Comparison of mixed- model sequencing in just-in-time production systems

by

Cheng-Kung Wu

A Thesis Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Department: Industrial and Manufacturing Systems Engineering
Major: Industrial Engineering

: University
, Iowa

1994

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

This study compares the performance of several algorithms used in mixed model sequencing. Mixed model sequencing is said to occur when more than one type of final product is produced and some combination of parts are used for more than one product. The procedure for designing a mixed-model assembly line involves the following steps (Monden 1993):

1. Determination of a cycle time.
2. Computation of a minimum number of processes.
3. Preparation of a diagram of integrated precedence relationships among elemental jobs.
4. Line balancing.
5. Determination of the sequence schedule for introducing various products to the line.
6. Determination of the length of the operations range of each process.

This study deals with the fifth step: The problem of sequencing various products models on the line. The algorithms studied include Goal Chasing I and II, Time Spread, Quick and Dirty and Miltenburg Algorithm 1 and 3. The performance measures include goal chasing D, weighted goal chasing D, objective functions of time spread, variance and computational effort required. Twenty-three problem sets with different demand patterns and eleven part structures are used to test the algorithms.

Just in time (JIT) is a total system philosophy that was developed and promoted by Toyota Motor Corporation. It has gained much attention from both researchers and industrial practitioners. JIT requires the production of only the necessary items in the necessary

quantities at the necessary time to keep in process inventory and its fluctuation to a minimum. It may not be overstating that JIT would be another revolutionary production management system that follows Taylor's scientific management and Ford's mass assembly system (Monden 1993). The primary goal of the JIT system is cost reduction through elimination of waste. Waste is defined as any surplus over the minimum amount of anything needed for production. To achieve this primary goal, three subgoals must be met, which are quantity control, quality assurance and respect of humanity.

There are several key concepts and requirements to implement the JIT philosophy, such as the pull system, utilization of Kanbans, setup reduction, new definition of vendor relationship, leveling production, etc. Among these techniques, mixed model sequencing is often applied to produce many different products without carrying large inventories (Miltenburg 1989). This technique helps many companies to maintain diversified small-lot production to satisfy customers' demands for a variety of products, without holding large inventories. Mixed model sequencing is said to occur when more than one type of final product is produced and some types of parts are used for more than one product (Joo and Wilhelm 1993). The complexity of the mixed model sequencing problem stems from the large number of final products to be sequenced and the differing usage patterns of various parts by each final product.

The difficulty of mixed model sequencing is more acute in an assembly line setting. An assembly line is essentially a production system where fabricated parts and assemblies are installed, or other work done, on units of a main product as they move form one station to another. The issue is complicated because the performance of the assembly line is influenced by the order in which products are sequenced during each day, and it makes the traditional assembly line balancing problem more complex.

Monden points out that the procedure to design a mixed-model assembly line is to determine the cycle time, compute a minimum number of processes, prepare a diagram of integrated precedence relationships among elemental jobs, balance the line, determine the sequence schedule for introducing various products to the line. Particularly, the mixed model sequencing is targeted at the fifth step: sequence to introduce the models into the assembly line.

Due to the different goals and purposes of controlling the assembly line, there are different sequences of introducing models. Two major goals suggested by Monden (1993) are:

1) Leveling the load (total assembly time) on each process within the line, and
2) Keeping a constant speed in consuming each part on the line

The first goal recognizes that all products do not have the same operation time at each work station on the line, and some workstations might have a longer than average cycle time. The second goal attempts to keep the quantity of each part used by the mixed-model assembly line per unit time constant. In Toyota, the second goal is considered to be more important and critical in a JIT production system.

The difficulty imposed by mixed model sequencing has prompted some research. Different sequencing algorithms have been proposed and developed to meet different requirements and objectives. This study surveys and classifies some existing mixed model sequencing algorithms, which include Goal Chasing I and II, Time Spread, Quick and Dirty sequencing, and Miltenburg's algorithm 1 and 3. Goal Chasing I and II, Time Spread and Miltenburg's third algorithm are selected because some existing literature has compared their results and concluded that they perform reasonably well. The Quick and Dirty algorithm is newly developed and is added to the comparison. Although Miltenburg's algorithm 1 does not always provide a feasible sequence, it does give a quick performance lower bound against

which other algorithms can be compared. Miltenburg's algorithm 2 is not included in the study. Although it guarantees feasibility, it is too slow to be considered for any practical purposes. The performance of these algorithms is compared based on several measurements, such as deviation from constant part consumption rate, total variation of the actual production from the desired production, ability to smooth the load on each workstation on an assembly line and the computational effort needed.

The following section reviews a collection of existing literature in mixed model sequencing. Section 2.2 gives a brief review of the algorithms under study. The measure of performance and problem sets used to evaluate the algorithms are presented in Section 3. Section 4 shows the results and finally in Section 5, concluding remarks are made.

2. LITERATURE REVIEW

2.1 Related Review

The importance of the order in which models are assembled on a mixed-model line has been recognized by practitioners and academics alike. Monden (1993) presented a goal chasing method, the first analytical scheduling model used in Toyota production. He also introduced a simplified algorithm, Goal Chasing II. Both algorithms try to keep a constant part consumption rate on the line.

Miltenburg (1989) extended the Goal Chasing algorithm and developed three additional algorithms. His first algorithm does not guarantee a feasible sequence but quickly provides a lower bound on performance. Inman and Bulfin (1992) proposed a quick and dirty sequencing method which only considers the due date of the final products. Sumichrast, et al. (1992) developed the Time Spread method which takes into consideration the processing time at each workstation.

Groeflin, et al. (1989) proposed an algorithm for sequencing the final assembly of highly customized end products. Their algorithm seeks to minimize the usage variability of component parts by giving priority to the parts with the largest variability among different models. Lee and Kim (1988) combined MRP with JIT to reduce work in process. They extended goal chasing and apply it in an environment with short delivery distance and low risk of accident during delivery. Kubiak and Sethi (1991) converted the mixed model sequencing problem into an assignment problem and claimed that an optimal solution can be found.

Because of the numerous algorithms proposed, literature exists which reports on the comparison of the performance of the algorithms. Sumichrast and Russell (1990) compared

Goal Chasing algorithms and Miltenburg's algorithms based on model usage rate and component usage rate. Miltenburg's algorithm 3, heuristic 2 produced the highest quality feasible solutions under all conditions tested. Both goal chasing algorithms were also compared to the mixed integer programming model, and Goal Chasing I was found to be more desirable.

Sumichrast, et al. (1992) compared the two Goal Chasing algorithms, Miltenburg's algorithm 3 and the Time Spread method, according to four measures of assembly line in efficiency, work not completed, worker idleness, worker station time and a measure of variability in uniform component usage using a simulation model. They conclude that Time Spread and Miltenburg's algorithm 3, heuristic 2 are most effective. The former is a better choice when the main concern is assembly efficiency (such factors as product quality and worker flexibility) and the latter is the preferred selection when fabrication efficiency (i.e. uniform parts usage) is more important.

This research further examines the two Goal Chasing algorithms, Time Spread, Quick and Dirty and Miltenburg's algorithms. These seven algorithms are reviewed in the next section.

2.2. Review of Mixed Model Sequencing Algorithms

Goal Chasing I and II focus on the objective of maintaining a constant usage of parts. Miltenburg's algorithms are designed to achieve uniform production rates for each model and thus indirectly achieve uniform parts usage rate. Time Spread is designed to smooth the work load at each station on the line to achieve efficient assembly. The Quick and Dirty algorithm is a newly developed algorithm which provides a sequence by using a simple and highly

efficient procedure based on minimizing the deviation between cumulative production and cumulative demand. These algorithms are discussed briefly in the following section.

2.2.1. Goal Chasing I

Goal Chasing I method was developed at Toyota and reported by Monden (1983). The objective of Goal Chasing I is to keep the actual usage rate and the ideal usage rate as close as possible. To achieve this goal, the withdrawal rate of each part must be held as constant as possible. This condition is described as follows:

$$\text{Min } D_k = \text{Min} \sqrt{\sum_{j=1}^{b} \left(\frac{KN_j}{Q} - X_{jk} \right)^2}$$

where Q = total production quantity of all products A_i ($i = 1,..., \alpha$)

$= \sum_{i=1}^{\alpha} Q_i$, (Q_i = production quantity of each product A_i)

N_j = Total necessary quantity of part a_j to be consumed for producing all products A_i ($i = 1,..., \alpha$, $j = 1,..., \beta$)

X_{jk} = Total necessary quantity of part a_j to be utilized for producing the products of determined sequence from first to Kth

The first item in the equation, KN_j/Q, represents the quantity of part a_j required to assemble the first k items in the sequence given a constant usage for a specific partial sequence. The squared differences between KN_j/Q and $X_{j,k}$ are summed over all component parts to provide a measure of non-uniform usage. Position k in the sequence is filled by the model which minimizes this measure. As the sequence is constructed, at each step the 'distance' between the expected usage of components and actual usage is minimized. Goal Chasing I assumes

that each time slot represents a time unit during which an end-product is assembled and that assembly time is the same for all types of products.

Procedure

Step 1 Initialization: let $k = 1$; $X_{j,0} = 0$, $j = 1,2 \ldots, \beta$; $S = 1,2,\ldots, \alpha$; and $M = \varnothing$.

Repeat the following steps until $S = \varnothing$.

Step 2 For $i \in S$ calculate D_{ki} such that :

$$D_{ki} = \text{Min} \sqrt{\sum_{j=1}^{b} \left(\frac{KN_j}{Q} - X_{j,k-1} - b_{ij} \right)^2}$$

where b_{ij} = necessary quantity of the part a_j, $j = 1, \ldots, b$, for producing one unit of product A_i, $i = 1, \ldots, a$

Select to produce r such that $D_{kr} = \text{Min} \{D_{ki}\}$, $i \in S$.

Append r to the scheduled list M.

Step 3 Set $Q_r = Q_r - 1$. If $Q_r = 0$, set $S = S - \{r\}$.

Step 4 If $S = \varnothing$, then stop; otherwise, set $X_{jk} = X_{j,k-1} + b_{rj}$, $j = 1,\ldots, \beta$, and $k = k+1$.

Numerical Example (Monden 1993)

Suppose the production quantities Q_i ($i = 1, 2, 3$) of each product A_1, A_2, and A_3, and the required unit b_{ij} ($i = 1, 2, 3$; $j = 1, 2, 3, 4$) of each part a_1, a_2, a_3, and a_4 for producing these products are as shown in Table 1.

Table 1 Production Quantities Q_i and Parts Condition b_{ij}.

	Product A_t	A_1	A_2	A_3	
	Planned Production Quantity Q_i	2	3	5	
		Parts a_i			
		a_1	a_2	a_3	a_4
Product A_i	A_1	1	0	1	1
	A_2	1	1	0	1
	A_3	0	1	1	0

Then, the total necessary quantity (N_j) of part a_j ($j = 1, 2, 3, 4$) for producing all products A_i ($i = 1, 2, 3$) can be computed as follows:

$$[N_j] = [Q_i][b_{ij}]$$
$$= [2, 3, 5]\begin{bmatrix}1011\\1101\\0110\end{bmatrix} = [5, 8, 7, 5]$$

Further, the total production quantity of all products A_i ($i = 1, 2, 3$) will be:

$$\sum_{i=1}^{3} Q_i = 2 + 3 + 5 = 10$$

Therefore,

$$[N_j/Q] = [5/10, 8/10, 7/10, 5/10]$$
$$(j = 1, 2, 3, 4)$$

Next, applying the values of $[N_j/Q]$ and $[b_{ij}]$ to the formula in step 2 of the above algorithm, when $K = 1$, the distances D_{ki} can be computed as follows:

$$\text{for } i = 1,\ D_{1,1} = \sqrt{(\frac{1\times5}{10}-0-1)^2+(\frac{1\times8}{10}-0-0)^2+(\frac{1\times7}{10}-0-1)^2+(\frac{1\times5}{10}-0-1)^2}$$
$$= 1.11.$$

for i = 2, $D_{1,2}$ =

$$\sqrt{(\frac{1\times5}{10}-0-1)^2+(\frac{1\times8}{10}-0-1)^2+(\frac{1\times7}{10}-0-0)^2+(\frac{1\times5}{10}-0-1)^2}$$

= 1.01

for i = 1, $D_{1,3}$ =

$$\sqrt{(\frac{1\times5}{10}-0-0)^2+(\frac{1\times8}{10}-0-1)^2+(\frac{1\times7}{10}-0-1)^2+(\frac{1\times5}{10}-0-0)^2}$$

= 0.79

Thus, D_{1,i^*} = min {1.11, 1.01, 0.79} = 0.79

$\therefore r^* = 3$

New Q_r (r = 3) = 5-1 = 4 ≠ 0

Therefore, the first order in the sequence schedule is the product A_3. Proceeding to Step 4 of the algorithm,

$X_{jk} = X_{j,\,k-1} + b_{3j}$:

$X_{1,1} = 0 + 0 = 0$ $\qquad$ $X_{3,1} = 0 + 1 = 1$

$X_{2,1} = 0 + 1 = 1$ $\qquad$ $X_{4,1} = 0 + 0 = 0$

Thus, the first line in Table 2 was written based on the above computations.

Similarly, when k = 2 , then

for i = 1, $D_{2,1}$ =

$$\sqrt{(\frac{2\times5}{10}-0-1)^2+(\frac{2\times8}{10}-1-0)^2+(\frac{2\times7}{10}-1-1)^2+(\frac{2\times5}{10}-0-1)^2}$$

= 0.85.

for i = 2, $D_{2,2}$ =

$$\sqrt{(\frac{2\times5}{10}-0-1)^2+(\frac{2\times8}{10}-1-1)^2+(\frac{2\times7}{10}-1-0)^2+(\frac{2\times5}{10}-0-1)^2}$$

= 0.57.

for i = 1, $D_{2,3}$ =

$$\sqrt{(\frac{2\times5}{10}-0-0)^2+(\frac{2\times8}{10}-1-1)^2+(\frac{2\times7}{10}-1-1)^2+(\frac{2\times5}{10}-0-0)^2}$$

= 1.59.

Thus, $D_{2,i^*} = \min\{0.85, 0.57, 1.59\} = 0.57$

$\therefore i^* = 2$

Therefore, the second order in the sequence schedule is the product A_2. Also, X_{jk} will be computed as:

$X_{jk} = X_{j,\,k-1} + b_{2j}$:

$X_{1,1} = 0 + 1 = 1$ $\quad$ $X_{3,1} = 1 + 0 = 1$

$X_{2,1} = 1 + 1 = 2$ $\quad$ $X_{4,1} = 0 + 1 = 1$

This procedure was used to developed the second line of Table 2. The remaining lines in Table 2 can also be written by following the same procedures. As a result, the complete sequence schedule of this example will be:

$A_3A_2A_1A_3A_2A_3A_3A_1A_2A_3$

Table 2 Sequence Schedule of Goal Chasing I Example

K	D_{k1}	D_{k2}	D_{k3}	Sequence Schedule	X_{1k}	X_{2k}	X_{3k}	X_{4k}
1	1.11	1.01	0.79*	A_3	0	1	1	0
2	0.85	0.57*	1.59	A_3A_2	1	2	1	1
3	0.82*	1.44	0.93	$A_3A_2A_1$	2	2	2	2
4	1.87	1.64	0.28*	$A_3A_2A_1A_3$	2	3	3	2
5	1.32	0.87*	0.87	$A_3A_2A_1A_3A_2$	3	4	3	3
6	1.64	1.87	0.28*	$A_3A_2A_1A_3A_2A_3$	3	5	4	3
7	0.93	1.21	0.82*	$A_3A_2A_1A_3A_2A_3A_3$	3	6	5	3
8	0.57*	0.85	1.59	$A_3A_2A_1A_3A_2A_3A_3A_1$	4	6	6	4
9	1.56	0.77*	1.01	$A_3A_2A_1A_3A_2A_3A_3A_1A_2$	5	7	6	5
10	-----	-----	0*	$A_3A_2A_1A_3A_2A_3A_3A_1A_2A_3$	5	8	7	5

* Indicates smallest distance D_{kj}.

2.2.2 Goal Chasing II

This method is a simplified version of Goal Chasing I, whose goal is to keep a constant speed in the utilization of each part on the mixed model assembly line. A product is assigned to position k in the assembly sequence that will

$$\text{Maximize} \sum_{j_i \in B_j} \left(\frac{KN_{ji}}{Q} - X_{j,k-1} \right)$$

where

B_j is a set of constituent parts a_{ij} for the product A_i

i.e., Goal Chasing II selects the model requiring those parts that most need to catch up to their average usage rate. The notations used are the same as those in Goal Chasing I.

To avoid introducing successively the same product requiring a longer operation time, all automobiles on the line are classified according to large (a_l), or medium (a_m), or small (a_s) total assembly times. Each a_j (j = 1, m, and s in the situation) must be introduced to the line so as to keep the speed constant on the line. This goal can be achieved by using the same simplified algorithm for keeping the speed constant of utilizing each part a_j on the line. (Monden 1993)

Procedure

Step 1 Initialization: let k = 1; $X_{j,0} = 0$, j = 1,2 ..., β, S = 1,2,..., α, and M= $\varnothing$.

Repeat the following steps until S = $\varnothing$.

Step 2 For i $\in$ S calculate E_{ki} such that :

$$E_{ki} = \sum_{j_i \in B_j} \left(\frac{KN_{ji}}{Q} - X_{j,k-1} \right)$$

Select product r such that $E_{kr} = \text{Max}\ \{E_{ki}\}$, i $\in$ S.

Append r to the scheduled list M.

Step 3 Set $Q_r = Q_r - 1$. If $Q_r = 0$, set $S = S - \{r\}$.

Step 4 If $S = \varnothing$, then stop; otherwise, set $X_{jk} = X_{j,k-1} + b_{rj}$, $j = 1,..., \beta$, and $k = k+1$.

Numerical Example (Monden 1993)

Production quantities Q_i and part condition b_{ij} are referred to Table 1.

$B_j = \{ j \mid b_{ij} \neq 0 \}$

$\therefore\ B_1 = \{ 1, 3, 4 \}$ $B_2 = \{ 1, 2, 4 \}$ $B_3 = \{ 2, 3, 4 \}$

for i = 1, $E_{1,1}$ =

$$\left(\frac{1\times5}{10}-0\right)+\left(\frac{1\times7}{10}-0\right)+\left(\frac{1\times5}{10}-0\right)$$

=1.7

for i = 2, $E_{1,2}$ =

$$\left(\frac{1\times5}{10}-0\right)+\left(\frac{1\times8}{10}-0\right)+\left(\frac{1\times5}{10}-0\right)$$

=1.8

for i = 1, $E_{1,3}$ =

$$\left(\frac{1\times8}{10}-0\right)+\left(\frac{1\times7}{10}-0\right)$$

=1.5

Thus, $E_{1,i^*} = \max \{1.7, 1.8, 1.5\} = 2$

$\therefore i^* = 2$

Therefore, the first order in the sequence schedule is product A_2.

$X_{jk} = X_{j,\,k-1} + b_{2j}$:

$X_{1,1} = 0 + 1 = 1$ $\qquad$ $X_{3,1} = 0 + 0 = 0$

$X_{2,1} = 0 + 1 = 1$ $\qquad$ $X_{4,1} = 0 + 1 = 1$

Thus, the first line in Table 3 was written based on the above computations.

