AN EMPIRICAL COMPARISON OF ADVANCED EVENT FILE SYNCHRONIZATION STRUCTURES

by

Luis C. Rodriguez, Texas A&M University
Gary L. Hogg, Texas A&M University
John H. Blackstone, Jr., Auburn University

ABSTRACT

Recent articles have presented new event file synchronization structures for discrete event simulation that project performance superior to the linked list structure used in most simulation systems. The three newly developed structures are the time indexed list, the Two List structure, and the TL structure. This provides an empirical comparison of these structures and the traditional linked list structure, within a simulation system. The structures were adapted to the GASP IV simulation language and tested with a set of discrete event simulations under various conditions. The results show that all three structures proved vastly superior to the linked list structure, with the TL structure outperforming its counterparts.

1. INTRODUCTION

In many cases the most important factor involved in the total execution time of a simulation model is the time required to file an event in the event file. Thus the computer running costs are often strongly influenced by the synchronization mechanism embedded in the simulation language. Most general purpose simulation systems utilize singly or double linked lists in order to maintain the event file during a simulation. A problem that users face in applying the traditional structure is that as the event file becomes larger, the time required to file or retrieve an event may become quite large. Over the past few years several new event file synchronization structures for use in discrete event simulations have been presented in the literature (1), (2), (3), (4), (7). These advanced synchronization mechanisms can increase the efficiency of the event filing process and reduce the computer time necessary to file or retrieve an event. Three major approaches have been suggested in order to better the linked list structure currently used in most simulation systems. These approaches are heaps, binary trees, and time indexed lists.

The use of heaps as the filing structure to utilize in simulation systems was suggested by Gonnet (5). Even though heaps were shown to be more efficient than linked lists, Vaucher and Duval (7) demonstrate that they destroy the ability to order simultaneous events by certain secondary ranking methods. Wyman (8) proposed the use of the time indexed list as an event file synchronization mechanism and proved that it was superior to the linked list filing structure. The possibility of using binary trees was studied by Vaucher and Duval. They demonstrated that binary trees were superior to the linked list; however, they also found that the tree structures were inferior to the time indexed list. Franta and Maly (3) introduced the TL (Two Level) structure, a more efficient time indexed list structure with two levels of indexing, and after comparing it with the time indexed list they concluded that the TL structure was superior, except for small event file sizes. Franta and Maly (3) also compared their structure with heaps and found it to be superior. Blackstone, Hogg, and Phillips (1) developed the Two List structure which utilized two linked lists with different filing priorities and compared it with the time indexed list in large scale simulations. They concluded that the two structures were comparable, with the Two List filing structure possibly somewhat faster. The Two List structure contains some unique features which make it particularly attractive when frequent event cancellation occurs.

The purpose of this paper is to make a comparison of three mechanisms for event file synchroniza-

Proceedings of the 1982
Winter Simulation Conference
Highland * Chao * Madrigal, Editors

82CH1844-0/82/0000-0189 $00.75 © 1982 IEEE

tion structures for discrete event simulation: (1) the time indexed list, (2) the two list method, and (3) the TL structure. The linked list structure will also be used in the study in order to illustrate the improvements made by the new structures over the traditional approach. The comparisons will be made by incorporating each of the filing structures into the GASP IV simulation language and running a set of discrete event simulations under varying conditions for each structure. The source listings of the GASP IV routines modified in order to incorporate each of the synchronization structures are available from the authors on request. A detailed description of each of the structures is first presented, then the theoretical merits of each are compared and finally the experimental results obtained by the study are presented.

2. THE EVENT LIST STRUCTURES

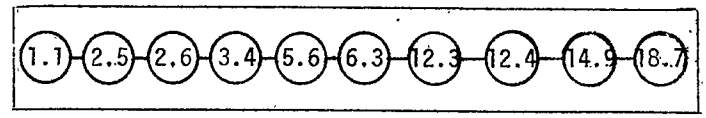
The Linked List Structure

This linked list structure is the synchronization procedure currently found in most simulation systems. It consists of a linear list of synchronized events to be carried out in future simulated time (Figure 1). GPSS, SIMSCRIPT, GASP IV, and SLAM use linked lists. SLAM II uses a binary tree. GEMS uses the two list structure.

