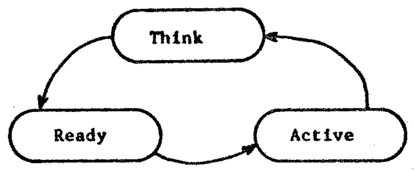
Figure 4.8 depicts the cycle which an interactive job traverses. Statistics concerning the length of time spent in each state are also included. It seems reasonable to assume that an improvement in the memory management would not change a job's think time and so that portion of a job's cycle can be assumed to remain constant. Improving the operation of the physical mechanism can reduce the time a job spends in each of the other two states.

A job begins the active state when it is attached to a control point. The job must then wait for the physical mechanism to allocate memory to the CP. It then executes until it finishes or is preempted. The active state terminates when the rollout process begins. During much of this state, the job requires the CPU to progress; however, it sometimes must wait to be allocated the CPU because the CPU is performing a storage move. If the physical mechanism performed fewer storage moves, jobs would not spend as much time in the active state. The maximum reduction in time in the active state would occur, if time waiting for CPU compaction of memory was eliminated. This is at most 6% to 8% of time in the state.

A job begins the ready state when the system monitor first notices that the job is requesting service. The job must then wait until the policy scheduler dictates that the physical mechanism should rollin the job. The ready state ends upon entering the active state. There is a time lag between the scheduler's designating the job for rollin and the initiation of the rollin. The physical mechanism must



TAPE	STAGE	RATE	MEAN(msec)
1	Active	3.555	895.557
	Think	1.790	44,905.173
	Ready	3.623	1,761.516
2	Active	2.933	893.650
	Think	1.963	20,464.905
	Ready	2.996	2,718.582
3	Active	3.374	795.416
	Think	2.270	27,729.164
	Ready	3.485	5,614.313

Figure 4.8

initiate the rollin and it cannot respond immediately. During this time, memory may be compacted so that desired storage increases can take place. Improving the physical mechanism can, therefore, reduce the time lag to initiation of the rollin. Statistics regarding the time spent in the stages of the ready state were not gathered. It will be assumed that improving the physical mechanism could reduce the time in all stages of the ready state, thus maximizing the possible reduction. This is at most 6% to 8% of the time in the state as this is the fraction of time which the CPU is compacting memory.

It has been shown that improving the physical mechanism can reduce a job's traversal time for the cycle shown in Figure 4.8. The time spent in both the ready and active states could be reduced by up to the amount of CPU time spent compacting memory. If a job is preempted, it will have a zero think time (effectively skip the state). Therefore, the number of occurences of the other two states is greater than the number of think states. By weighting the times spent in these two states, the time spent in each state per complete cycle is obtained and thus the total time per cycle. The reduction in cycle time due to improvement of the physical mechanism is obtained by reducing the time in the ready and active states by the fraction of time spent compacting memory. For all three tapes, the reduction in average cycle time was less than 2%.

The primary factor in determining the frequency with which the scheduler must run is the rate at which jobs complete the cycle

shown in Figure 4.8. It has been shown that the largest effect which a change in the physical mechanism can have on the rate at which a job completes this cycle is 2%. This reduction in time will be only marginally noticeable in its effect on a given job. However, the system executes many such job cycles. Over the long term, such an increase in throughput could represent a substantial increase in system performance. Thus it can be seen that changing the physical mechanism can not increase the coupling effect between policy scheduler and physical mechanism significantly, but should provide a substantial decrease in CP overhead and a corresponding increase in user CPU utilization when viewed over the long term.

On the basis of the results presented in this and the previous section it was decided to perform the simulation in the manner alluded to previously. The policy scheduling portion of the memory management function is implicitly performed by using the decisions of the job scheduler as recorded on the trace tape. The physical mechanism task is explicitly simulated, keeping track of the location and storage requirements of all jobs in memory, simulating the placement of new jobs according to various strategies and compacting memory by performing storage moves using the same algorithm as the system.

4.1.3 Necessary Events

The majority of the events recorded on the trace tape are unnecessary for the operation of the simulation models. Examples of some of unneeded information are allocation and deallocation of I/O channels and assignment of the CPU to different user control points.

The trace-driven simulator of this study, therefore, retains and utilizes only a limited set of the events which are recorded on the trace. Specifically, the events which are retained for operating of the simulation are those events which concern assignment and release of PPs to a control point and also the events which indicate the reply from MTR to a PP concerning a storage request.

The events detailing PP assignment and release are necessary only because of the way the information is presented on the event trace. The RST reply indicates which PPs request for storage has been completed. Thus, it is necessary to track the assignment of PPs to determine which control point the PP was attached to and therefore, which CP received the indicated new field length. The reply event is the event which indicates that a change in a CP's field length has been completed. This is the event to which the simulation is keyed as is explained in the following section.

4.2 Model Operation

The simulation model operates directly from the event trace tape recorded by the software monitor. The method used is very simple in concept. It is not necessary to retain any sequence of events for the purpose of determining some particular process. Instead, the structure is such that each event is viewed individually. If the event is one of those mentioned in the previous section as being

necessary for the simulation, then the required routines are called as dictated by the particular event. After completion of the simulation processing, the event is simply discarded. If the event is one of the unnecessary events it is discarded immediately.

The performance metric of primary interest to this study is overhead on the CPU due to the necessary periodic compaction of memory, also called "storage move overhead." There are only two conditions which can bring about the necessity to relocate a job in memory. These conditions are:

1) The rolling in of a new job at a time when the total of all available memory would be sufficient to satisfy the job's memory requirement, but the hole designated for the placement of the job is not of sufficient size.

2) The request of an already resident job to increase its memory requirements at a time when the hole immediately above the requesting job is of insufficient size to satisfy the request.

The detection of these two conditions implies the necessity of knowing the current memory configuration including the field length and location of each job in memory.

The software monitor dumps several of the system's tables onto the first portion of the tape before initiating the recording of events begins. One of these tables is the Control Point Status Table. This table maintains the field length and the location of every control point. If the events supply the necessary information to update the table as needed, it is possible to maintain a record of the configuration of memory throughout the simulation. The manner in which this is done is now presented.

Since no job can change field length (including the system jobs) without issuing a request for storage (RST), it is possible to maintain a completely faithful record of the amount of memory occupied by each CP by changing the CP's memory allocation when an RST is recorded to indicate the new field length for that CP. Thus, the process for correctly maintaining a record of CP field lengths is very easily implemented.

It is also necessary to maintain a record of each CP's location. This process is not as easy as that of maintaining the field length record. It is allowable to simply accept the changes in field length for control points, as this constitutes the earlier mentioned process of "accepting the decisions of the job scheduler as recorded on the trace tape." It is not, however, possible to simply accept and process the events which indicate that a job has been moved. These events represent the physical mechanism's implementation of the job scheduler's decisions, and this is the portion of the memory management function which is to be explicitly simulated. It is therefore, necessary that the simulation model somehow determine the location of the CPs without specific knowledge of what the system's physical mechanism has done. This involves simulating two of the system's algorithms:

1) the algorithm for placement of new jobs in memory, and
2) the algorithm for relocation of jobs resident in memory.

These algorithms are described in sections 3.2.3.2 and3.2.3.3 respectively. The determination that either of these algorithms must be executed by the simulation model is made when the occurence of an RST is noted on the event trace tape.

When an RST is read which indicates the new field length for a CP is less than the old field length, neither of the two algorithms need be executed. No change in location could possibly be called for by the occurrence of this event.

When an RST is read which indicates an increase in field length for a CP which is already present in central memory, the expansion algorithm of section 3.2.3.3 is executed. The presence of this event indicates that the system has successfully completed the requested increase for the job. Since the simulation model is maintaining all CPs at the same field length as the system, this implies that the simulator can also satisfy the requested storage increase. By executing the same algorithm as the system, the simulator determines and duplicates any "storage moves" which might have been necessary to complete the RST. In this way, the simulation model determines any CP relocation which must have taken place to accomplish the RST, without actual reference to the recorded events indicating the "storage moves."

When an RST which is an increase in field length of a CP from zero is read, this indicates a rollin. The placement algorithm is executed to determine a location for the job and the expansion algorithm is then called as described in section 3.2.3.2. To simulate different algorithms for placing jobs in memory, it is only necessary to change the routine which chooses a location for a job rolling in. From this point the processing corresponds to that for jobs which were already resident in memory and requested a storage increase.

Thus, it can be seen that the entire operation of the simulation model is keyed to the RST events. These are the replies from MTR to the PPs indicating the new field lengths for the associated CPs. The simulator operation consists of maintaining all the CPs at the same field length as the system and determining the location of each CP based on the known interactions between CPs caused by RSTs. Upon reading an event which indicates that the system has completed some change in the memory configuration, the simulation model determines what the system must have done (or would have done in the case of experiments) in the way of relocating jobs to have successfully completed the changes. Any changes necessary in the locations of CPs are made and, based on the field lengths of the CPs moved, the time necessary to make those moves is determined and recorded.

4.3 Model Validation

Before the results of any experimental study based upon a trace-driven simulation can be used for simulation experiments, it is

necessary to determine that the simulation model does in fact accurately emulate the function being modeled. The importance of validation is stressed in Brice (6), Sherman (4) and Anderson (7). This chapter presents the explanations necessary to show that the behavior of the simulation model is well defined, well understood and a good representation of the memory management function.

The trace-driven simulator of this study retains and utilizes only a limited set of the events available, as delineated in section 4.1.3. Thus, a completely faithful reproduction of the system's activities is not to be expected. The validation of the simulation must, however, account for the deviations of the simulation from the actual system measurements for the performance metrics of interest.

4.3.1 Design Consequences

The simulation method of processing RSTs individually and independently yields a very close approximation to system behavior, but it does in fact differ from true system behavior in some respects. Of course, the correct field lengths for all CPs are maintained, but the lcoations determined occasionally differ from the locations of the same CPs as recorded for the actual system. The events described in section 4.1.3 are all the input used for the operation of the simulation model. This data is incomplete, with respect to the domain of memory system modeling, in two areas:

1) The system often overlaps the execution of a number of functions, i.e., an RST to a larger field length may overlap one or

even more than one RST to a smaller field length and a 1SJ may overlap an RST to a larger field length. This is not fundamental but is a property of the system being modeled.

2) Before a control point can be relocated, it must be idle as described in section 3.2.3.4. For a CP to be idle it must either have no PPs attached or all PPs must have paused. The event which indicates that a PP has paused is a PFR (Pause for Relocation). The information on the trace tape records those times at which a PP did pause because of existing conditions; however, information concerning when a PP would have paused, had conditions been different, is not available.

It is the policy of the UT-2D memory manager to process only a single "RST up" at a time. This is not done instantaneously and, in fact, the system frequently overlaps the processing of more than one "RST down" while waiting to move jobs so that the pending "RST up" may be allocated. While overlapping these processes, the processing of the "RST up" may require the movement of a CP which, before reaching an idle stage, issues an "RST down" to zero. This "RST down" makes it unnecessary to move the CP, but the system is unable to cancel its past decision and so the storage move will be performed anyway. This inability of the system to cancel past decisions is a primary cause of the discrepancies between system and simulation behavior. Note also, that were it not for the overlapping of processes, the need to cancel decisions would not arise.

The information recorded on the event trace tapes includes both the request by a PP to change the memory allocation of a CP, and the reply from MTR to the PP of what action was taken. Both of these events were available to the simulation model, but the decision was made to use only the reply events for two reasons.

1) The request indicates the new field length desired for the requesting CP. It is not uncommon that the system is unable to satisfy a CP's request. One example of this is what is referred to as a "punt." There is a time lapse between the scheduler's decision to rollin a job and the beginning of the processing of the "RST up" for the rollin. It is possible for a CP in memory to request and receive a storage increase during this time lapse. If this should occur, the "RST up" for the rollin might not be possible, since the amount of memory available will have decreased since the scheduler decision. The cancelling of the attempted rollin is referred to as a punt. The request for memory will appear on the trace tape, but will not be satisfied, therefore, to assume that every request was satisfied, and to simulate the granting of every request is not possible. On this basis it can be seen that simulating by using only the requests was infeasible.

2) The length of time passing between the request for memory and the reply indicating the amount of memory actually allocated can vary considerably. This is especially true in those cases where storage moves are performed to accomplish the "RST up." The possibility exists of using both the request and the reply events in the

the simulation model. The purpose of this is to better indicate the extent of the overlapping of processing RSTs and better represent the effect where a job wishes to rollout but is forced to "idle down" and be moved first. Using the recorded events indicating the pausing of PPs to determine when a CP is idle, actually represents a better method of simulating the system as it currently performs. However, it is here that the second point, concerning the recording of PFRs, is of importance. This study wishes to compare several different simulations to each other and to the existing system, all on an equal basis. Since different placement algorithms would cause different sets of CPs to be moved, in different order from the system, the necessity of PPs "idling down" would occur at completely different times from the current system. The information concerning when the related PPs would pause is not available and therefore, it would be impossible to simulate new algorithms in the same manner. Since simulation for all algorithms in a comparable manner was not possible using both request and reply events, this method was also rejected.

The method of simulation employed uses only the reply events. This method also has its shortcomings. The simulations which result are somewhat optimistic because of the added information which results. Stated briefly, the simulation model effectively has information concerning the future which the system did not have at the time it began processing the "RST up." The main effect of this is that the simulator can avoid starting a move sequence which will not have to be completed. An explanation of how the system will

sometimes find this action necessary is given in section 3.2.3.4. Since this was done because of system characteristics and not for some fundamental property of all systems, it would seem that eliminating direct references to the effect provides a model which is more general. In any case, using the reply events only, provides a method for a normalized comparison between algorithms.

The remainder of this chapter presents a qualitative description of the deviations of the simulation from system behavior, a discussion of the method used to detect the occurence of the deviations and a quantitative accounting of the deviations.

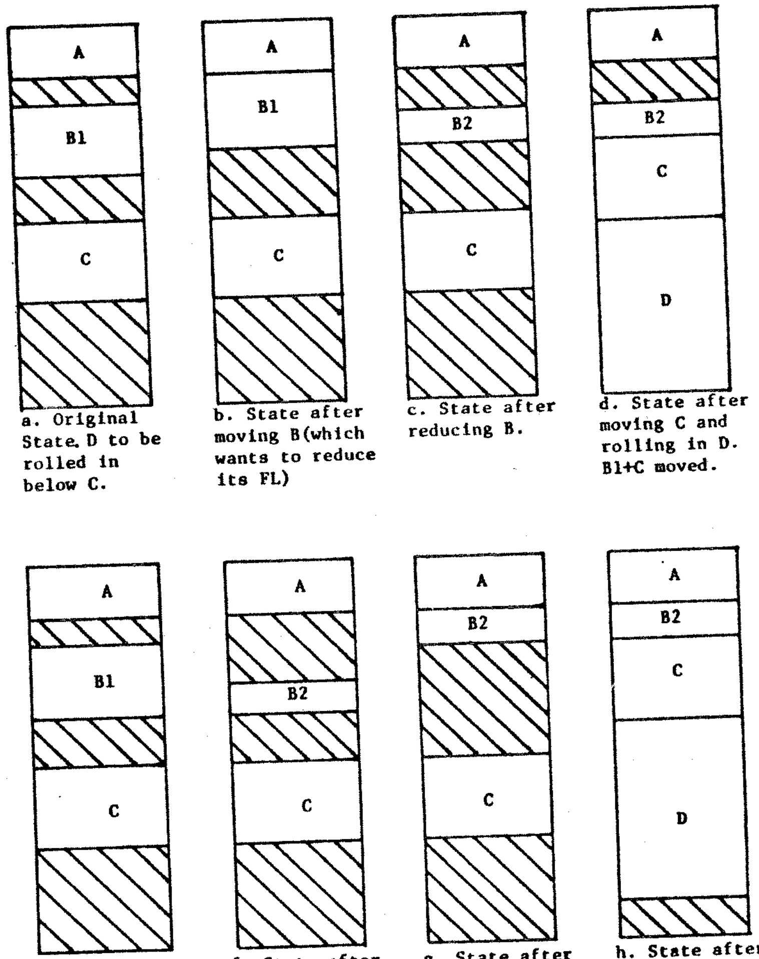
4.3.2 Qualitative Description of Deviations

The validation process consists of running the simulation model in the same manner in which the experiments will be run. The same algorithm for placement and expansion is used by the simulation model as is used by the system to accomplish the validation. The behavior of the simulator and system are systematically compared to detect any deviations which occur and to determine the exact nature of all deviations.