Next, when k = 2, then

for i = 2, $E_{2,1}$ =

$$\left(\frac{2\times5}{10}-1\right)+\left(\frac{2\times7}{10}-0\right)+\left(\frac{2\times5}{10}-1\right)$$

$=1.4$

for i = 2, $E_{2,2}$ =

$$\left(\frac{2\times5}{10}-1\right)+\left(\frac{2\times8}{10}-1\right)+\left(\frac{2\times5}{10}-1\right)$$

$= 0.6$

for i = 1, $E_{2,3}$ =

$$\left(\frac{2\times8}{10}-1\right)+\left(\frac{2\times7}{10}-0\right)$$

$= 2$

Table 3 Sequence schedule of Goal Chasing II Example

K	E_{k1}	E_{k2}	E_{k3}	Sequence Schedule	X_{1k}	X_{2k}	X_{3k}	X_{4k}
1	1.7	1.8*	1.5	A_2	1	1	0	1
2	1.4	0.6	2*	A_2A_3	1	2	1	1
3	2.1*	1.4	1.5	$A_2A_3A_1$	2	2	2	2
4	0.8	1.2	2*	$A_2A_3A_1A_3$	2	3	3	2
5	1.5	2*	1.5	$A_2A_3A_1A_3A_2$	3	4	3	3
6	1.2	0.8	2*	$A_2A_3A_1A_3A_2A_3$	3	5	4	3
7	1.9*	1.6	1.5	$A_2A_3A_1A_3A_2A_3A_1$	4	5	5	4
8	0.6	1.4	2.0*	$A_2A_3A_1A_3A_2A_3A_1A_3$	4	6	6	4
9	1.3	2.2*	1.5	$A_2A_3A_1A_3A_2A_3A_1A_3A_2$	5	7	6	5
10	1.0	1.0	2.0*	$A_2A_3A_1A_3A_2A_3A_1A_3A_2A_3$	5	8	7	5

* Indicates smallest distance E_{kj}.

Thus, $E_{2,i^*} = \max \{1.4, 0.6, 2\} = 2$

$\therefore i^* = 3$

Therefore, the second order in the sequence schedule is product A_3. Also, X_{jk} will be computed as:

$X_{jk} = X_{j,\, k-1} + b_{3j}$:

$X_{1,2} = 1 + 0 = 1$ $\qquad$ $X_{3,2} = 0 + 1 = 1$

$X_{2,2} = 1 + 1 = 2$ $\qquad$ $X_{4,2} = 1 + 0 = 1$

This procedure was used to developed the second line of Table 3. The remaining lines in Table 3 can be determined by following the same procedures. As a result, the complete sequence schedule of this example will be:

$A_2A_3A_1A_3A_2A_3A_1A_3A_2A_3$

2.2.3 Time Spread

Time Spread was proposed by Sumichrast et al. (1992) to smooth the work load at each station on the line to achieve efficient assembly. The mixed model assembly line is assumed to be balanced so that each station can at least produce the average amount of work without requiring additional, non-assigned workers. The structure of this function is similar to that in Goal Chasing I except that the individual terms representing quantities of parts and products in Goal Chasing I have been replaced by their individual corresponding assembly (process) times. A product is assigned to position k in the sequence that will

$$\text{Min} \sum_{l=1}^{s} \left(\frac{kT_l}{T} - AT_{l(k-1)} - t_{il} \right)^2$$

where

$AT_{l(k-1)}$ = the actual time required at station l to assemble the first k-1 units of the sequence

s = the number of assembly stations

t_{il} = the assembly time required by product i at station l

T = the total time to assemble all parts in the sequence at all stations

T_l = the total time to assemble all parts in the sequence at station l

b_{ij} = the quantity of components j required to assemble one end item i

N_i = the total demand quantity of product i

Procedure

Step 1 Calculate the time to assemble all items at station i,

$T_j = b_{ij} \times N_i$, $i = 1,2,..., \alpha$,

$j = 1,..., \beta$.

Calculate the sum of the assembly time at each station, $T = \sum_{j=1}^{\beta} T_j$

Step 2 Initialization: let $k = 1$; $X_{j,0} = 0$, $j = 1,2 ..., \beta$; $S = 1,2,..., \alpha$; and $M = \varnothing$.

Repeat the following steps until $S = \varnothing$.

Step 3 For $i \in S$ calculate T_{ki} such that :

$$T_{ki} = \text{Min} \sum_{l=1}^{s} \left(\frac{kT_l}{T} - AT_{l(k-1)} - t_{il} \right)^2$$

Select to produce r such that $T_{kr} = \text{Min} \{T_{ki}\}$ $i \in S$.

Append r to the scheduled list M.

Step 4 Set $Q_r = Q_r - 1$. If $Q_r = 0$, set $S = S - \{r\}$.

Step 5 If $S = \varnothing$, then stop; otherwise, set $X_{jk} = X_{j,k-1} + b_{rj}$, $j = 1,..., \beta$, and $k = k+1$.

Numerical Example (Sumichrast et al. 1992)

Using the data shown in Table 4, the time to assemble all items at station 1 is computed as follows:

$$T_1 = (3)(4) + (2)(5) + (2)(4) = 30$$

Similarly, for the remaining stations:

$$T_2 = (3)(2) + (2)(3) + (2)(6) = 24$$

$$T_3 = (3)(0) + (2)(2) + (2)(1) = 6$$

$$T_4 = (3)(5) + (2)(2) + (2)(0) = 19$$

Total time is the sum of the assembly time at each station.

Table 4 Assembly Times by Station and Demand

	Assembly time by station				
Products	1	2	3	4	Demand
1	4	2	0	5	3
2	5	3	2	2	2
3	4	6	1	0	2

$$T = 30 + 24 + 6 + 19 = 79$$

T_{ki} is now calculated for each model. Since this will determine the model to sequence in position 1; k = 1 and $AT_{l(k-1)} = 0$ for each station, *l*.

$$\left(1\times\frac{30}{79}-0-4\right)^2+\left(1\times\frac{24}{79}-0-2\right)^2+\left(1\times\frac{6}{79}-0-0\right)^2+\left(1\times\frac{19}{79}-0-5\right)^2=38.64$$

The objective function T_{ki} is calculated for product 2 and 3 in the same way.

Stage 2		
Objective function value by product		
1	2	3
38.64	35.41	46.46

Method TS would select product 2 as the first one in the assembly sequence since it minimizes the objective function.

The procedure is followed again as the second position in the sequence is filled. Now k = 2 and $AT_{l(k-1)}$ is

Actual time required by station before stage 2			
1	2	3	4
5	3	2	2

T_{ki} is recomputed for each product. For product 1, its value is

$$\left(2\times\frac{30}{79}-5-4\right)^2+\left(2\times\frac{24}{79}-3-2\right)^2+\left(2\times\frac{6}{79}-2-0\right)^2+\left(2\times\frac{19}{79}-2-5\right)^2=133.11$$

Performing this calculation for all products gives the following results.

Stage 1		
Objective function value by product		
1	2	3
133.11	141.66	148.76

Product 1 minimizes the objective function at this stage and is therefore selected.

Continuing in this manner. TS selects the following product sequence.

Position	1	2	3	4	5	6	7
Product	2	1	3	1	2	1	3

2.2.4. Quick and Dirty

Quick and Dirty algorithm was developed by Inman and Bulfin (1991) to provide a new formulation and solution procedure to sequence a mixed model just-in-time (JIT) assembly system. The purpose of the algorithm is to identify an alternative and equally approximate pair of objectives for which an optimal schedule may be determined by a simple and highly efficient procedure. The problem may be interpreted as a single-machine scheduling problem with each unit of product treated as a separate job, where t_{ik} is defined as the due date of job of product i in stage k. The problem becomes significantly more difficult when the processing times are different. The objectives can be expressed as:

$$\text{Min} \sum_{i=1}^{\alpha} \sum_{t=1}^{Q} (X_{it} - d_{it})^2 \qquad \text{or} \qquad \text{Min} \sum_{i=1}^{\alpha} \sum_{t=1}^{Q} \left| X_{it} - d_{it} \right|$$

where

X_{ik} denotes the time at which *k*th unit of product i is actually produced

Q: total demand ($=\Sigma Q_i$, Q_i is production quantity of each product A_i)

d_{it} : the cumulative demand for product i at time t

The algorithm assumes that the unit production times of all items are identical and without loss of generality, equal to one time unit. This implies that $Q = \Sigma Q_j$. Also, there are no setups when switching production from one product to another. The cumulative demand for product i at time t equals $d_{it} = tQ_i/(\Sigma Q_i)$. The purpose is to minimize the deviation between cumulative production and cumulative demand. Let t_{ik} denote the time at which the kth unit of product i is needed, which will be used as its due date. Then

$$t_{ik} = \frac{[(K - 1/2)T]}{Q_i} \quad i = 1,..., \alpha; \qquad k = 1,..., Q_i,$$

where T: the total demand of all products

Procedure

Step 1 Calculate every value of $t_{ik} = \frac{[(k-1/2)T]}{Q_i}$

Step 2 Sort the due date by ascending order.

Ties can be broken up arbitrarily. However, scheduling the product with the largest demand seems to generate good schedules.

Numerical Example (Inman and Bulfin 1991)

n = 3 product types with demands 6, 6 and 1 units. There are a total 13 jobs to be assigned to 13 positions. Table 5 gives the due dates according to equation in step 1.

Ordering these jobs results in the optimal schedule 2-1-2-1-2-1-3-1-2-1-2-1-2.

2.2.5. Miltenburg's Algorithm

Miltenburg (1989) proposed several algorithms to solve the mixed model sequencing problem. The purpose of the algorithms are to develop a theoretical basis for scheduling mixed model assembly lines in JIT production systems, with minimum variation in model production rate. It was assumed that all products require approximately the same number and mix of parts. Using the following notation:

Q_i = the production quantity of each product A_i

$Q = \sum_{i=1}^{\alpha} Q_i$

k: stage number

$m_{i,k}$: the nearest integer point to the k points,

m_k: scheduled products in stage k,

$x_{i,k} = kr_i$,

$r_i = Qi/Q$: the ratio of the production quantity of product A_i to the total demand.

Table 5 Due Dates of Quick and Dirty Example

Product i	Unit k	Due Date		t_{ik}
1	1	13/12	=	1.083
	2	39/12	=	3.250
	3	65/12	=	5.417
	4	91/12	=	7.583
	5	117/12	=	9.750
	6	143/12	=	11.917
2	1	13/12	=	1.083
	2	39/12	=	3.250
	3	65/12	=	5.417
	4	91/12	=	7.583
	5	117/12	=	9.750
	6	143/12	=	11.917
3	1	13/2	=	6.500

2.2.5.1. Algorithm 1

In the first algorithm developed by Miltenburg, the sequence produced is not necessarily feasible. At each stage of mixed model sequencing the total production should be increased by one. However, algorithm 1 occasionally selects to produce two items and then destroys one. There is no potential for practical use due to the infeasibility of the sequence, but it can quickly create a lower bound for variability against which other schedules may be compared. The procedure can be described by using the following notation:

Procedure

Point $X = (Q_1, Q_2, \ldots, Q_n) \in R^n$ where $\sum_{i=1}^{n} m_i = \sum_{i=1}^{n} x_i = k$. And the nearest integer. The following algorithm finds nearest integer point $M = (m_1, m_2, \ldots, m_n) \in Z^n$ to a point in the production schedule.

Step 1 Calculate $k = \sum_{i=1}^{\alpha} Q_i$.

Step 2 Find the nearest non-negative integer m_i to each coordinate Q_{ij}, that is, find m_i such that $\left|m_i - Q_i\right| \leq \frac{1}{2}$, i = 1, 2,..., a.

Step 3 Calculate $k_m = \sum_{i=1}^{\alpha} m_i$

Step 4 (a) If $k - k_m = 0$ stop . The nearest integer point is $M = (m_1, m_2, \ldots, m_n)$

(b) If $k - k_m > 0$, go to step 5.

(c) If $k - k_m < 0$, go to step 6.

Step 5 Find the coordinate Q_i with the smallest $m_i - Q_i$.

Increment the value of this m_i; $m_i \rightarrow m_i + 1$. Go to step 3.

Step 6 Find the coordinate x_i with the largest $m_i - x_i$.

Decrement the value of this m_i; $m_i \rightarrow m_i - 1$. Go to step 3.

Step 7 Stop when all stages are scheduled.

Numerical Example (Miltenburg 1989)

There are n = 3 products with D = (6, 6, 1) to be assembled on a mixed-model assembly line. Hence the vector of demand ratios, is r = (6/13, 6/13, 1/13). Algorithm 1 gives the schedule shown in Table 6.

The optimal production over 5 stages is (2, 2, 1) while the optimal production over 6 stages is (3, 3, 0). During the sixth stage one unit of product 1 and one unit of product 2 must be produced while one unit of product 3 must be destroyed. Of course this is impossible. Only one product can be assembled during a stage and products assembled earlier cannot be destroyed.

2.2.5.2. Algorithm 3 heuristic 1

This is an algorithm that is fast enough to solve problems approaching a realistic size and which guarantees a feasible solution. This heuristic starts by scheduling using Algorithm 1 and then revises the sequence produced to obtain feasibility. The choosing criteria is very similar to Goal Chasing II when all models use the same parts.

Procedure

Step 1 Use algorithm 1 to schedule a sequence.

Step 2 For the infeasible schedule determined in Step 1, find the first (or next) stage l where $m_{i,l} - m_{i,l-1} < 0$.

Step 3 Set ∂ = number of product i, for which $m_{i,l} - m_{i,l-1} < 0$ and beginning at stage $l-\partial$

Step 4 Schedule stages $l-\partial$, $l-\partial+1$,..., $l+w$ by selecting the product with the lowest $x_{i,k-1} - kr_i$; where $w \geq 0$.

($l+w$ is the first stage where the schedule determined by the heuristic matches the schedule determined in step 1)

Table 6 Example Data of Miltenburg's Algorithm 1

Stage k	X_k			M_k	Product Scheduled	$\sum_{i=1}^{n}(m_{i,k}-x_{i,k})^2$	Total Variation
1	6/13	6/13	1/13	1 0 0	1	0.5088	0.5088
2	12/13	12/13	2/13	1 1 0	2	0.0355	0.5444
3	18/13	18/13	3/13	2 1 0	1	0.5799	1.1243
4	24/13	24/13	4/13	2 2 0	2	0.1420	1.2663
5	30/13	30/13	5/13	2 2 1	3	0.5680	1.8343
6	36/13	36/13	6/13	3 3 0	1, 2, -3	0.3195	2.1538
7	42/13	42/13	7/13	3 3 1	3	0.3195	2.4734

Step 5 Repeat steps 2, 3 and 4 for other stages where $m_{i,k} - m_{i,k-1} < 0$ or until the m_k values are the same as those obtained in algorithm 1. Then stop.

Numerical Example (Miltenburg 1989)

There are n = 7 products with demands $d_1 = d_2 = d_3 = d_4 = 25$ and $d_5 = d_6 = d_7 = 4$. Hence $r1 = r_2 = r_3 = r_4 = 25/112$ and $r_5 = r_6 = r_7 = 4/112$.

The schedule in Table 7 was obtained by using Algorithm 1. For each stage k, a point X_k is calculated (where $x_{ik} = kr_i$). The nearest integer point is calculated (M_k-initial). If $\sum_{i=1}^{n} mi \neq k$ the point is adjusted to give M_k-final. The schedule for stage 1 to 10 is 1-2-3-4-5-(1, 2, -5)-3-4-5-6 giving a total variation 8.078. Unfortunately this is not a feasible schedule. In stage 6 one unit of product 5 is destroyed ($m_{5,6} - m_{5,6} = 0 - 1 < 0$) while both product 1 and product 2 are scheduled. Suppose we use Heuristic 1 to calculate an entire schedule (see Table 8). For each stage k, $x_{i,k-1} - kr_i$, i = 1, 2, ..., n, is calculated and the product i with the smallest value is scheduled. Ties are arbitrarily broken. (Note that $x_{i,0} = 0$, for all i.) The schedule is 1-2-3-4-5-1-2-3-4-6 for 10 stages. The total variation is 9.953 which is considerably more than Algorithm 1's schedule. This is not the optimal schedule.

In Tables 9 and 10, Algorithm 3 with Heuristic 1 is used to determine a schedule. It begins by using Algorithm 1. Stage 6 is found to be infeasible and so Heuristic 1 is used to reschedule stages 5, 6, ..., until a stage is reached where M_k matches the original schedule. As we see in the second part of Table 9, stages 5 to 9 are scheduled with Heuristic 1. At stage 9, M_9 from Heuristic 1 matches the original schedule - namely 2 2 2 2 1 0 0. The final schedule is 1-2-3-4-5-1-2-3-4-6 (shown in Table 11). Notice that this schedule is the same as the schedule produced by Heuristic 1 alone.

Table 7 Scheduling Example Using Miltenburg Algorithm1

Stage k	$X_k = (x_1, x_2, ..., x_7)$							Nearest $Mk = (m_1, m_2, ..., m_7)$ Initial	Final	Product Scheduled	Var.	Total Var.
1	0.223	0.223	0.223	0.223	0.036	0.036	0.036	0000000	1000000	1	0.757	0.757
2	0.446	0.446	0.446	0.446	0.072	0.072	0.072	0000000	1100000	2	1.027	1.784
3	0.670	0.670	0.670	0.670	0.107	0.107	0.107	1111000	1110000	3	0.810	2.594
4	0.893	0.893	0.893	0.893	0.143	0.143	0.143	1111000	1111000	4	0.107	2.701
5	1.116	1.116	1.116	1.116	0.179	0.179	0.179	1111000	1111100	5	0.792	3.493
6	1.339	1.339	1.339	1.339	0.214	0.214	0.214	1111000	2211000	1 2 -5**	1.241	4.734
7	1.563	1.563	1.563	1.563	0.250	0.250	0.250	2222000	2221000	3	1.078	5.812
8	1.786	1.786	1.786	1.786	0.286	0.286	0.286	2222000	2222000	4	0.429	6.241
9	2.009	2.009	2.009	2.009	0.321	0.321	0.321	2222000	2222100	5	0.667	6.908
10	2.232	2.232	2.232	2.232	0.357	0.357	0.357	2222000	2222110	6	1.170	8.078

Table 8 Scheduling Example Using Miltenburg Algorithm 3 with Heuristic 1 (I)

Stage k	$x_{1,k-1}$-kr_1	$x_{2,k-1}$-kr_2	$x_{3,k-1}$-kr_3	$x_{4,k-1}$-kr_4	$x_{5,k-1}$-kr_5	$x_{6,k-1}$-kr_6	$x_{7,k-1}$-kr_7	Product Scheduled	variation	Total Variation
1	-0.223	-0.223	-0.223	-0.223	-0.036	-0.036	-0.036	1	0.757	0.757
2	0.554	0.554	0.554	0.554	-0.071	-0.071	-0.071	2	1.027	1.784
3	0.330	0.330	0.330	0.330	-0.107	-0.107	-0.107	3	0.810	2.594
4	0.107	0.107	0.107	0.107	-0.143	-0.143	-0.143	4	0.107	2.701
5	-0.116	-0.116	-0.116	-0.116	-0.176	-0.176	-0.176	5	0.792	3.493
6	-0.339	-0.339	-0.339	-0.339	0.786	-0.214	-0.214	1	1.491	4.984
7	0.437	0.437	0.437	0.437	0.750	-0.250	-0.250	2	1.703	6.687
8	0.214	0.214	0.214	0.214	0.714	-0.286	-0.286	3	1.429	8.116
9	-0.009	-0.009	-0.009	-0.009	0.676	-0.312	-0.312	4	0.667	8.783
10	-0.232	-0.232	-0.232	-0.232	0.642	-0.357	-0.357	6	1.170	9.953

Table 9 Scheduling Example Using Miltenburg Algorithm 3 with Heuristic 1 (II)

(From stage l-∂, ∂ = number of product i, for which $m_{i,l} - m_{i,l-1} < 0$)

Main Procedure - Algorithm 1

Algorithm 1 (from Table 3.5.2.1)			Heuristic 1			
Stage k	M_k	Product Scheduled	M_k	Product Scheduled	Variation	Total Variation
1	1000000	1			0.757	0.757
2	1100000	2			1.027	1.784
3	1110000	3			0.810	2.594
4	1111000	4			0.107	2.701
5	1111100	5	1111100	5	0.792	3.493
6	2211000	1 2 -5**	2111100	1	1.491	4.984
7	2221000	3	2211100	2	1.703	6.687
8	2222000	4	2221100	3	1.429	8.116
9	2222100	5	2222100	4	0.667	8.783
10	2222110	6			1.170	9.953

** Infeasible. Using Heuristic 1 to reschedule stages 5, 6, ...

Table 10 Subprogram - Correcting Infeasibilities Using Miltenburg Algorithm 3 with Heuristic 1

Stage k	$x_{1,k-1}$-kr_1	$x_{2,k-1}$-kr_2	$x_{3,k-1}$-kr_3	$x_{4,k-1}$-kr_4	$x_{5,k-1}$-kr_5	$x_{6,k-1}$-kr_6	$x_{7,k-1}$-kr_7	Product Scheduled	M_k-Final	variation	Total Variation
4									1111000		2.701
5	-0.116	-0.116	-0.116	-0.116	-0.176	-0.176	-0.176	5	1111100	0.792	3.493
6	-0.339	-0.339	-0.339	-0.339	0.786	-0.214	-.214	1	2111100	1.491	4.984
7	0.437	-0.562	-0.562	-0.562	0.750	-0.250	-0.250	2	2211100	1/703	6.687
8	0.214	0.214	-0.786	-0.786	0.714	-0.286	-0.286	3	2221100	1.429	8.116
9	-0.009	-0.009	-0.009	-1.009	0.676	-0.321	-0.321	4	2222100	0.667	8.783

Table 11 Schedule Produced by Miltenburg Algorithm 3 with Heuristic 1

Stage, k	1 2 3 4 5 6 7 8 9 10
Algorithm 1	1 2 3 4 6
Heuristic 1	5 1 2 3 4

2.2.5.3. Algorithm 3 heuristic 2

Miltenburg claimed that Heuristic 2 improves the performance from Heuristic 1 but it takes more effort computationally . Heuristic 2 also begins by creating a trial schedule and revises it to obtain feasibility. It uses the same criteria as Heuristic 1 except a product is selected to be produced which minimizes the approximate variation over stages k and k+1.