In a forward search linked list structure, when a new event is created the new event time is compared with the event time of the first entry in the events list. If the event time of the new event is less than that of the first entry, then it is placed at the front of the list. If the event time of the new event is greater than the event time of the first event, subsequent events are checked in ascending order of event time. When the first event having a larger time is located, the new event is inserted in front of this event, and the appropriate file pointers are updated.

Assuming a uniform event time distribution and a constant file length n, approximately n/2 entries are searched on the average in order to file one event. Englebrecht-Wiggans and Maxwell (2) demonstrate that for non-uniform distribution the search length of n/2 is an approximation. If the shape of the distribution is known in advance one can exploit the shape to choose either forward or backward searches in order to achieve a search length of less than n/2.

Figure 1: The Linked List Structure



The Time Indexed List Structure

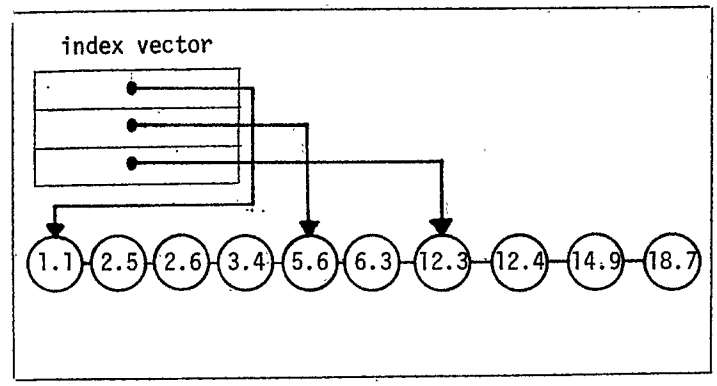
This filing structure consists of a linked list of events and a vector of index pointers indicating the locations of events that subdivide the events list into a set of unique time intervals (Figure 2).

When a new event is created, the index pointers are searched in order to determine the time interval in the events list where the new event will be filed. Upon determination of the interval, the events internal to that interval are scanned in the same fashion as the linked list procedure and the new event location is determined. The new event is then properly placed into the events list.

When all the events in the first interval have been exhausted by the simulation, the index pointer associated with the first interval is redefined. The existing final interval is scanned in order to locate the first entry of what will be the new final interval. Then the first interval index pointer is redefined as the new last interval index pointer.

Assuming uniform event time distribution and a constant file length n, Wyman shows that the optimal number of index pointers is $n^{1/2}$ and that on the average each of the intervals should contain $n^{1/2}$ events (8). These results indicate that approximately $n^{1/2}$ entries are searched on the average in order to file one event in the event file. Englebrecht-Wiggans and Maxwell have extended these results to non-uniform distributions, and present a general formula for optimizing the number of intervals. [As a part of this research the authors have created an internal mechanism for creating a near optimal number of indices, thereby eliminating the need for the user to make this computation.]

Figure 2. The Time Indexed List Structure



The TL Structure

This structure is based in part on the time indexed list and in part on the notion of balanced trees (3). The structure consists of: (1) linked list of events that is subdivided into sublists, the last of which is an overflow sublist; (2) a linked list of primary and secondary keys, where the primary keys point to dummy events that bound sublists of equal-sized time intervals in the events list, and the secondary keys point to normal events that bound sublists created by the balancing mechanism of the structure; and (3) a vector of index pointers that indicate the primary keys in the key list (Figure 3).

When a new event is to be filed, first the index vector is utilized to find out between which two primary keys in the key list the new event time occurs. Then the key list is scanned between the two primary keys in order to locate the proper sublist where the new event will be filed. Finally, the sublist of the events list is scanned and the new event is properly linked into the list.

The notion of balancing the sublists is incorporated into the structure in order to limit the number of entries which must be scanned for the insertion of a new event. During the insertion process, if a sublist becomes too large then the structure is balanced by either moving the new event into an adjacent sublist or by creating a new sublist with an associated secondary key. These secondary keys are created and deleted as necessary.

