Model of Flexible Production Systems with Sub-Lines and Their GA Expressions

Jaber Abu Qudeiri†, Hidehiko Yamamoto† and Rizauddin Bin Ramli†

† Intelligent Manufacturing Systems Laboratory, Gifu University
1-1 Yanagido, Gifu Shi, 501-1193, Japan

Summary

The production engineer's decision regarding buffer size in a flexible production system is one of the most important factors in maximizing production efficiency. In this paper, we propose a model Flexible Production System with Sub-lines (FPSS). By using the model, we can find the buffer size of FPSS by using GA. Also, we propose a new GA expression, referred to as a Matrix Encoding Method (MEM). This paper deals with an FPSS which consists of 5 main component lines, i.e. a main production line, parallel lines, rework path, feed-forward line and feeder line. Numerical examples show that after a number of operations based on the GA with MEM expression, the nearest optimal buffer size of FPSS could be found. The results of this study can be used to improve the production plant, and production engineers can use these results in their decisions on buffer size when they develop FPSS.

Key words:
Flexible production system, Buffer size, Genetic algorithm.

1. Introduction

Deciding buffer size for production systems has gained more and more importance because of growing production lines' complexity and because the buffer size has a great impact on production efficiency. The buffer size is still one of the major optimization problems faced by production engineers. Many articles and researches related to buffer size have been published [1-4]. To solve the buffer size problem, two requirements need to be considered. The first requirement is a search method used to solve the buffer size problem, and the second requirement is a model or approach used to evaluate and measure production system performance. The buffer size is optimized by various techniques, such as functional approximation and evaluation [5], knowledge based methods [6], simulated annealing [7], dynamic programming method [8], and other search methods. One of the search methods that can be used for studying the buffer size in production systems is a genetic algorithm (GA) [9,10]. GA is an evolutionary technique that uses crossover and mutation operators to solve optimization problems using a survival of the fittest idea. GA has been used successfully for various optimization problems, such as the buffer size problem. The performance evaluation of the production system is needed to calculate the performance measure of the production system which has to be optimized. Performance evaluation of a production system has garnered close attention in recent years, and most work done in this field can be found in a number of books [11-16], and in reviewed papers [17, 18], among others.

However, despite the many existing buffer size methodologies, many research studies have focused on the buffer size problem for a serial production line and there is a lack in the literature of studies that look at the buffer size of complicated production systems such as a Flexible Production System with Sub-lines (FPSS). This paper deals with FPSS which consists of 5 component lines, a main production line, parallel lines, rework path, feed-forward line and feeder line. In this paper, we define FPSS model. By using the model, we can find the buffer size of FPSS by using GA. In solving the buffer size, we propose a new GA expression, referred to as the Matrix Encoding Method (MEM) to carry out the GA.

2. Definition of FPSS Model

2.1 Structural Definition

The flexibility of the production process has become very important in new manufacturing industries [19]. The importance of flexibility increases the production systems complexity. Thus, recently, complicated production systems such as FPSS are being widely used in industries that require a high variety of production.

One of this paper goals is to model FPSS. After modeling, we use the FPSS model to study its buffer size decision-making. In general, many kinds of FPSS structures can be

Manuscript received January 5, 2007
Manuscript revised January 25, 2007

considered. This research defines the following FPSS as our FPSS model. The FPSS has 5 main components lines, the main production line and 4 sub-lines, parallel lines, rework path, feed-forward line and feeder line. The research adopts the structure of the 5 main components lines as shown in Figure 1.

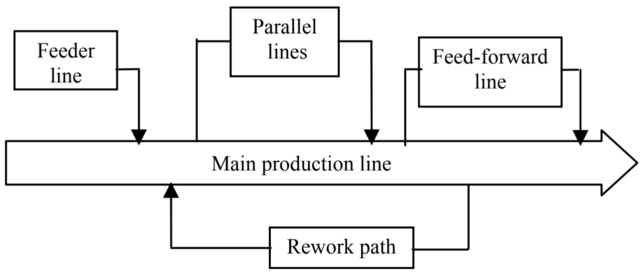

Fig. 1 FPSS components lines

The details of the 5 main components lines are described below.