Procedure

Step 1 Use algorithm 1 to schedule a sequence.

Step 2 For the infeasible schedule determined in Step 1, find the first (or next) stage *l* where $m_{i,l} - m_{i,l-1} < 0$.

Step 3 Set h =1.

Step 4 Tentatively schedule product h to be produced in stage k.

Calculate the variation for stage k and call it $V1_h$.

Step 5 Schedule the product i with the lowest $Q_{i,k}-(k+1)r_i$ for stage k+1. It is the same decision rule as Heuristic 1

Calculate the variation for stage k+1 and call it $V2_h$.

Calculate $V_h = V1_h + V2_h$.

Step 6 Increment h; h → h+1.

If h > n go to Step 7, otherwise go to Step 3. (n is the number of products.)

Step 7 Schedule the product h with the lowest V_h.

Step 8 Stop when all m_k values are the same as those obtained in algorithm 1.

Numerical Example (Miltenburg 1989)

Tables 12 and 13 shows how Heuristic 2 would be used with Algorithm 3. In the first section Algorithm 1 is used to calculate the k nearest integer points M_k. Stage 6 is found to be infeasible. The algorithm backtracks to stage 5, and uses Heuristic 2 to schedule stages 5, 6, ..., until a stage is reached where M_k matches the original schedule. For the problem stages 5 and 6 are scheduled with Heuristic 2. At stage 6 the schedule produced by Heuristic 2 matches the original schedule-namely 2 2 1 1 0 0 0.

The next part of the table shows the detailed calculations for Heuristic 2. First stage k = 5 is considered. The first row shows that the variation in stage 5 when product 1 is scheduled is V1 = 0.917. Given this schedule for the current stage, product 2 should be scheduled for the next stage because product 2 has the lowest $x_{i,k} - (k+1)r_i$. If product 2 is scheduled for stage 6, the variation in stage 6 is V2 = 1.241. Hence the variation over stages 5 and 6, when product 1 is scheduled for stage 5, is approximately 0.917 + 1.241 = 2.158. When this procedure is repeated for each product we see that it is best to schedule product 1 for stage 5 because V1 + V2 is the lowest.

Schedule:

Stage, k	1 2 3 4 5 6 7 8 9 10
Algorithm 1	1 2 3 4 3 4 5 6
Heuristic 1	1 2

Table 12 Scheduling Example Using Algorithm 3 with Heuristic 2

Main Procedure - Algorithm 1						
Algorithm 1 (from Table 7)			Heuristic 2			
Stage k	M_k	Product Scheduled	M_k	Product Scheduled	Variation	Total Variation
1	1000000	1			0.757	0.757
2	1100000	2			1.027	1.784
3	1110000	3			0.810	2.594
4	1111000	4			0.107	2.701
5	1111100	5	2111000	1	0.917	3.618
6	2211000	1 2 -5**	2211000	2	1.241	4.859
7	2221000	3			1.078	5.937
8	2222000	4			0.429	6.366
9	2222100	5			0.667	7.033
10	2222110	6			1.170	8.203

** Infeasible. Using Heuristic 2 to reschedule stages 5, 6, ...

Table 13 Subprogram - Correcting Infeasibilities Using MA3H2

	Stage k		Stage k+1						
Stage k	Product	V1	Lowest $X_{i\,k}$-$(k+1)r_i$	Product	V2	V1+V2	Product Scheduled	Variation	Total Variation
5	1	0.917	-0.339	2	1.214	2.158	1	0.917	3.618
	2	0.917	-0.339	1	1.214	2.158			
	3	0.917	-0.339	1	1.214	2.158			
	4	0.917	-0.339	1	1.214	2.158			
	5	0.792	-0.339	1	1.491	2.283			
	6	0.792	-0.339	1	1.491	2.283			
	7	0.792	-0.339	1	1.491	2.283			
6	1	3.241	-0.562	2	3.078	6.319			
	2	1.241	-0.562	3	1.078	2.319	2	1.241	4.859
	3	1.241	-0.562	2	1.078	2.319			
	4	1.241	-0.562	2	1.078	2.319			
	5	1.491	-0.562	2	1.703	3.194			
	6	1.491	-0.562	2	1.703	3.194			
	7	1.491	-0.562	2	1.701	3.194			

* $V1 = \sum_{i=1}^{n}(X_{i,k} - kr_i)^2$ $\qquad V2 = \sum_{i=1}^{n}(X_{i,k+1} - (k+1)r_i)^2$

** MA3H2: Miltenburg Algorithm 3 with Heuristic 2

2.3. Summary of Algorithms

The objective function and the criterion considered by each algorithm are summarized and presented in the following sections.

2.3.1. Objective Function

Table 14 is a summary of the objective function used by each algorithm.

Table 14 Summary of Objective Function

Algorithms	Objective Function
Goal Chasing I (GC1)	$\min \sqrt{\sum_{j=1}^{\beta} \left(\frac{KN_j}{Q} - X_{jk} \right)^2}$
Goal Chasing II (GC2)	$\max \sum_{j_i \in B_j} \left(\frac{KN_{ji}}{Q} - X_{ji,k-1} \right)$
Time Spread (TS)	$\min \sum_{l=1}^{s} \left(\frac{kT_l}{T} - AT_{l(k-l)} - t_{il} \right)^2$
Quick and Dirty (Q&D)	$\min \sum_{i=1}^{\alpha} \sum_{t=1}^{Q} (X_{it} - d_{it})^2$
Miltenburg Algorithm 1 (MA1)	$\min \sum_{k=1}^{Q} \sum_{i=1}^{\alpha} (x_{i,k} - kr_i)^2$
Miltenburg Algorithm 3 Heuristic 1(MA3H1)	$\min \sum_{k=1}^{Q} \sum_{i=1}^{\alpha} (x_{i,k} - kr_i)^2$
Miltenburg Algorithm 3 Heuristic 2(MA3H2)	$\min \sum_{k=1}^{Q} \sum_{i=1}^{\alpha} (x_{i,k} - kr_i)^2$

2.3.2. Criterion Considered

Table 15 summarizes some of the specific criterion considered by each algorithm in this study. Both Goal Chasing I and II consider the part structure. Time Spread explicitly considers processing time at each workstation. Quick and Dirty algorithm considers due date. All algorithms except Miltenburg's Algorithm 1 consider the feasibility of the sequence generated. Miltenburg's Algorithm 1 does provide a quick lower bound of variability. In addition, when the algorithm creates a feasible sequence, it performs relatively well when measured against the performance discussed.

Table 15 Criterion Considered by Each Algorithm

	GC 1	GC 2	Time Spread	Q&D	MA1	MA3H1	MA3H2
Part Structure	√	√					
Processing Time			√				
Due Date				√			
Feasibility	√	√	√	√		√	√

3. EVALUATION

In this section, the measures of performance used to compare the algorithms are discussed, which are Goal Chasing D, Weighted Goal Chasing D, objective function of Time Spread, variance and the computational effort required. The problem sets used in this study are presented. The problem sets consist of twenty-three different demand patterns and eleven different product structures.

3.1. Measure of Performance

The measure of performance considered in this study are primarily the objective function of each algorithm. They are selected to test different aspects and robustness of the algorithms, such as the ability to maintain a constant part consumption rate, minimize model usage variance, etc.

3.1.1. Goal Chasing D

The first performance measure used in this study is the ability to maintain a constant speed in consuming each part on the line. This is similar to minimizing the deviation, D, of the number of each part actually consumed to the ideal value. This is an important evaluation criteria especially in the just in time environment utilizing a pull system. Under the pull system, the variation in production quantities or conveyance times at preceding processes must be minimized. As a result, the quantity used for each part (consumption rate) in the line must be kept as constant as possible. The sequences produced by various algorithms are substituted into the Goal Chasing I calculation and

$$D = \sqrt{\sum_{j=1}^{\beta} \left(\frac{KN_j}{Q} - X_j\right)^2}$$

is calculated for each product. The D's corresponding to the products schedule to be produced are summed up. This sum is then used as the measure of performance.

3.1.2. Weighted Goal Chasing D

This measure is similar to Goal Chasing D, except a different weight is assigned to each part. Goal Chasing D implicitly assumes each part is of the same importance. By assigning a different weight, the importance of different parts can be considered. In this study, the weight is calculated as the ratio of the number of parts to the total number of parts, i.e. heaviest weight is assigned to a part which is required by most products. Similarly, the sequences produced by each algorithm are subject to calculation.

To evaluate these algorithms, the mean of the total average X_j for part a_j is computed as

$$X_j = \sum_{k=1}^{Q} \left\| \frac{KN_j}{Q} - X_{jk} \right\| \text{ for each part } a_j$$

The ratios of the part number to total part number are used as the weights, i.e.

$$W_j = \frac{N_j}{N}$$

$$D_W = \sum_{j=1}^{\beta} (W_j \times X_j), \text{ for each method.}$$

At each stage, the weighted D_W values corresponding to the scheduled products are summed up.

3.1.3. Objective Function of Time Spread

This performance measure evaluates the ability of the algorithms to prevent a continuous stream of unfinished work from the station. Under this measure, the processing time at each workstation is taken into consideration. Again, all the sequences generated by each algorithm are put into the Time Spread calculation,

$$D_{ts} = \sum_{k=1}^{Q} T_k$$

where D_{ts} = the measure value obtained by using the objective function of Time Spread.

T_k = the minimum value obtained by using the Time Spread algorithm at each stage

Q = the total demand

3.1.4. Variance

This measure of performance evaluates the variance, i.e., how close the proportion of product i produced (over a time period) to the total production r_i (for all time periods), where $r_i = Q_i/Q$. Similarly, all sequences obtained by each algorithm are subject to variance calculation. At each stage, the variance, defined as:

$$\sum_{i=1}^{\alpha} (m_{i,k} - x_{i,k})^2$$

is calculated and summed up. The total variance thus obtained is then used as the performance measure.

3.1.5. Computational Effort

Most real life production line problems involve a minimum of thousands of products and parts, thus an efficient algorithm in terms of computational complexity is a very important consideration for the algorithm to have any practical usage. Some mathematical programming

techniques such as linear programming or mixed integer programming can guarantee optimality but are often too slow to be considered for any real world applications. Therefore, most algorithms are heuristic in nature and require different amounts of computational effort.

When scheduling large problems, most good algorithms will produce a short sequence and simply repeat this short sequence a number of times to meet the production demand. In practice, this can be done by taking the largest common divisor among the product demand rates. This can greatly reduce the computational effort while creating feasible sequences.

3.2. Problem Sets

The problem sets used in this study are designed to test the algorithm on their versatility against different demand patterns (i.e. uniformity of product demands) and different part structures (i.e. uniformity of part demands).

3.2.1. Demand Patterns

This project studies and compares the seven algorithms based on the problem sets shown in Table 16, 17 and 18. The problem sets are designed to test the mixed model sequences against different product demands. All problem sets have the same total demand of 20 products. Demand patterns with 5, 10 and 15 different models of products are considered to test these algorithms. The coefficient of variation (standard deviation divided by the mean) is used as an attribute of each problem set. Problem set 1 to problem set 6 have 5 models of products and whose coefficients of variations shift gradually from 2.2 to 0. Problem set 7 to problem set 14 have 10 models of products and whose coefficients of variations shift gradually from 3.2 to 0. Problem set 15 to problem set 23 have 15 models of products and whose coefficients of variations shift gradually from 3.9 to 0.4.

Table 16 Product Demand Pattern (5 kinds of products)

Problem Set	Product 1	2	3	4	5	Coefficient of Variation
PS 1	20	0	0	0	0	2.2
PS 2	18	1	1	0	0	2.0
PS 3	14	5	1	0	0	1.5
PS 4	11	3	3	2	1	1
Ps 5	6	6	4	2	2	0.5
PS 6	4	4	4	4	4	0

Table 17 Product Demand Pattern (10 kinds of products)

Problem Set	Product 1	2	3	4	5	6	7	8	9	10	C.V.
PS 7	20	0	0	0	0	0	0	0	0	0	3.2
PS 8	19	1	0	0	0	0	0	0	0	0	3.0
PS 9	16	2	1	1	0	0	0	0	0	0	2.5
PS 10	12	6	1	1	0	0	0	0	0	0	2.0
PS 11	11	1	1	1	1	1	1	1	1	1	1.6
PS 12	7	4	2	1	1	1	1	1	1	1	1.0
PS 13	3	3	3	3	2	2	1	1	1	1	0.5
PS 14	2	2	2	2	2	2	2	2	2	2	0

** C.V. : Coefficient of Variation

Table 18 Product Demand Pattern (15 kinds of products)

Problem Set	Product 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	C.V.
PS 15	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3.9
PS 16	18	1	1	0	0	0	0	0	0	0	0	0	0	0	0	3.5
PS 17	16	1	1	1	1	0	0	0	0	0	0	0	0	0	0	3.1
PS 18	13	2	1	1	1	1	1	0	0	0	0	0	0	0	0	2.5
PS 19	10	5	1	1	1	1	1	0	0	0	0	0	0	0	0	2.0
PS 20	8	2	1	1	1	1	1	1	1	1	1	1	0	0	0	1.4
PS 21	6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1.0
PS 22	3	3	2	1	1	1	1	1	1	1	1	1	1	1	1	0.5
PS 23	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	0.4

** C.V. : Coefficient of Variation

3.2.2. Part Structures

Eleven different product structures (as shown in Table 19 to Table 29) are designed to test the algorithms robustness against part usage variation. Part structures a and e are simple structures such that all products require a similar number of parts and similar number of parts requires by all products. Part structures b, f and i are moderate structures such that each part is required at most by 2 products and every demand is only 1. Part structures c, g and j are another kind of moderate structure that each part is required at most by 2 products but each demand is not necessarily only 1. Part structures d, h and k are complex structures that have larger variance among each part requirement, where each product requires a very different number of parts.

In this study, it is assumed that each part will take one unit of time to be assembled, and each workstation only assembles one part. Therefore, the total assembly time at each workstation is equal to the total number of parts required.

Table 19 Part Structure a

		Part									
		1	2	3	4	5	6	7	8	9	10
Products	1	0	0	0	0	1	1	0	0	0	0
	2	0	0	0	1	0	0	1	0	0	0
	3	0	0	1	0	0	0	0	1	0	0
	4	0	1	0	0	0	0	0	0	1	0
	5	1	0	0	0	0	0	0	0	0	1

Table 20 Part Structure b

		Part									
		1	2	3	4	5	6	7	8	9	10
Products	1	0	0	0	0	0	1	0	0	0	0
	2	0	1	1	0	0	0	1	0	0	0
	3	1	0	1	0	0	0	1	1	1	0
	4	0	0	0	1	0	0	0	0	1	0
	5	0	0	0	0	0	0	0	0	0	1

Table 21 Part Structure c

		Part									
		1	2	3	4	5	6	7	8	9	10
Products	1	0	0	0	0	0	7	0	0	0	0
	2	0	3	6	0	0	0	6	0	0	0
	3	4	0	2	0	0	0	5	8	9	0
	4	0	0	0	5	0	0	0	0	4	0
	5	0	0	0	0	0	0	0	0	0	7

Table 22 Part Structure d

		Part									
		1	2	3	4	5	6	7	8	9	10
Products	1	3	2	5	5	0	1	9	3	2	2
	2	2	1	3	1	8	0	1	0	4	3
	3	1	0	6	2	3	4	4	2	5	0
	4	4	2	1	8	0	0	7	3	0	4
	5	7	3	2	0	5	3	3	6	4	0

Table 23 Part Structure e

		Part									
		1	2	3	4	5	6	7	8	9	10
Products	1	1	0	0	0	0	0	0	0	0	0
	2	0	1	0	0	0	0	0	0	0	0
	3	0	0	1	0	0	0	0	0	0	0
	4	0	0	0	1	0	0	0	0	0	0
	5	0	0	0	0	1	0	0	0	0	0
	6	0	0	0	0	0	1	0	0	0	0
	7	0	0	0	0	0	0	1	0	0	0
	8	0	0	0	0	0	0	0	1	0	0
	9	0	0	0	0	0	0	0	0	1	0
	10	0	0	0	0	0	0	0	0	0	1

Table 24 Part Structure f

		Part									
		1	2	3	4	5	6	7	8	9	10
Products	1	0	0	0	0	1	0	0	0	0	0
	2	0	1	1	0	0	0	1	0	0	0
	3	1	0	0	0	0	0	1	1	1	0
	4	0	0	0	1	0	0	0	0	1	0
	5	0	0	0	0	0	0	0	0	0	1
	6	1	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	1	0	0	0	0	0
	8	0	0	0	0	0	0	0	0	0	1
	9	0	0	0	1	0	0	0	0	0	0
	10	0	1	0	0	0	0	0	0	0	0

Table 25 Part Structure g

		Part									
		1	2	3	4	5	6	7	8	9	10
	1	0	0	0	0	5	0	0	0	0	0
	2	0	4	6	0	0	0	7	0	0	0
	3	8	0	0	0	0	0	9	2	8	0
Products	4	0	0	0	9	0	0	0	0	3	0
	5	0	0	0	0	0	0	0	0	0	4
	6	3	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	5	0	0	0	0	0
	8	0	0	0	0	0	0	0	0	0	3
	9	0	0	0	6	0	0	0	0	0	0
	10	0	7	0	0	0	0	0	0	0	0