When the first interval is exhausted, the dummy event and its associated keys are redefined by incrementing their present time frames by a fixed time quantity. The redefined entries are then relocated in the structure where their new frames are located. Secondary keys are destroyed when their sublists are exhausted.

Franta and Maly point out that this filing structure is more efficient than the time indexed list; therefore, under ideal conditions the average number of entries scanned for an insertion is less than $n^{1/2}$, where n is constant file size (2).

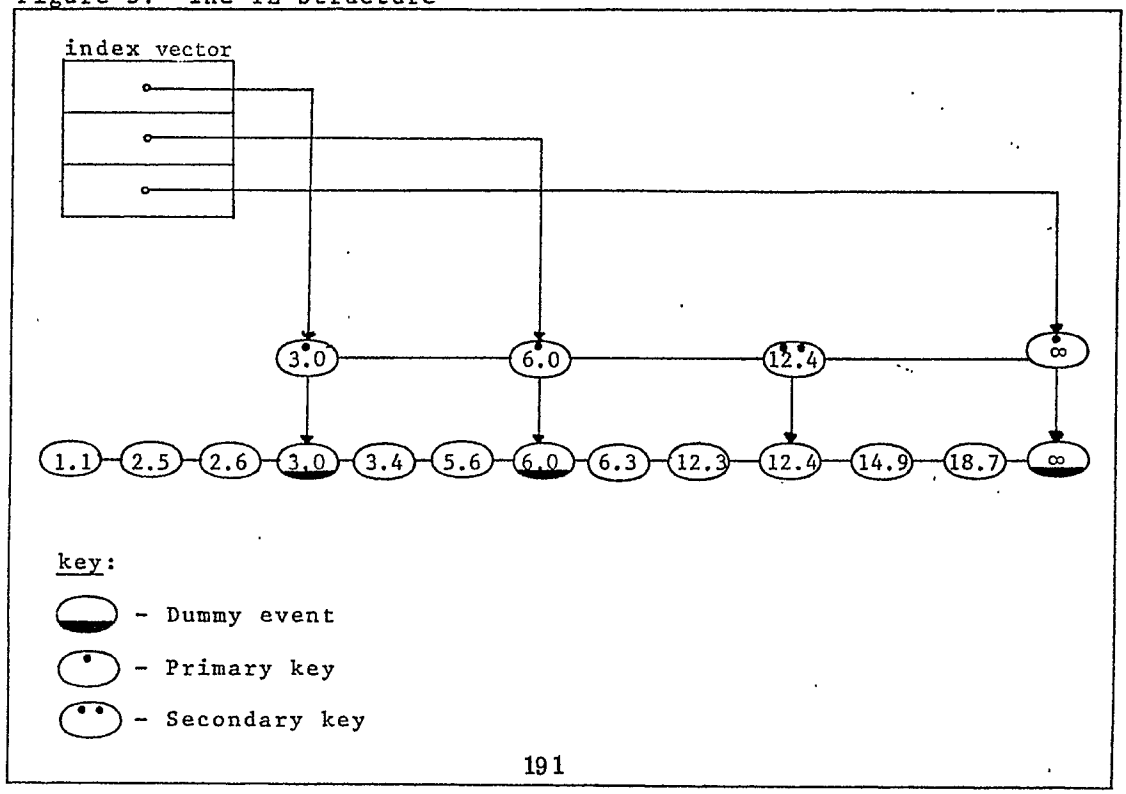
The Two List Structures

This structure consists of two linked lists with different filing priorities. The first file contains a synchronized list of all the events with times less than a previously determined break time. The second file contains all the events with times greater than or equal to the break time in FIFO order (that is, the order in which they were created). Thus, placing an entry in this file requires no search (Figure 4).

Figure 4: The Two List Structure

File 1:	1.1	2.5	2.6				
File 2:	18.7	5.6	3.4	14.9	6.3	12.4	12.3
Break Time:	3.0						

Figure 3: The TL Structure



The filing process for a new event is simple. First the new event time is compared against the break time. If the event time is less than the break time then the event will be filed in the first file, and this process is that of a traditional linked list. If the event time is greater than or equal to the break time, the event is placed at the front of the second file (no synchronization).

