# Long Term Memory Strategies for Solving the Early/Tardy Scheduling Problem

Ross James
Department of Management
University of Canterbury
New Zealand
r.james@mang.canterbury.ac.nz

## Abstract

Long term memory is an effective way of improving the performance of any search heuristic. In this paper we examine different ways of incorporating long term memory strategies into a tabu search for the early/tardy machine scheduling problem. Computational experiments are carried out on a set of test problems and the performance of the different strategies is reported.

## 1. Introduction

Glover first proposed the use of long and intermediate term memory in a search framework when he introduced Tabu Search (TS) [3] [4]. Many researchers using tabu search have only used TS with short term memory and found it to be successful. Those that have also used long and intermediate term memory have found them to produce superior results.

Little research has been done which compares different forms of long and intermediate term memory and how they perform with respect to a specific class of problem. This paper attempts to answer the following questions in the context of a relatively well structured NP-Hard scheduling problem.

1. Do long and intermediate term memories give a significant performance increase in the chosen problem context?
2. What forms of long and intermediate term memory are more effective?
3. How much needs to be recorded in long term memory for it to be effective?
4. Does the starting solution generation procedure affect the search performance?

The scheduling problem used in this research is the distinct due date early/tardy scheduling problem with job dependent earliness and tardiness penalties as described in James and Buchanan [7]. This problem can be formally defined as:

$$\textbf{min} \sum_{i=1}^{n} \left( a_i \left| d_i - c_i \right|^+ + b_i \left| c_i - d_i \right|^+ \right)$$

where:

$n$ = the number of jobs to be scheduled,
$c_i$ = the completion time of job $i$,
$d_i$ = the due date of job $i$,
$a_i$ = the penalty per unit of time when job $i$ is produced early,

$b_i$ = the penalty per unit of time when job $i$ is produced late and

$|x|^+$ = $x$ if $x > 0$, or 0 otherwise.

The problem has several features that do provide some structure to the problem. These features were used by James and Buchanan [7] to develop a heuristic which can generate a schedule solely from specifying whether each job is to be scheduled early or tardy. We use this heuristic to help generate a starting point and also in some of the long-term memory implementations.

## 2. Tabu Search Implementation

The tabu search is implemented by first determining the sequence of jobs to be processed, then inserting idle time in a fashion described by James and Buchanan [7]. The search uses using an insert neighbourhood scheme, whereby one job can be taken and inserted in front of or behind another job. At each move the job that is moved and the position that it was moved from is made tabu. In situations where the move made can be reversed by the movement of a different job, the alternative moves are also made tabu. All searches used a fixed tabu list size. To speed up candidate selection, the candidate selection procedure called Reeves-Window-Candidate List proposed by James and Buchanan [8] was implemented. This procedure was found to be very effective by James and Buchanan [8] in solving this same problem. The aspiration criterion used was that a tabu move would be accepted if it produced a solution that was better than the best solution found to date.

The long and intermediate term memory components were organised so that after every iteration the memory would be able to record information on the current solution to the problem. After a specified amount of time, the long term memory was called to diversify the search by finding a solution which was different from the current solutions examined and became the new starting point for the search. After the same amount of time the intermediate term memory was then called to intensify the search by finding a new starting point which tended to share common features solutions examined. The way in which these solutions were generated was the primary difference between the searches.

The total search time was given, as was the number of long and intermediate term memory interactions during this time. The interactions were evenly spaced throughout the time span. All searches were first diversified, then intensified.

## 3. Memory Implementations

Several different types of memory structures were implemented for the long and intermediate term memories. A couple of “random memories” were also included to see how the quality of the search was affected by the quality of what was held in memory.

The memories could be classified into three broad categories with varying levels of sophistication and memory requirements.