● Main production line: The main production line is typically the principle production line which is used to manufacture products and is shared by most of the products produced by FPSS. The main production line terminates at the end of FPSS.
● Parallel lines: The parallel lines are a splitting of a main production line into two or more parallel lines and merging into a main line again. Parallel lines are used to increase production capacity or product varieties.
● Feed-forward line: The feed-forward line is used to bypass some special operations and is terminated by re-inserting the manufactured product into the main production line.
● Rework path: The rework path is included for reprocessing the job to repair some defaults in the part to attain quality standards. The damaged parts are rejected from FPSS by this line.
● Feeder line: The feeder line is used to bring two or more parts together to form a single part.

By taking into account the component lines described above, the FPSS model this paper studies is defined as shown in Figure 2. The rectangles of the model represent the machine tools and the circles represent the buffer size. Each pair of machine tools in the model is separated by a buffer size. However, there is no buffer size in front of the first machine tool in each main production line and feeder line. Also, there is no buffer size next to the last machine tool in the model.

The notations in Figure 2 are described as follows.

M_j^i	Machine tool j in Line i.
$S[i]$	Number of buffer sizes in Line i.
M_a, M_b and M_e	Feeder, rework path and feed-forward merge machine tools, respectively.
M_c and M_d	Feed-forward and rework path split machine tools, respectively.
B_s and B_m	Parallel split and merge buffer sizes, respectively.
k	Number of parallel lines

2.2 Digital Code Definition

The other goal of this paper is to find the buffer size of the FPSS model of Figure 2. To find it, we need to define FPSS buffer size as a digital expression. In order to do this, we adopt three procedures. The first one is that FPSS is divided into $11+k$ lines as shown in Figure 3. The second is the FPSS buffer size is expressed as a matrix. The matrix element values represent the buffer size. Each buffer size is located in the matrix according to its location in FPSS. The third is that we make the buffer size matrix by using an algorithm to make the buffer size matrix. Before describing the algorithm, the divided $11+k$ lines are defined as below.

Line 1: starts from the beginning of the main production line until the merge point of the main production line and the feeder line.
Line 2: is the feeder line.
Line 3: starts from the merge point of the main production line and the feeder line until the merge point of the main production line and the rework path.
Line 4: starts from the merge point of the main production line and rework path until the start point of the parallel lines.
Line 5: starts from the end point of the parallel lines until the start point of the feed forward line.
Line 6: represents the part of the main production line from the split of the feed forward line until the split of the rework path.
Line 7: represents the part of the main production line from the split of the rework path until the merge of the feed forward line.
Line 8: represents the part of the main production line from the merge of the feed forward line until the end of FPSS.
Line 9: is the feed forward line.
Line 10: is the part of the rework path until the point just before the scrap point.
Line 11: is the part of the rework path before the scrap point.
Lines 12 to 11+k: are the parallel lines.

