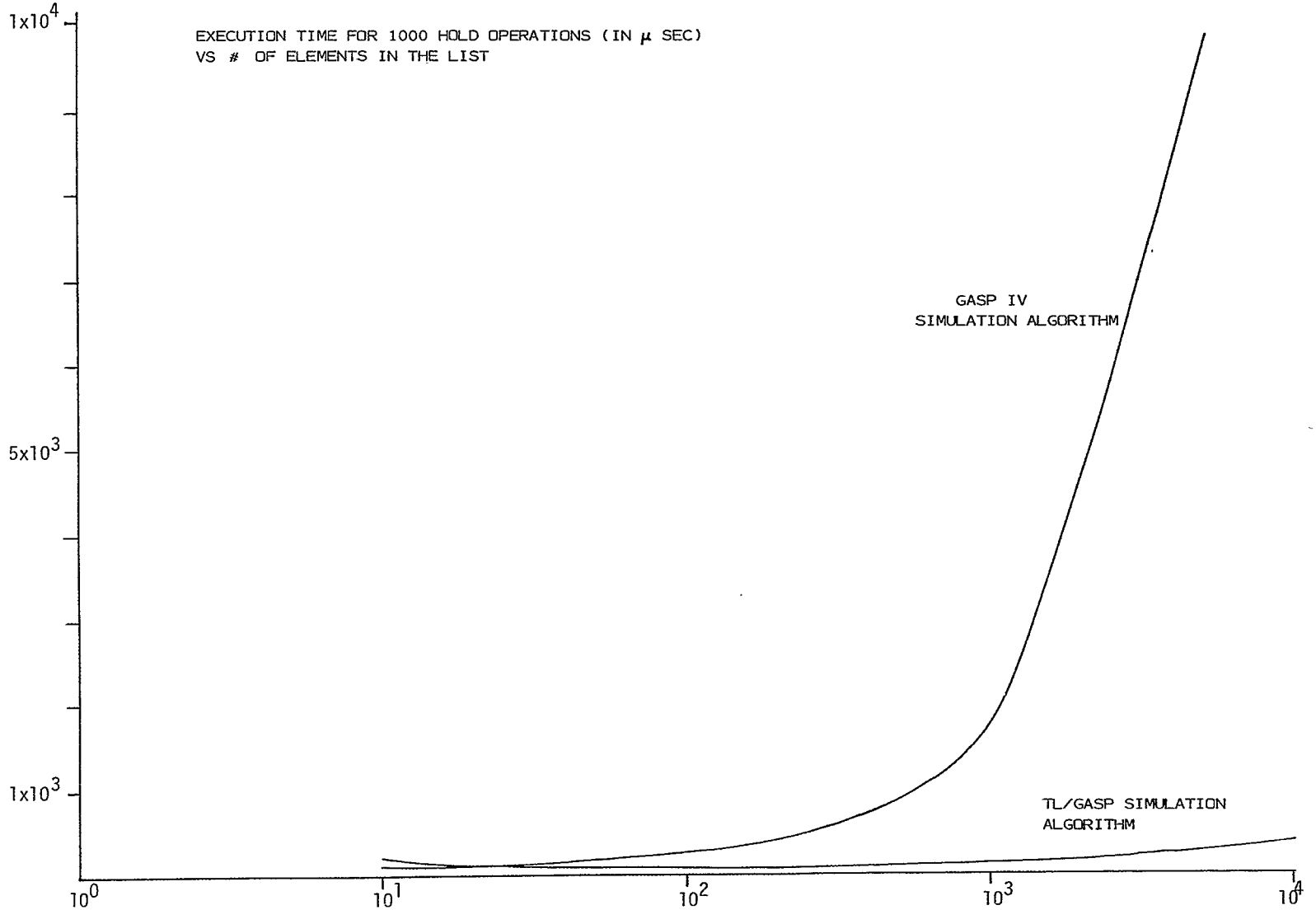

EXECUTION TIME FOR 1000 HOLD OPERATIONS (IN μ SEC)
VS # OF ELEMENTS IN THE LIST
1x10^4
5x10^3
1x10^3
GASP IV
SIMULATION ALGORITHM
TL/GASP SIMULATION
ALGORITHM
10^0
10^1
10^2
10^3
10^4