This section presents a detailed description, including examples, of each deviation from system behavior which the system exhibits. A brief summary of the conditions which can cause a discrepancy and a recap of the design decisions which lead to these conditions is first presented.

If the simulator and system are in the same state at the completion of the processing of one RST, then they should be in the same state after processing the next RST. In fact, this is the case if the second RST is for a storage decrease. The difficulty which arises is due to the fact that the simulator processes all RSTs one at a time as they are finished by the system, whether they are increases or decreases. This gives the appearance of instant processing by the simulator. The system, as was previously stated, processes one "RST up" at a time. This is not done instantaneously and in fact, as described earlier, the system processes field length reductions while still waiting to move jobs so that the pending "RST up" may be allocated. It is this policy of the system to process more than one RST at a time which causes the deviations of the simulator from system behavior. This policy is not necessarily better and occasionally is decidedly inferior. The following subsections will describe the different deviations which occur.

The first deviation is directly attributable to the exception stated in section 3.2.3.4 for immediate processing of an "RST down." The system, in processing an "RST up," might mark a job to be moved. Should the marked job request a reduction in memory, it will first be moved and then reduced (Figure 4.9 A-D). This reduction will quite possibly occur before the satisfaction of the "RST up." The resultant effect to the simulator is the reduction of the CP before it has received any knowledge of the pending "RST up." This yields the obvious deviation that the memory which was taken from the



a. Original State. D to be rolled in below C.

b. State after moving B (which wants to reduce its FL)

c. State after reducing B.

d. State after moving C and rolling in D. B1+C moved.

e. Original State. D to be rolled in below C.

f. State after reducing B.

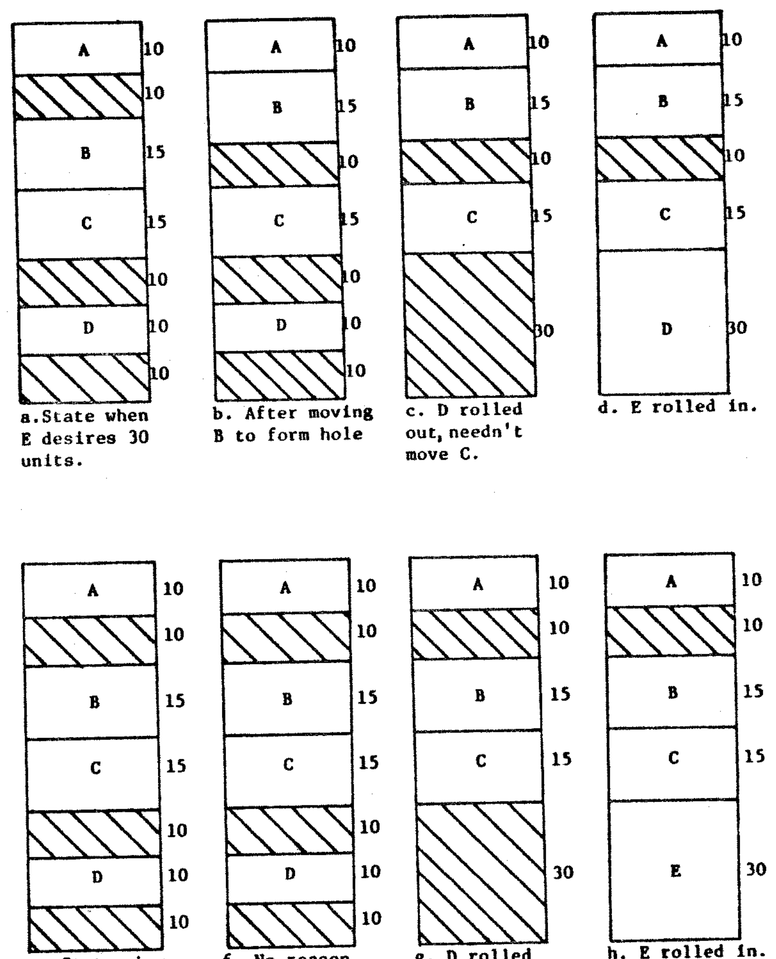
g. State after moving B.

h. State after moving C and rolling in D. B2+C moved.

Figure 4.9

reduced CP is no longer available to be moved by the simulator upon its receipt of the completed "RST up." When the simulator looks back now, to see what the system must have done to accomplish the "RST up," it is in a different state than the system was when the system first began to process the "RST up" (Figure 4.9 E-H). When it now simulates the system it will do less work simply because one CP has reduced and there are fewer occupied words in memory which it may choose to move. Note that the job whose CP was reduced is possibly gone completely since the reduction which took place could have been a rollout. (Detailed trace indicates that in fact this is the predominant case.) In any case, a substantial portion of "lost work" is due to this one discrepancy.

A second discrepancy also results from the processing of an "RST down" while an "RST up" is pending. As the algorithm shows, rather than deciding on a complete sequence of moves to take place for an "RST up," only the next move is determined. Should no requests for memory reductions occur before the desired hole is formed, the result of this one at a time method would be the same as if a full sequence were determined. However, by making decisions individually it becomes possible that the remainder of the sequence be made unnecessary by the opportune arrival of an appropriate "RST down" (Figure 4.10 A-D). Since the simulator only processes storage requests one at a time, it gains an advantage over the system. Upon the arrival of this opportune "RST down," which allowed the system to



a. State when E desires 30 units.
b. After moving B to form hole
c. D rolled out, needn't move C.
d. E rolled in.
e. State when E desires 30 units.
f. No reason to move B.
g. D rolled out.
h. E rolled in. B unmoved.

Figure 4.10

truncate the sequence of moves, the simulator will not yet have begun the sequence. Thus, when notified of the completed "RST up," since its state is once again different from that which the system had, the simulator will not realize that the system had started the sequence and will not move that portion of the sequence which took place before the "RST down" (Figure 4.10 E-H). This "lost work" is different from the first discrepancy, though, in that it would better be called "deferred work." There are actually three ways in which this "deferred work" may be disposed of in the simulator.

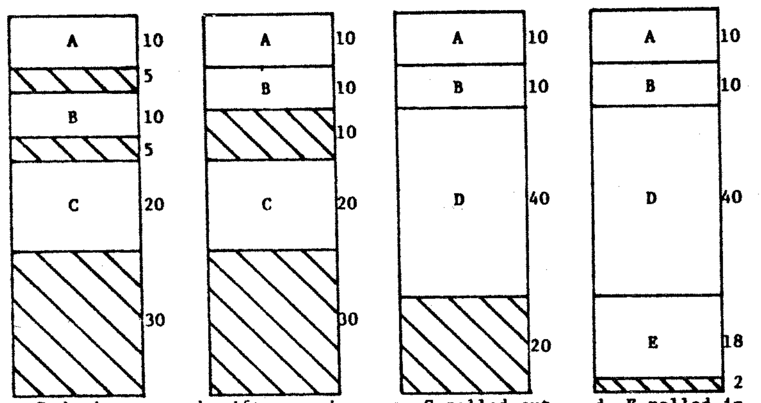
The first way to dispose of this work is to actually recover it. The "deferred work" is so called because it consists of a CP which the system has moved and the simulator has not. The CP is still occupying the same amount of memory as when it was moved by the system. The effect may simply be that the system has predone some work, which will later be done by the simulator. It is entirely possible that the system will be able to satisfy a later "RST up" without moving the CP while the simulator will now find it necessary to move the CP and thus, recover this "deferred work." If a new CP were to request a rollin of size 10 in Figure 4.10, the system would have to move only CPs "C" and "E". The simulator, before moving CPs "C" and "E", would have to move CP "B", thus recovering the "deferred work."

The second manner of disposing of this work is that the work may be lost in a manner similar to the first deviation. If the CP now reduces its memory requirement, before the simulator finds it

necessary to move the job, then that work is "lost." The only difference between this loss and the one first described is the presence of an "RST up" between the system's move of the CP and the "RST down" of the moved CP. Referring again to Figure 4.10, if CP "B" were to reduce its storage, the work which the system did in moving that CP is lost in a manner exactly analogous to the "lost work" in Figure 4.9.

The third way to dispose of this "deferred work" is for the system to move the CP another time before the simulator moves it the first time. Just as the system might move a CP to satisfy an "RST up" while the simulator does not, it is possible that the system could move the same CP again, to satisfy another "RST up," before the simulator moves the CP. Even if the simulator should now recover this move, the first move is lost. The system has moved the CP twice and the simulator has moved the CP only once. Should the CP rollout before the second "RST up" is satisfied, both moves would be lost to the simulator.

It is also possible for the simulator to do work which the system will find unnecessary. If there is some "deferred work" such that the system is in the state shown in Figure 4.11C and the simulator is in the state shown in Figure 4.11G, then an "RST up" for a new CP desiring more than 15 units of memory would require the simulator to move CPs "B" and "D". The movement of CP "B" is another example of simulator recovery of "deferred work." The movement of

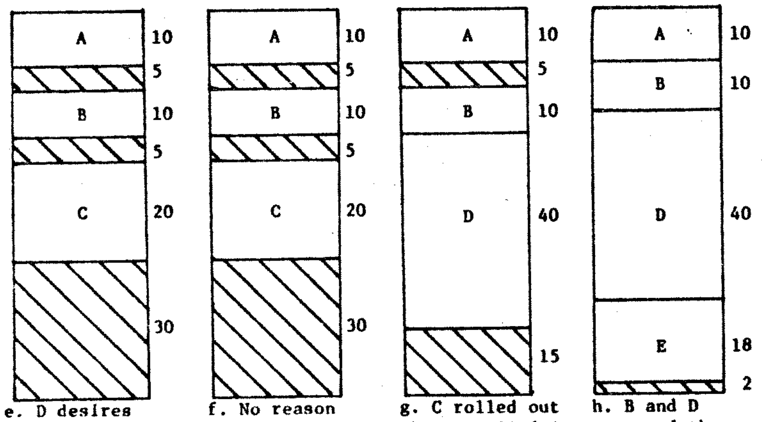


a. D desires 40 units.

b. After moving B to form hole.

c. C rolled out then D rolled in.

d. E rolled in.



e. D desires 40 units.

f. No reason to move B.

g. C rolled out then D rolled in.

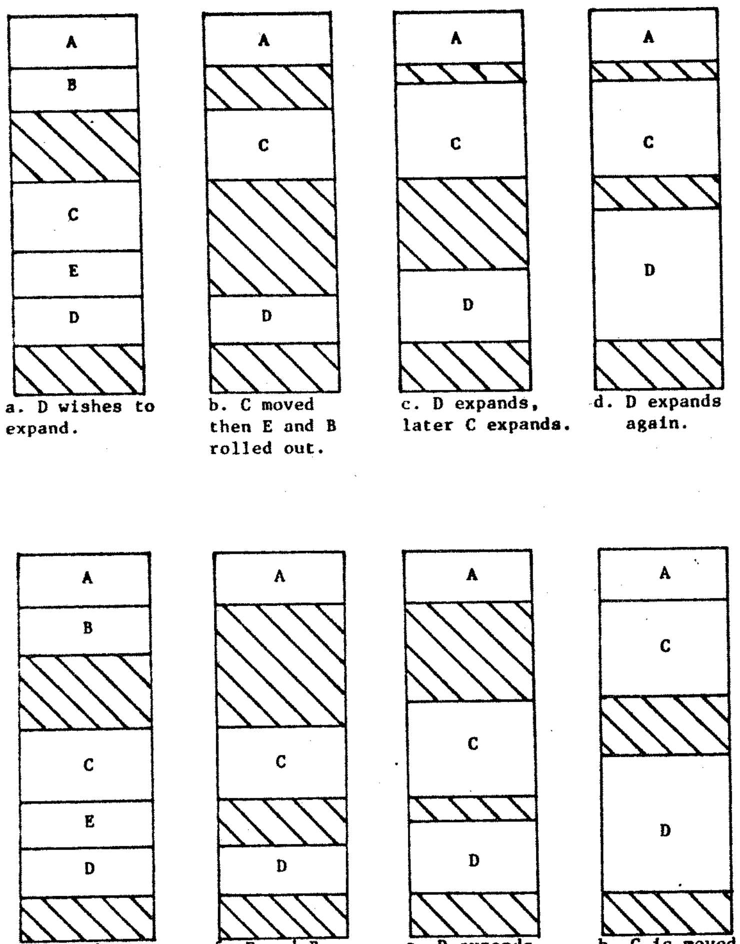
h. B and D moved then E rolled in.

Figure 4.11

CP "D", however, is extra work for the simulator. Furthermore, there can be no "recovery" of this work by the system, since both simulator and system are in the same state as shown in Figure 4.11.

Another way in which the simulator can perform more work than the system is to over-recover some "deferred work." This can occur if a CP which has been moved by the system, but not by the simulator, should complete an "RST up" and then be moved by the simulation model. The likelihood of this happening is extremely small, but nonetheless,occurrences were detected. Figure 4.12 shows one possible sequence which could result in an over-recovery by the simulator.

One final deviation was detected which bears special mention. This deviation occurs due to the method of operation of the software event recorder. Its effect on the statistics is negligible since it occurs so infrequently and the existence of this deviation is noted only to complete the discussion of all discrepancies which exist. The occurrence of this event appears to the simulator as the movement of the same CP more than once during the process of satisfying a single "RST up." The error is due to the hierarachical manner in which CPM satisfies its duties. When the event recorder of UT-2D is activated, all recording of events is performed by CPM. If the following sequence of events occurs, the two events indicated to be recorded will be recorded on a single run of CPM, in reverse order:



a. D wishes to expand.
b. C moved then E and B rolled out.
c. D expands, later C expands.
d. D expands again.
e. D wishes to expand.
f. E and B rolled out.
g. D expands, later C expands.
h. C is moved and over-recovered so D can expand.

Figure 4.12

1) MTR replies to a CP that the requested "RST up" is completed and leaves an event indicating this, to be recorded by CPM.

2) MTR begins processing a new "RST up"; determines that a CP needs relocating; determines that the CP is idle and can be moved and, therefore, calls CPM to move the CP.

3) CPM moves the CP, as requested by MTR, and records an event indicating that the storage move has taken place; continuing its loop, it finds and records the event from MTR indicating the completed "RST up."

Note that the events were recorded in reverse order from the order in which they occurred. Furthermore, if the CP which was moved to satisfy the new "RST up" was also moved to satisfy the old "RST up," it will appear to be moved twice during the process of satisfying the old "RST up." Since it will be moved twice before the "RST up" informs the simulator of the completion, the first move will be "lost work" in the same manner as the "deferred work" which becomes "lost work" because it is moved a second time.

This is all of the deviations which exist between the system and the simulation model. The following section describes the method whereby each of the deviations was detected and the section after that presents quantitative results indicating the magnitude of each deviation was well as the totals necessary showing that all deviations were detected.

4.3.3 Detection of Deviations

Having described each deviation which occurs between system and simulation model behavior, this section describes the mechanism implemented in the program, by which each discrepancy is detected.

System behavior was determined by changing the field lengths of each CP as indicated by the RST reply events from MTR to the PP, informing the PP of the new field length of the CP. CPs which were indicated to have increased from zero field length, were placed in memory according to the known system algorithm. There is no event on the tape which indicates the location of a CP, so rollins were effectively simulated for the system. There is, however, no doubt that the jobs were placed correctly since the correct system state was maintained at all times.

Relocation of jobs was performed in the system only when indicated by the reading of a storage move event. The storage move event records which CP is moved, how far it is moved and in what direction. Note that the event does not record the new location explicitly; the change in location is recorded instead. This is what makes it necessary to simulate rollins for the system. Using both RST events and storage move events provides complete knowledge of all changes in the system's memory configuration.