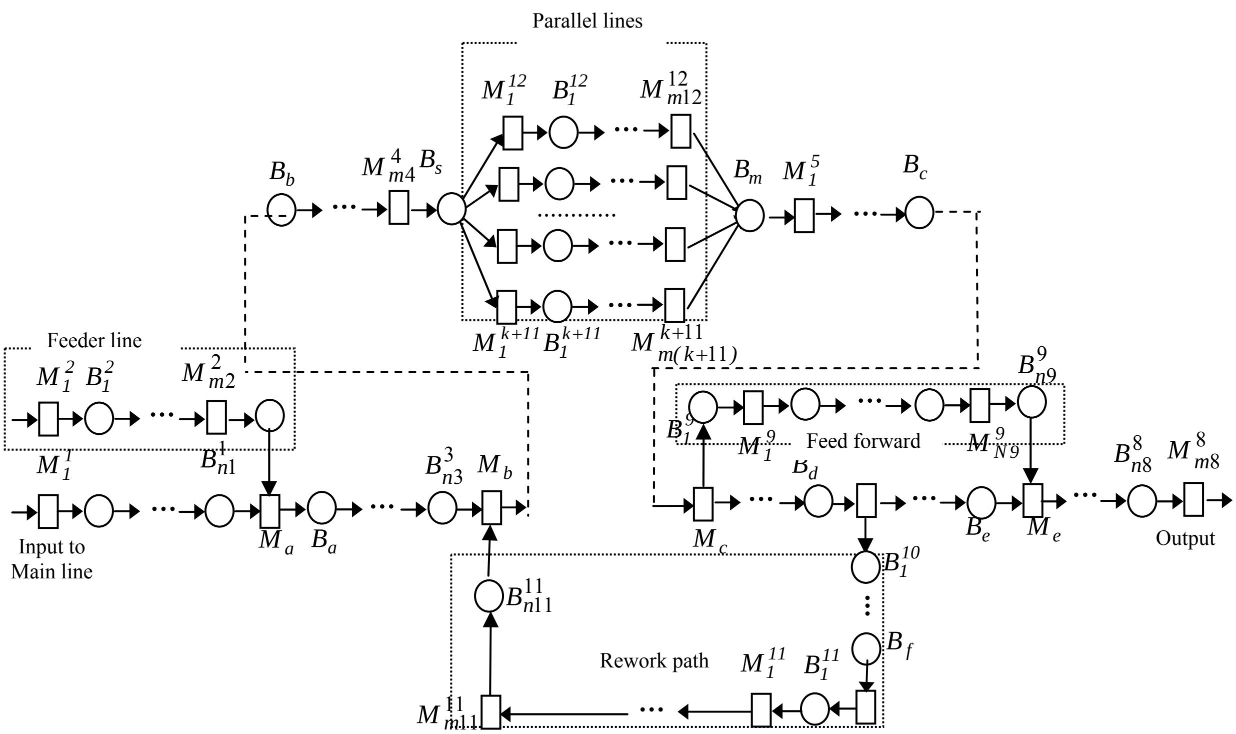

Fig. 2 FPSS model

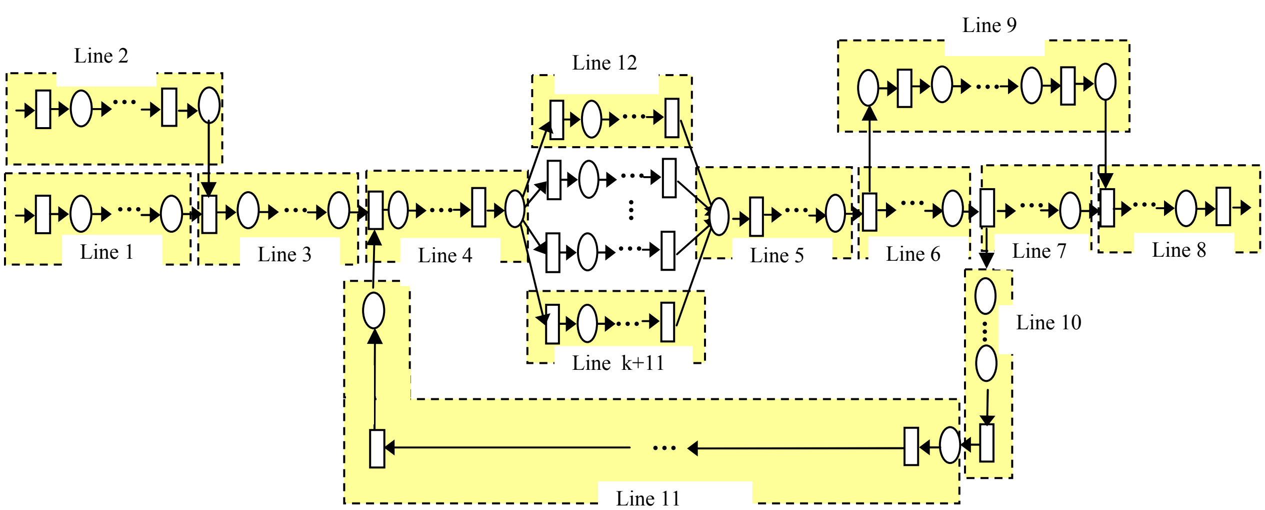

Fig. 3 11+k lines of FPSS

[Algorithm to make buffer size matrix]

Step 1: Find the number of buffer sizes in Line i for each $i = 1,2,\dots, 11+k$.
Step 2: Find N. N is the maximum buffer size number.
Step 3: Reserve $M \times N$ matrix. M is equal to the divided FPSS lines number.
Step 4: Set $i = 1$.
Step 5: Put B_i^1 into the first element of row i. B_i^1 is the first buffer size in Line i. Put B_i^2 into the second element of row i and so on.
Step 6: Set $i = i+1$.
Step 7: If $i < 11+k$, *go to* Step 5, [*Else*] *go to* Step 8.
Step 8: Put zero into all other matrix elements and end the algorithm.