Table 26 Part Structure h

		Part									
		1	2	3	4	5	6	7	8	9	10
	1	3	2	5	5	0	1	9	3	2	2
	2	2	1	3	1	8	0	1	0	4	3
	3	1	0	6	2	3	4	4	2	5	0
Products	4	4	2	1	8	0	0	7	3	0	4
	5	7	3	2	0	5	3	3	6	4	0
	6	0	2	4	2	3	1	0	4	3	6
	7	2	3	2	1	3	2	8	4	5	0
	8	3	4	6	7	8	9	0	2	1	1
	9	3	3	1	1	0	2	3	4	8	9
	10	2	1	3	0	6	4	3	0	0	5

Table 27 Part Structure i

		Part									
		1	2	3	4	5	6	7	8	9	10
	1	0	0	0	0	1	0	0	0	0	0
	2	0	1	1	0	0	0	1	0	0	0
	3	1	0	0	0	0	0	1	1	1	0
Products	4	0	0	0	1	0	0	0	0	1	0
	5	0	0	0	0	0	0	0	0	0	1
	6	1	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	1	0	0	0	0	0
	8	0	0	0	0	0	0	0	0	0	1
	9	0	0	0	1	0	0	0	0	0	0
	10	0	1	0	0	0	0	0	0	0	0
	11	1	0	0	0	0	0	0	0	0	0
	12	0	0	0	0	0	1	0	0	0	0
	13	0	0	0	0	0	1	0	0	0	0
	14	0	0	0	0	0	0	0	1	0	0
	15	0	0	1	0	0	0	0	0	0	0

Table 28 Part Structure j

		Part									
		1	2	3	4	5	6	7	8	9	10
	1	0	0	0	0	6	0	0	0	0	0
	2	0	4	5	0	0	0	7	0	0	0
	3	4	0	0	0	0	0	8	7	5	0
Products	4	0	0	0	9	0	0	0	0	8	0
	5	0	0	0	0	0	0	0	0	0	6
	6	4	0	0	0	0	0	0	0	0	0
	7	0	0	0	0	3	0	0	0	0	0
	8	0	0	0	0	0	0	0	0	0	2
	9	0	0	0	4	0	0	0	0	0	0
	10	0	8	0	0	0	0	0	0	0	0
	11	2	0	0	0	0	0	0	0	0	0
	12	0	0	0	0	0	7	0	0	0	0
	13	0	0	0	0	0	3	0	0	0	0
	14	0	0	0	0	0	0	0	8	0	0
	15	0	0	9	0	0	0	0	0	0	0

Table 29 Part Structure k

		Part									
		1	2	3	4	5	6	7	8	9	10
Products	1	3	2	5	5	0	1	9	3	2	2
	2	2	1	3	1	8	0	1	0	4	3
	3	1	0	6	2	3	4	4	2	5	0
	4	4	2	1	8	0	0	7	3	0	4
	5	7	3	2	0	5	3	3	6	4	0
	6	0	2	4	2	3	1	0	4	3	6
	7	2	3	2	1	3	2	8	4	5	0
	8	3	4	6	7	8	9	0	2	1	1
	9	3	3	1	1	0	2	3	4	8	9
	10	2	1	3	0	6	4	3	0	0	5
	11	0	5	2	1	9	2	0	4	2	3
	12	9	2	1	6	4	3	2	4	5	7
	13	7	0	4	3	2	3	4	7	5	2
	14	6	8	5	0	1	2	5	3	2	3
	15	2	7	3	3	5	2	0	6	4	8

4. COMPUTATIONAL EXPERIENCE

Using the measure of performance discussed in the previous section, and the different problem sets created, the performance of the algorithms are evaluated and the results are shown in the following sections. The algorithms that performed well under each measure were identified, the relations between part structures and performance were also revealed.

4.1. Comparison by Goal Chasing D

From Figure 1 to Figure 11, when using goal chasing D as the measurement, Goal Chasing I has the best performance as evident from the lowest D values obtained. Time Spread algorithm results in the highest D value amount in all the algorithms. It is noted that, the performance of Miltenburg Algorithm 3 Heuristic 1 is very good, steady and close to that of Goal Chasing I. Quick and Dirty, Heuristic 1 of Miltenburg's Algorithm 3 have similar performances in part structures a, b, c, d, g, i and j. Goal Chasing II performs well except part structure b, c and d. Quick and Dirty has very good performance except problem set 14 of structure e. And Miltenburg Algorithm 3 Heuristic 2 has unsteady performance.

Comparing the D values obtained for eleven part structures, it is also found that basically the performance of all algorithms worsen when the variance of part structure increases, i.e., it is harder to achieve ideal parts usage rate when the parts requirement by each product varies. But there are some exceptions, i.e., structures b and f.