For each control point, two flags are maintained. Each time a storage move is performed on a CP, the first flag is set. When an "RST up" is read, the simulation model executes the necessary

routines, as explained in the previous section, and for any CPs which the simulator moves a flag is set by the simulator. At this time, comparisons are made of the flags set in the system and simulation model, respectively. If the system and simulator have performed identically, all flags will match. As pairs of matching flags are determined, both are cleared. The existence of "leftover" flags indicates either the existence of the "deferred work" condition described earlier or it may signal the actual occurrence of a deviation. The method by which each deviation is detected is now presented.

Detection of the first deviation described is straightforward and simple. When the CP is moved in the system, the first flag is set and the field length of the job is recorded as being recoverable. If an "RST down" for the CP is now detected, and the flag is still set, then an occurrence of the first deviation has been detected. The difference between the field length moved and the new field length is recorded as lost. If the new field length is zero, the flag is also cleared. If the new field length is non-zero, it is recorded as the amount of work which is now recoverable.

If, after comparing flags between the system and the simulation model, there are flags remaining for the system, these indicate a "deferred work" condition. This is recorded by setting the second flag for the CP and clearing the first flag. Methods of disposing of the "deferred work" are detected by examining the second flag.

When moving a CP in the simulation, if the first flag is not set for the system, but the second flag is, the simulator has recovered the "deferred work." The second method of disposing of the "deferred work," losing it, is detected when an "RST down" is performed while the second flag is set. This is recorded as "lost work." If the "RST down" is a rollout, the flag is cleared. If the system moves a CP while the second flag is set, the first move is lost and is recorded as such. The second flag is cleared and the first flag is set, therefore, allowing the possibility of the simulator also performing the move.

The deviation where the simulation model performs work which is unnecessary to the system is also detected using the flags. If the simulation model's flag is set for a CP, but neither system flag is set for the same CP, the simulator has moved a job which the system has not. This is recorded and the simulator flag is cleared.

The over-recovery by the simulator of "deferred work" is a special case of the earlier described recovery. When a CP is moved by the system, the field length recoverable is recorded. Whem comparing flags to detect the recovery, the field length being moved by the simulation model is also compared to that which was recorded as recoverable. If the simulator over-recovers, the extra field length moved is recorded.

The special error caused by the system's mis-recording of events is detected solely by use of the first of the system flags.

When a storage move is performed, if the first flag is already set, then the CP has already been moved since the last "RST up" and this can only be caused by the event's being recorded out of sequence.

All deviations from system behavior, by the simulator are detected by the use of the flags. As an occurrence of each deviation is detected, the magnitude of the discrepancy is accumulated. If all of the difference between the sum of all words moved by the system, and all words moved by the simulator are accounted for in the accumulated individual deviations, then the correctness of the simulation model's behavior is established.

4.3.4 Quantitative Accounting of Deviations

This section presents the quantitative accounting of the deviations described in section 4.3.2. The numbers reported are those gathered by the detection mechanism of section 4.3.3. Memory is allocated in blocks of 100_8. All values reported in this section are in blocks of 100_8.

During the course of the 13 minute and 58 second (13:58) interval recorded on Tape 1, a total of 274,133 blocks of memory were moved by the physical mechanism of the system. Of these, 60,512 blocks were moved by the system without the simulator "simultaneously" following suit. In simulating the same workload, the simulator moved a total of 238,799 blocks of memory of which 25,178 blocks were moved without "simultaneous" corroboration from the system. Note that the difference in the total number of words moved is the same as the difference between the number of uncorroborated words. Therefore, if those words moved by the system but not by the simulator, and vice versa, are accounted for, the total discrepancy between the system measurements and the simulation model are accounted for. This proves that the relation between system and simulator is well defined and that all deviations in behavior are detected.

The number of blocks lost to the first deviation described in section 4.3.2 is 24,914. Of these, 24,774 were lost to rollouts. This means that 70% of the total discrepancy between system and simulator is due to jobs which are moved and then reduced (usually rolled out) before the simulator is notified of the "RST up" which caused the move.

The second discrepancy described is the recovery of "deferred work." The system moves 9,186 blocks which are later recovered by the simulator. The system reduces jobs a total of 17,622 blocks after the simulator has deferred this work. The system moves these jobs but the simulator does not find it necessary to do so at the same time. Before the simulator finds it necessary to move these jobs and recover this "deferred work," the system reduces the job and the work is lost. The second way in which "deferred work" becomes "lost work" is for the system to move the job a second time before the simulator moves it once. The number of blocks for which this occurred is 8,462.

The last discrepancy for which the system is credited with performing more work than the simulator is due to the recording error described in section 4.3.2. The "lost work" due to recording errors is 328 blocks.

These were all detected discrepancies which resulted in extra work being performed by the system. They are summarized in Table 4.6. They sum to 60,512 blocks, the total number of blocks which were moved by the system without the simulator "simultaneously" following.

The simulator, as earlier described, recovers 9,186 blocks of memory which it had "deferred." It moves 15,674 blocks which the system did not move, as described in section 4.3.2. Finally, the simulator "over-recovers" 318 blocks of memory. These discrepancies are also summarized in Table 4.6. The total of these discrepancies which result in extra simulator work is 25,178. This is the total number of blocks which the simulator moved without "simultaneous" corroboration from the system.

Thus, the total discrepancy between the total number of words moved by the system and the total number of words moved by the simulator is accounted for.

The above analysis has used the values obtained from Tape 1. A similar analysis can be performed for Tapes 2 and 3. The values of the discrepancies for the tapes are also shown in Table 4.6.

TAPE	1	2	3
MOVED BY SYSTEM	274,133	432,868	280,996
MOVED BY SIMULATOR	238,799	416,472	264,383
TOTAL DIFFERENCE	35,334	16,396	16,613
LOST WORK	24,914	13,804	12,596
DEFERRED...RECOVERED	9,186	10,780	8,320
DEFERRED...LOST	17,622	8,423	9,445
DEFERRED...REDEFERRED	8,462	3,833	4,959
MIS-RECORDED	328	168	272
TOTAL SYSTEM EXCESS	60,512	37,008	35,592
DEFERRED...RECOVERED	9,186	10,780	8,320
EXTRA WORK	15,674	9,824	10,589
OVER RECOVERED	318	8	70
TOTAL SIMULATOR EXCESS	25,178	20,612	18,979
DIFFERENCE OF EXCESS	35,334	16,396	16,613

Table 4.6

CHAPTER 5

EXPERIMENTAL STUDIES

The goal of this study is an analysis of different algorithms for choosing the location in main memory to which incoming jobs are assigned. Several measurements are made so that a quantitative comparison of the different algorithms' effects can be made.

5.1 The Algorithms

Two algorithms for placement of jobs in memory were simulated, first-fit and best-fit. In first-fit allocation, available blocks of storage are examined in the order of their starting addresses. The job being allocated is placed in the first hole encountered which is sufficiently large to accept it. In best-fit allocation, the job is allocated the smallest available block in which it will fit.

One further condition must be specified, resolution of the overflow condition when no single block is sufficiently large to accomodate the new job. No previous work has explicitly allowed for reclaiming of holes by compaction. Overflow has been treated as a halt condition which ends whenever a job adjacent to a hole releases memory (11, 13, 14, 17, 18). Through the use of compaction, overflow is overcome by relocation of certain jobs in memory so that a hole of the desired size (or larger) is created. This is done by designating an already existing hole as the hole which is to be expanded for allocation to the new job. Occasionally the hole may be of size 0.

It is possible to determine which of the holes would require the fewest words to be moved to expand the hole to sufficient size. This would require checking every hole in memory together with an algorithm for determining which hole(s) should be coalesced. This algorithm would be optimal in a local sense. It would move the fewest words to resolve this overflow. There is no guarantee that it would move the fewest number of words over the length of the simulation. Furthermore, it would be quite complex and probably too slow to actually implement on a running system. It is desired that instead some very fast and simple algorithm be chosen.

Two of these "simple" strategies come to mind for resolving the situation where no hole is of sufficient size: 1) Designate the "low hole" (the hole immediately above SATURN) as the hole to be expanded. This will tend to cause jobs which are resident in memory the longest to rise to the top as is the case in the current system algorithm for job placement. 2) Designate the "largest hole" as the hole to be expanded. This will hopefully require the fewest number of holes to be coalesced into a single hole.

Therefore, there are, in effect, four different placement strategies to be tested: 1) First-fit-1 (FF1), first-fit with overflow resolved by designating the "low hole" to be expanded. 2) First-fit-2 (FF2), first-fit with overflow resolved by designating the "largest hole" to be expanded. 3) Best-fit-1 (BF1), best-fit with overflow resolved by designating the "low hole" to be expanded. 4) Best-fit-2 (BF2), best-fit with overflow resolved by designating the "largest hole" as the hole to be expanded.

5.2 The Performance Metrics

The following metrics are gathered for use in the study of compaction overhead.

1) Number of jobs moved. While performing the expansion algorithm, the number of jobs which it was necessary to move is tabulated. Since a job must stop all processing before it can be moved, it is desirable to move as few jobs as possible so as to reduce the amount of time which other resources are held idle by the job being moved.

2) Field Length moved. The total of the field lengths of all jobs moved is accumulated. It is desired that the amount of compaction overhead be reduced. If fewer words are moved because one large job is not moved or because several smaller jobs are not moved, the CP overhead is reduced simply because the total number of words moved is less.

3) CPU time spent moving jobs. This metric is essentially the same as the second. However, since there is some small initial overhead involved in starting up the move package, the movement of J jobs of size M/J will take slightly longer than moving one job of size M. This metric is included also so that comparisons on the

basis of fraction of CP time moving jobs can be compared with system measurements of the same metric.

These three metrics are presented on a "per storage increase" basis. The expansion algorithm of section 3.2.3.3 is called for every storage increase. The desired metrics are accumulated while performing the expansion algorithm.

A fourth metric of performance, the number of times a job is moved per memory residence, is also tabulated. The current UT-2D placement algorithm was partially chosen because it included one desirable effect. Jobs which remain resident in memory for long periods of time will tend to be pushed to the top of memory, where they can hopefully remain without being moved. This is a desirable aspect in this system as it should reduce lost resource time due to these longer resident jobs being forced to hold resources while they are idling down to be moved. This metric is therefore, also included so that the impact of the new algorithms may be measured.

The reduction of CP overhead in the performance of storage moves is the major study of this project. The four measures indicated above are used as the basis for comparison of the placement algorithms in an effort to judge which is better from the point of view of minimizing compaction overhead.

A second basis for comparison of system performance is memory utilization. The manner in which memory utilization is studied is presented in section 5.4. The data which are used in that study will be detailed here.

The two data sets necessary to obtain estimates of memory utilization for different schedulers are the field length distribution of ready jobs and the distribution of hole sizes in memory.

An approximation to the field length distribution of ready jobs is given in Chapter 4. This was gathered in a manner which allowed an easily presented distribution for purposes of indicating the scheduler's effect on the distribution of jobs which run. For studying memory utilization, a truer expression of the field length distribution of ready jobs is needed. This is done by polling the Job Status Table (JST). When an event occurs which indicates a change in the JST, the JST is scanned to determine how many jobs of each field length are ready (but not allocated memory), and how long it has been since the last change in the JST was recorded. This time interval is added to an accumulator for each state, indicating that K jobs of field length range L are present. At the end of the run, these accumulators are divided by the total length in seconds of the run and a distribution is obtained for the probability of K jobs of range L being ready. Although this distribution still shows some impact due to the system scheduler, this impact is minimized. The distribution obtained is a complete ready job probability distribution for the steady-state probability of K jobs of field length range L being present in memory for all values of K and L which occur.

The second data set used in the study of memory utilization is a distribution of the hole sizes generated by the simulation model running with the different placement algorithms. The distribution obtained is the steady-state probability distribution for K holes in the range L being present in memory. This data is recorded after every change in the field length for a control point. An example of each of these distributions is given in the appendix.

Results of the study of placement algorithm effects on CP compaction overhead are presented separately from results of the memory utilization study for ease of exposition.

5.3 Compaction Overhead Results

Only a very limited amount of information was available at the time this study was begun, by which the compaction overhead could be characterized. Statistics indicating mean storage move time, maximum time and variance were the only available figures. Furthermore, these were gathered by considering each move independently and not on a per "RST up" basis.

Five sets of data for each metric and for each of three tapes were gathered. Data was obtained by simulating the current system placement strategy and each of the four proposed strategies discussed in section 5.1.

The chief difference between tapes is in the load placed on the system. Tape 1 is a moderately low workload. The mean number of jobs ready for, but not in central memory is 3.0. Tape 2 is a slightly heavier workload. The mean number of ready jobs is 7.3. Tape 3 is an exceptionally heavy workload. The mean number of ready jobs is 17.3 and there were about 80 users on line.

Although the demands placed on the system vary from tape to tape, the results of the experiments were very consistent from tape to tape. Therefore, in the following sections, only a representative sample of figures is included.

In the histograms presented in the following sections, the FROM-TO columns indicate the range of values recorded in that column of the histogram. They indicate the range FROM $\leq$ value < TO. For those metrics where the values are integers, only the correct value is shown.

5.3.1 Jobs Moved per Storage Increase

The figures presented in this section were obtained from Tape 2.

5.3.1.1 System Algorithm

Figure 5.1A shows a histogram of the number of jobs moved per storage increase when the system algorithm is simulated.

The first figure of interest was the fraction of jobs which completed storage increases without causing a storage move. Over 64% of the increases were possible with no compaction overhead. When rolling a job in, the system algorithm occasionally compacts memory even though a hole of sufficient size exists. This is because it requires the particular hole above SATURN as explained in section 3.2.3.2. The fact that over 64% of storage increases, the vast majority of these being rollins, were accomplished without compacting

```
16.24.48. 06 AUG 75   UNIVERSITY OF TEXAS 66/6400   UT 2D
NUMBER OF JOBS MOVED. FOR EACH ROLLIN OR EXPANSION. BY SIMULATED MEMORY MANAGER
  FROM        TALLY   SCALING:  1 PRINT COLUMN = 0.0100

    0          3389   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    1          1053   XXXXXXXXXXXXXXXXXXXX
    2           496   XXXXXXXXX
    3           192   XXX
    4            67   X
    5            24
    6             4

NUMB =     5225     NUMANZ =     1836
MEAN =    0.580     NZMEAN =    1.651
K**2 =    2.738     NZK**2 =    0.313
MAX  =        6     MAXTIN =      377
```

Figure 5.1A

```
16.24.48. 06 AUG 75   UNIVERSITY OF TEXAS 66/6400   UT 2D
NUMBER OF JOBS MOVED. FOR EACH ROLLIN OR EXPANSION. BY SIMULATED MEMORY MANAGER
  FROM        TALLY   SCALING:  1 PRINT COLUMN = 0.020

    0          4336   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    1           473   XXXX
    2           237   XX
    3           110   X
    4            52
    5            15
    6             2

NUMB =     5225     NUMANZ =      889
MEAN =    0.300     NZMEAN =    1.768
K**2 =    6.803     NZK**2 =    0.327
MAX  =        6     MAXTIN =      381
```

Figure 5.1B

```
16.24.48. 06 AUG 75   UNIVERSITY OF TEXAS 66/6400   UT 2D
NUMBER OF JOBS MOVED. FOR EACH ROLLIN OR EXPANSION. BY SIMULATED MEMORY MANAGER
  FROM        TALLY   SCALING:  1 PRINT COLUMN = 0.020

    0          4364   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    1           475   XXXX
    2           248   XX
    3            93
    4            33
    5             8
    6             4

NUMB =     5225     NUMANZ =      861
MEAN =    0.276     NZMEAN =    1.679
K**2 =    6.937     NZK**2 =    0.308
MAX  =        6     MAXTIN =      525
```

Figure 5.1C

```
16.24.48. 06 AUG 75   UNIVERSITY OF TEXAS 66/6400   UT 2D
NUMBER OF JOBS MOVED. FOR EACH ROLLIN OR EXPANSION. BY SIMULATED MEMORY MANAGER
  FROM        TALLY   SCALING:  1 PRINT COLUMN = 0.020

    0          4367   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    1           464   XXXX
    2           228   XX
    3           102
    4            44
    5            18
    6             2

NUMB =     5225     NUMANZ =      858
MEAN =    0.287     NZMEAN =    1.752
K**2 =    7.132     NZK**2 =    0.335
MAX  =        6     MAXTIN =      381
```

Figure 5.1D

```
16.24.48. 06 AUG 75   UNIVERSITY OF TEXAS 66/6400   UT 2D
NUMBER OF JOBS MOVED. FOR EACH ROLLIN OR EXPANSION. BY SIMULATED MEMORY MANAGER
  FROM        TALLY   SCALING:  1 PRINT COLUMN = 0.020

    0          4373   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    1           470   XXXX
    2           248   XX
    3            87
    4            37
    5             8
    6             2

NUMB =     5225     NUMANZ =      852
MEAN =    0.273     NZMEAN =    1.674
K**2 =    6.978     NZK**2 =    0.301
MAX  =        6     MAXTIN =      525
```

Figure 5.1E

memory was a surprise. This, of course, means that all the compaction is caused by only 36% of the storage increases. Note that the mean number of jobs per storage increase was .580 but with a fairly large coefficient of variation. The number of moves per "RST up" given that at least one job was moved was considerably more, 1.65.