By using the above algorithm, we can make the buffer size matrix. The buffer size matrix, *BM*, is given as blow.

$$BM = \begin{bmatrix} B_1^1 & B_2^1 & \cdots & B_{S[1]-1}^1 & B_{S[1]}^1 \\ B_1^2 & B_2^2 & \cdots & B_{S[2]-1}^2 & B_{S[2]}^2 \\ & \vdots & \vdots & \vdots & \\ B_1^{K+11} & B_2^{K+11} & \cdots & B_{S[K+11]-1}^{K+11} & B_{S[K+11]}^{K+11} \end{bmatrix} \quad (1)$$

where $S[i]$ is the number of buffer sizes in Line i.
The values of the element B_j^i in the matrix in Eq. (1) indicates the buffer size j in Line i in FPSS.

The above algorithm example is introduced. Consider the example of FPSS shown in Figure 4. The example FPSS is divided into 15 lines. The buffer size matrix of the example can be expressed as the following.

[1] $S[1], S[2], \dots$ and $S[15]$ of Step 1 become 3, 3, 2, 2, 2, 2, 2, 3, 4, 2, 4, 2, 2, 2 and 2 respectively.
[2] N of Step 2 equals 4.
[3] 15 and 4 correspond to M and N of Step 3.
[4] When $i=1$, the number of buffer sizes is 3. The buffer sizes are 2, 5 and 8. Put 2 into the first element of row *1*, put 5 into the second element of row 2 and put 8 into the third element of row 1. Following Step 5 to Step 7, the result as shown in Figure 5-B is acquired.
[5] Figure 5-C is the acquired matrix of Step 8.

$$\begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \quad \begin{bmatrix} 2 & 5 & 8 & * \\ 5 & 3 & 9 & * \\ 1 & 3 & * & * \\ 5 & 2 & * & * \\ 9 & 5 & * & * \\ 4 & 6 & * & * \\ 8 & 1 & * & * \\ 9 & 4 & 5 & * \\ 5 & 2 & 7 & 5 \\ 1 & 3 & * & * \\ 3 & 5 & 3 & 5 \\ 5 & 3 & * & * \\ 4 & 3 & * & * \\ 5 & 2 & * & * \\ 7 & 4 & * & * \end{bmatrix} \quad \begin{bmatrix} 2 & 5 & 8 & 0 \\ 5 & 3 & 9 & 0 \\ 1 & 3 & 0 & 0 \\ 5 & 2 & 0 & 0 \\ 9 & 5 & 0 & 0 \\ 4 & 6 & 0 & 0 \\ 8 & 1 & 0 & 0 \\ 9 & 4 & 5 & 0 \\ 5 & 2 & 7 & 5 \\ 1 & 3 & 0 & 0 \\ 3 & 5 & 3 & 5 \\ 5 & 3 & 0 & 0 \\ 4 & 3 & 0 & 0 \\ 5 & 2 & 0 & 0 \\ 7 & 4 & 0 & 0 \end{bmatrix}$$