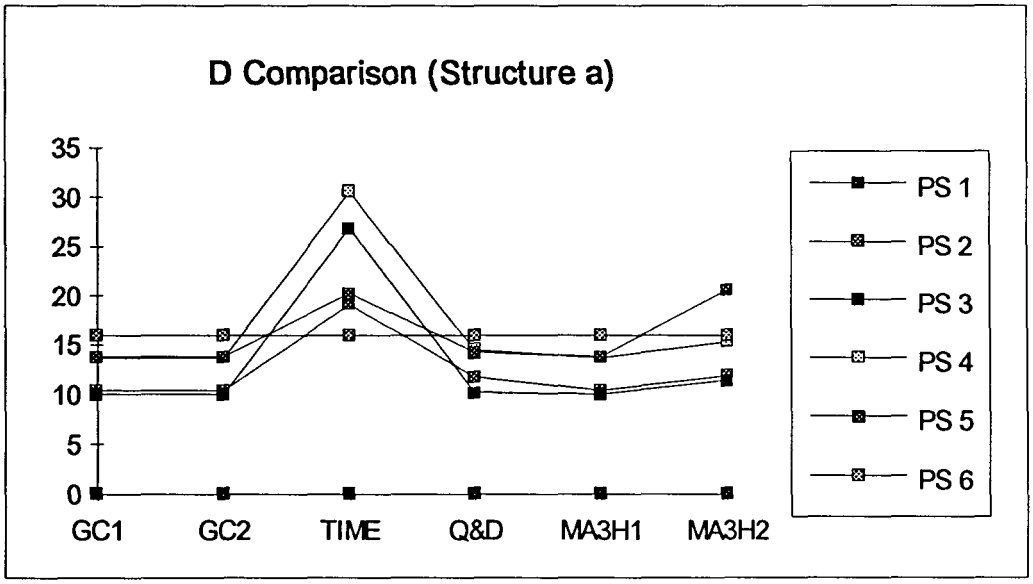

Figure 1 D Comparison for Part Structure a

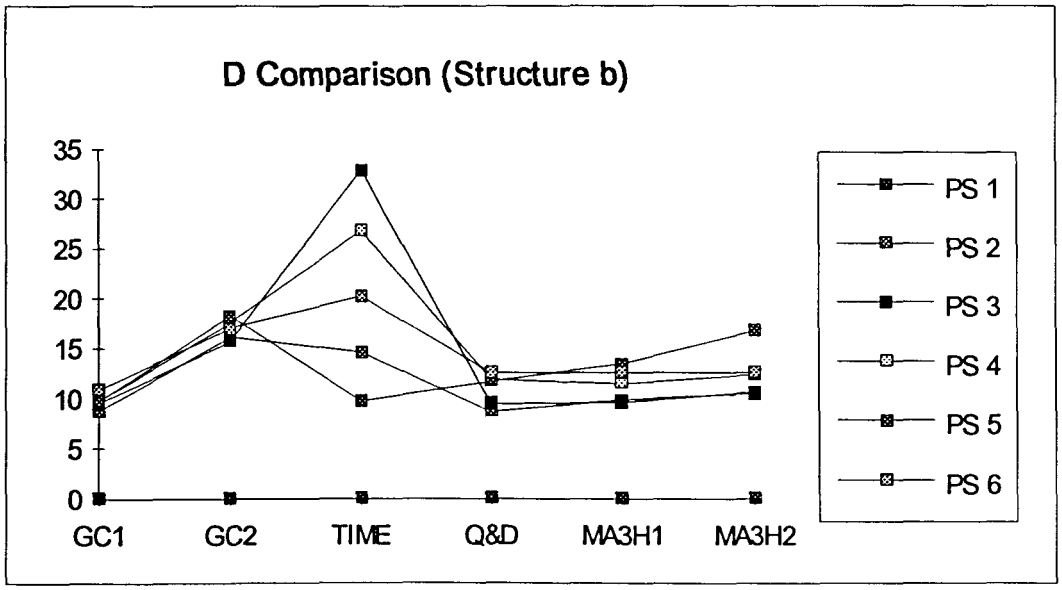

Figure 2 D Comparison for Part Structure b

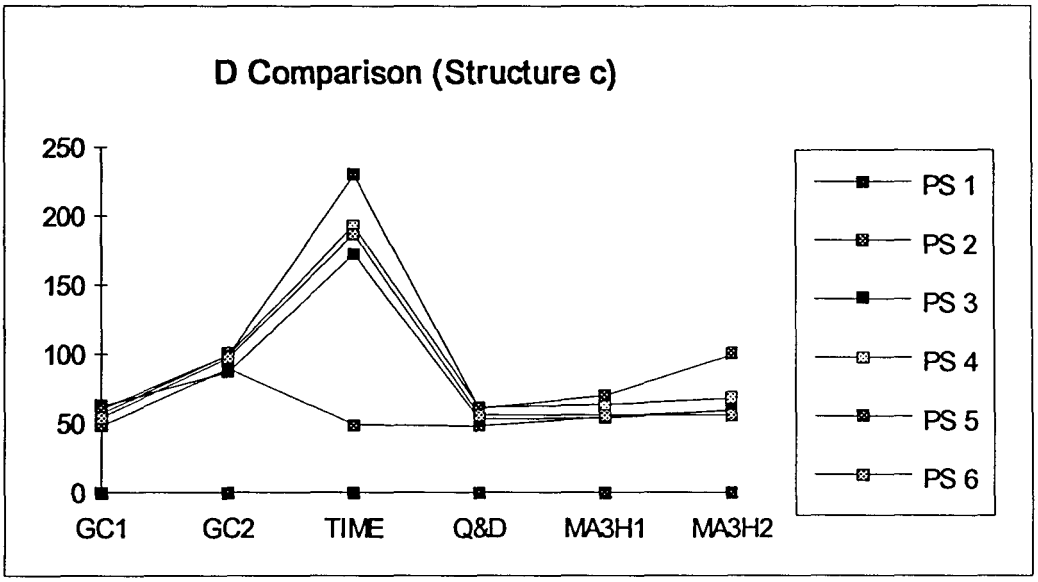

Figure 3 D Comparison for Part Structure c

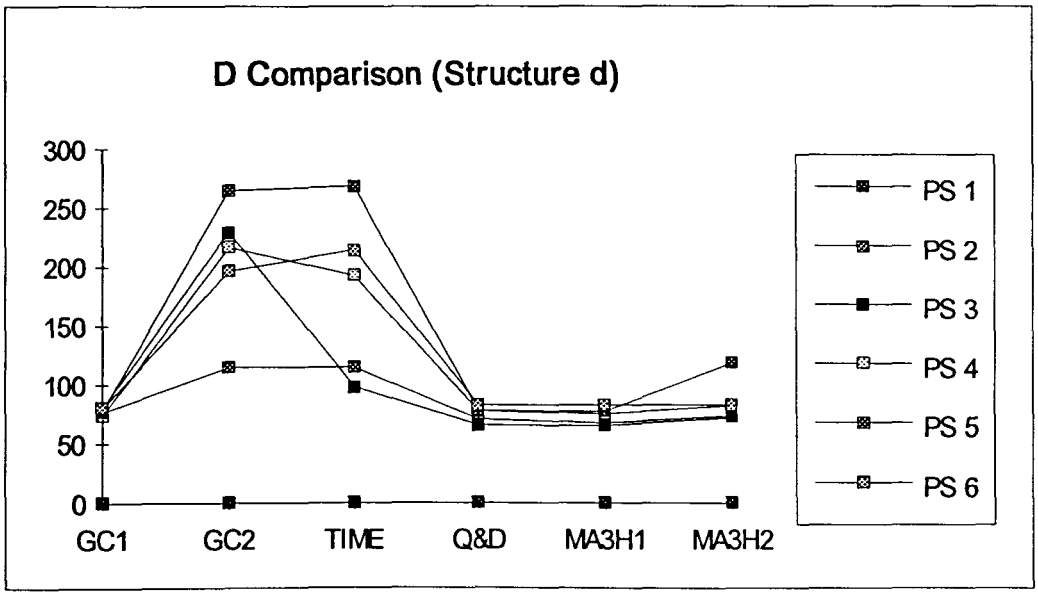

Figure 4 D Comparison for Part Structure d

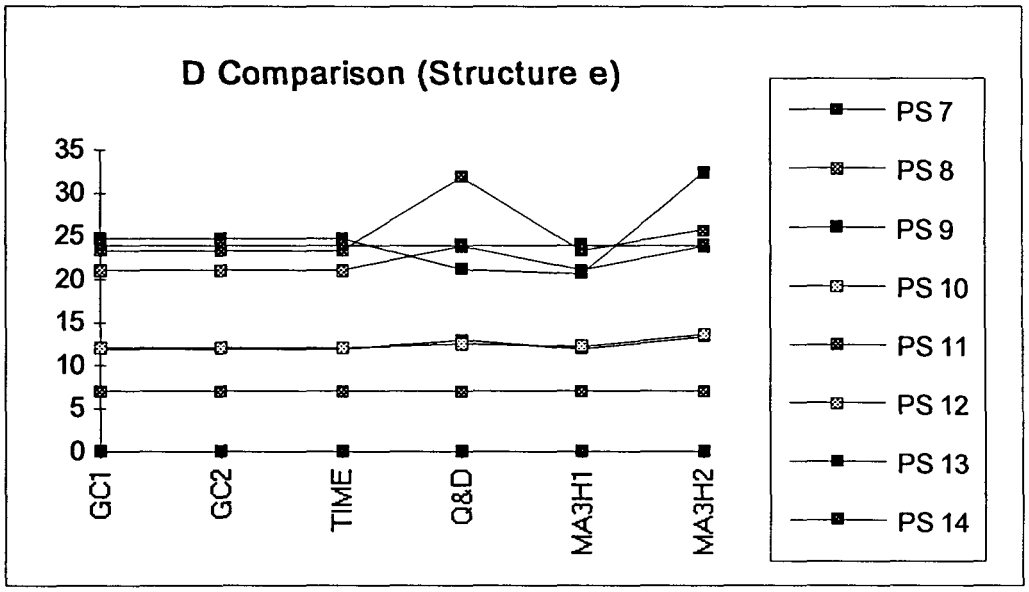

Figure 5 D Comparison for Part Structure e

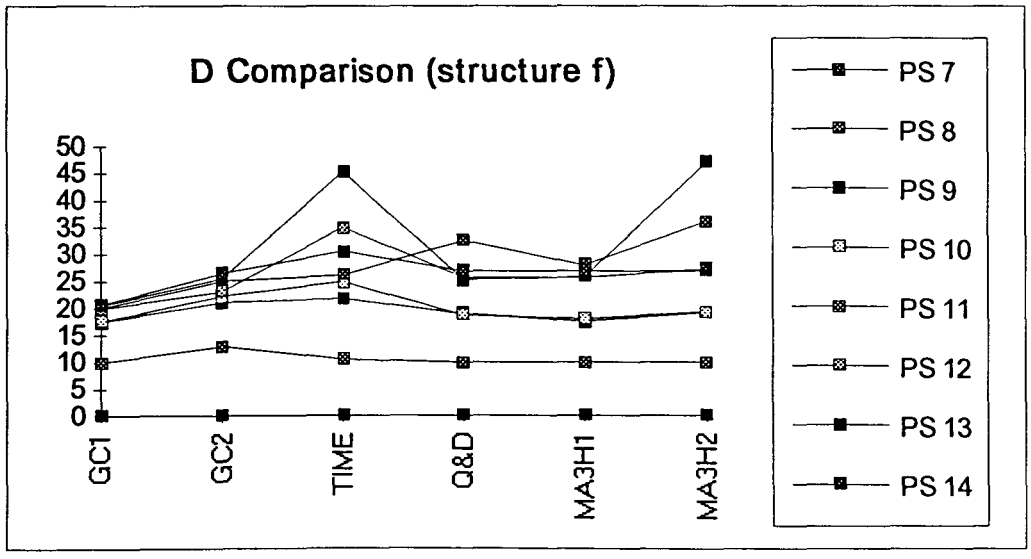

Figure 6 D Comparison for Part Structure f

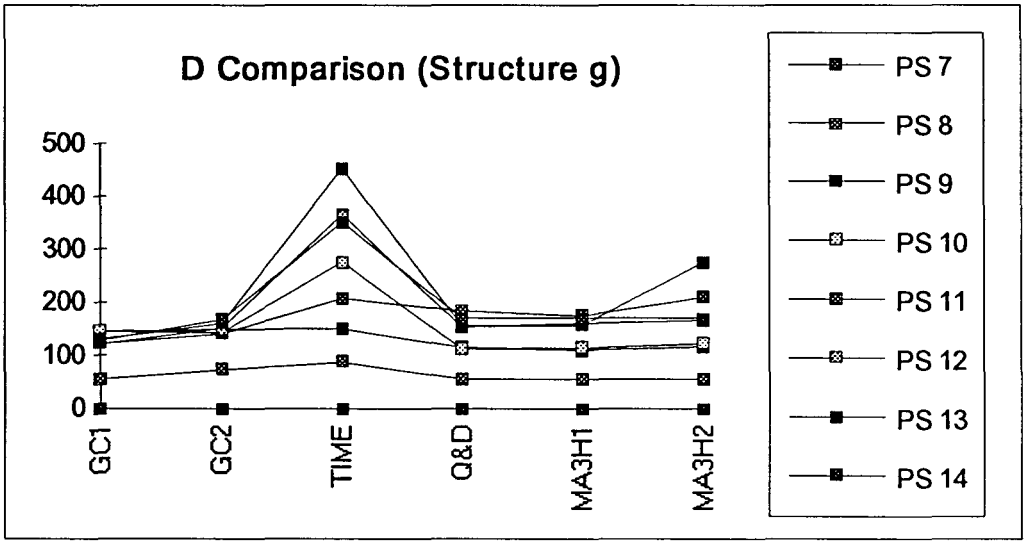

Figure 7 D Comparison for Part Structure g

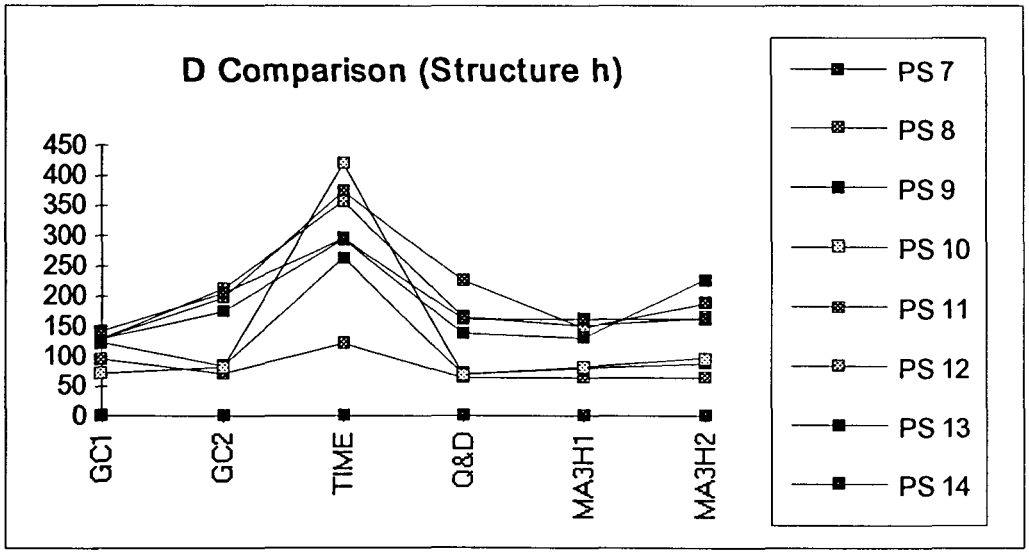

Figure 8 D Comparison for Part Structure h

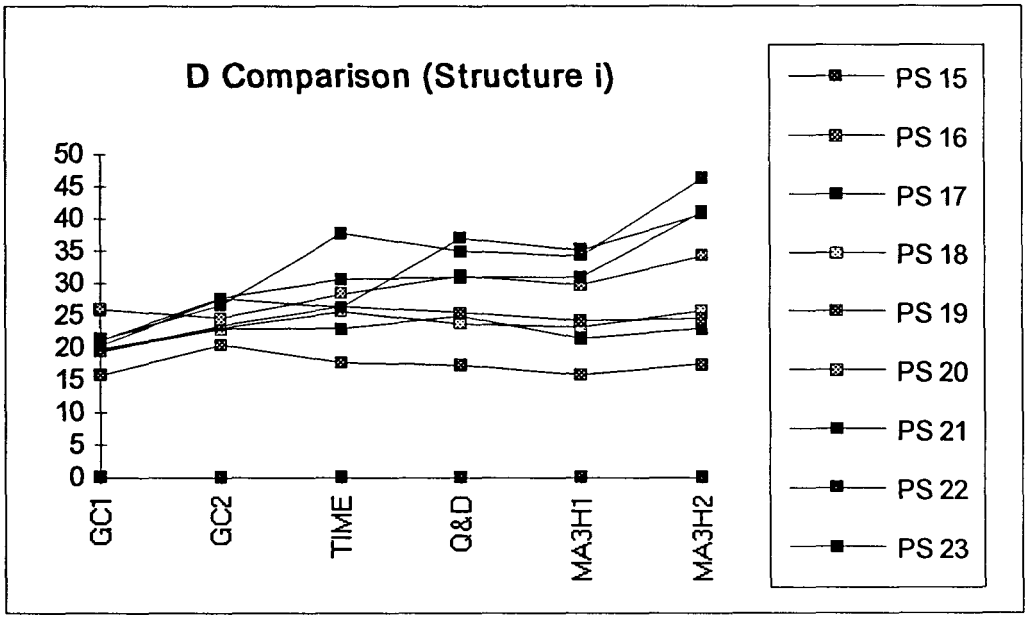

Figure 9 D Comparison for Part Structure i

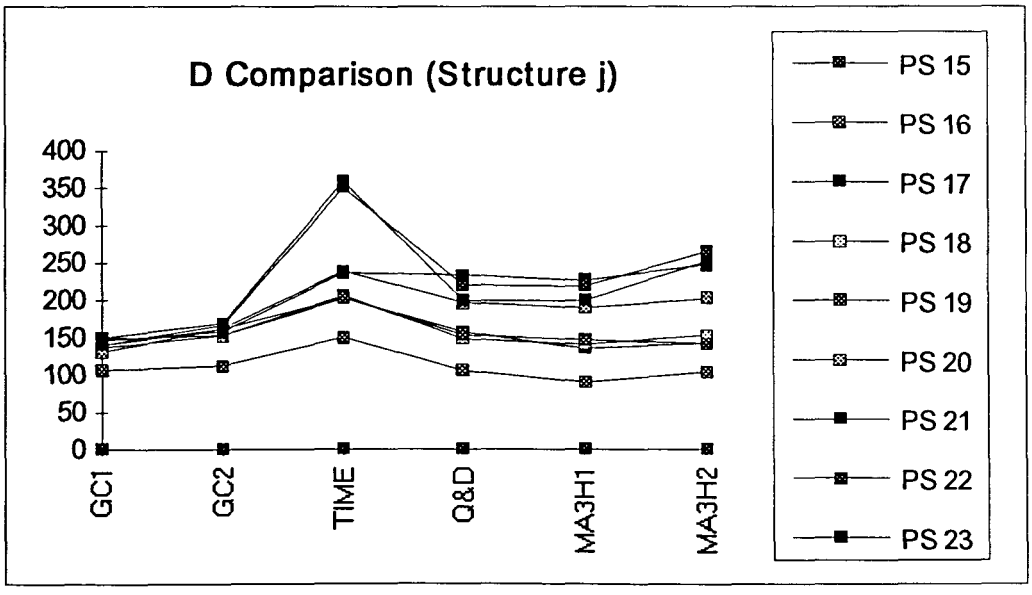

Figure 10 D Comparison for Part Structure j

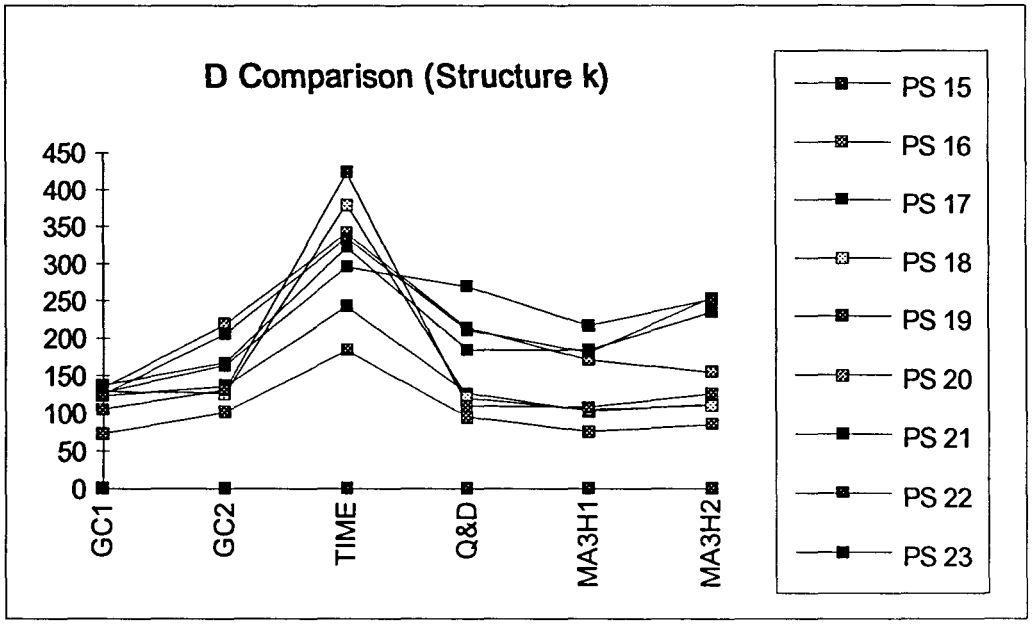

Figure 11 D Comparison for Part Structure k

4.2. Comparison by Weighted Goal Chasing D

From Figure 12 to Figure 22, it is obvious that, Goal Chasing I performed best among all algorithms when compared against Weighted-D, because of the lowest values obtained. Miltenburg Algorithm 3 Heuristic 1 is steady and close to the best. Time Spread Algorithm has good performance in structure e , fair performance in structure i and bad performance in other structures. Quick and Dirty has very good performance generally except structures e and f. Goal Chasing II's performance is all right except structure d. Some performance of algorithms worsen when the part structure becomes more complex, this again leads to the conclusion that basically the ideal parts usage rate is harder to achieve when the parts requirement varies among products.