5.3.1.2 First-Fit-1

The number of moves per storage increase for algorithm FF1 is markedly different from the system algorithm as is shown in Figure 5.1B. The FF1 algorithm never compacts unless every available hole is too small. Almost 83% of requested increases are completed without requiring compaction.

This algorithm resolves the overflow condition by forcing the new job into memory immediately above the SATURN control point. This was so that the system's bias, of forcing the jobs which remain in memory longer, to the top of memory, was maintained. It was also hoped that doing so would leave fewer jobs available for movement when compaction was necessary. This was not accomplished. The mean number of jobs moved, given that at least one job was moved has increased over the system algorithm. The mean number of jobs moved for all storage increases is reduced by almost 50%. This means that FF1 succeeds in postponing compaction, but once overflow is reached, more work is required to resolve the condition.

5.3.1.3 First-Fit-2

Figure 5.1C shows the same metric for algorithm FF2. The fraction of jobs which increased without causing compaction is once again slightly increased and the mean number of jobs moved per "RST up" is also improved. FF2, which expands the largest hole when necessary, is better than FF1 in all aspects of this metric. FF2 still moves slightly more jobs than the system on those occasions when a non-zero number of jobs is moved.

5.3.1.4 Best-Fit-1

This version of the best-fit algorithm uses the "low-hole" for rollins when no hole is of sufficient size. Once again, the superiority of this algorithm to the system algorithm is evident (see Figure 5.1D). It does, however, have a larger mean number of jobs moved when a non-zero number of moves was made. Comparisons with FF1 and FF2 are not as conclusive. The fraction of jobs requiring no compaction to complete an "RST up" is virtually identical to FF2 in the figure shown, but varies from best to worst of the algorithm for the other tapes. However, when comparisons of the mean number of jobs moved for all expansions and the mean number of moves given that compaction was necessary are considered, the placement of BF1 with respect to the first-fit algorithms becomes clear. BF1 is marginally better than FF1 but it is also decidedly inferior to FF2.

5.3.1.5 Best-Fit-2

The results for the BF2 algorithm, shown in Figure 5.1E, compare very closely to the FF2 algorithm. BF2 is better than FF2 in every aspect of the metric. However, the numbers compare so closely that results for BF2 and FF2 are virtually indistinguishable for this metric.

5.3.1.6 Summary

On the basis of this metric, it can be said that all four of the proposed placement algorithms are superior to the current system algorithm. It also appears that expanding to the largest hole rather than the low hole is desirable as FF2 is better than FF1 and BF2 is better than BF1. It does not, however, seem clear whether FF2 or BF2 is superior since they compare so closely. Further comparison will be needed to make this judgement as well as to confirm the other trends of comparison between algorithms.

5.3.2 Total Field Length Moved per Storage Increase

This data is the most direct measurement of the actual compaction overhead. As shown in the previous section, a very large fraction of the storage increases required no movement of jobs and therefore, zero overhead. It was desired to look at the distribution of compactions actually performed. Therefore, for reasons of scaling, the data presented in this section is actually a distribution of the non-zero compactions performed. It must be realized during the course of

this discussion that there is an extremely large spike at "0" which is not shown.

The average number of words moved per storage increase, the average non-zero number of words moved per increase and the total number of words moved are the three most important aspects of this metric.

The data presented in this section was obtained from Tape 3.

5.3.2.1 System Algorithm

The results for the simulation of the system algorithm are given in Figure 5.2A. The distributions are very spiked, more than might intuitively have been expected. The reason, though, is not obscure. Figure 4.6 shows the distribution of the field lengths for jobs which run. The general similarity of shape is unmistakable. The spikes, though of different relative heights, are in the same place. From the figures of section 5.3, it can be seen that for those cases where at least one job is moved, nearly half are exactly one job. Furthermore, if two 4K jobs are moved, this totals 8K; if two 8K jobs, it totals to 16K and so on. It is easily seen why the spikes appear to coincide.

5.3.2.2 First-Fit-1

As was anticipated from the results of the previous section, the mean number of words moved per "RST up" was dramatically decreased, due primarily to the increased number of storage increases for which no compaction was necessary. The total number of words

```
 15.35.12. 26 SEP 75   UNIVERSITY OF TEXAS 66/6400   UT 2D
FIELD LENGTH MOVED IN SIMULATED MEMORY MANAGER

   FROM        TO          TALLY    SCALING:  1 PRINT COLUMN = 0.00100

    0          0.500          5     X
    0.500      1              0
    1          1.500         31     XXXXXXXX
    1.500      2             37     XXXXXXXXX
    2          2.500          4     X
    2.500      3              5     X
    3          3.500         20     XXXXX
    3.500      4              3
    4          4.500        210     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    4.500      5             17     XXXX
    5          5.500         11     XX
    5.500      6             33     XXXXXXXX
    6          6.500          7     X
    6.500      7              8     XX
    7          7.500         24     XXXXXX
    7.500      8              8     XX
    8          8.500         23     XXXXXX
    8.500      9             70     XXXXXXXXXXXXXXXXXX
    9          9.500         31     XXXXXXXX
    9.500     10             27     XXXXXXX
   10         10.500          8     XX
   10.500     11             30     XXXXXXX
   11         11.500         19     XXXX
   11.500     12             10     XX
   12         12.500         16     XXXX
   12.500     13             33     XXXXXXXX
   13         13.500         11     XX
   13.500     14              9     XX
   14         14.500         24     XXXXXX
   14.500     15              8     XX
   15         15.500          7     X
   15.500     16              9     XX
   16         16.500         50     XXXXXXXXXXXXX
   16.500     17             26     XXXXXX
   17         17.500         15     XXX
   17.500     18             20     XXXXX
   18         18.500         16     XXXX
   18.500     19             16     XXXX
   19         19.500          7     X
   19.500     20             10     XX
   20         20.500         58     XXXXXXXXXXXXXXX
   20.500     21             19     XXXX
   21         21.500          7     X
   21.500     22             23     XXXXXX
   22         22.500         12     XXX
   22.500     23             26     XXXXXX
   23         23.500         16     XXXX
   23.500     24              9     XX
   24         24.500          9     XX
   24.500     25             27     XXXXXXX    NUMA =      3824     NUMANZ =     1259
   25         25.500          6     X          MEAN =     4.321     NZMEAN =   13.124
   25.500     26             10     XX         K**2 =     3.291     NZK**2 =    0.412
   26         26.500         20     XXXXX      MAX  =        33     MAXTIM =     1654
   26.500     27             13     XXX
   27         27.500         10     XX
   27.500     28             13     XXX        TOTAL FIELD LENGTH MOVED IN SIMULATIONS: 16523.937
   28         28.500          9     XX
   28.500     29             14     XXX
   29         29.500          7     X
   29.500     30              7     X
   30         30.500          9     XX
   30.500     31              2
   31         31.500          4     X
   31.500     32              2
   32         32.500          1
   32.500     33              3
   33         33.500          2
   33.500     34              3
```

Figure 5.2A

```
 15.35.12. 26 SEP 75   UNIVERSITY OF TEXAS 66/6400   UT 2D
FIELD LENGTH MOVED IN SIMULATED MEMORY MANAGER

   FROM        TO          TALLY    SCALING:  1 PRINT COLUMN = 0.00050

    0          0.500          6     XXX
    0.500      1              1
    1          1.500         30     XXXXXXXXXXXXXXX
    1.500      2             18     XXXXXXXXX
    2          2.500          0
    2.500      3              0
    3          3.500         15     XXXXXXX
    3.500      4              1
    4          4.500         74     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    4.500      5              8     XXXX
    5          5.500          3     X
    5.500      6             21     XXXXXXXXXX
    6          6.500          4     XX
    6.500      7              2     X
    7          7.500         10     XXXXX
    7.500      8              3     X
    8          8.500         12     XXXXXX
    8.500      9             20     XXXXXXXXXX
    9          9.500         15     XXXXXXX
    9.500     10             16     XXXXXXXX
   10         10.500          4     XX
   10.500     11             13     XXXXXX
   11         11.500          6     XXX
   11.500     12              5     XX
   12         12.500         13     XXXXXX
   12.500     13             17     XXXXXXXX
   13         13.500          3     X
   13.500     14              4     XX
   14         14.500         14     XXXXXXX
   14.500     15              3     X
   15         15.500          5     XX
   15.500     16              7     XXX
   16         16.500         22     XXXXXXXXXXX
   16.500     17              6     XXX
   17         17.500          5     XX
   17.500     18             12     XXXXXX
   18         18.500          4     XX
   18.500     19             15     XXXXXXX
   19         19.500          4     XX
   19.500     20              6     XXX
   20         20.500         20     XXXXXXXXXX
   20.500     21              9     XXXX
   21         21.500          5     XX
   21.500     22             11     XXXXX
   22         22.500          5     XX
   22.500     23             14     XXXXXXX
   23         23.500          3     X
   23.500     24              5     XX
   24         24.500          4     XX
   24.500     25              9     XXXX
   25         25.500          2     X          NUMB =      3824     NUMBNZ =      565
   25.500     26              5     XX         MEAN =     1.880     NZMEAN =   12.728
   26         26.500          9     XXXX       K**2 =     8.732     NZK**2 =    0.438
   26.500     27              3     X          MAX  =        33     MAXTIM =     2203
   27         27.500          4     XX
   27.500     28              3     X          TOTAL FIELD LENGTH MOVED IN SIMULATIONS:  7191.375
   28         28.500          2     X
   28.500     29              8     XXXX
   29         29.500          2     X
   29.500     30              3     X
   30         30.500          1
   30.500     31              2     X
   31         31.500          2     X
   31.500     32              1
   32         32.500          0
   32.500     33              2     X
   33         33.500          2     X
   33.500     34              2     X
```

Figure 5.2B

```
15.35.12. 26 SEP 75   UNIVERSITY OF TEXAS 66/6400   UT 2D
FIELD LENGTH MOVED IN SIMULATED MEMORY MANAGER

  FROM        TO         TALLY   SCALING!  1 PRINT COLUMN = 0.00050

   0          0.500        7     XXX
   0.500      1            2     X
   1          1.500       30     XXXXXXXXXXXXXXX
   1.500      2           23     XXXXXXXXXXXX
   2          2.500        1
   2.500      3            2     X
   3          3.500       22     XXXXXXXXXXX
   3.500      4            1
   4          4.500      105     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   4.500      5           11     XXXXX
   5          5.500        5     XX
   5.500      6           22     XXXXXXXXXXX
   6          6.500        5     XX
   6.500      7            4     XX
   7          7.500       11     XXXXX
   7.500      8            1
   8          8.500       17     XXXXXXXX
   8.500      9           17     XXXXXXXX
   9          9.500       15     XXXXXXX
   9.500     10           14     XXXXXXX
  10         10.500        8     XXXX
  10.500     11           14     XXXXXXX
  11         11.500        5     XX
  11.500     12            5     XX
  12         12.500        5     XX
  12.500     13           18     XXXXXXXXX
  13         13.500        5     XX
  13.500     14            5     XX
  14         14.500       10     XXXXX
  14.500     15            4     XX
  15         15.500        2     X
  15.500     16            7     XXX
  16         16.500       18     XXXXXXXXX
  16.500     17            7     XXX
  17         17.500        7     XXX
  17.500     18            6     XXX
  18         18.500        4     XX
  18.500     19           10     XXXXX
  19         19.500        3     X
  19.500     20            2     X
  20         20.500       20     XXXXXXXXXX
  20.500     21           10     XXXXX
  21         21.500        4     XX
  21.500     22            7     XXX
  22         22.500        5     XX
  22.500     23            8     XXXX
  23         23.500        6     XXX
  23.500     24            1
  24         24.500        5     XX
  24.500     25            6     XXX       NUMD =      3824     NUMBNZ =       566
  25         25.500        0               MEAN =     1.612     NZMEAN =    10.896
  25.500     26            1               K**2 =     9.413     NZK**2 =     0.541
  26         26.500        5     XX        MAX  =        33     MAXTIM =        87
  26.500     27            5     XX
  27         27.500        3     X         TOTAL FIELD LENGTH MOVED IN SIMULATIONS!  6167.562
  27.500     28            4     XX
  28         28.500        0
  28.500     29            2     X
  29         29.500        4     XX
  29.500     30            1
  30         30.500        2     X
  30.500     31            1
  31         31.500        0
  31.500     32            1
  32         32.500        1
  32.500     33            2     X
  33         33.500        2     X
```

Figure 5.2C

```
15.35.12. 26 SEP 75   UNIVERSITY OF TEXAS 66/6400   UT 2D
FIELD LENGTH MOVED IN SIMULATED MEMORY MANAGER

  FROM        TO         TALLY   SCALING!  1 PRINT COLUMN = 0.00050

   0          0.500        5     XX
   0.500      1            0
   1          1.500       28     XXXXXXXXXXXXXX
   1.500      2           19     XXXXXXXXX
   2          2.500        0
   2.500      3            1
   3          3.500       14     XXXXXXX
   3.500      4            1
   4          4.500       84     XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   4.500      5            6     XXX
   5          5.500        3     X
   5.500      6           19     XXXXXXXXX
   6          6.500        5     XX
   6.500      7            3     X
   7          7.500       10     XXXXX
   7.500      8            0
   8          8.500       10     XXXXX
   8.500      9           20     XXXXXXXXXX
   9          9.500       16     XXXXXXXX
   9.500     10           13     XXXXXX
  10         10.500        6     XXX
  10.500     11           12     XXXXXX
  11         11.500        8     XXXX
  11.500     12            3     X
  12         12.500        9     XXXX
  12.500     13           17     XXXXXXXX
  13         13.500        4     XX
  13.500     14            4     XX
  14         14.500       14     XXXXXXX
  14.500     15            5     XX
  15         15.500        6     XXX
  15.500     16            3     X
  16         16.500       22     XXXXXXXXXXX
  16.500     17           11     XXXXX
  17         17.500        5     XX
  17.500     18           10     XXXXX
  18         18.500        7     XXX
  18.500     19           14     XXXXXXX
  19         19.500        5     XX
  19.500     20            5     XX
  20         20.500       25     XXXXXXXXXXXXX
  20.500     21            8     XXXX
  21         21.500        5     XX
  21.500     22            9     XXXX
  22         22.500        5     XX
  22.500     23           11     XXXXX
  23         23.500        3     X
  23.500     24            6     XXX
  24         24.500        5     XX
  24.500     25            8     XXXX
  25         25.500        5     XX        NUMB =      3824     NUMBNZ =       573
  25.500     26            5     XX        MEAN =     1.944     NZMEAN =    12.978
  26         26.500        7     XXX       K**2 =     8.613     NZK**2 =     0.448
  26.500     27            3     X         MAX  =        33     MAXTIM =      1654
  27         27.500        4     XX
  27.500     28            3     X         TOTAL FIELD LENGTH MOVED IN SIMULATIONS!  7436.562
  28         28.500        3     X
  28.500     29            7     XXX
  29         29.500        4     XX
  29.500     30            2     X
  30         30.500        2     X
  30.500     31            3     X
  31         31.500        3     X
  31.500     32            1
  32         32.500        2     X
  32.500     33            2     X
  33         33.500        2     X
  33.500     34            3     X
```