When the first file is exhausted a search procedure is invoked. This procedure first increments the break time by some predetermined or internally calculated time interval. This time interval may be updated on each occasion that the search procedure is invoked. The second file is then searched from the front and all the entries having event times less than the new break time are relocated and properly linked in synchronized order in the first file.

Assuming a constant file size of length n, Blackstone, et. al. prove that the average number of entries scanned per event is $n^{1/2}$ or less (1).

3. FILING STRUCTURE COMPARISON

Methodology

In order for an event list filing structure to be incorporated as part of a general purpose simulation system, it is required to operate under a wide variety of conditions. It is difficult to determine theoretically how a method will perform in actual practices since many simplifying assumptions must be incorporated into a rigorous mathematical analysis. An alternative method of comparison is to adapt these structures into a general purpose simulation system and to run a typical set of simulations for each structure under varying conditions. This latter approach was employed in this paper.

The GASP IV simulation was utilized as the test vehicle not only because of the authors' familiarity with it, but also because it is FORTRAN based as are many simulation languages and programs. Further GASP IV, its forerunners and more recent extension such as SLAM are commonly used.

Three discrete event simulations were chosen as the test set: an inventory system simulation, an oil tanker fleet simulation, and a drive-in bank simulation. Source listings and more detailed information on these simulations can be found in the GASP IV textbook (6). Each simulation was run under varying event file sizes for each of the filing structures thus providing more information as to the behavior of each one than the example would provide without modification.

Structure Comparison

The theoretical comparisons in the literature lead one to believe that the structures should have comparable performances. Nevertheless, many characteristics of real problems would appear to present some hardships to some of the structural formats presented previously. These are discussed below.

One problem encountered in real world problems is that skewness in the event time distribution is generally expected (contrary to most theoretical comparisons). The time indexed list structure is at a disadvantage, unlike the other structures, when this occurs. Since its interval sizes are fixed, skewness in the event time distribution will cause some intervals to be more densely populated than others. These same intervals are those where most of the filing will take place, thus hampering the performance of the structure. The TL structure, by way of its balancing mechanism, handles this problem well. As the sublists grow too large it creates new sublists, thus maintaining a reduced filing time. Meanwhile, the Two List structure can dynamically update its break time increment after each first list exhaustion. In this manner it can increase or decrease the increment so that the first list will approach a near optimal number of entries.

Another problem is that of dynamic changes in the number of events found in the event file. Again, the structure that would appear most handicapped is the time indexed list. As the number of events in the list grows, the accomodation of the new events in the fixed intervals will cause some of the intervals to be densely populated. This should in turn result in longer search times. The adaptiveness of the other two structures helps to avoid the problem of dynamic event file changes, to some degree. Clearly the TL structure should perform better than the Two List structure since the Two List structure must wait for the first file to be exhausted each time before it can dynamically update the break time increment size. The dynamic balancing of the TL structure is virutally immediate.

Finally, a potential difficulty found in two of the structures (time indexed and TL) is that they require user defined static parameter estimates which give shape and form to the filing structure. Utilizing static user defined parameters to develop

the structure is not a pleasing philosophy since: (1) the user will then need to be familiar with the filing structure utilized; and (2) depending on the complexity of his simulation, the required parameters may be difficult to obtain and the performance may be good or bad depending on whether good choices for the parameters were made. Instead, a better form is to make the user unaware of the filing structure used in the simulation system by permitting the system to provide some 'ball park' estimates in order to develop the structural format. The authors revised the time indexed and TL methods to include an internal estimation of the required parameters. The manner in which this was accomplished was to utilize a linked list synchronization structure until the first 100 events were removed from the event file. At that time an event was created that caused the system to invoke a procedure that would estimate the required parameters (e.g., number of pointers) and then create the appropriate filing structure. In this manner both the structures were made into more general purpose structures at a very low cost in terms of exeuction time. The two list method was initially designed to include this user transparency and thus no modification was required.