FIGURE #2

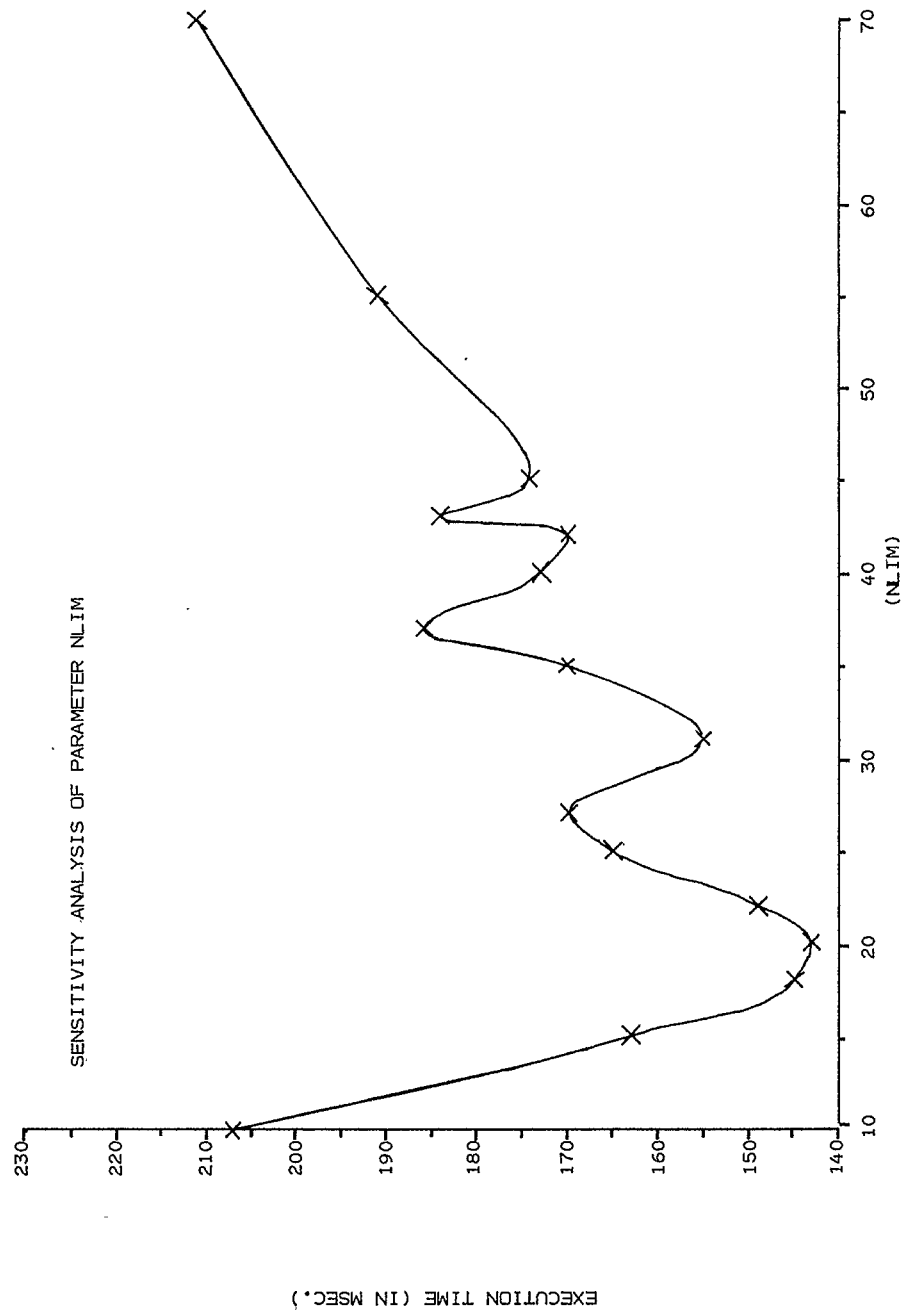

SENSITIVITY ANALYSIS OF PARAMETER NLIM
EXECUTION TIME (IN MSEC.)
(NLIM)
230
220
210
200
190
180
170
160
150
140
10
20
30
40
50
60
70