Figure 5.2D

```
15.35.12. 26 SEP 75   UNIVERSITY OF TEXAS 66/6400   UT 20
FIELD LENGTH MOVED IN SIMULATED MEMORY MANAGER

 FROM        TO        TALLY   SCALING!  1 PRINT COLUMN = 0.00050

  0          0.500        7    XXX
  0.500      1            0
  1          1.500       29    XXXXXXXXXXXXXX
  1.500      2           22    XXXXXXXXXXX
  2          2.500        1
  2.500      3            2    X
  3          3.500       20    XXXXXXXXXX
  3.500      4            1
  4          4.500      104    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  4.500      5            9    XXXX
  5          5.500        5    XX
  5.500      6           14    XXXXXXX
  6          6.500        6    XXX
  6.500      7            3    X
  7          7.500       11    XXXXX
  7.500      8            1
  8          8.500       22    XXXXXXXXXXX
  8.500      9           16    XXXXXXXX
  9          9.500       11    XXXXX
  9.500     10           12    XXXXXX
 10         10.500        8    XXXX
 10.500     11           14    XXXXXXX
 11         11.500        8    XXXX
 11.500     12            2    X
 12         12.500        3    X
 12.500     13           16    XXXXXXXX
 13         13.500        4    XX
 13.500     14            6    XXX
 14         14.500       14    XXXXXXX
 14.500     15           10    XXXXX
 15         15.500        2    X
 15.500     16            5    XX
 16         16.500       20    XXXXXXXXXX
 16.500     17            8    XXXX
 17         17.500        6    XXX
 17.500     18            7    XXX
 18         18.500        4    XX
 18.500     19           11    XXXXX
 19         19.500        0
 19.500     20            2    X
 20         20.500       19    XXXXXXXXX
 20.500     21           11    XXXXX
 21         21.500        2    X
 21.500     22            7    XXX
 22         22.500        4    XX
 22.500     23            5    XX
 23         23.500        5    XX
 23.500     24            1
 24         24.500        4    XX
 24.500     25            7    XXX     NUMB =      3024     NUMBNZ =      551
 25         25.500        2    X       MEAN =     1.607     NZMEAN =   11.158
 25.500     26            1            K**2 =     9.597     NZK**2 =    0.527
 26         26.500        8    XXXX    MAX  =        33     MAXTIM =       87
 26.500     27            4    XX
 27         27.500        4    XX      TOTAL FIELD LENGTH MOVED IN SIMULATIONS!  6148.062
 27.500     28            6    XXX
 28         28.500        0
 28.500     29            1
 29         29.500        3    X
 29.500     30            1
 30         30.500        2    X
 30.500     31            1
 31         31.500        0
 31.500     32            1
 32         32.500        2    X
 32.500     33            2    X
 33         33.500        2    X
```

Figure 5.2E

moved, to accomplish the same set of expansions, was less than half the system algorithm for all tapes. For Tape 3, even the average number of words per non-zero move is decreased. For Tapes 1 and 2 this was not found to be true. For those two tapes, less work was required overall, because it was necessary much less frequently than in the system algorithm. However, when finally required to compact, more compaction was necessary.

5.3.2.3 First-Fit-2

Figure 5.2C presents the results for algorithm FF2 for this metric. The general shape of curve is the same as the previous two algorithms. Although the relative heights of the spikes varies, the same spikes remain predominant.

Comparison to FF1 parallels the comparison of section 5.3.1.3. In all aspects of this metric, FF2 is superior to FF1. The total number of words moved is about 14% less than FF1 which was itself less than 45% of the system algorithm. This algorithm is also better than the system algorithm in the mean number of words moved given that compaction was necessary. FF2 must compact less frequently than both FF1 and the system, and even when it must compact, fewer words are moved.

5.3.2.4 Best-Fit-1

Comparisons of BF1 to FF1 and FF2 are inconclusive. In the figure shown (Figure 5.2D) BF1 is inferior to both first-fit algorithms. For the other tapes, however, BF1 was between the first-fit algorithms. Still, the difference between BF1 and FF1 is relatively small and so, no judgement is possible yet.

5.3.2.5 Best-Fit 2

The results, as presented in Figure 5.2E, are consistent for all tapes. Of the 5 algorithms simulated, BF 2 moves the fewest words during the simulation run. As such, it also has the lowest mean number of words moved per storage increase, moving about 60% fewer words per expansion than the current system algorithm. However, the margin over FF2 is so slight as to be negligible. BF2 moves less than 1% fewer words per storage increase than FF2. The relative difference in the total number of words moved is the same. Comparison of the mean number of words moved given that compaction was necessary, show that FF2 is slightly superior. The range of difference is from less than .5% for Tape 2 to 3.5% for Tape 1. This difference is somewhat larger than the differences for the other aspects of the metric.

5.3.2.6 Summary

The results of this metric follow the same trends as the first. Those algorithms which expand the largest hole are definitely superior to those algorithms which choose the "low hole" as they cause less compaction. A judgement between Best-fit and First-fit is still not reasonable. It seems that if no greater difference between the two can be detected, then First-Fit can be recommended simply because it is a less complex algorithm to implement.

5.3.3 CPU Time Spent Moving Jobs

As stated in section 5.2, this metric is essentially the same as the previous metric. It was thought that there might be some slight difference since the time required to move "M" words varies slightly, depending on how many jobs made up the "M" words. This is not the case. Although there is some slight difference in the relative percent between algorithms, the results of comparing algorithms by this metric are the same as for the previous metric. Therefore, a detailed discussion of this metric is not included. Figure 5.3 shows the results of this metric. The figures in this section are from Tape 3 so that the correlation between this metric and the previous metric can be directly compared.

The close correlation of the total field length moved with the CPU time spent moving jobs implies that the importance of the number of jobs moved is reduced. The number of jobs which contain the "M" words moved, is of no consequence. The total number of words per storage increase is the factor of greatest importance. Still, the number of jobs moved per storage increase provides data which supports the results of the other metrics.

One significant result is produced from this metric. The CPU utilization for storage moves is obtained. Dividing the total CPU time spent moving memory, by the total time interval of the tape, produces the fraction of time the CPU was compacting memory. Since CPU utilization is a very common statistic, this represents the most readily understood single metric for quick comparisons of the

```
15.35.12. 26 SEP 75   UNIVERSITY OF TEXAS 66/6400   UT 2D
TIME SPENT MOVING JOBS IN SIMULATED MEMORY MANAGER

  FROM       TO        TALLY  SCALING!  1 PRINT COLUMN = 0.00100

    0         5          36   XXXXXXXXX
    5        10          42   XXXXXXXXXX
   10        15          27   XXXXXXX
   15        20         238   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   20        25          45   XXXXXXXXXXX
   25        30          34   XXXXXXXX
   30        35         120   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   35        40          47   XXXXXXXXXXXX
   40        45          52   XXXXXXXXXXXXX
   45        50          53   XXXXXXXXXXXXX
   50        55          44   XXXXXXXXXXX
   55        60          16   XXXX
   60        65          91   XXXXXXXXXXXXXXXXXXXXXXX
   65        70          40   XXXXXXXXXX
   70        75          31   XXXXXXXX
   75        80          81   XXXXXXXXXXXXXXXXXXXXX
   80        85          47   XXXXXXXXXXXX
   85        90          43   XXXXXXXXXXX
   90        95          40   XXXXXXXXXX
   95       100          35   XXXXXXXXX
  100       105          31   XXXXXXXX
  105       110          30   XXXXXXX
  110       115          19   XXXX
  115       120           8   XX
  120       125           4   X
  125       130           5   X

NUMA  =     3824    NUMBNZ =      1259   TOTAL TIME SPENT MOVING JOBS IN SIMULATION!  61849
MEAN  =   16.173    NZMEAN =    49.125
K**2  =    3.304    NZK**2 =     0.417   UTILIZATION!  0.058
MAX   =      127    MAXTIM =      1654
```

Figure 5.3A

```
15.35.12. 26 SEP 75   UNIVERSITY OF TEXAS 66/6400   UT 2D
TIME SPENT MOVING JOBS IN SIMULATED MEMORY MANAGER

  FROM       TO        TALLY     SCALING!  1 PRINT COLUMN = 0.00050

    0         5          37   XXXXXXXXXXXXXXXXXXX
    5        10          18   XXXXXXXXX
   10        15          16   XXXXXXXX
   15        20          85   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   20        25          26   XXXXXXXXXXXXX
   25        30          14   XXXXXXX
   30        35          45   XXXXXXXXXXXXXXXXXXXXXXX
   35        40          24   XXXXXXXXXXXX
   40        45          22   XXXXXXXXXXX
   45        50          31   XXXXXXXXXXXXXXXX
   50        55          21   XXXXXXXXXX
   55        60          10   XXXXX
   60        65          35   XXXXXXXXXXXXXXXXXX
   65        70          18   XXXXXXXXX
   70        75          24   XXXXXXXXXXXX
   75        80          32   XXXXXXXXXXXXXXXX
   80        85          24   XXXXXXXXXXXX
   85        90          17   XXXXXXXX
   90        95          14   XXXXXXX
   95       100          17   XXXXXXXX
  100       105           8   XXXX
  105       110          12   XXXXXX
  110       115           4   XX
  115       120           5   XX
  120       125           2   X
  125       130           4   XX

NUMA  =     3824    NUMBNZ =       565   TOTAL TIME SPENT MOVING JOBS IN SIMULATION!  26903
MEAN  =    7.035    NZMEAN =    47.615
K**2  =    8.775    NZK**2 =     0.444   UTILIZATION!  0.025
MAX   =      126    MAXTIM =       416
```

Figure 5.3B

```
15.35.12. 26 SEP 75   UNIVERSITY OF TEXAS 66/6400   UT 2D
TIME SPENT MOVING JOBS IN SIMULATED MEMORY MANAGER

  FROM       TO        TALLY     SCALING!  1 PRINT COLUMN = 0.00050

    0         5          39   XXXXXXXXXXXXXXXXXXXX
    5        10          24   XXXXXXXXXXXX
   10        15          25   XXXXXXXXXXXXX
   15        20         120   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   20        25          31   XXXXXXXXXXXXXXXX
   25        30          13   XXXXXX
   30        35          47   XXXXXXXXXXXXXXXXXXXXXXXX
   35        40          27   XXXXXXXXXXXXXX
   40        45          21   XXXXXXXXXX
   45        50          24   XXXXXXXXXXXX
   50        55          20   XXXXXXXXXX
   55        60          10   XXXXX
   60        65          32   XXXXXXXXXXXXXXXX
   65        70          12   XXXXXX
   70        75          14   XXXXXXX
   75        80          32   XXXXXXXXXXXXXXXX
   80        85          19   XXXXXXXXX
   85        90          11   XXXXX
   90        95          11   XXXXX
   95       100           8   XXXX
  100       105          10   XXXXX
  105       110           4   XX
  110       115           5   XX
  115       120           2   X
  120       125           3   X
  125       130           2   X

NUMA  =     3824    NUMBNZ =       566   TOTAL TIME SPENT MOVING JOBS IN SIMULATION!  23051
MEAN  =    6.027    NZMEAN =    40.726
K**2  =    9.471    NZK**2 =     0.550
MAX   =      125    MAXTIM =        87   UTILIZATION!  0.021
```

Figure 5.3C

```
15.35.12. 26 SEP 75   UNIVERSITY OF TEXAS 66/6400   UT 2D
TIME SPENT MOVING JOBS IN SIMULATED MEMORY MANAGER

  FROM       TO        TALLY     SCALING!  1 PRINT COLUMN = 0.00050

    0         5          33   XXXXXXXXXXXXXXXXX
    5        10          19   XXXXXXXXX
   10        15          16   XXXXXXXX
   15        20          93   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   20        25          26   XXXXXXXXXXXXX
   25        30          11   XXXXX
   30        35          43   XXXXXXXXXXXXXXXXXXXXXX
   35        40          23   XXXXXXXXXXXX
   40        45          22   XXXXXXXXXXX
   45        50          27   XXXXXXXXXXXXXX
   50        55          23   XXXXXXXXXXXX
   55        60          11   XXXXX
   60        65          37   XXXXXXXXXXXXXXXXXXX
   65        70          19   XXXXXXXXX
   70        75          23   XXXXXXXXXXXX
   75        80          36   XXXXXXXXXXXXXXXXXX
   80        85          22   XXXXXXXXXXX
   85        90          15   XXXXXXX
   90        95          15   XXXXXXX
   95       100          16   XXXXXXXX
  100       105           8   XXXX
  105       110          14   XXXXXXX
  110       115           5   XX
  115       120           7   XXX
  120       125           4   XX
  125       130           5   XX

NUMA  =     3824    NUMBNZ =       573   TOTAL TIME SPENT MOVING JOBS IN SIMULATION!  27830
MEAN  =    7.277    NZMEAN =    48.568
K**2  =    8.653    NZK**2 =     0.447
MAX   =      127    MAXTIM =      1654   UTILIZATION!  0.026
```

Figure 5.3D

```
15.35.12. 26 SEP 75   UNIVERSITY OF TEXAS 66/6400   UT 20
TIME SPENT MOVING JOBS IN SIMULATED MEMORY MANAGER

  FROM      TO       TALLY   SCALING:  1 PRINT COLUMN = 0.00050

    0        5          36   XXXXXXXXXXXXXXXXXX
    5       10          23   XXXXXXXXXXXX
   10       15          23   XXXXXXXXXXXX
   15       20         117   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   20       25          23   XXXXXXXXXXXX
   25       30          13   XXXXXX
   30       35          48   XXXXXXXXXXXXXXXXXXXXXXXXX
   35       40          24   XXXXXXXXXXXX
   40       45          21   XXXXXXXXXX
   45       50          20   XXXXXXXXXX
   50       55          29   XXXXXXXXXXXXXXX
   55       60          10   XXXXX
   60       65          33   XXXXXXXXXXXXXXXXX
   65       70          13   XXXXXX
   70       75          12   XXXXXX
   75       80          31   XXXXXXXXXXXXXXXX
   80       85          16   XXXXXXXX
   85       90           8   XXXX
   90       95          11   XXXXX
   95      100          13   XXXXXX
  100      105          11   XXXXX
  105      110           3   X
  110      115           5   XX
  115      120           2   X
  120      125           4   XX
  125      130           2   X

NUMB  =     3824    NUMBNZ  =      551   TOTAL TIME SPENT MOVING JOBS IN SIMULATION:  22981
MEAN  =    6.009    NZMEAN  =   41.707
K**2  =    9.655    NZK**2  =    0.536
MAX   =      125    MAXTIM  =       87   UTILIZATION:  0.021
```

Figure 5.3E

algorithms. This of course ignores side effects, but the other metrics should be used for judging algorithms which are indistinguishable on a basis of CPU utilization.

Looking at the CPU utilizations, the comparisons of the previous section are evident. Version 2 of each algorithm is superior to version 1, but Best-fit and First-fit for the same version are very close, even identical for version 2. All four new algorithms are at least 50% better than the current system algorithm.

5.3.4 Number of Moves per Residence

This metric is not a direct measure of compaction overhead. It's included as a measure of impact on the other system resources. Whenever a control point is moved, those other resources which are allocated to the job, must essentially be held in an idle state. Most importantly, any PPs which are assigned must either terminate or pause for relocation.

One of the specific design goals of the current system placement and expansion algorithms was the reduction of this metric. Under the current algorithm, the longer a job is resident, the more likely that it will be forced up in memory. As a job reaches the top of memory, it should be moved less often since the system placement algorithm forces most activity of movement of jobs to be at the bottom of memory.

Those aspects of this metric which seem most useful are similar to those used in section 5.3.1:

1) The fraction of jobs which are not moved,

2) The mean number of times a job is moved,

3) The mean number of times a job is moved, given that it is moved at least once.

The results presented were obtained from Tape 1.

5.3.4.1 System Algorithm

Figure 5.4A shows that only 45% of jobs complete their residences without being relocated for compaction. This figure is higher for Tapes 2 and 3. Tape 1 was chosen because it allowed the most room for improvement. The mean number of moves per residence is .81 and the coefficient of variation is relatively low. On those residences when a job is moved, it is moved an average of 1.5 times. The system method of always choosing the low hole seems to achieve its goal of moving a job to where it will no longer have to be moved, by moving a large portion of jobs at least once. Only those jobs which rollin when memory is mostly empty, or which have very short residence times escape without being moved.