Looking at the different strucutres and the manner in which they operate, one can come to the conclusion that all three new structures should perform much better than the linked list structure. The adaptive abilities of the TL structure make it the theoretical choice as the best of the filing structures presented in this study. It seems that the time indexed list should not perform as well as the other two newly developed structures because of its static nature. The experimental results of this study are presented in the next section.

Results

As mentioned previously, three examples from the GASP IV test were chosen to benchmark the procedures. These include two queueing systems and an inventory system. The parameters of the arrival distributions in the models were adjusted to produce three different average event file sizes for each of the discrete event simulations. Table 1 summarizes the execution times in seconds for all the runs made on an AMDAHL 470/V6 at Texas A&M University.

Table 1: Execution Times

Inventory Simulation

	Event File Size			
Structure	10	100	500*	500
Linked List	9.68	220.00	341.20	3412.0
Two List	10.11	140.98	155.70	1557.0
Indexed List	11.00	97.69	124.67	1246.7
TL Structure	10.51	98.07	38.35	383.5

Tanker Simulation

	Event File Size			
Structure	10	100	500*	500
Linked List	12.18	252.50	393.77	3937.7
Two List	12.57	169.69	106.69	1066.9
Indexed List	12.92	136.54	126.72	1267.2
TL Structure	12.45	134.68	75.09	750.9

Bank Simulation

	Event File Size			
Structure	10	100	450*	450
Linked List	11.83	178.79	363.15	2269.7
Two List	12.29	157.73	294.10	1838.1
Indexed List	13.18	133.48	88.56	553.5
TL Structure	11.83	113.00	75.91	474.4

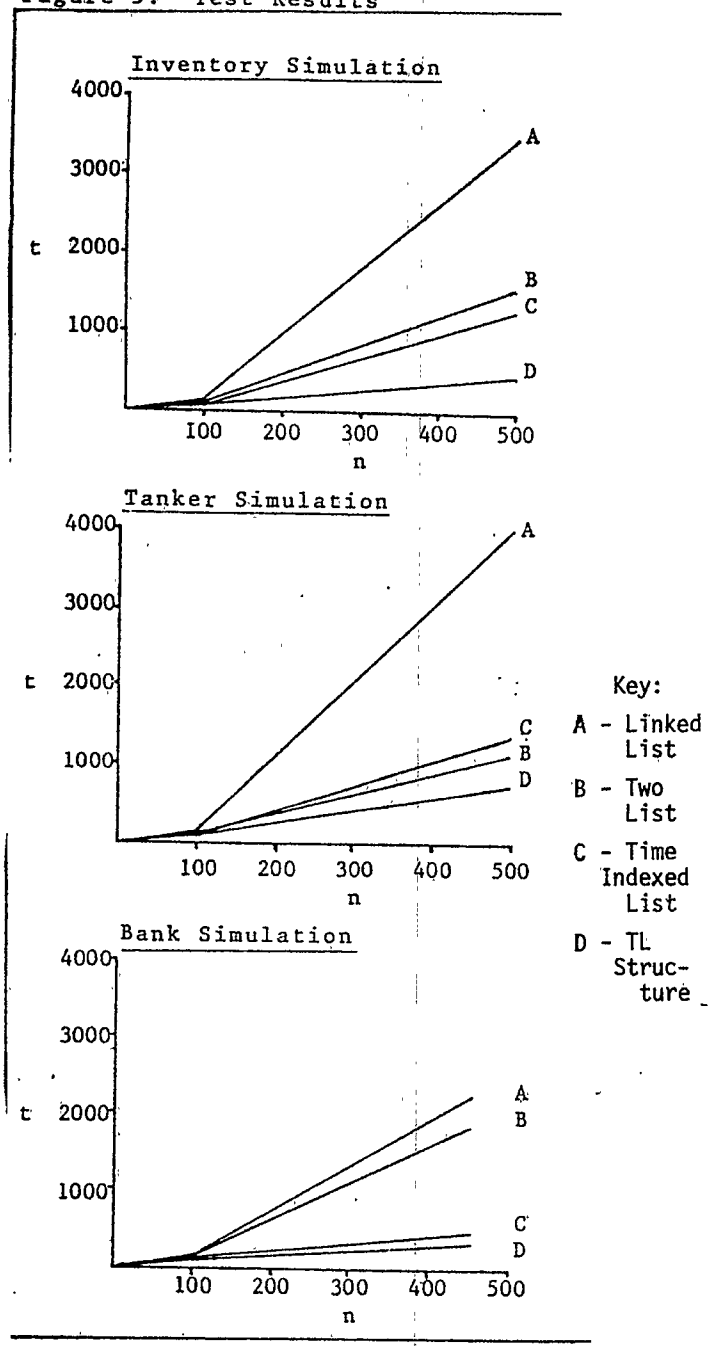
* Runs with shorter simulation times, last column contains corrected times.