A B C

Fig. 5 Buffer size matrix examples

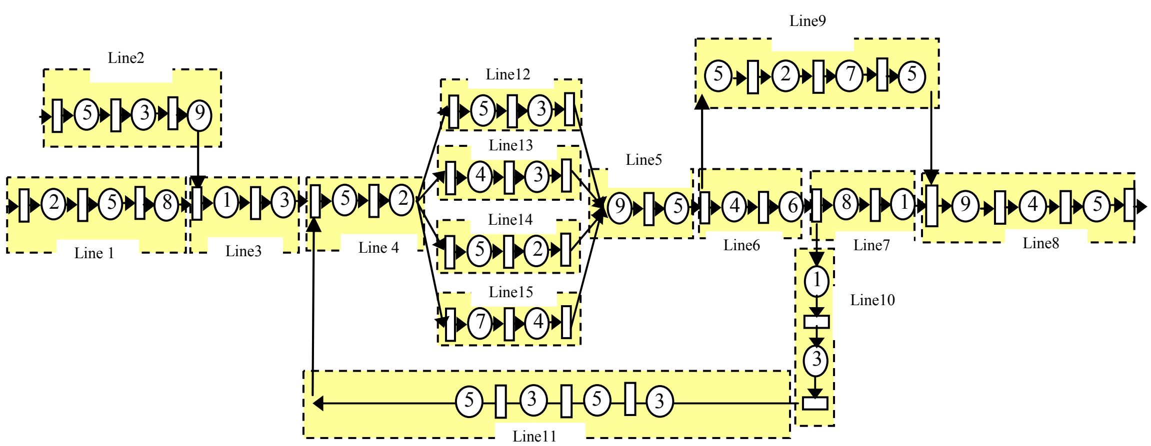

Fig. 4 FPSS example

3. MEM and its GA

The second goal of this paper is to find the buffer size of FPSS utilizing GA. To use GA, we adopt each buffer size between machine tools as a gene. To express the genes, we propose a new gene expression method called MEM. One of MEM's characteristics is that it expresses the genes as a matrix.

3.1 Matrix Encoding Method

The conventional GA expresses a gene with a linear gene expression method. In the case we are studying here, it is difficult to use a linear gene expression method to find the FPSS buffer size. This is because the buffer sizes in FPSS are arranged as an $M \times N$ matrix as we described in section 2.2, and it is impossible to express the matrix with conventional gene expression methods. In order to solve this problem, we propose our MEM as a new gene expression method. The MEM codes the gene expression according to the buffer size matrix. The MEM gene expression is a $M \times N$ matrix similar to the buffer size matrix. The MEM codes the gene expression represented by B_j^i in Eq. (1) as a gene, G_j^i , for all values of *i* and *j*. The general MEM gene expression is shown in Eq. (2). Hereafter, *individual* is used instead of gene expression.

$$Individual = \begin{bmatrix} G_1^1 & G_2^1 & \dots & G_{S[1]-1}^1 & G_{S[1]}^1 \\ G_1^2 & G_2^2 & \dots & G_{S[2]-1}^2 & G_{S[2]}^2 \\ & \vdots & \vdots & \vdots & \\ G_1^{K+11} & G_1^{K+11} & \dots & G_{S[K+11]-1}^{K+11} & G_{S[K+11]}^{K+11} \end{bmatrix} \quad (2)$$

The number of columns of the matrix is not limited, there for, the MEM can deal with any FPSS with any number of machine tools in each sub-line.

3.2 Crossover by MEM

The crossover operations by MEM are different from the crossover operations using a conventional gene expression method. The main difference between our MEM crossover and conventional crossover is that our MEM crossover operation is applied using a crossover line instead of a crossover point used in conventional expression methods.

The crossover by our MEM is carried out using the following steps.

Step1: Randomly select two individuals from the current population according to their fitness.
Step2: Select *11+k* crossover points, CP_i, as follows.

$$CP_i = Random(1,\dots,S[i]-1)\ ,\ \forall i = 1,2,\dots,(11+k)$$

Step3: Define the imaginary crossover stepped line that links each crossover point.
Step4: Swap the genes after the imaginary crossover stepped line between the two individuals

The following example describes the steps of the crossover operations.

[1] Figure 6 shows two selected individual of Step 1.

$$\begin{bmatrix} 2 & 6 & 4 & 9 & 0 & 0 & 0 \\ 4 & 7 & 5 & 8 & 10 & 2 & 5 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 7 & 3 & 9 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 6 & 1 & 4 & 4 & 0 & 0 & 0 \\ 2 & 5 & 3 & 8 & 8 & 3 & 5 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 9 & 6 & 7 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Fig. 6 Two selected individuals

[2] CP_1, CP_2, … and $CP_{(11+k)}$ of Figure 7 are the crossover points selected from Step 2.

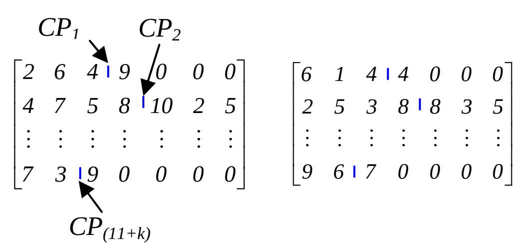

Fig. 7 Selecting crossover points

[3] Figure 8 shows the imaginary crossover stepped line of Step 3.