5.3.4.2 First-Fit-1

The results for algorithm FF1 are given in Figure 5.4B. They show that even this algorithm, which has shown to be the least successful of the proposed algorithms, is a definite improvement over the system algorithm. In the figure shown, 69% more jobs were not moved. The relative improvement for the other tapes is not a great

```
 15.48.36. 18 OCT 74   UNIVERSITY OF TEXAS 66/6400   UT 2D
NUMBER OF MOVES PER RESIDENCE IN SIMULATED MEMORY MANAGER

   FROM  TALLY    SCALING:  1 PRINT COLUMN = 0.0100

    0     1502    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    1     1073    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    2      315    XXXXXXXXXX
    3      100    XXX
    4       25
    5        6
    6        0
    7        1

  NUMA =      3022     NUMANZ =     1520
  MEAN =     0.708     NZMEAN =    1.407
  K**2 =     1.529     NZK**2 =    0.272
  MAX  =         7     MAXTIM =     1284
```

Figure 5.4A

```
 15.48.36. 18 OCT 74   UNIVERSITY OF TEXAS 66/6400   UT 2D
NUMBER OF MOVES PER RESIDENCE IN SIMULATED MEMORY MANAGER

   FROM  TALLY  SCALING:  1 PRINT COLUMN = 0.0100

    0     2290  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    1      549  XXXXXXXXXXXXXXXXXX
    2      137  XXXX
    3       34  X
    4       10
    5        2

  NUMA =      3022     NUMANZ =      732
  MEAN =     0.322     NZMEAN =    1.332
  K**2 =     4.124     NZK**2 =    0.242
  MAX  =         5     MAXTIM =      451
```

Figure 5.4B

```
 15.48.36. 18 OCT 74   UNIVERSITY OF TEXAS 66/6400   UT 2D
NUMBER OF MOVES PER RESIDENCE IN SIMULATED MEMORY MANAGER

   FROM  TALLY  SCALING:  1 PRINT COLUMN = 0.0100

    0     2397  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    1      487  XXXXXXXXXXXXXXXX
    2       96  XXX
    3       27
    4       10
    5        2
    6        2
    7        0
    8        0
    9        1

  NUMA =      3022     NUMANZ =      625
  MEAN =     0.274     NZMEAN =    1.329
  K**2 =     5.465     NZK**2 =    0.337
  MAX  =         9     MAXTIM =     1106
```

Figure 5.4C

```
 15.48.36. 18 OCT 74   UNIVERSITY OF TEXAS 66/6400   UT 2D
NUMBER OF MOVES PER RESIDENCE IN SIMULATED MEMORY MANAGER

   FROM  TALLY SCALING:  1 PRINT COLUMN = 0.0100

    0      2351 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    1       521 XXXXXXXXXXXXXXXXX
    2       113 XXX
    3        32 X
    4         5

 NUMB =     3022     NUMANZ =      671
 MEAN =    0.285     NZMEAN =    1.286
 K**2 =    4.442     NZK**2 =    0.208
 MAX  =        4     MAXTIM =      445
```

Figure 5.4D

```
 15.48.36. 18 OCT 74   UNIVERSITY OF TEXAS 66/6400   UT 2D
NUMBER OF MOVES PER RESIDENCE IN SIMULATED MEMORY MANAGER

   FROM TALLY SCALING:  1 PRINT COLUMN = 0.0100

    0     2407 XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    1      471 XXXXXXXXXXXXXXX
    2      118 XXX
    3       20
    4        4
    5        2

 NUMA =     3022     NUMANZ =      615
 MEAN =    0.262     N7MEAN =    1.289
 K**2 =    4.946     N7K**2 =    0.210
 MAX  =        5     MAXTIM =      445
```

Figure 5.4E

but is still substantial. The system method of rolling all jobs into memory as the lowest job, is too great a sacrifice.

Even the mean number of non-zero moves is reduced showing that the reduction was not just the elimination of compaction for jobs which were moved only once. FF1 achieves the stated system goal of reducing the number of times a job is moved, better than the system.

5.3.4.3 First-Fit-2

The First-fit-2 algorithm is once again better than FF1. The fraction of jobs which are never moved is slightly increased and the mean number of moves per residence is once again reduced as is the mean number of non-zero moves per residence. However, the reduction in the non-zero mean is much less. The maximum number of times a job is moved is increased greatly. Although the frequency of this occurrence is low, this is part of what a "desirable" algorithm should avoid and is a feature of this algorithm which is undesirable.

5.3.4.4 Best-Fit-1

Comparison of results for BF1 given in Figure 5.4D yields no startling results. On Tape 2 it yields the lowest mean non-zero number of moves per residence but also results in a high maximum number of moves per residence. In 5.4D, it can be seen that for Tape 1, BF1 has the lowest mean number of moves per residence, given that at least one move occurred, of all algorithms. Based on the mean number of moves per residence, BF1 falls between FF1 and FF2 as it has for the other metrics.

5.3.4.5 Best-Fit-2

Comparison of Best-Fit-2 to the other algorithms yields results which follow the same established trend. The fraction of jobs which are moved is greater than for any other algorithm though by less than 0.5% of FF2. The mean number of moves per residence for BF2 is less than FF2 by over 4%. For Tapes 2 and 3 (not shown) this algorithm suffers the same flaw as algorithm FF2. The maximum number of times a job is moved is 10. This occurence is so infrequent (.03%) that it is easily outweighed by other advantages.

5.3.4.6 Summary

The results for this metric closely mirror those of the other metrics. BF2 seems best but only marginally over FF2. Version 2 of an algorithm is better than Version 1 but either version of both First-Fit and Best-Fit are better than the current system algorithm.

5.3.5 Further Studies

The results of the previous sections have shown that much better results than the current system can be obtained by changing the algorithm for placement of jobs. All of the simulations changed only the placement algorithm for rolling in new jobs. The algorithm for expanding a hole is unchanged.

The system expansion algorithm is biased toward pushing jobs up in memory. The algorithm is described in section 3.2.3.3. When moving a job up in memory, the job is moved completely to the top of the adjacent hole. When a job is moved down, it is sometimes moved only partially down into the adjacent hole.

Algorithms FF2 and BF2 have no bias toward one end or the other when selecting a hole to be expanded. It seemed possible that the bias of the system expansion algorithm might adversely affect the performance of these two algorithms. A change was made in the expansion algorithm to move a job completely down into an adjacent hole instead of partially. No other change was made in the expansion algorithm. The results which were obtained follow.

It did not seem that this change would enhance algorithms FF1 and BF1. Simulations showed that the effect of this change on algorithms FF1 and BF1 was slightly negative. Results for FF1 and BF1 were not interesting and discussion is omitted.

The results of this experiment were as conjectured. The system algorithm for expansion had been adversely affecting the performance of those algorithms which expand the largest hole. These results are shown in Figures 5.5 through 5.8. Only the results from Tape 2 are shown. In some metrics the changes are insignificant, but in others, the change is as large as was obtained by a change in placement algorithms. In general, the results are superior to those obtained by any placement algorithm used with the system expansion

```
16.24.48. 06 AUG 75   UNIVERSITY OF TEXAS 66/6400   UT 20
NUMBER OF JOBS MOVED, FOR EACH ROLLIN OR EXPANSION, BY SIMULATED MEMORY MANAGER

  FROM                  TALLY   SCALING:  1 PRINT COLUMN = 0.020

    0                    4422   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    1                     457   XXXX
    2                     223   XX
    3                      80
    4                      32
    5                       9
    6                       2

 NUMB =      5225     NUMBNZ =       803
 MEAN =     0.254     NZMEAN =     1.653
 K**2 =     7.528     NZK**2 =     0.310
 MAX  =         6     MAXTIM =       915
```

Figure 5.5A

```
16.24.48. 06 AUG 75   UNIVERSITY OF TEXAS 66/6400   UT 20
NUMBER OF JOBS MOVED, FOR EACH ROLLIN OR EXPANSION, BY SIMULATED MEMORY MANAGER

  FROM                  TALLY   SCALING:  1 PRINT COLUMN = 0.020

    0                    4440   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    1                     457   XXXX
    2                     227   XX
    3                      69
    4                      23
    5                       9

 NUMB =      5225     NUMBNZ =       785
 MEAN =     0.240     NZMEAN =     1.598
 K**2 =     7.557     NZK**2 =     0.285
 MAX  =         5     MAXTIM =        80
```

Figure 5.5B

```
FIELD LENGTH MOVED IN SIMULATED MEMORY MANAGER

  FROM       TO        TALLY   SCALING:  1 PRINT COLUMN = 0.00050

   0         0.500         3   X
   0.500     1             2
   1         1.500        18   XXXXXX
   1.500     2            39   XXXXXXXXXXXXXX
   2         2.500         8   XXX
   2.500     3             7   XX
   3         3.500        14   XXXXX
   3.500     4             3   X
   4         4.500       106   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   4.500     5            12   XXXX
   5         5.500        23   XXXXXXXX
   5.500     6            19   XXXXXXX
   6         6.500         5   X
   6.500     7            11   XXXX
   7         7.500         7   XX
   7.500     8             6   XX
   8         8.500         5   X
   8.500     9            12   XXXX
   9         9.500        12   XXXX
   9.500    10            16   XXXXXX
  10        10.500         8   XXX
  10.500    11            81   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  11        11.500         5   X
  11.500    12            10   XXX
  12        12.500        17   XXXXXX
  12.500    13            12   XXXX
  13        13.500         4   X
  13.500    14             8   XXX
  14        14.500         2
  14.500    15             4   X
  15        15.500        40   XXXXXXXXXXXXXXX
  15.500    16             9   XXX
  16        16.500        26   XXXXXXXXX
  16.500    17            10   XXX
  17        17.500         8   XXX
  17.500    18             6   XX
  18        18.500        19   XXXXXXX
  18.500    19            12   XXXX
  19        19.500         9   XXX
  19.500    20             9   XXX
  20        20.500        30   XXXXXXXXXXX
  20.500    21            10   XXX
  21        21.500         5   X
  21.500    22            14   XXXXX
  22        22.500        10   XXX
  22.500    23            15   XXXXX
  23        23.500         5   X
  23.500    24             4   X
  24        24.500         7   XX
  24.500    25             7   XX       NUMB =      5225     NUMBNZ =       803
  25        25.500         3   X        MEAN =     1.941     NZMEAN =    12.634
  25.500    26             6   XX       K**2 =     8.327     NZK**2 =     0.433
  26        26.500         4   X        MAX  =        37     MAXTIM =       922
  26.500    27             4   X
  27        27.500         6   XX
  27.500    28             1             TOTAL FIELD LENGTH MOVED IN SIMULATIONS: 10145.125
  28        28.500         4   X
  28.500    29             4   X
  29        29.500         8   XXX
  29.500    30             6   XX
  30        30.500         1
  30.500    31             2
  31        31.500         5   X
  31.500    32             2
  32        32.500         1
  32.500    33             2
  33        33.500         1
  33.500    34             0
  34        34.500         0
  34.500    35             2
  35        35.500         1
  35.500    36             4   X
  36        36.500         1
  36.500    37             0
  37        37.500         0
  37.500    38             1
```

Figure 5.6A

```
16.24.48. 06 AUG 75   UNIVERSITY OF TEXAS 66/6400   UT 2D
FIELD LENGTH MOVED IN SIMULATED MEMORY MANAGER

  FROM        TO          TALLY  SCALING:  1 PRINT COLUMN = 0.00050

   0          0.500          6   XX
   0.500      1              1
   1          1.500         18   XXXXXX
   1.500      2             34   XXXXXXXXXXXXX
   2          2.500          4   X
   2.500      3              4   X
   3          3.500         13   XXXX
   3.500      4              2
   4          4.500        115   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   4.500      5             12   XXXX
   5          5.500         21   XXXXXXXX
   5.500      6             20   XXXXXXX
   6          6.500          5   X
   6.500      7             11   XXXX
   7          7.500          6   XX
   7.500      8              7   XX
   8          8.500          4   X
   8.500      9             13   XXXX
   9          9.500         13   XXXX
   9.500     10             12   XXXX
  10         10.500          9   XXX
  10.500     11             82   XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  11         11.500          7   XX
  11.500     12             10   XXX
  12         12.500         18   XXXXXX
  12.500     13             10   XXX
  13         13.500          8   XXX
  13.500     14             10   XXX
  14         14.500          4   X
  14.500     15              7   XX
  15         15.500         37   XXXXXXXXXXXXXX
  15.500     16              9   XXX
  16         16.500         26   XXXXXXXXX
  16.500     17             11   XXXX
  17         17.500          4   X
  17.500     18              5   X
  18         18.500         18   XXXXXX
  18.500     19              6   XX
  19         19.500         11   XXXX
  19.500     20              8   XXX
  20         20.500         26   XXXXXXXXX
  20.500     21              8   XXX
  21         21.500          7   XX
  21.500     22             14   XXXXX
  22         22.500          8   XXX
  22.500     23             20   XXXXXXX
  23         23.500          3   X
  23.500     24              5   X
  24         24.500          7   XX
  24.500     25              6   XX   NUMB =      5225     NUMBNZ =      785
  25         25.500          1        MEAN =     1.864     NZMEAN =   12.408
  25.500     26              7   XX   K**2 =     0.490     NZK**2 =    0.426
  26         26.500          5   X    MAX  =        37     MAXTIM =     2562
  26.500     27              3   X
  27         27.500          4   X
  27.500     28              0          TOTAL FIELD LENGTH MOVED IN SIMULATIONS:  9740.687
  28         28.500          3   X
  28.500     29              4   X
  29         29.500          4   X
  29.500     30              7   XX
  30         30.500          2
  30.500     31              2
  31         31.500          7   XX
  31.500     32              1
  32         32.500          1
  32.500     33              2
  33         33.500          2
  33.500     34              1
  34         34.500          0
  34.500     35              0
  35         35.500          1
  35.500     36              1
  36         36.500          1
  36.500     37              0
  37         37.500          1
```

Figure 5.6B

```
16.24.48. 06 AUG 75   UNIVERSITY OF TEXAS 66/6400   UT 2D
TIME SPENT MOVING JOBS IN SIMULATED MEMORY MANAGER

  FROM      TO      TALLY   SCALING:  1 PRINT COLUMN = 0.00050

    0        5        20    XXXXXXX
    5       10        50    XXXXXXXXXXXXXXXXXXX
   10       15        24    XXXXXXXXX
   15       20       140    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   20       25        26    XXXXXXXXX
   25       30        20    XXXXXXX
   30       35        29    XXXXXXXXXXX
   35       40        27    XXXXXXXXXX
   40       45        96    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   45       50        32    XXXXXXXXXXXX
   50       55        11    XXXX
   55       60        49    XXXXXXXXXXXXXXXXXX
   60       65        45    XXXXXXXXXXXXXXXXX
   65       70        30    XXXXXXXXXXX
   70       75        26    XXXXXXXXX
   75       80        44    XXXXXXXXXXXXXXXX
   80       85        33    XXXXXXXXXXXX
   85       90        18    XXXXXX
   90       95        14    XXXXX
   95      100        13    XXXX
  100      105        11    XXXX
  105      110        13    XXXX
  110      115        12    XXXX                          NUMB =     5225     NUMBNZ =      803
  115      120         5    X                             MEAN =    7.253     NZMEAN =   47.208
  120      125         5    X                             K**2 =    0.369     NZK**2 =    0.440
  125      130         1                                  MAX  =      141     MAXTIM =      922
  130      135         7    XX
  135      140         1
  140      145         1

TOTAL TIME SPENT MOVING JOBS IN SIMULATION:  37902          UTILIZATION:  0.022
```