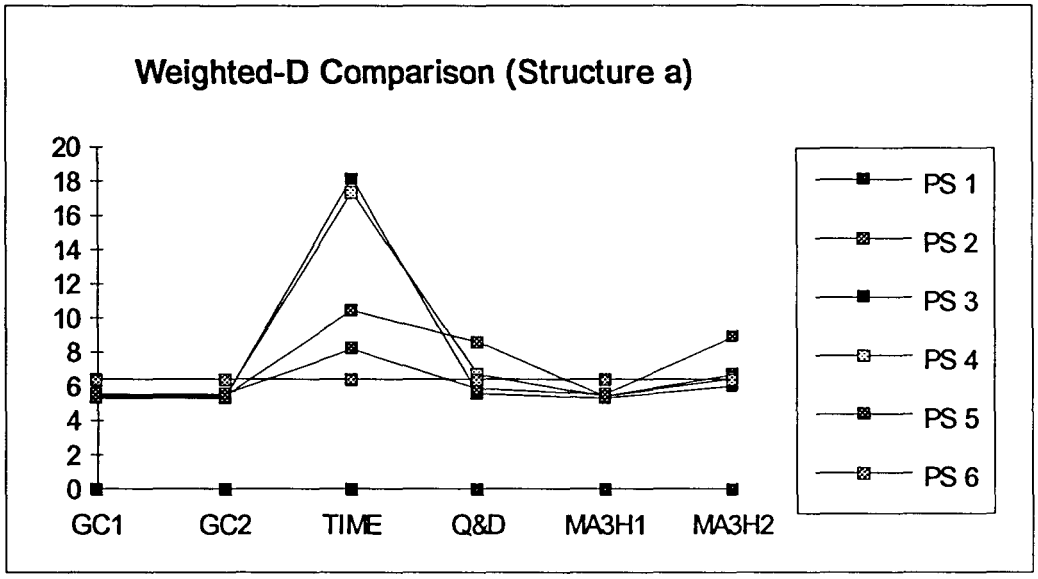

Figure 12 Weighted-D Comparison for Part Structure a

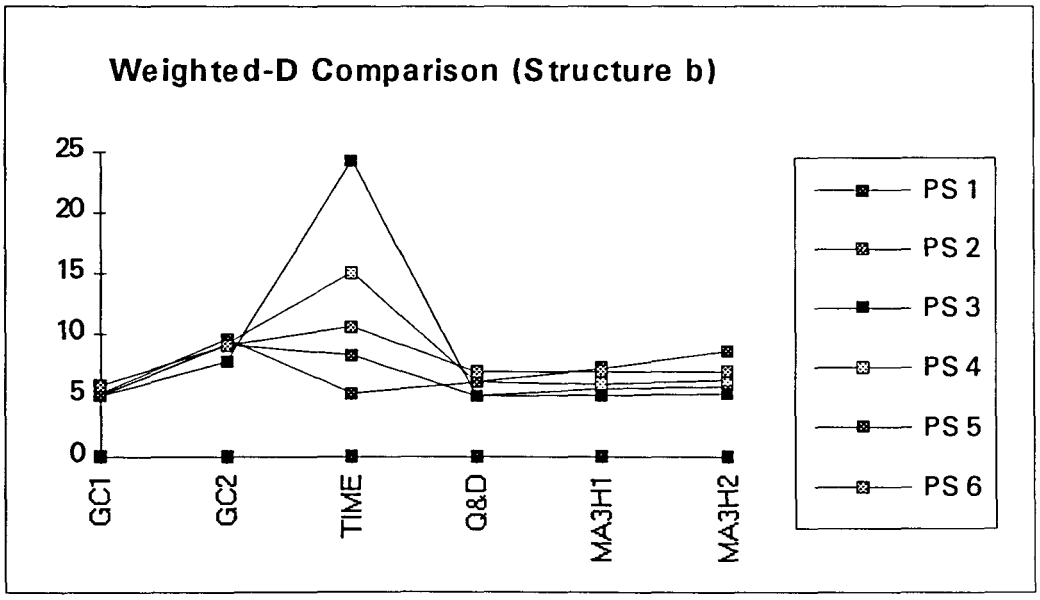

Figure 13 Weighted-D Comparison for Part Structure b

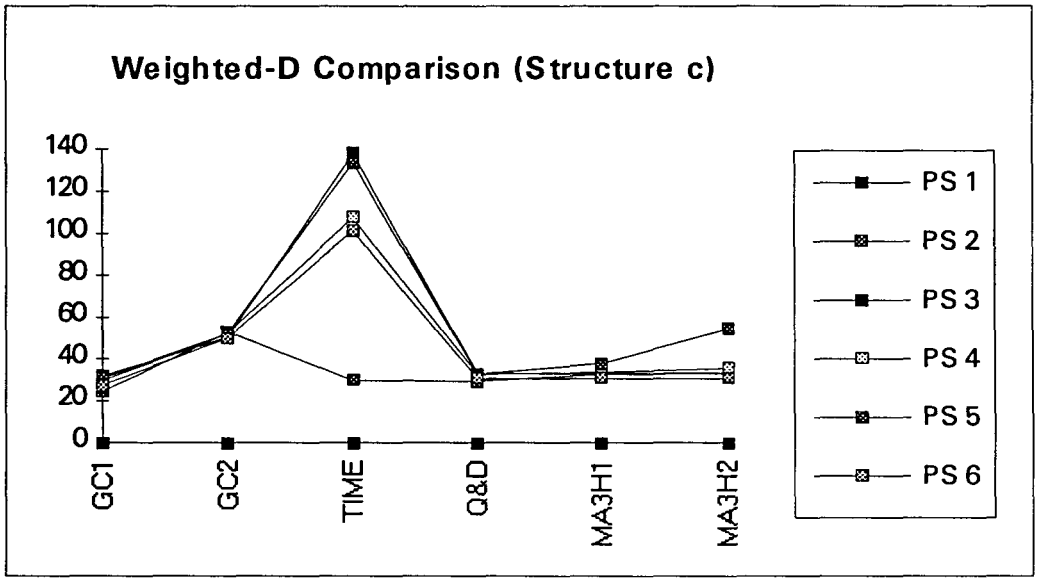

Figure 14 Weighted-D Comparison for Part Structure c

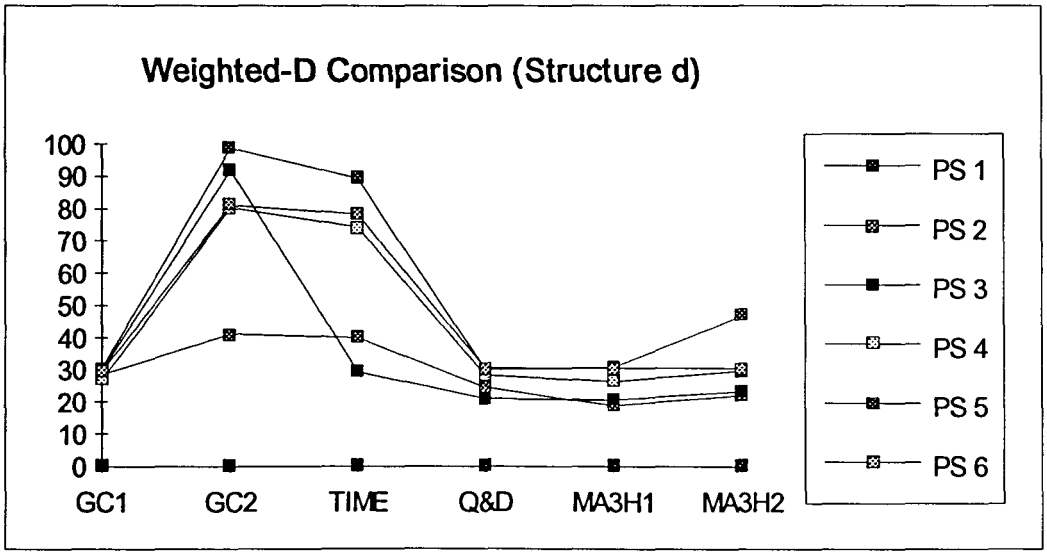

Figure 15 Weighted-D Comparison for Part Structure d

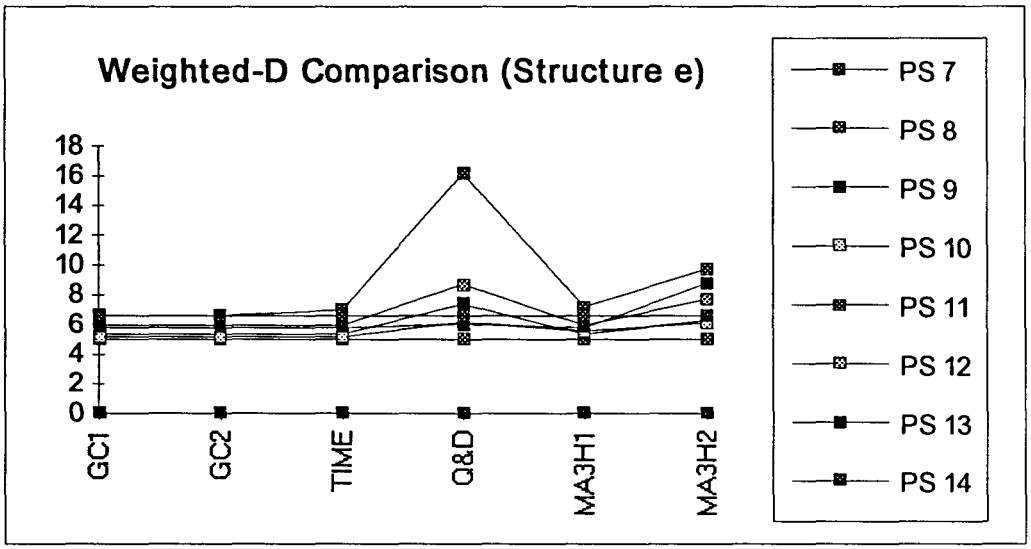

Figure 16 Weighted-D Comparison for Part Structure e

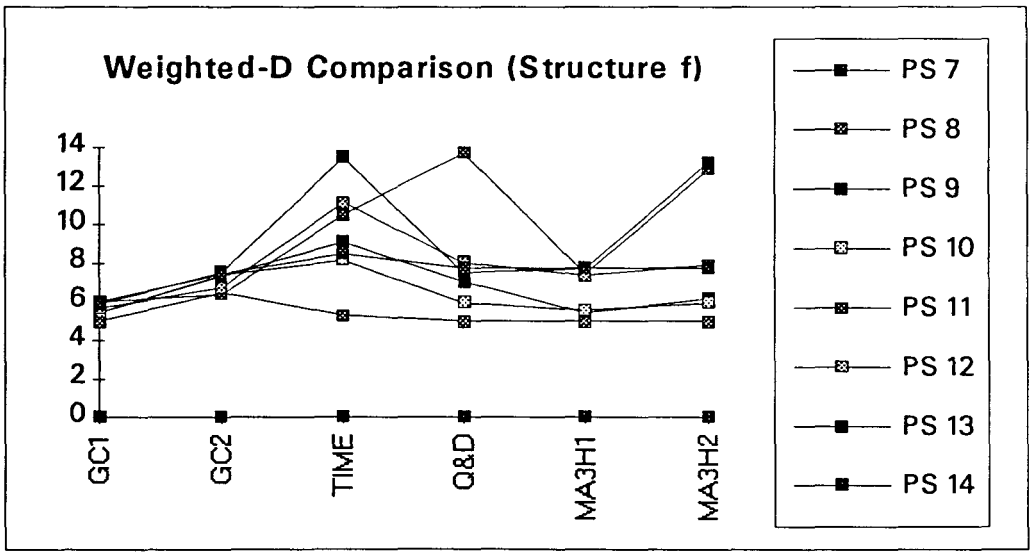

Figure 17 Weighted-D Comparison for Part Structure f

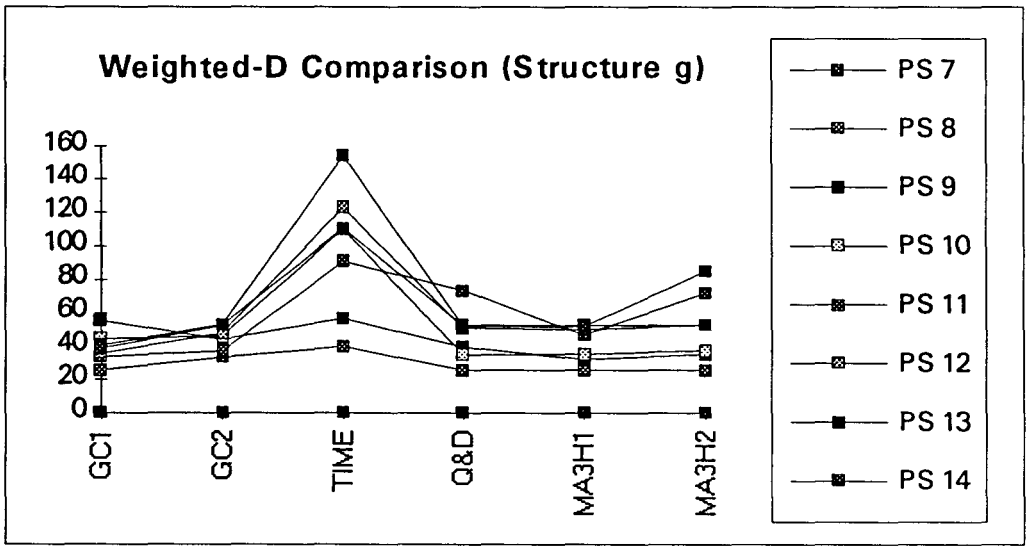

Figure 18 Weighted-D Comparison for Part Structure g

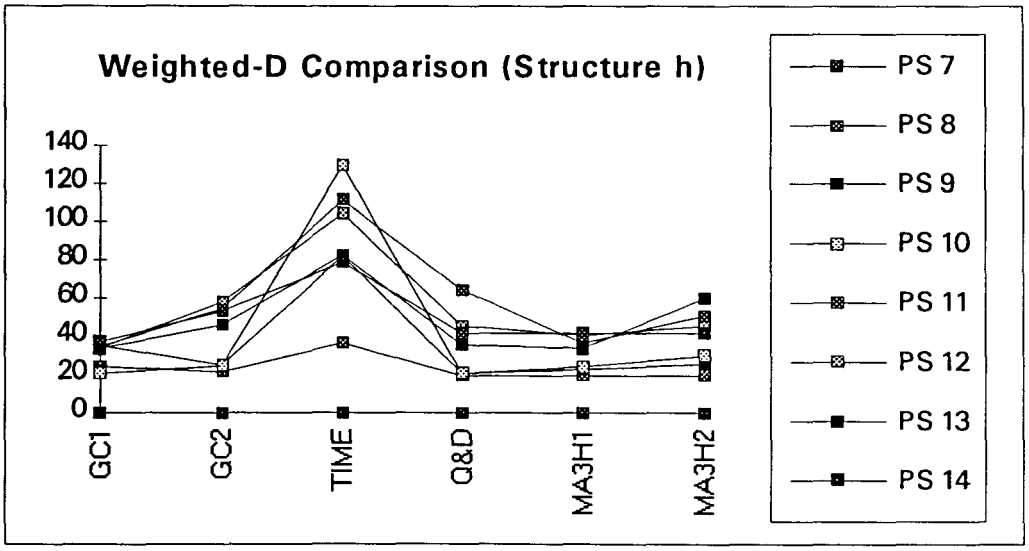

Figure 19 Weighted-D Comparison for Part Structure h

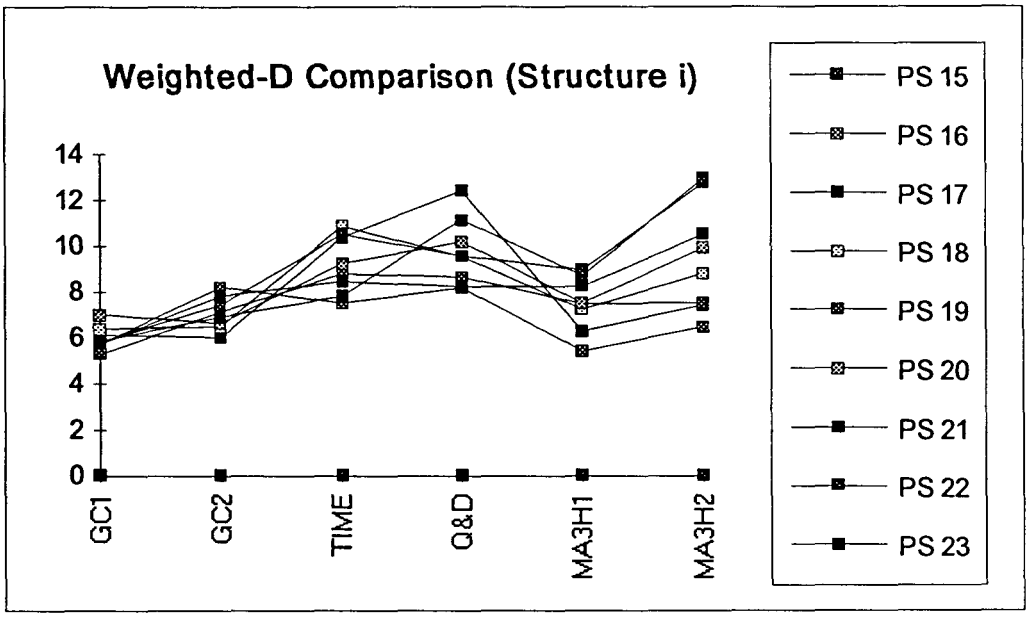

Figure 20 Weighted-D Comparison for Part Structure i

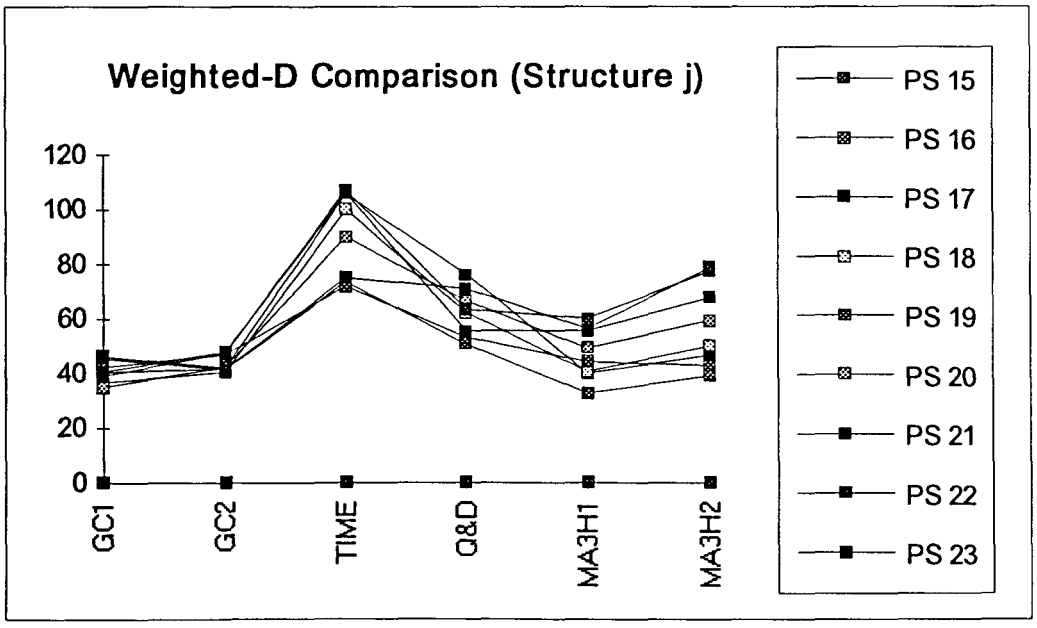

Figure 21 Weighted-D Comparison for Part Structure j

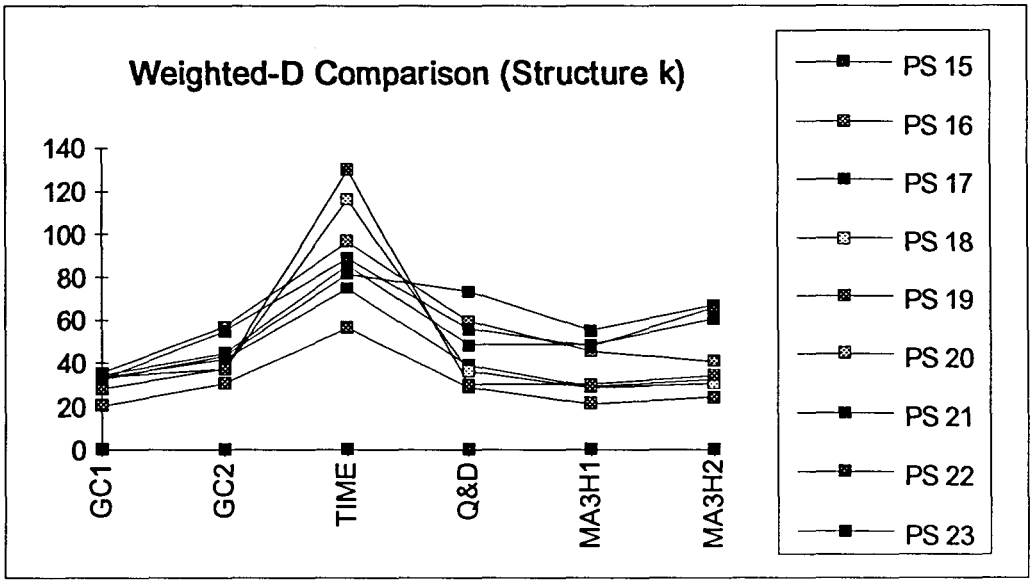

Figure 22 Weighted-D Comparison for Structure k

4.3. Comparison by Objective Function of Time Spread

From Figure 23 to Figure 33, when using the objective function of Time Spread Algorithm as the measure of performance, the processing time at each workstation becomes an important issue. As expected, Time Spread Algorithm performs well except structure a because it is the only algorithm that takes process time into consideration when generating the mixed model sequence. Miltenburg Algorithm 3 Heuristic 1's performance is steady and close to the best. Goal Chasing I has very good performance except problem set 22 of structure i. Goal Chasing II's performance is not steady. And Miltenburg Algorithm 3 Heuristic 2 has very good performance except problem set 14 of structure e and problem set 19 of structure i

It is also noted that the performance of some algorithms deteriorates from simple part structure to complex part structure, i.e., it is harder to smooth the work load at each workstation when there is a large variance of part requirements.