The first is a sequence based frequency memory recorded the number of times a particular job was in a given location in the sequence as explained by Glover and Laguana [5]. Every time the search moved to a new solution the memory was updated with the job position. The memory recorded the total number of times a job was positioned in a given location. The diversification strategy chose a job for a particular

position by finding the job that had been in that position the least number of times. In the case of ties, the jobs were randomly selected among tied jobs. Similarly the intensification strategy chose a job for a particular position by finding the job that had been in that position the highest number of times. Clearly, once a job had been selected for a position, it could no longer be chosen for further positions. Once the sequence had been determined, idle time was inserted to create a schedule.

The second form of memory used a frequency-based memory that recorded the number of times a particular job was scheduled on or before its due date (early) or after its due date (tardy). For the diversification phase, jobs that were normally tardy were scheduled early while those jobs that were normally early were scheduled so they were tardy. This early/tardy information was then feed into the heuristic of James and Buchanan [7] to devise a schedule. As it is possible for a given set of early/tardy job settings to be infeasible due to having too many early jobs, the algorithm would change the early job with the most times early into a late job in order to create a feasible schedule. The inverse procedure was implemented for the intensification strategy.

The third form of memory structure investigated was the most extensive in that it recorded the most information from the search. This memory we called "Total Recall" memory. This memory recorded the sequence of jobs that formed the solution, the objective function value for the solution and the moves that were made from this solution. This memory records every move made during the search, and does not "strategically forget" any moves made during the search as with standard tabu search. Clearly this form of memory requires an efficient method of storing this information, otherwise it could slow down considerably as the number of solutions visited and moves stored increased. Previous researchers have discouraged tabu lists which stores points rather than moves because they "may be extremely space consuming" [6]. Battiti and Tecchiolli [2] showed with their development of Reactive Tabu Search that with the increases in memory available and appropriate memory structures that it was now practical to store all this information in a single memory. In this research a special memory structure was used which comprises of a hash table and a self height balancing trinary (three branch) tree as outlined in Figure 1. Those unfamiliar with these data structures are referred to a computer science book on data structures such as [1]. The advantage with this memory structure is that it does not slow down significantly as the number of iterations increases as long as the hash table is sufficiently large. Solutions were indexed into the hash table and the tree according to their objective function value hence the top best solutions could always be found by scanning the best values in each of the trees pointed to by the hash table. At any time the search had a record of whether it had visited a specific solution or not which, if required, could be incorporated into the search. This memory was used to find the top $x$ solutions which were then used to diversify and intensify the search in the same manner as the frequency based early/tardy memory structure outlined above. A simpler array memory structure of could have been used for this purpose, however this would have been at the expense of speed and flexibility for future diversification and intensification options. This structure also allowed the diversification and intensification algorithms to be integrated with the main search, in that a specific point and move could be checked to see if it have ever been carried out from that point.