Figure 5.7A

```
16.24.48. 06 AUG 75   UNIVERSITY OF TEXAS 66/6400   UT 2D
TIME SPENT MOVING JOBS IN SIMULATED MEMORY MANAGER

  FROM      TO      TALLY   SCALING:  1 PRINT COLUMN = 0.00050

    0        5        23    XXXXXXXX
    5       10        40    XXXXXXXXXXXXXXX
   10       15        19    XXXXXXX
   15       20       147    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   20       25        27    XXXXXXXXXX
   25       30        20    XXXXXXX
   30       35        29    XXXXXXXXXXX
   35       40        25    XXXXXXXXX
   40       45        99    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
   45       50        33    XXXXXXXXXXXX
   50       55        19    XXXXXXX
   55       60        48    XXXXXXXXXXXXXXXXXX
   60       65        43    XXXXXXXXXXXXXXXX
   65       70        25    XXXXXXXXX
   70       75        22    XXXXXXXX
   75       80        41    XXXXXXXXXXXXXXX
   80       85        34    XXXXXXXXXXXXX
   85       90        18    XXXXXX
   90       95        13    XXXX
   95      100        13    XXXX
  100      105         7    XX
  105      110         9    XXX
  110      115        13    XXXX                          NUMB =     5225     NUMBNZ =      785
  115      120         8    XXX                           MEAN =    6.963     NZMEAN =   46.349
  120      125         4    X                             K**2 =    0.531     NZK**2 =    0.432
  125      130         2                                  MAX  =      140     MAXTIM =     2542
  130      135         2
  135      140         1
  140      145         1

TOTAL TIME SPENT MOVING JOBS IN SIMULATION:  36384          UTILIZATION:  0.021
```

Figure 5.7B

```
16.24.48. 06 AUG 75   UNIVERSITY OF TEXAS 66/6400   UT 20
NUMBER OF MOVES PER RESIDENCE IN SIMULATED MEMORY MANAGER

FROM                   TALLY    SCALING:  1 PRINT COLUMN = 0.020

  0                     4081    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  1                      734    XXXXXXX
  2                      137    X
  3                       39
  4                       11
  5                        2
  6                        0
  7                        2
  8                        1
  9                        1
 10                        0
 11                        1

NUMB =     5009    NUMBNZ =     928
MEAN =    0.243    NZMEAN =   1.315
K**2 =    6.470    NZK**2 =   0.384
MAX  =       11    MAXTIM =     141
```

Figure 5.8A

```
16.24.48. 06 AUG 75   UNIVERSITY OF TEXAS 66/6400   UT 2D
NUMBER OF MOVES PER RESIDENCE IN SIMULATED MEMORY MANAGER

FROM                   TALLY    SCALING:  1 PRINT COLUMN = 0.020

  0                     4114    XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
  1                      704    XXXXXXX
  2                      138    X
  3                       35
  4                       12
  5                        2
  6                        1
  7                        1
  8                        1
  9                        0
 10                        0
 11                        1

NUMB =     5009    NUMBNZ =     895
MEAN =    0.234    NZMEAN =   1.312
K**2 =    6.558    NZK**2 =   0.350
MAX  =       11    MAXTIM =     141
```

Figure 5.8B

algorithm. Both FF2 and BF2 outperformed the best of the previously modeled memory managers when the changed expansion algorithm was substituted.

It was thought that the new expansion algorithm might benefit either best-fit or first-fit more than the other. No trend could be detected. On Tape 1, the improvement for first-fit was 3% more than for best-fit. On Tape 3, the improvement was 30% more than the improvement for best-fit and actually resulted in first-fit doing less work. However, on Tape 2, the improvement for best-fit was 31% better than for first-fit. The final result was consistent with previous results, best-fit generally requires less compaction but only marginally.

5.3.6 Summary

Certain results of the modeling are clearly evident:

1) The current job load of the system could be managed with much less compaction than current levels.

2) If no available hole is large enough to allow rolling in a new job without expansion, an algorithm which expands the largest hole will perform better than the same algorithm expanding the lowest hole.

3) The largest whole algorithm will even cause a job to be moved fewer times per residence (on the average) than its low hole counterpart.

4) For algorithms which have no bias toward one end or the other of memory, an expansion algorithm which tends to move jobs completely to the end of a hole, will yield better results.

Other results are not as clear. Best-fit seems consistently better than first-fit. However, the margin is always small. This study does not take into account the actual cost of running the algorithms to determine placement. Best-fit is more complex and somewhat slower. The margin of difference between compaction overhead is so small, that the implementation of the first-fit rather than best-fit might well be advisable, since the extra cost of running a best-fit algorithm might not be recoverable from the reduced compaction.

5.4 Memory Utilization

A metric commonly used for the comparison of placement algorithms is memory utilization.

The simulation model satisfies the same stream of memory requests, at the same rate for each algorithm simulated using the same tape. Thus, the measured memory utilization obtained for a given tape is constant and does not change with the algorithm. Different algorithms fragment memory into different numbers of holes of different sizes.

The system function of dominant influence on memory utilization, the job scheduler, was not explicitly modeled in the simulation model. Therefore, the results produced in this study of memory

utilization are intended only as a basis for insight into possible improvement. The method of simulation used does not provide resolution of changes in the memory utilization which might be obtained by the use of different placement algorithms. The data gathered for this study can be used to study one property of the job scheduler. That is, the effects on memory utilization of a scheduler which will skip down the priority list if the top priority job will not fit in available memory.

The current job scheduler for the UT-2D system does not skip. If there is a job which will not currently fit in memory, but which has a higher priority for memory than any other job which is not in memory, the job scheduler will not skip this job and designate a smaller, lower priority job for rollin. This strategy reduces memory utilization if there are smaller jobs which could be allocated memory. A scheduler with skip could achieve higher memory utilization by designating the highest priority job which requires less memory than the total of currently available free memory. Such a scheduler would then designate this job and assign the physical mechanism to allocate the desired memory, performing whatever compaction was necessary.

If this scheduler with skip had more specific knowledge of the configuration of memory, it could use this knowledge to further enhance its decision. The scheduler might have available the knowledge of exactly how many holes were present and what the sizes of these holes were. Then, instead of skipping to the first job which requires less memory than is available, the scheduler could skip to the first job which will fit into an existing hole and increase memory utilization even further. If no job will fit into an existing hole, then a job which would fit if compaction were performed could be selected and the physical mechanism would perform whatever management functions are required.

5.4.1 Distribution of Holes

The job scheduler, in the performing of its scheduling function, does not reschedule jobs each time memory is reconfigured. The memory configuration is transient during the course of compacting memory so that a job may be rolled in. Holes are formed and then reformed because only one job at a time is moved. In studying the steady-state configuration of memory, the existence of these transient holes will be ignored. What is desired is a distribution of the holes avialability to the scheduler. These holes change only on specific occasions:

1) when a job rolls in,
2) when a job rolls out, and
3) when a memory resident job changes its field length but remains resident.

Taken as a whole, these three cases are all occurences of a control point changing field length.

To measure the hole distribution generated by a placement algorithm, measurement of time spent at a given configuration of memory is recorded after every change in a CP's field length. The distribution thus generated is of the holes available to the job scheduler for allocation.

5.4.2 Approximation Technique

The technique employed uses two distributions. The first distribution, the ready job distribution, is described in section 5.2. It supplies a distribution of jobs from which the scheduler might select. Each entry is the probability, over the length of the tape, that J or more jobs in the field length range l are ready for, but not allocated, central memory. This distribution differs from tape to tape but not for different algorithms.

The second distribution is the available space or hole distribution. It supplies the distribution of holes generated by the simulation model using different placement algorithms. Each entry is the probability that there are exactly L holes of size K free in memory (as defined in the previous section). This distribution is different for each algorithm and each tape.

The hole distribution can be used to determine the approximate memory utilization for a given tape. Field length ranges are not infinitely small and so the answers obtained are not exact. Each range is of 1/2K (K=1024) words. To approximate, the mid-point of each range is used as the value for the jobs which were of a given range. To obtain the mean amount of vacant space one uses the formula

$$\bar{V} = \sum_{i}\sum_{j} j * \bar{m}(i) * P_{ij}(H) \qquad (1)$$

where,

$\bar{V}$ = mean amount of vacant space,

$\bar{m}(i)$ = value of midpoint of field length range i,

j = number of holes of range i present and,

$P_{ij}(H)$ = probability of j holes of range i.

Memory utilization is then

$$U = \frac{T - \bar{V}}{T} \qquad (2)$$

where,

U = memory utilization

T = Total available space and,

$\bar{V}$ = mean amount of vacant space.

For the system scheduler and any placement algorithm using the simulation model memory utilization is constant for any given tape.

This portion of the study is an attempt to estimate an upper bound on memory utilization which could be achieved by using a scheduler with skip and knowledge of the available holes in memory. Certain

assumptions are implicit in the order in which operations are performed. These assumptions are made so that the upper bound is maximized.

The hole distributions generated by each of the simulations are used as the distribution of holes available to the scheduler for allocation to jobs. An attempt is made to place the largest jobs first into the smallest available holes of sufficient size. Placing the largest jobs first will maximize the achievable packing density of memory.

The mean amount of vacant space available to the scheduler does not change by changing the placement algorithm. This is due to the method of simulation. The mean amount of vacant space would probably vary in reality. Since this factor is not included in the hole distributions used as available to the scheduler, some detail is lost. The estimates of the upper bound of memory utilization obtainable are useful estimates of improvements over the current system. However, sufficient detail does not exist to allow comparisons between algorithms of the same tape. The mean vacant space in each hole distribution is very nearly constant when the algorithm to estimate the upper bound is performed, and it remains very nearly constant after completion of the program. A description of the algorithm used to model the scheduler's operation and estimate the memory utilization obtained by its use follows.

An iteration is performed over four variables.

1) The jobs are "used" from largest to smallest.

2) The number of jobs of the given range is varied from the maximum number present to the minimum. The job distribution used is a cumulative distribution so $P_{ij}(J)$ represents the probability of i or more jobs of range j present in memory.

3) The holes are "used" in increasing order from j to the largest holes size.

4) The number of holes of a given range is varied from the maximum number of holes of the same size present to the minimum. The hole distribution is not cumulative and so $P_{kl}(H)$ represents the probability of exactly k holes of range l being present.

The four variables are nested so that each succeeding variable is looped over its entire range before its containing loop is changed. The iteration is used to form a set of joint probabilities

$$t = P_{ij}(J) * P_{kl}(H).$$

This is the probability that i jobs of size j are placed in k holes of size l. This implies that the following changes must be made to "update" the distributions:

1) $P_{ij}(J)$ must be reduced by t. The probability of i jobs of size j is reduced by the probability of the indicated event's occurence.

2) $P_{kl}(H)$ must be reduced by t.

Once again, in order that the memory utilization is maximized, it is assumed that the jobs are placed in memory in the most favorable fashion. Thus, as many jobs as possible, are assumed to be placed in a hole. No case is considered where all i jobs cannot be placed in the k holes. In Figure 5.9 is shown an example of the most general case. It can be seen that at most three kinds of holes are formed.

1) Those (m-1) holes left of size "x" from the holes which were filled as much as possible.

2) Those (k-m) holes which were not needed.

3) The 0 or 1 hole left of size nj+x if there was a partially needed hole. Under the assumption that jobs are placed as tightly as possible, no more than one of these holes can exist.

Since these holes are formed, the hole distribution must be "updated" to show this. The probability of m-1 holes of size x,$P_{m-1,x}(H)$ must be increased by t. The probably of k-m holes of size l,$P_{k-m,l}(H)$ must be increased by t and the probability of 1 hole of size nj+x,$P_{1,nj+x}(H)$ must also be increased by t. After all adjustments to the distribution are completed, the next iteration to obtain the next joint probability is initiated.

Upon completion of this iterative procedure, a new distribution of holes remains. These are the holes which remain because the scheduler could not utilize them. The compaction which would arise is included in the original hole distribution from which the iteration process began. Since the effects of both compaction and scheduling

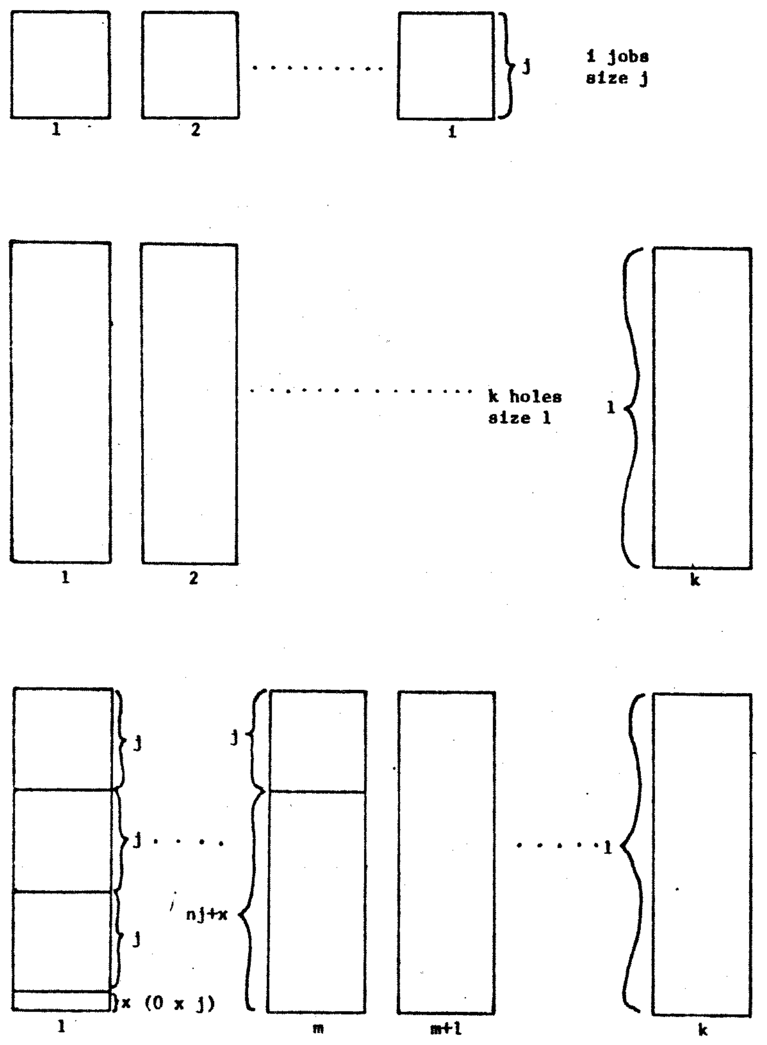


Figure 5.9

with skip including knowledge of hole sizes have been included, the remaining space is the vacant space which could not be used. The mean vacant space for this scheduler is then obtained from Equation (1) and the completely updated hole distribution.

The metric presented will be memory utilization rather than vacant space. Memory utilization for this study is based on that portion of memory which is available to those control points which alter their storage requirements. These include all user CPs and the two system CPs, PISCES and GEMINI. The portion of the memory occupied by the operating system, CMR, and the SATURN control point are not included in the available memory since they are permanently occupying memory and do not change size for a given tape. Memory utilization is thus, the mean amount of the user available memory which is occupied.

5.4.3 Results

The mean memory utilization for the simulations of the system running with different placement algorithms should be constant. However, using Equation (1) to determine these values, a slight error is introduced because the values are measured in widths of a finite size. Equation (1) uses the midpoints of these ranges as the value for the hole sizes. The memory utilizations, therefore, vary slightly.

Table 5.1 presents the memory utilizations obtained by the current system. The workload represented by each of these three

TAPE	1	2	3
FF1	.620	.654	.603
BF1	.618	.654	.603
FF2(old expansion)	.616	.653	.603
BF2	.618	.653	.603
FF2(new expansion)	.618	.653	.603
BF2	.618	.653	.603
MEAN	.618	.653	.603

Table 5.1 Current System Memory Utilization

TAPE	1	2	3
FF1	.797	.893	.914
BF1	.802	.894	.916
FF2(old expansion)	.795	.892	.917
BF2	.799	.893	.919
FF2(new expansion)	.801	.895	.920
BF2	.804	.895	.922
MEAN	.800	.894	.918

Table 5.2 Estimated Upper Bound Of Memory Utilization

tapes varies considerably, but the memory utilization does not. Since the scheduler does not skip, it does not take advantage of the possible improvements of placing smaller, lower priority jobs in memory. Thus, even though Tape 3 is much busier than the other tapes and therefore, has a much richer set of jobs from which to select, the memory utilization is of the same order as the other tapes.