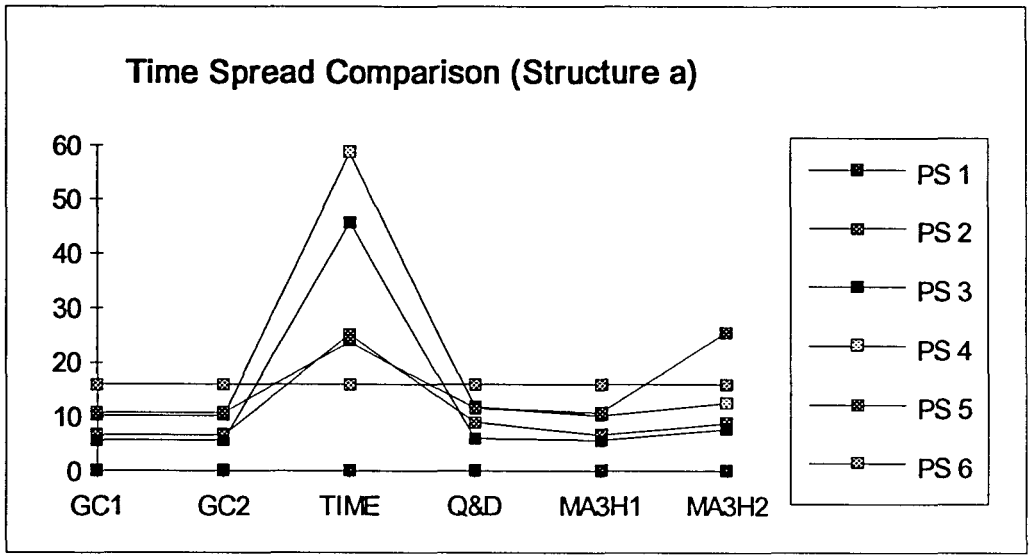

Figure 23 Time Spread Comparison for Part Structure a

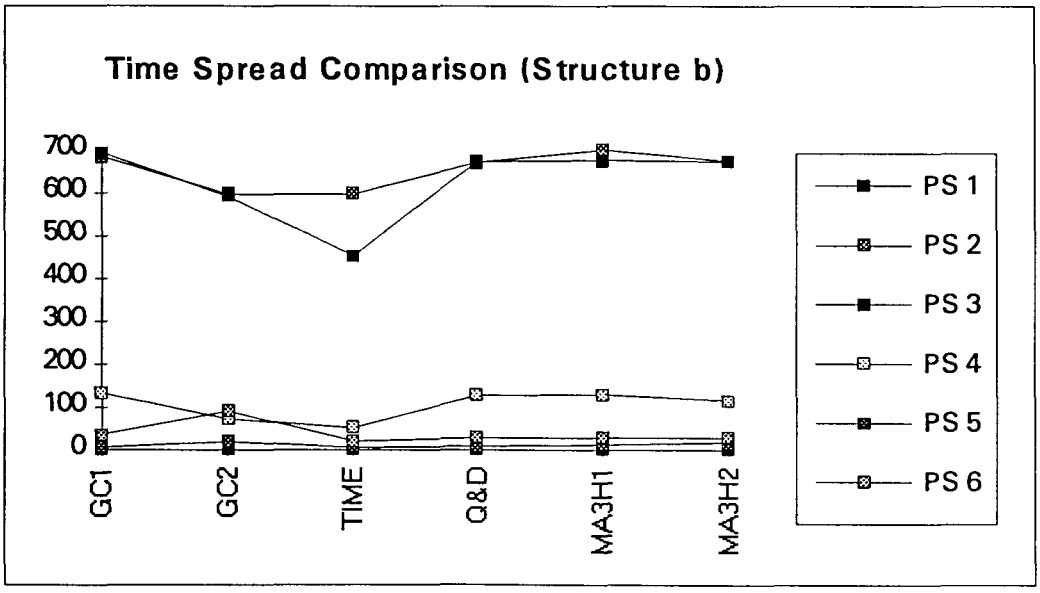

Figure 24 Time Spread Comparison for Part Structure b

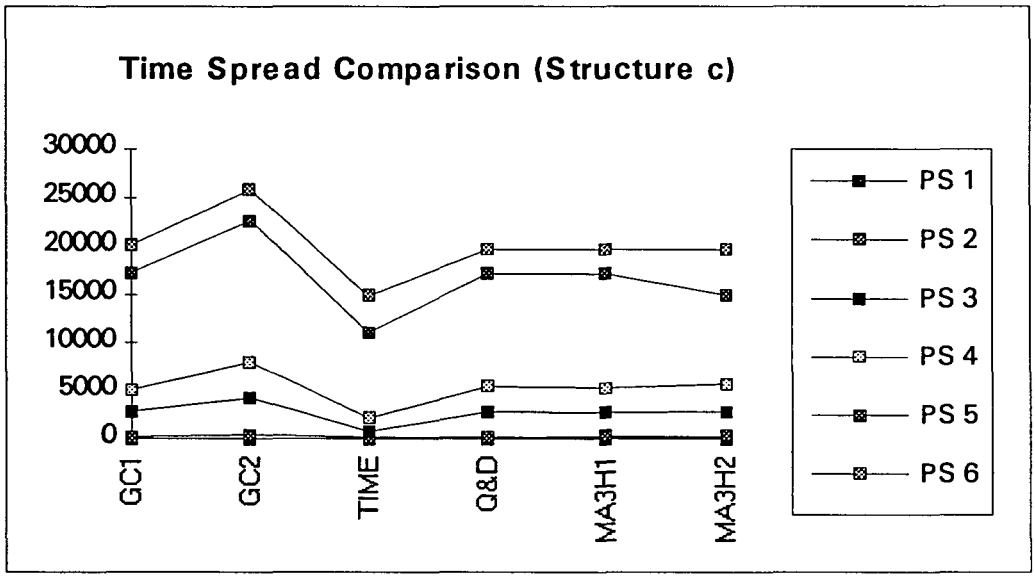

Figure 25 Time Spread Comparison for Part Structure c

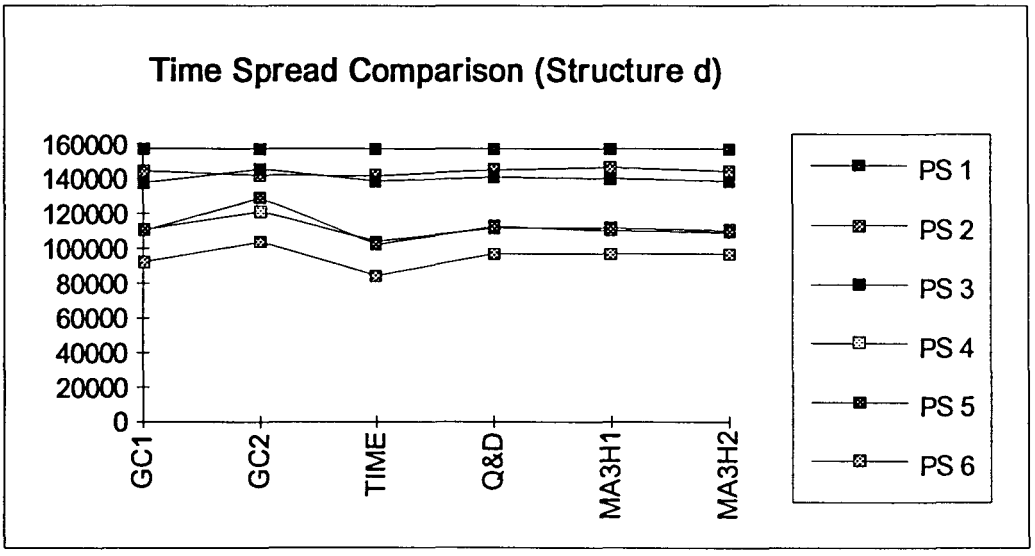

Figure 26 Time Spread Comparison for Part Structure d

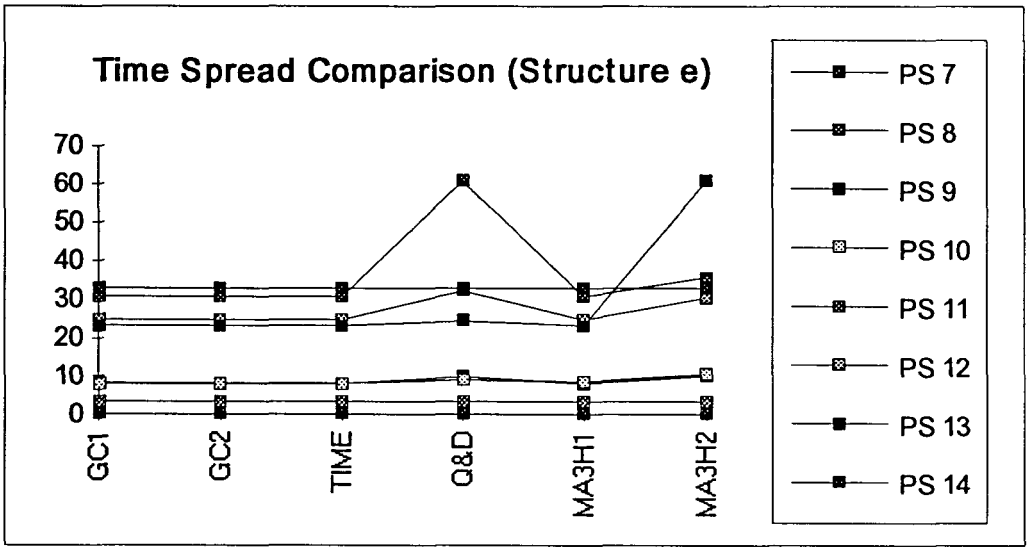

Figure 27 Time Spread Comparison for Part Structure e

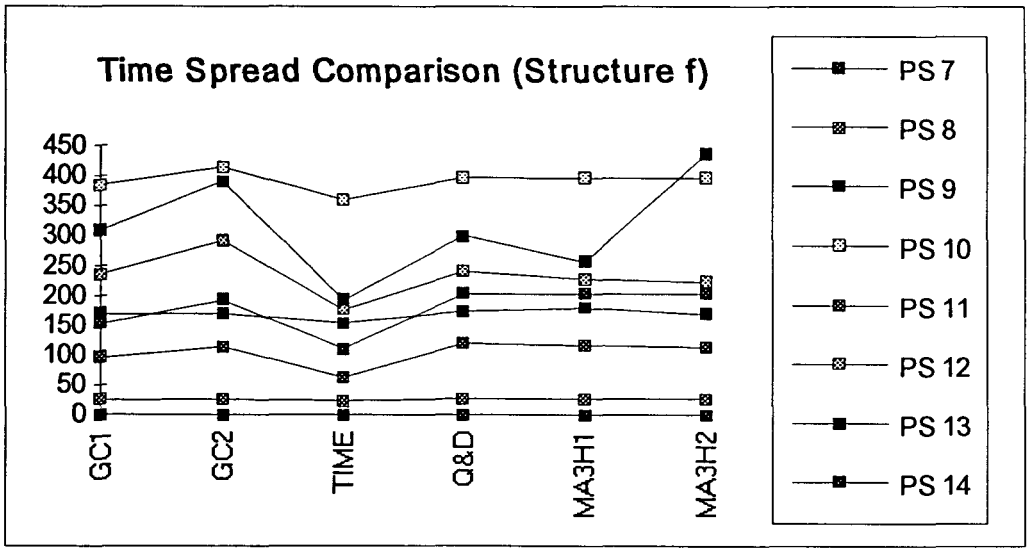

Figure 28 Time Spread Comparison for Part Structure f

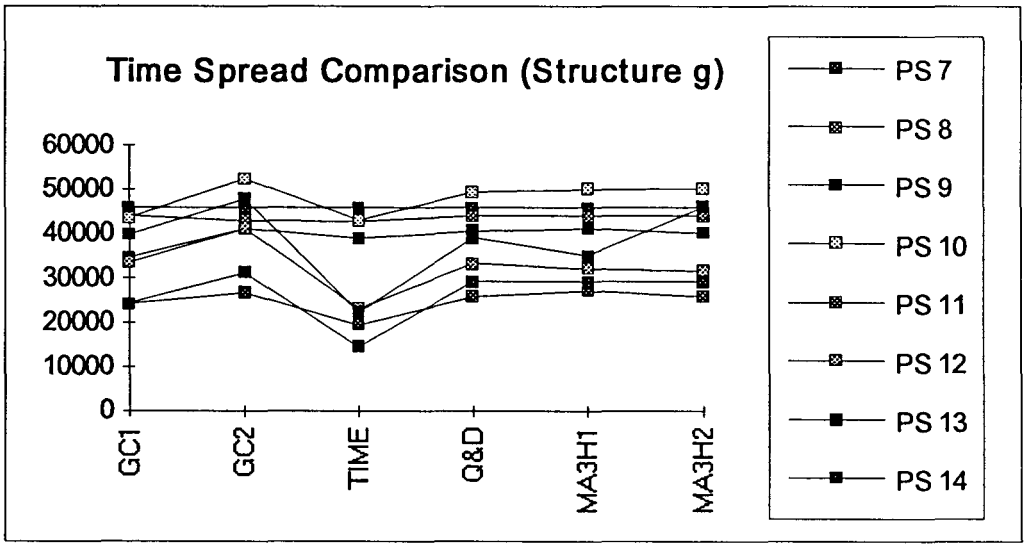

Figure 29 Time Spread Comparison for Part Structure g

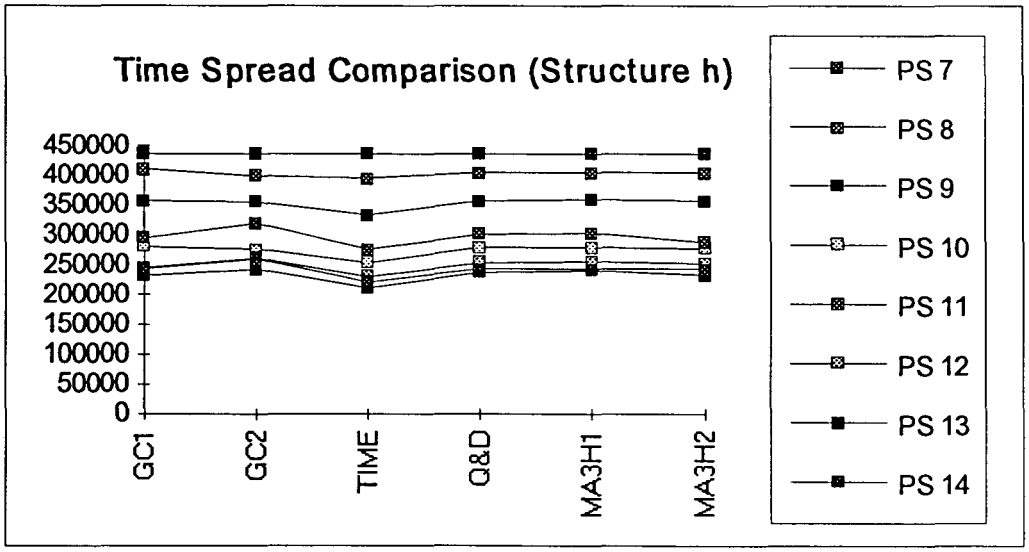

Figure 30 Time Spread Comparison for Part Structure h

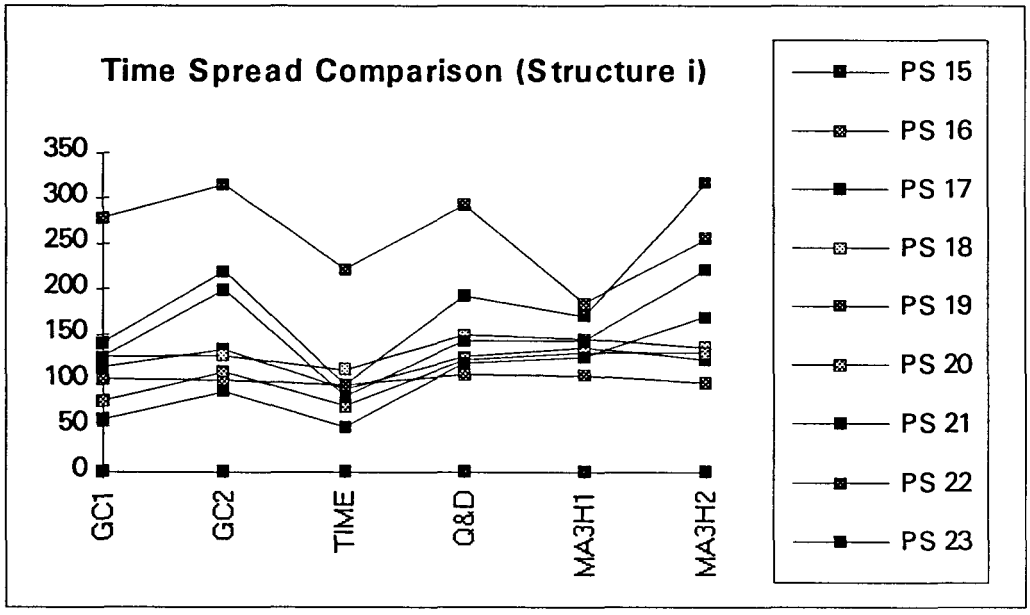

Figure 31 Time Spread Comparison for Part Structure i

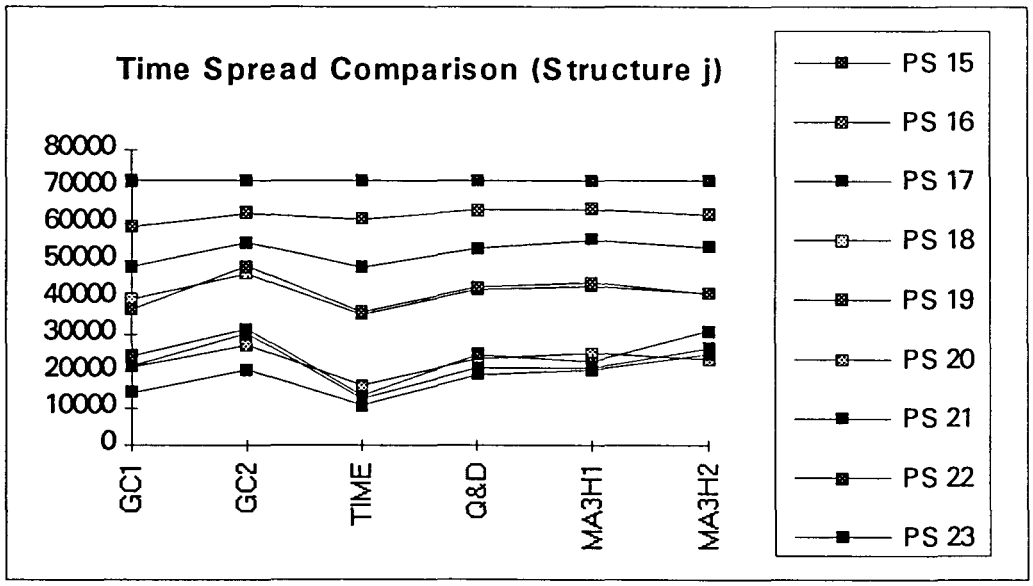

Figure 32 Time Spread Comparison for Part Structure j

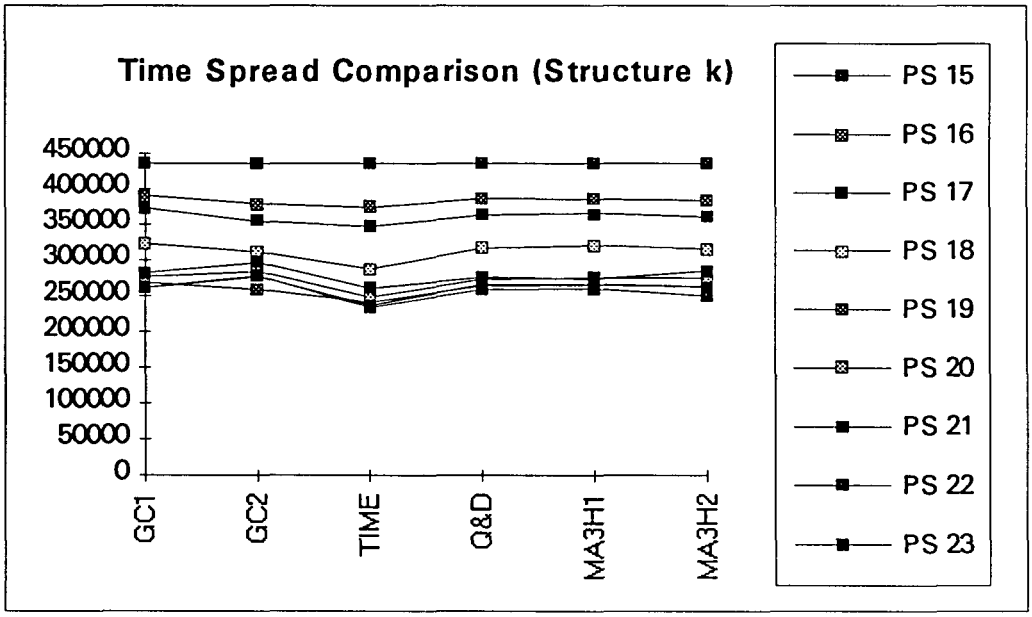

Figure 33 Time Spread Comparison for Structure k

4.4. Comparison by Variance

From Figure 34 to Figure 44, by using total variance as the measure of performance , Miltenburg Algorithm 1 has the lowest total variance. However, the sequence generated may be infeasible. When Miltenburg Algorithm 1 failed to generate a sequence, Heuristic 1 of Miltenburg Algorithm 3 has the best performance. Goal Chasing I, Goal Chasing II and Miltenburg Algorithm 3 Heuristic 2 have unsteady performance. Quick and Dirty has very good performance in part structure a, b, c and d. Again Time Spread algorithm has the worse performance.

Basically when the variance of part structure increases, the performance gets worse but there are some exceptions.