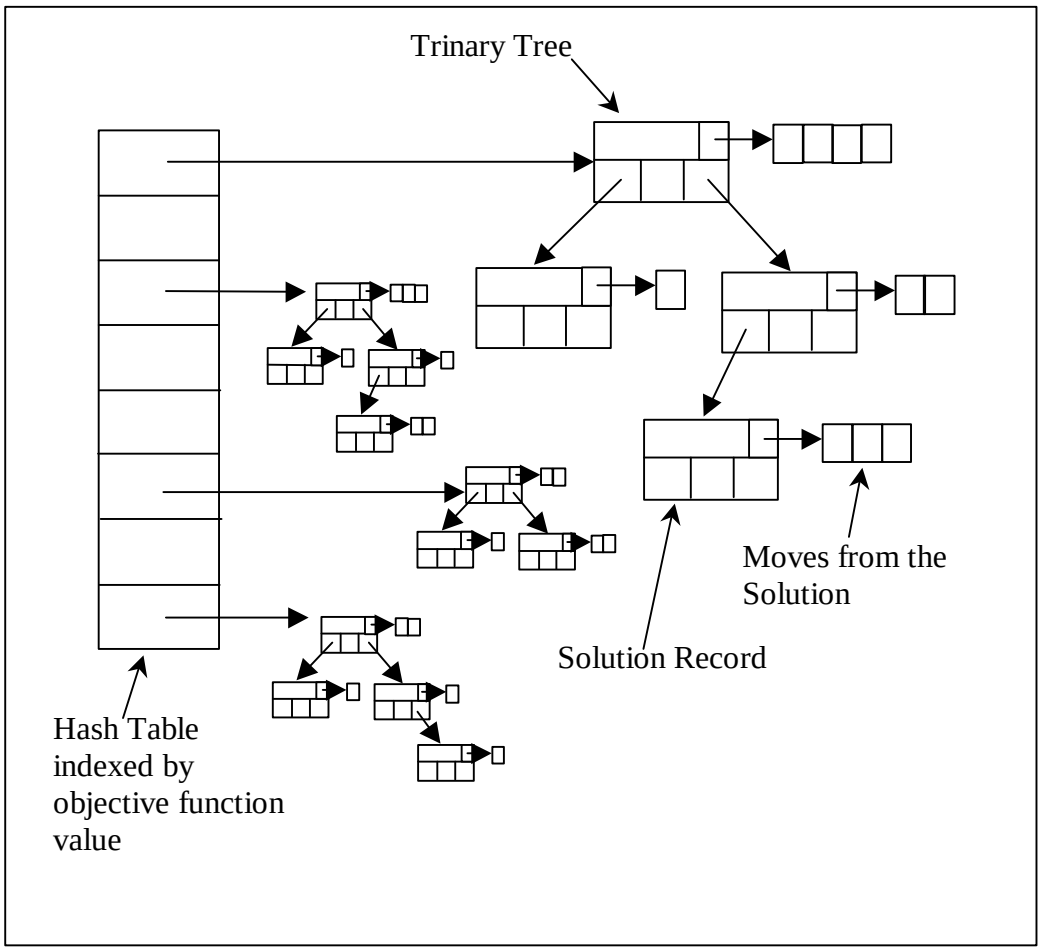


Figure 1. Memory Structure for Total Recall Memory.

## 4. Search Descriptions

Seven different combinations of memories and searches were compared.

A Standard Tabu Search without long or intermediate term memory was initially run to test the hypothesis that there is a difference in the performance of the search if long and intermediate term memory are implemented in the search for this problem.

Random Long Term Frequency Tabu Search (Random LT Freq TS) did not use a memory at all for its intensification and diversification. It simply generated a random sequence every time the search was to be either intensified or diversified. This allowed the hypothesis to be tested that what is stored in sequence frequency memory is important to the performance of the search.

Frequency Based Long Term Tabu Search (Freq LTTS) used a frequency base memory that recorded the number of times a job was in a particular position in the sequence. To intensify the search jobs that were often in a particular position were kept in that position. For diversification jobs that were seldom in a particular position were placed in that position.

Random Early/Tardy Long Term Memory (Random ETLTTS) is similar to Random LT Freq TS in that it does not use any actual memory but allows us to test the hypothesis that what is stored in early/tardy frequency memory does make a difference to the performance of the search. When requested to intensify or diversify the search this search randomly allocates each job to being either an early or a tardy job. The early/tardy heuristic of James and Buchanan [7] is then used to generate a new starting

point. If the allocation is infeasible, then a new set of allocations are made with more probability of a job being set as tardy rather than early.

Early/Tardy Long Term Memory Tabu Search (ETLTTS) records the number of times a job has been scheduled later than its due date. From this the number of times the job has been schedule on or earlier that its due date can be calculated by taking the number of iterations less the number of times it had been scheduled late. This information is used to define whether the job should be defined as being early or tardy by the intensification and diversification routines. Jobs which are normally early would be set early to intensify the search, or late to diversify the search. Once each job had been allocated an early/tardy setting the James and Buchanan [7] early/tardy heuristic can be used to create the new schedule.

Early/Tardy Total Recall Long Term Memory with Tabu Search Candidate Selection (ETTRLTTSCS) is similar to ETLTTS, except that it uses the Total Recall memory structure. This means that it can selectively specify how many of the best top solutions it wants to consider when generating its new starting point. The procedure is exactly the same as ETLTTS except that is considers only the specified number of top solutions. ETLTTS, on the other hand, considers every point visited, whether good or bad.