Table 5.2 presents the predicted upper bounds on memory utilization obtained using a scheduler with skip and knowledge of existing hole sizes. The improvements are as expected ranging from 20% to 30%, and this seems entirely plausible. For the scheduler using skip, the memory utilization is much more dependent on workload. The more jobs demanding service, the greater the memory utilization. This is to be expected. When the system is busy, as on Tape 3, the scheduler with skip can usually find a ready job of the correct size for any hole. If no exact fit is available, the scheduler can select the nearest fit and then attempt to fill the smaller remaining hole with a small job. Of course, some holes will occur which simply cannot be filled, but these results indicate that a significant improvement in memory utilization is possible by implementing a scheduler with skip and knowledge of holes.

The tabulations in Table 5.1 are not precise predictions but estimations. There are coupling effects between the job scheduler and the memory management mechanism which are ignored in this simple model.

CHAPTER VI

SUMMARY

The goal of this study was to investigate the impact of placement algorithms on memory management in a non-paged environment. The primary metric of performance by which the impact on the system is determined, is the amount of CPU processing time consumed in the performance of compaction. A determination was made that the explicit simulation of the policy scheduling task was not necessary to achieve the desired results. A model was constructed in which the policy scheduling task of memory management is implicitly performed by accepting the decision made by the system as recorded on the event trace tape. The mechanism level of memory management is explicitly modeled. The model is validated for three sets of data.

Different algorithms for the placement of jobs being rolled into memory are simulated. Results are produced which allow the quantitatively accurate assessment of the impact of the different algorithms. Model results indicated:

1. compaction overhead can be reduced by over 50% from current system levels;

2. algorithms which expand the largest hole, when such an expansion is necessary, perform the best of those tested;
3. for algorithms which expand the largest hole, a minor modification of the current system expansion algorithm produces a further reduction in compaction overhead;
4. for the workload of this system, which is predominantly small jobs (jobs smaller than 15% of available memory), but which also allows jobs as large as 75% of available memory, the difference between the same version of best-fit and first-fit is virtually negligible. Best-fit consistently performs less compaction than first-fit. The difference is merely a fraction of 1% and it is unclear that best-fit would outperform first-fit to a degree sufficient to outweigh the extra overhead of performing the best-fit algorithm.

As a by-product of this study, results are shown which indicate that duration and size of memory requests are not independent.

A secondary result is presented which shows that increasing the flexibility of the policy scheduler to allow it to designate some job other than the top priority job, and giving the scheduler more detailed knowledge of the configuration of memory, can result in large increases in memory utilization. Such a scheduler also increases memory utilization as greater demands are placed on the system.

This work has used a trace-driven simulation model of memory management to investigate the impact of placement strategies on system performance. The metric of performance is the amount of CPU processor time consumed in the task of performing memory compaction. Under the conditions tested, it was shown that significant improvement over the current system was possible. Quantitative results are produced to substantiate these claims. These results were presented as the basis for suggested changes to the UT-2D operating system for the CDC 6000's at the University of Texas at Austin.

APPENDIX

```
 15.35.12. 26 SEP 75   UNIVERSITY OF TEXAS 66/6400   UT 20
PROBABILITY DISTRIBUTION

 READY JOBS

    FROM       TO   1          2          3          4          5

     0          0.5 0          0          0          0          0
     0.5        1   0.00230562 0          0          0          0
     1          1.5 0.00295195 0.00007978 0          0          0
     1.5        2   0.01407448 0.00082402 0          0          0
     2          2.5 0.01690927 0          0          0          0
     2.5        3   0.00005777 0          0          0          0
     3          3.5 0.00609765 0.00038014 0          0          0
     3.5        4   0.00162425 0          0          0          0
     4          4.5 0.04262503 0.17175469 0.22495185 0.15964650 0.31177085
     4.5        5   0.00949304 0.00010890 0          0          0
     5          5.5 0.01469513 0.00001421 0          0          0
     5.5        6   0.00136076 0          0          0          0
     6          6.5 0.00368679 0          0          0          0
     6.5        7   0.00279169 0          0          0          0
     7          7.5 0          0          0          0          0
     7.5        8   0.03627091 0          0          0          0
     8          8.5 0.26447279 0.00520415 0.00038083 0          0
     8.5        9   0.00964275 0          0          0          0
     9          9.5 0.19750586 0.00464953 0          0          0
     9.5       10   0.02077743 0.00003485 0          0          0
    10         10.5 0.00005800 0          0          0          0
    10.5       11   0.02814643 0.00048859 0          0          0
    11         11.5 0.03734302 0.00386654 0          0          0
    11.5       12   0.00032259 0          0          0          0
    12         12.5 0.12149871 0.18341304 0          0          0
    12.5       13   0          0          0          0          0
    13         13.5 0.00101157 0          0          0          0
    13.5       14   0.01482604 0          0          0          0
    14         14.5 0.04029085 0.00421642 0          0          0
    14.5       15   0.40148012 0.00178309 0          0          0
    15         15.5 0          0          0          0          0
    15.5       16   0          0          0          0          0
    16         16.5 0.40120224 0.40708965 0.05898758 0.00569871 0.00986653
    16.5       17   0.11180139 0.01106749 0          0          0
    17         17.5 0.24840107 0.00304252 0          0          0
    17.5       18   0          0          0          0          0
    18         18.5 0.34795945 0.02663251 0          0          0
    18.5       19   0.02746410 0          0          0          0
    19         19.5 0          0          0          0          0
    19.5       20   0          0          0          0          0
    20         20.5 0.05498507 0.09777918 0.38128345 0.34437239 0.11946502
    20.5       21   0          0          0          0          0
    21         21.5 0          0          0          0          0
    21.5       22   0.06586203 0.55452686 0.24741952 0.07634578 0
    22         22.5 0.05169653 0          0          0          0
    22.5       23   0.56824527 0.32584051 0          0          0
    23         23.5 0          0          0          0          0
    23.5       24   0          0          0          0          0
 15.35.12. 26 SEP 75   U\IVERSITY OF TEXAS 66/6400   UT 20
PROBABILITY DISTRIHUTION

    24        24.5 0.64340958 0          0          0          0
    24.5      25   0.30192770 0          0          0          0
    25        25.5 0.41653637 0          0          0          0
    25.5      26   0.85884800 0          0          0          0
    26        26.5 0.59522445 0.02093930 0          0          0
    26.5      27   0          0          0          0          0
    27        27.5 0          0          0          0          0
    27.5      28   0          0          0          0          0
    28        28.5 0.04086840 0          0          0          0
    26.5      29   0          0          0          0          0
    29        29.5 0          0          0          0          0
    29.5      30   0          0          0          0          0
    30        30.5 0          0          0          0          0
    30.5      31   0          0          0          0          0
    31        31.5 0          0          0          0          0
    31.5      32   0          0          0          0          0
    32        32.5 0.03699199 0          0          0          0
    32.5      33   0.00933139 0          0          0          0
 MEAN NUMBER OF READY JOBS = 17.33546
```

```
 15.35.12. 26 SEP 75   UNIVERSITY OF TEXAS 66/6400   UT 20
PROBABILITY DISTRIBUTION

 HOLE SIZES

    FROM      TO   1          2          3          4          5          6          7

    0         0.5 0.05174559 0          0          0          0          0          0
    0.5       1   0.22950991 0.00739537 0          0          0          0          0
    1         1.5 0.05685254 0.00035010 0          0          0          0          0
    1.5       2   0.08052026 0.00312758 0.00009125 0          0          0          0
    2         2.5 0.04565802 0.00019511 0          0          0          0          0
    2.5       3   0.03999095 0.00038702 0          0          0          0          0
    3         3.5 0.05863357 0.00344192 0          0          0          0          0
    3.5       4   0.03565324 0          0          0          0          0          0
    4         4.5 0.14470237 0.01274993 0.00040811 0          0          0          0
    4.5       5   0.03621497 0.00025908 0          0          0          0          0
    5         5.5 0.03643118 0.00024991 0          0          0          0          0
    5.5       6   0.04511441 0.00075455 0          0          0          0          0
    6         6.5 0.02181583 0          0          0          0          0          0
    6.5       7   0.02858000 0.00011922 0          0          0          0          0
    7         7.5 0.02318118 0          0          0          0          0          0
    7.5       8   0.01810404 0.00048767 0          0          0          0          0
    8         8.5 0.03630737 0.00027719 0          0          0          0          0
    8.5       9   0.04079090 0.00058672 0          0          0          0          0
    9         9.5 0.03383598 0          0          0          0          0          0
    9.5      10   0.02634316 0          0          0          0          0          0
   10        10.5 0.02826520 0          0          0          0          0          0
   10.5      11   0.01799605 0          0          0          0          0          0
   11        11.5 0.01583533 0          0          0          0          0          0
   11.5      12   0.01423635 0          0          0          0          0          0
   12        12.5 0.01861831 0          0          0          0          0          0
   12.5      13   0.02570095 0          0          0          0          0          0
   13        13.5 0.01896108 0          0          0          0          0          0
   13.5      14   0.01495674 0          0          0          0          0          0
   14        14.5 0.03630714 0.00003209 0          0          0          0          0
   14.5      15   0.01089003 0.00004447 0          0          0          0          0
   15        15.5 0.01232027 0          0          0          0          0          0
   15.5      16   0.00920437 0          0          0          0          0          0
   16        16.5 0.02166428 0          0          0          0          0          0
   16.5      17   0.01117365 0          0          0          0          0          0
   17        17.5 0.01035764 0          0          0          0          0          0
   17.5      18   0.00925963 0          0          0          0          0          0
   18        18.5 0.01875955 0          0          0          0          0          0
   18.5      19   0.00919406 0          0          0          0          0          0
   19        19.5 0.00999034 0          0          0          0          0          0
   19.5      20   0.00534356 0          0          0          0          0          0
   20        20.5 0.01704019 0          0          0          0          0          0
   20.5      21   0.00382826 0          0          0          0          0          0
   21        21.5 0.00638929 0          0          0          0          0          0
   21.5      22   0.00929998 0          0          0          0          0          0
   22        22.5 0.00577529 0          0          0          0          0          0
   22.5      23   0.00569412 0          0          0          0          0          0
   23        23.5 0.00331123 0          0          0          0          0          0
   23.5      24   0.00450852 0          0          0          0          0          0
```

```
15.35.12. 26 SEP 75   UNIVERSITY OF TEXAS 66/6400   UT 2D
PROBABILITY DISTRIBUTION

      24         24.5 0.00272290 0         0         0         0         0         0
      24.5       25   0.00475706 0         0         0         0         0         0
      25         25.5 0.00581426 0         0         0         0         0         0
      25.5       26   0.00292054 0         0         0         0         0         0
      26         26.5 0.01568195 0         0         0         0         0         0
      26.5       27   0.00559851 0         0         0         0         0         0
      27         27.5 0.00181611 0         0         0         0         0         0
      27.5       28   0.00390002 0         0         0         0         0         0
      28         28.5 0.00305696 0         0         0         0         0         0
      28.5       29   0.00335594 0         0         0         0         0         0
      29         29.5 0.00699940 0         0         0         0         0         0
      29.5       30   0.00154327 0         0         0         0         0         0
      30         30.5 0.03832731 0         0         0         0         0         0
      30.5       31   0.00108471 0         0         0         0         0         0
      31         31.5 0.01163289 0         0         0         0         0         0
      31.5       32   0.00229071 0         0         0         0         0         0
      32         32.5 0.00270823 0         0         0         0         0         0
      32.5       33   0.01179866 0         0         0         0         0         0
      33         33.5 0.00493108 0         0         0         0         0         0
      33.5       34   0.00080018 0         0         0         0         0         0
      34         34.5 0.00015866 0         0         0         0         0         0
      34.5       35   0.03365760 0         0         0         0         0         0
      35         35.5 0.00466535 0         0         0         0         0         0
      35.5       36   0          0         0         0         0         0         0
      36         36.5 0          0         0         0         0         0         0
      36.5       37   0          0         0         0         0         0         0
      37         37.5 0          0         0         0         0         0         0
      37.5       38   0          0         0         0         0         0         0
      38         38.5 0          0         0         0         0         0         0
      38.5       39   0          0         0         0         0         0         0
      39         39.5 0          0         0         0         0         0         0
      39.5       40   0          0         0         0         0         0         0
      40         40.5 0          0         0         0         0         0         0
      40.5       41   0          0         0         0         0         0         0
      41         41.5 0          0         0         0         0         0         0
      41.5       42   0          0         0         0         0         0         0
      42         42.5 0          0         0         0         0         0         0
      42.5       43   0          0         0         0         0         0         0
 MEAN NUMBER OF HOLES = 1.69750
```

BIBLIOGRAPHY

1. Randell, "A Note on Storage Fragmentation and Program Segmentation," CACM, 12, 7 (July 1969), 365-372.

2. Sherman, S., J. C. Brown, and F. Baskett, "Trace Driven Modeling and Analysis of CPU Scheduling in a Multiprogramming System," Proceedings of ACM Workshop on Performance Evaluation, Cambridge, Massachusetts (April 1971), 173-199.

3. Sherman, S., J. H. Howard, and J. C. Browne. "A Comparison of Deadlock Prevention Schemes Using A Trace Driven Model," Proceedings of the 6th Princeton Conference on Information Sciences and Systems, Princeton, New Jersey (March 1972).

4. Sherman, S. W., Trace-Driven Modeling Studies of the Performance of Computer Systems, Computation Center and Department of Computer Sciences TSN-30 (Ph.D. Dissertation), The University of Texas at Austin (August 1972).

5. Browne, J. C., J. Lan, and F. Baskett. "The Interaction of Multiprogramming Job Scheduling and CPU Scheduling," Proceedings FJCC AFIPS (December 1972), 13-21.

6. Brice, R. S., A Study of Feedback Coupled Resource Allocation Policies in a Multiprocessing Computer Environment, Computation Center TSN-35 (Ph.D. Dissertation), The University of Texas at Austin (August 1973).

7. Anderson, J. W., Primitive Process Level Modeling and Simulation of a Multiprocessing Computer System, Department of Computer Sciences TR-32 (Ph.D. Dissertation), The University of Texas at Austin (May 1974).

8. Newell, A., and Shaw, J. C., "Programming the Logic Theory Machine," Proceedings of the Western Joint Computer Conference, 1957.

9. Comfort, W. T., "Multiword List Items," CACM 7,6 (June 1964), 357-362.

10. Knowlton, K. C., "A Fast Storage Allocator," CACM 8,10 (October 1965), 623-625.

11. Knuth, D. E., The Art of Computer Programming, Volume 1, Fundamental Algorithms, Addison-Wesley, Reading, Massachusetts, 1968.

12. Collins, G. O., "Experience in Automatic Storage Allocation," CACM 4,10 (October 1961), 436-440.

13. Margolin, B. H., Parmlee, R. P., and Schatzoff, M., "Analysis of Free-Storage Algorithms," IBM Systems Journal 4 (1971).

14. Fenton, J. S., and Payne, D. W., "Dynamic Storage Allocation of Arbitrary Sized Segments," Proceeding IFIP 74, North-Holland Publishing Company, Amsterdam (1974), 344-348.

15. Batson, A., Shy-Ming Ju, and Wood, D. C., "Measurement of Segment Size," Proceedings of 2nd Symposium on Operating Systems Principles (1969), 25-29.

16. Totschek, R. A., "An Empirical Investigation into the Behavior of the SDC Time-Sharing System," Report SP2191, System Development Corporation, Santa Monica, California.

17. Shore, J. E., "On the External Storage Fragmentation Produced by First-Fit and Best-Fit Allocation Strategies," CACM 18,8 (August 1975), 433-440.

18. Agrawal, A. K., and Bryant, R. M., "Models of Memory Management," Proceedings of 5th Symposium on Operating Systems Principles (1975), 217-222.

19. Thornton, J. E., Design of a Computer--the Control Data 6600, Scott, Foresman, and Company, 1970.

20. Wedel, W. M., An Introduction to UT-2D for Systems Programmers, Computation Center TIMS-7, The University of Texas at Austin (October 1973).

21. Howard, J. H., "A Large-Scale Dual Operating System," Proceedings ACM 1973, Atlanta, Georgia (August 1973), 242-248.

22. Howard, J. H., and Wedel, W. M., The UT-2D Operating System Event Recorder, Computation Center TSN-37, The University of Texas at Austin (February 1974).

23. Madnick, S. E., and Donovan, J. J., Operating Systems, McGraw-Hill Book Company, 1974.