FIGURE #3

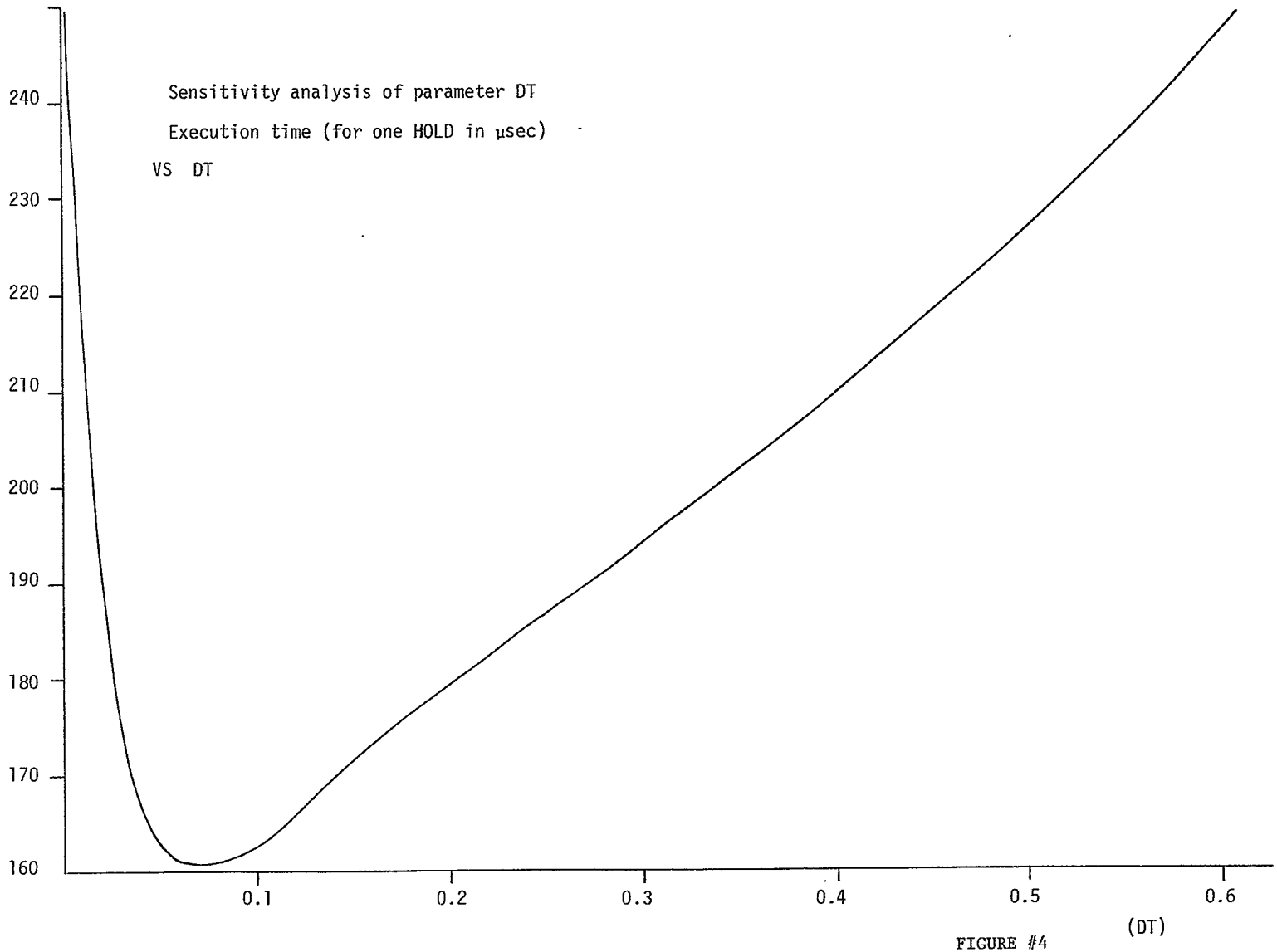

Sensitivity analysis of parameter DT
Execution time (for one HOLD in µsec)
VS DT
240
230
220
210
200
190
180
170
160
0.1
0.2
0.3
0.4
0.5
0.6
(DT)

FIGURE #4