Early/Tardy Total Recall Long Term Memory Tabu Search (ETTRLTTS) is the same as ETTRLTTSCS except that the information in the Total Recall memory is also incorporated into the search itself. This means that if a move from a solution is found in the Total Recall memory then it is defined as being tabu throughout the search, and can not be carried out again. The tabu list is still used to provide a diversification tool for the search.

## 5. Data Generation

Data was generated in the same manner as described by James and Buchanan [7]. The classes of data used are shown in Table 1.

| | Tardiness Factor | Due Date Range | Early Penalty Range | Tardy Penalty Range |
|---|---|---|---|---|
| 1 | 0.2 | 0.2 | [1,5] | [6,10] |
| 2 | 0.2 | 0.95 | [1,5] | [6,10] |
| 3 | 0.2 | 1.15 | [1,5] | [6,10] |
| 4 | 0.6 | 0.2 | [1,5] | [6,10] |
| 5 | 0.6 | 0.95 | [1,5] | [6,10] |
| 6 | 0.6 | 1.15 | [1,5] | [6,10] |

Table 1. Data classes used.

Each problem contained 50 jobs and 20 instances of each class of problem were generated.

## 6. Computational Results

Each problem was run on a 166Mhz Pentium machine running NT4.0 for 5 minutes. The code was developed in Borland C++ V5.0 which, being object oriented, allowed for most of the main code to be shared between the different “versions” of the search. This

meant that the comparative performance of the searches differs only by the effect of the change made in terms of memory, rather than any other changes which may effect computational speed.

For each search, all the parameters were tuned and only the results that gave the best mean and standard deviation are reported. The parameters reported in Table 2 are in the following order: tabu list size, source window size, destination window size, number of interactions from long and intermediate term memory during the search. For the total recall searches the next two parameters indicate the number of entries to consider in the memory to determine new starting points in diversification, and the number for intensification. The final parameter is the starting fraction to be used by the search if they have to determine whether to classify a job as being early or tardy. In the best searches it was found that if a point was early (or tardy) 80% or more of the time then we classify this as being an "early" (respectively tardy) job. The performance measure is the average percentage deviation from the best-known solution for each problem.

Table 2 shows the searches ordered from worst to best search performance in terms of mean deviation from the best known solution.

| Search | Mean Deviation from Best Known | Standard Deviation |
|---|---|---|
| Tabu Search 7,0.5,0.5 | 2.654% | 3.113% |
| Random LT Freq TS 7,0.5,0.2,5 | 0.209% | 0.573% |
| Freq LTTS 7,0.5,0.2,5 | 0.176% | 0.395% |
| Random ETLTTS 7,0.5,0.2,5,0.8 | 0.168% | 0.454% |
| ETTRLT TSCS 7,0.5,0.2,5,50,50,0.8 | 0.156% | 0.345% |
| ETLTTS 7,0.5,0.2,5,0.8 | 0.143% | 0.321% |
| ETTRLT TS 7,0.5,0.2,5,50,50,0.8 | 0.117% | 0.303% |

Table 2. Search Performance Results

From these results we can draw several conclusions about the issues we were wanting to investigate.

From the computational results it is clear that those searches that incorporate some form of intensification and/or diversification scheme did give a significant performance increase in this problem context. If we compare the standard Tabu Search with a "naïve" multi-restart procedure, Random LT Freq TS, we see that the standard Tabu Search has a mean deviation from best known that is that is more than 12 times that of Random LT Freq TS. When the Random LT Freq TS is compared with the Freq LTTS it would appear that the majority of this gain is actually made by simply restarting the search from a new point rather than carefully selecting the point that it restarts from. This, however, is not to say that selecting the point carefully does not have any impact on the performance of the search, as clearly Freq LTTS does have both a better mean and standard deviation than its random counterpart.