Since the simulations with large event file size require very large execution times, the simulation run length for these cases was reduced. These execution times therefore had to be corrected in order to make comparisons to the other cases. These cases were run 1/10 of the run length and thus the execution time is multiplied by 10. This will result in some magnification of the experimental error. In the table, the corrected execution times appear as the last entries in each row.

It can be observed from the tabulated results that the three suggested structures performed vastly superior to the linked list structure. This observation is better illustrated by the graphs of execution time, t, (in seconds) versus event file length, n (Figure 5). These graphs distinguish clearly the differences in efficiency between the structures. The results suggest that the TL structure is far superior to all its counterparts. The time indexed list and Two List structures performed about the same, except in the drive-in bank (multi-

channel queue) simulation. This seems a little surprising, since the theoretical foundations tend to lean towards the Two List structure. After close examination, it appears that the drive-in bank simulation provided very good initial parameter estimates for the development of the time indexed list structure and the simulation also behaves in a rather uniform fashion, as the intervals in the time indexed list approach contained about the same number of entries. These characteristics permitted the time indexed list structure to perform nearly at its ideal for this example such that it approached the TL structure.

Figure 5: Test Results



4. CONCLUSION

Theoretical examination of the different filing structures examined in this study suggested that they would outperform the linked list structure currently utilized in most simulation systems, with the TL structure probably outperforming its competition. The experimental results for some realistic examples obtained in this study support the theoretical conclusions. The only apparent discrepancy between the theory and the experiments occurred when the time indexed list vastly outperformed the Two List structure for one case. Both of these structures should perform about the same according to the theory.

There is a need to incorporate filing structures that are more efficient than the linked list into simulation systems. This report comprises the first computational study of advanced event filing structures and provides strong evidence that these methods should be incorporated in general simulation programs. It is hoped that this paper will motivate others to consider incorporation of advanced filing structures. It seems apparent that substantial amounts of computer time and money can be saved through more widespread use of these mechanisms.

REFERENCES

1. Blackstone, J. H., G. L. Hogg, and D. T. Phillips, "A Two List Synchronization Procedure for Discrete Event Simulation," to appear in Comm. ACM 24:12, Dec. 1981.

2. Engelbrecht-Wiggens, R. and W. L. Maxwell, "Analysis of the Time Indexed List Procedure for Synchronization of Discrete Event Simulations," Manage. Sci. 24:13, Sept. 1978.

3. Franta, W. and K. Maly, "An Efficient Data Structure for the Simulation Event Set," Comm. ACM 20:8, Aug. 1977.

4. Franta, W. and K. Maly, "A Comparison of Heaps and the TL Structure for the Simulation Event Set," Comm. ACM 21:10, Oct. 1978.

5. Gonnet, G. H., "Heaps Applied to Event Driven Mechanisms," Comm. ACM 19:7, Jul. 1976.

6. Pritsker, A. A. B., The GASP IV Simulation Language, John Wiley & Sons, Inc., New York, 1974.

7. Vaucher, J. G. and P. Duval, "A Comparison of Simulation Event List Algorithms," Comm. ACM 18:4, Apr. 1975.

8. Wyman, F. P., "Improved Event Scanning Mechanisms for Discrete Event Simulation," Comm. ACM 18: 6, Jun. 1975.