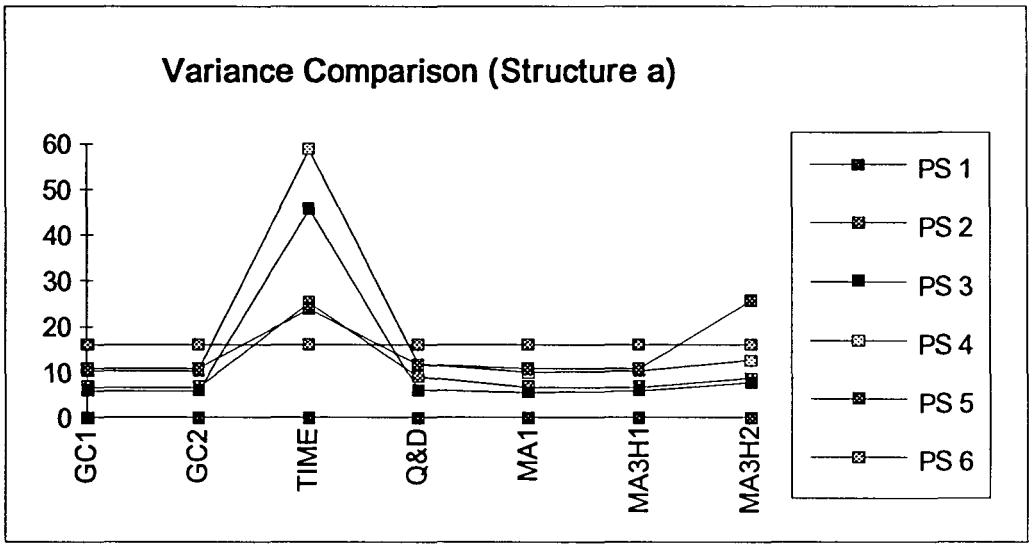

Figure 34 Variance Comparison for Part Structure a

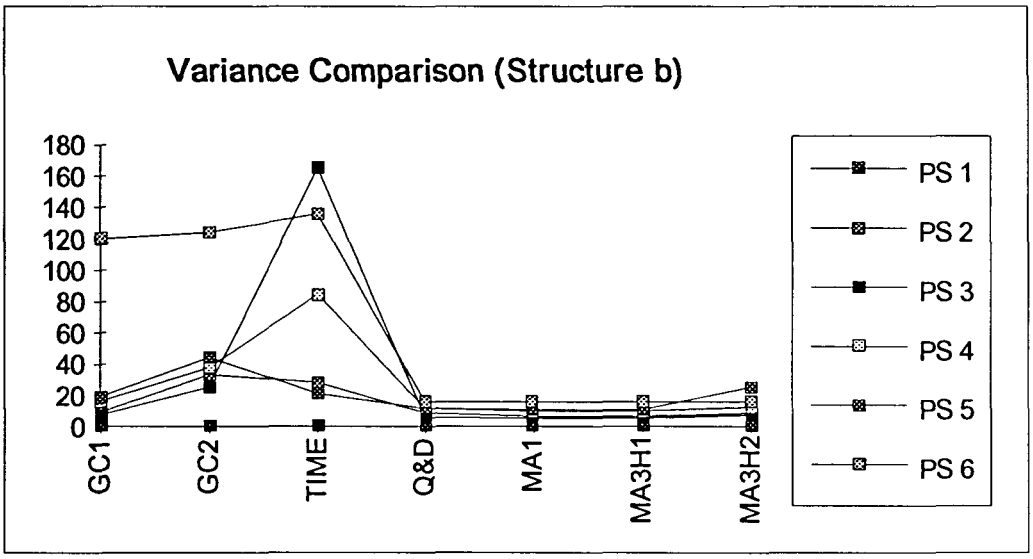

Figure 35 Variance Comparison for Part Structure b

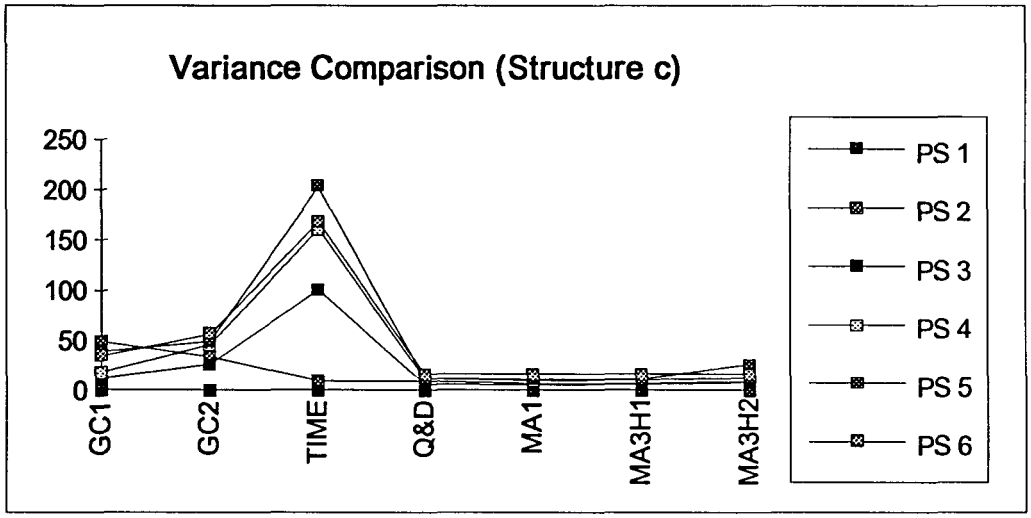

Figure 36 Variance Comparison for Part Structure c

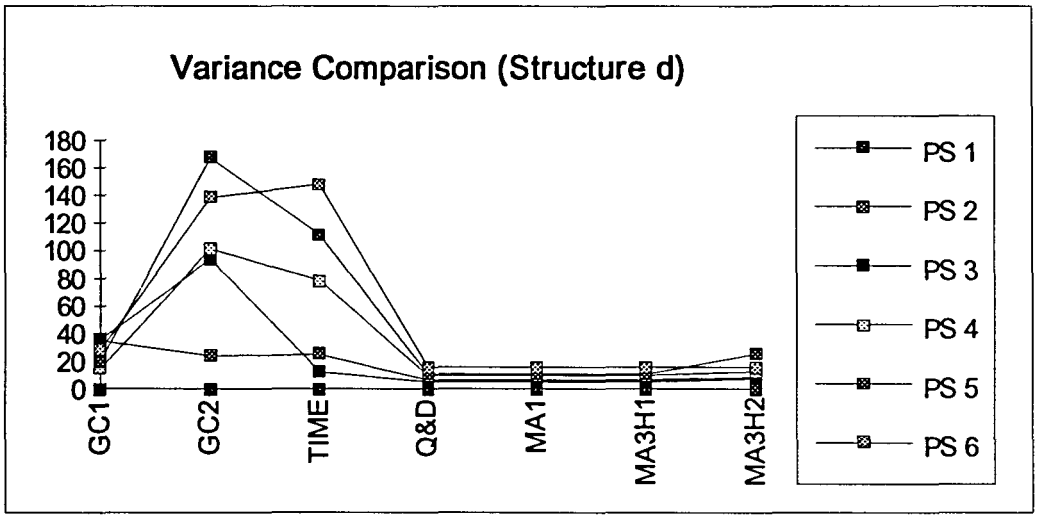

Figure 37 Variance Comparison for Part Structure d

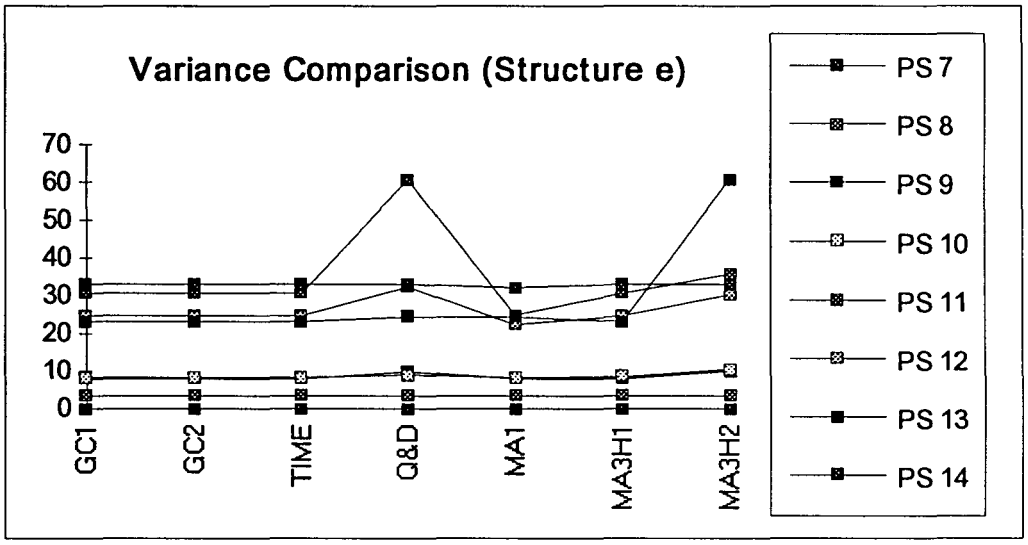

Figure 38 Variance Comparison for Part Structure e

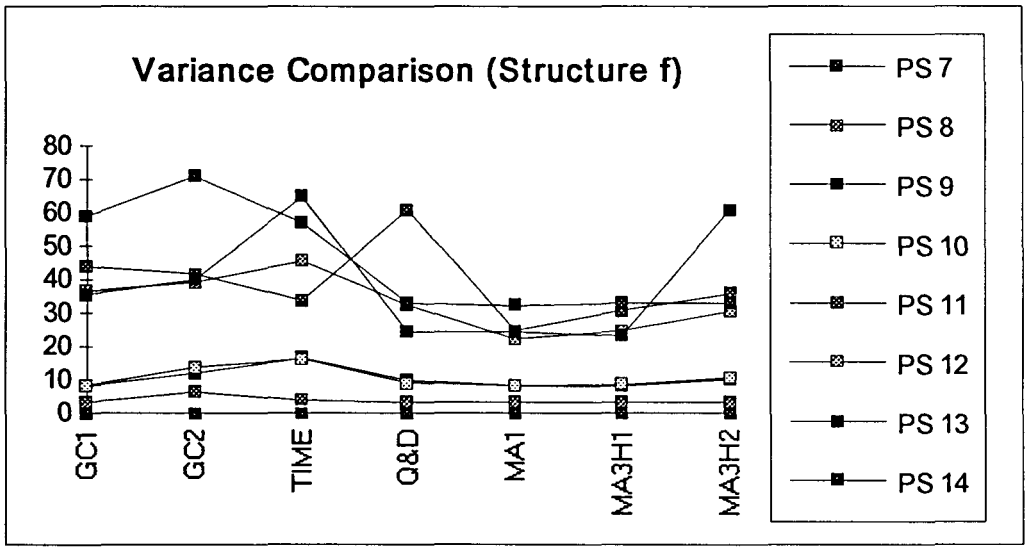

Figure 39 Variance Comparison for Part Structure f

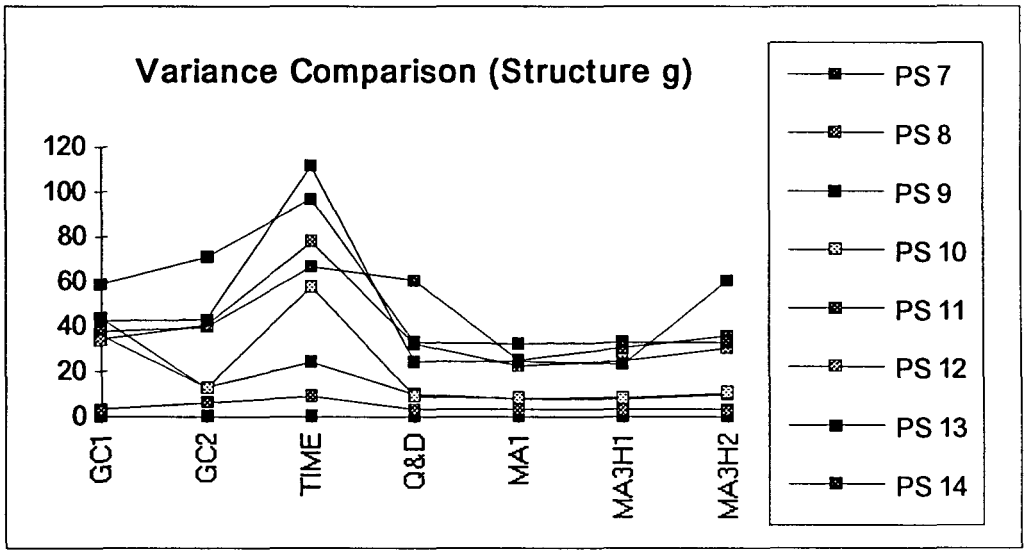

Figure 40 Variance Comparison for Part Structure g

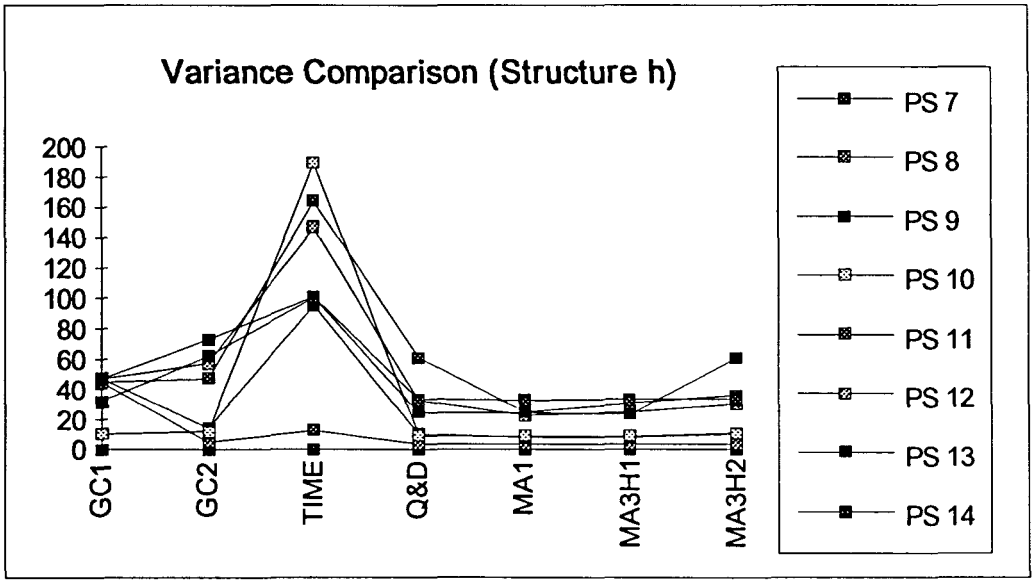

Figure 41 Variance Comparison for Part Structure h

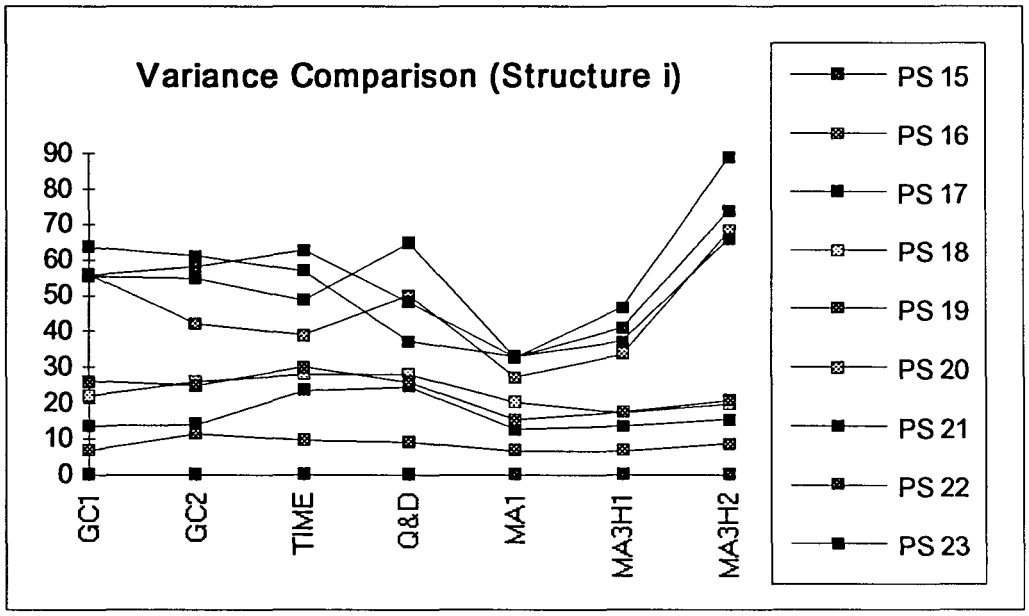

Figure 42 Variance Comparison for Part Structure i

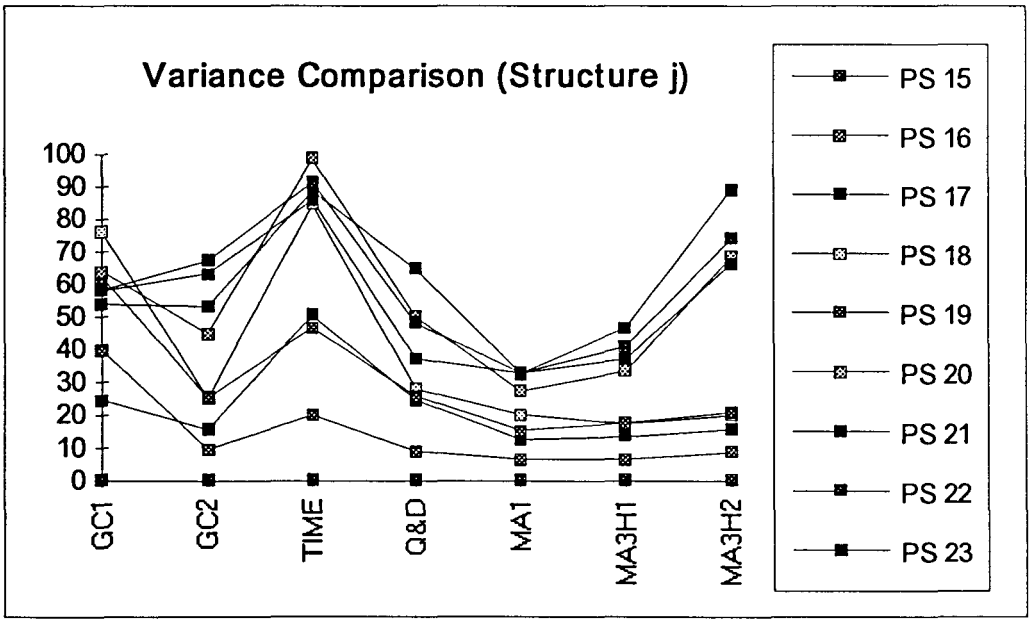

Figure 43 Variance Comparison for Part Structure j

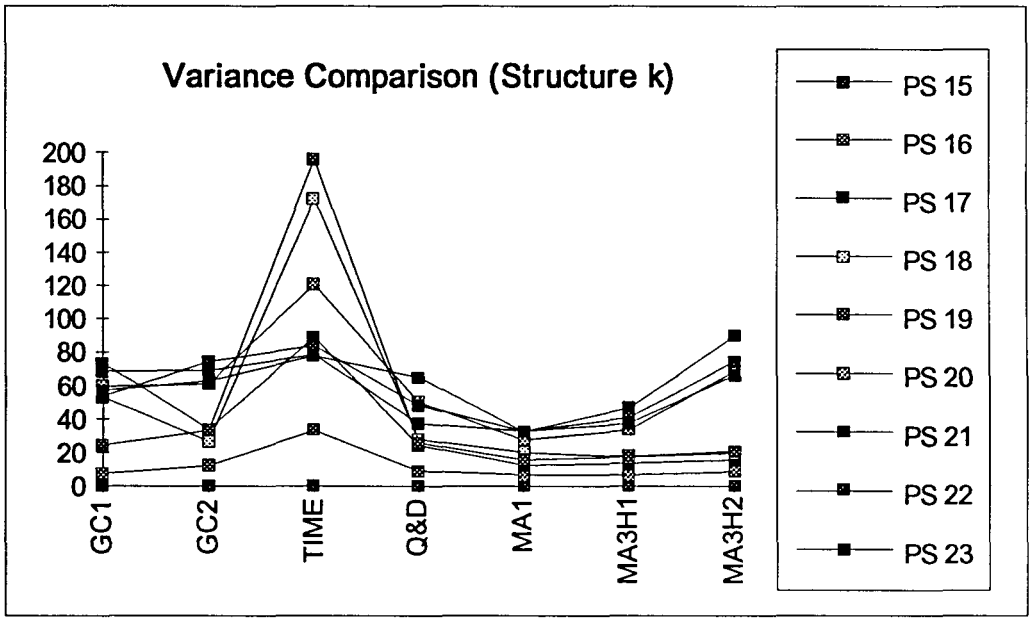

Figure 44 Variance Comparison for Structure k

4.5. Computational Efforts

The computational effort was measured quantitatively by the degree of difficulty in programming and summarized in Table 30. Among these algorithms, Quick and Dirty and Time Spread algorithm require the least amount of effort. Both Goal Chasing I and II require moderate computational time and Miltenburg's Algorithm 3 Heuristic 1 & 2 require the most calculations and iterations. But when testing the running time of each program, the interval of running time is only 1 second or less.

Table 30 Comparison of Computational Effort

	GC 1	GC 2	Time	Q&D	MA1	MA3H1	MA3H2
Easy			√	√			
Average	√	√			√		
Hard						√	√

5. CONCLUSIONS

Seven mixed model sequencing algorithms, Goal Chasing I and II, Time Spread, Quick and Dirty and three algorithms suggested by Miltenburg, have been studied and compared. Several performance measures are suggested and used to compare the efficiency of these algorithms. Almost all algorithms perform well when evaluated against their own objective function.

This study can be used as a basis of several future research efforts, and has demonstrated the procedure for comparison of mixed model sequencing techniques. When using the problem sets and part structures suggested, this study concludes that Heuristic 1 of Miltenburg Algorithm 3 generates the most efficient mixed model sequence. Goal Chasing I performs well in D, weighted-D and time spread comparisons. Quick and Dirty has some good performance in each evaluation but its performance is not steady all the time. Goal Chasing II and Miltenburg Algorithm 3 Heuristic 2 have unsteady performance. And Time Spread Algorithm generates the worst results. Basically when the variance of part structure increases, it is harder to achieve the ideal production objectives.

In practical applications, it would be very important for the managers to decide the pay off between computational complexity and performance and select an algorithm that best suits the needs. In addition, the objectives of the mixed model sequencing should be well identified. Different algorithms perform significantly different when evaluated under different performance measures. Some criterion may be developed to allow managers to quickly determine whether the sequence generated is acceptable. If the managers are interested in applying mixed-model sequence to their companies and their objectives are maintaining a constant speed in consuming each part on the line, preventing a continuous stream of

unfinished work from the station and reducing the variance, Miltenburg Algorithm 3 Heuristic 1 is a very good choice for them. Goal Chasing I is also a good choice for the managers if they do not care about the variance very much. Goal Chasing I takes less effort to get the sequences than Miltenburg Algorithm 3 Heuristic 1.

The first extension of this study is to include the evaluation of more newly developed algorithms and other heuristics that have performed well either theoretically or practically. Secondly, more comprehensive measures of performance may be suggested so that all algorithms can be evaluated under the same and fair basis, and to satisfy different objectives of maintaining a mixed model sequencing, such as assembly line efficiency, ability to satisfy customers' demand in a timely manner, reducing lot size and setup time, etc. Thirdly some existing good algorithms can be improved by involving the new criteria, i.e., considering the part structure, processing time and due date in Miltenburg Algorithm 3 Heuristic 1. In the real world, if the managers have some other objectives, they can create their own suitable methods.

REFERENCES

1. Abbott, Robert K., and Garcia-Molina, Hector, 1992. Scheduling real-time transactions: *performance evaluation, ACM Transactions on Database Systems,* Vol 17, September, pp 512-560.

2. Burns, Lawrence D. and Daganzo Carlos F., 1987. Assembly line job sequencing principles, *International Journal of Production Research*, Vol. 25, No. 1, pp 71-99.

3. Ding, Fong-Yuen and Cheng, Liping, 1993. An effective mixed-model assembly line sequencing heuristic for just-in-time production systems, *Journal of Operations Management*, Vol. 11, pp 45-50.

4. Groeflin H., Luss, H., Rosenwein, M. B., and Wahls, E. T., 1989. Final assembly sequencing for just-in-time manufacturing, *International Journal of Production Research*, Vol. 27, No. 2, pp 199-213.

5. Inman, R. R. and Bulfin, R. L., 1991, Sequencing JIT mixed-model assembly lines, *Management Science*, Vol. 37, No. 7, pp 901-904.

6. Inman, R. R. and Bulfin, R. L., 1992, Quick and dirty sequencing for mixed-model multi-level JIT systems, *International Journal of Production Research*, Vol. 30, No. 9, pp 2011-2018.

7. Joo, Sang-Ho and Wilhelm Wilbert E., 1993. A review of quantitative approaches in just-in-time manufacturing, *Production Planning & Control*, Vol. 4, No. 3, pp 207-222.

8. Miltenburg, John, 1989, Level schedules for mixed -model assembly lines in just-in- time production systems, *Management Science*, Vol. 35, No. 2, pp 192-207.

9. Miltenburg John and Goldstein, T., 1991. Developing production schedules with balance part usage and smooth production loads for just-in time production systems, *Naval Research Logistics*, Vol. 38, pp 893-910.

10. Miltenburg John and Sinnamon Gordon, 1989, Scheduling mixed-model multi-level just-in-time production systems, *International Journal of Production Research*, Vol. 27, No. 9, pp 1487-1509.

11. Miltenburg John and Sinnamon Gordon, 1992, Algorithms for scheduling multi-level just-in-time production systems, *IIE Transactions*, Vol. 24, pp 121-130

12. Monden, Y., 1993. *Toyota production system: An integrated approach to just-in-time*, Industrial Engineering and Management Press, Atlanta, Georgia.

13. Steiner, George and Yeomans, Scott, 1993. Level schedules for mixed-model, just-in-time processes, *Management Science*, Vol. 39, No. 6, June, pp 728-735.

14. Sumichrast, Roberta T., and Russell, Robert T., 1990. Evaluating mixed-model assembly line sequencing heuristics for just-in-time production systems, *Journal of Operations Management*, Vol. 9, No. 3, August 1990, pp 371-390.

15. Sumichrast, Robert T., Russell Roberta S., and Taylor, Bernard W. III, 1992. A comparative analysis of sequencing procedures for mixed-model assembly lines in a just-in-time production system, *International Journal of Production Research*, Vol. 30, No. 1, pp 199-214.

16. Wieslaw, Kubiak and Sethi Suresh, 1991. A note on level schedules for mixed-model assembly lines in just-in-time production systems, *Management Science*, Vol. 37, No. 1, January, pp 121-123.