Another interesting comparison is between Random ETLTTS and Freq LTTS. The Random scheme is not using an "knowledge" about the search, but instead uses the knowledge about the problem which is incorporated into the heuristic itself. In this situation the results are very similar although the Random scheme provides a better mean deviation from best know, it does suffer from greater variability than the Freq LT TS. This results shows that there are two dimensions that are equally important when developing a longer term memory. The first is the incorporation of knowledge gained

by the search process itself. The second is the incorporation of problem specific knowledge which can exploit the know properties of the problem being solved.

Note too that all the searches that used the early/tardy memory along with the heuristic performed better than the frequency based searches. This indicates that using a memory structure that records search process to complement a heuristic that is designed to incorporate problem specific knowledge into the solution, rather than a more general scheme which does not provide such knowledge, is likely to produce better results.

The use of different types of memory provided some important insight. If we compare ETTRLTTSCS and ETLTTS we note that the total recall memory does not improve the performance of the search. This implies that being able to perform intensification or diversification on a top band of solutions does not improve the performance of the search. From the mean deviations computed, it appears to be slightly detrimental to the performance of the search, however the difference is not significant enough to draw this conclusion.

Total Recall memory, however, does appear to perform better when we compare ETLTTS and ETTRLTTS. ETTRLTTS incorporates the long-term memory information into the search itself by recognising moves made from specific solutions and making these tabu permanently producing a slightly more diverse search. This has produced results that are slightly better than those searches that do not incorporate their long-term memory information in the main search.

## 7. Conclusion

It is important to note that there is little performance difference between any of the search methods that incorporate some form of restart strategy through the search, whether this is randomly or through the use of a memory and the creation of a more "considered" restarting point. The search performance does seem to improve with better knowledge and memory being incorporated into the restarting procedure. Incorporation of knowledge of the problem type via a heuristic appears to be just as important as knowledge from the search itself. Combining both sources of information appears to increase the search performance.

In terms of our research questions we can draw the following conclusions:

1. Long and intermediate term memory does give a significant performance increase in a specific problem context.
2. Long and intermediate term memories which combine problem specific and search information appear to outperform those memories that just contain one of these elements.
3. The more information that is recorded in long and intermediate term memory does not necessarily mean better performance, although long and intermediate term memory which has been incorporated into the search process itself appears to be a promising avenue for investigation.
4. The starting solution generation procedure used by the longer-term memories does affect the search performance, but not to a significant extent. The largest gains are made when the search is restarted several times from different starting points.

## 8. Future Research

There are many different research opportunities that need to be investigated from this research. These include:

1. Trialling the same memory structures on different types of problems and determining if the same conclusions hold for less structured problems.
2. Exploring the different uses of the total recall memory within the search.
3. Exploring how different starting point generation schemes affect the search performance using, for example, different information from the total recall memory.

## References


[1] R.J. Baron, C.G. Shapiro, *Data Structures and their Implementation*, Van Nostrand Reinhold, New York, (1980), pp 150-165

[2] R. Battiti, G. Tecchiolli, *The Reactive Tabu Search*, ORSA Journal on Computing, 6 (1994), pp 126-140.

[3] F. Glover, *Tabu Search - Part I*, ORSA Journal of Computing, 1 (1989), pp 190-206.

[4] F. Glover, *Tabu Search - Part II*, ORSA Journal of Computing, 2 (1990), pp 4-32.

[5] F. Glover, M. Laguna, *Tabu Search*, Kluwer Academic Publishers (1997)

[6] F. Glover, E. Talliard , D. de Werra, *A Users Guide to Tabu Search*, Annals of Operations Research, 41 (1993), pp 3-28.

[7] R.J.W. James, J.T. Buchanan, *Performance enhancements to tabu search for the early/tardy scheduling problem*, European Journal of Operational Research, 106 (1998), pp 254-265

[8] R.J.W. James, J.T. Buchanan, *A neighbourhood scheme with a compressed solution space for the early/tardy scheduling problem*, European Journal of Operational Research, 102 (1997), pp 513-527