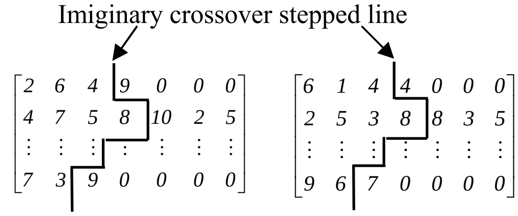

Fig. 8 Imaginary crossover line

[4] Figure 9 is an example from Step 4. Figure 10 shows the individuals after crossover.

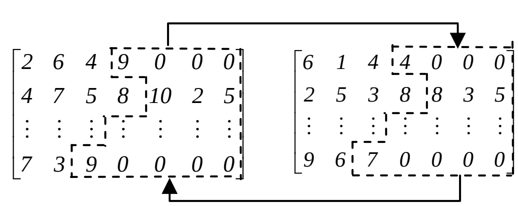

Fig. 9 Swap genes between the two individuals

$$\begin{bmatrix} 2 & 6 & 4 & 4 & 0 & 0 & 0 \\ 4 & 7 & 5 & 8 & 8 & 3 & 5 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 7 & 3 & 7 & 0 & 0 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 6 & 1 & 4 & 9 & 0 & 0 & 0 \\ 2 & 5 & 3 & 8 & 10 & 2 & 5 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 9 & 6 & 9 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Fig. 10 Individual after crossover

3.3 Mutation by MEM

The mutation by our MEM is also different. The characteristic of the mutation is to change the value of one gene for each row in the individual. The mutation is carried out using the following steps.

Step 1: Randomly select an individual from the current population.
Step 2: Select the *11+k* mutation locations, MP_i, as follows.

$$MP_i = Random\left(1, \ldots, S[i]\right),\ \forall i = 1,2,\ldots,(11+k)$$

Step 3: Replace the values in the selected locations by a new value; the new values are randomly selected from (*1* ~ *S*), where *S* is the maximum capacity of the buffer size.

The following example describes the crossover by our MEM.

[1] Figure 11 shows the individual of Step1.

$$\begin{bmatrix} 6 & 1 & 4 & 9 & 0 & 0 & 0 \\ 2 & 5 & 3 & 8 & 10 & 2 & 5 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 2 & 5 & 9 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Fig. 11 Selected individual

[2] MP_1, MP_2, … and $MP_{(11+k)}$ of Figure 12 are the locations selected from Step 2.

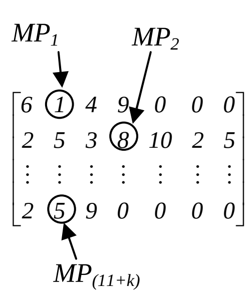

Fig. 12 Selecting the mutation places

[3] Assume that the new values of MP_1, MP_2, … and $MP_{(11+k)}$ selected according to Step 3 are 3, 7, … and 2. As a result, Figure 13 is acquired.

$$\begin{bmatrix} 6 & \underline{3} & 4 & 9 & 0 & 0 & 0 \\ 2 & 5 & 3 & \underline{7} & 10 & 2 & 5 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 2 & \underline{2} & 9 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Fig. 13 Individual after mutation

3.4 Fitness Calculation

The fitness represents the throughput of FPSS and it can be calculated by using Eq. (3).

$$\text{Fitness} = \frac{\text{Actual number of parts produced}}{\text{Theoretical number of pars that can be produced}} \qquad (3)$$

4. Numerical Example

As an example, we applied FPSS with 5 main component lines as shown in Figure 3. The FPSS we adopted contains 180 machine tools. The numbers of machine tools in the 5 component lines are as shown in Figure 14.

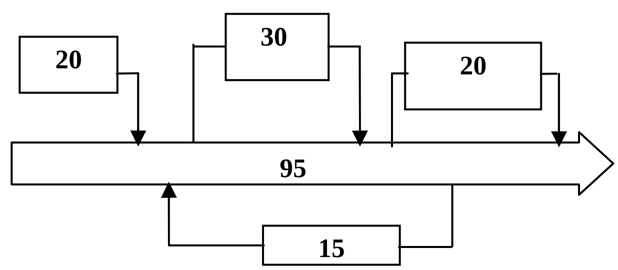

Fig. 14 Number of machine tools of FPSS component lines example

4.1 FPSS production conditions

The FPSS we adopted in the example has the following characteristics:
1. The number of machine tools for each line is given below.
Lines 1, 2, 8 and 9 have 20 machine tools for each. Line 3 has 15 machine tools. Lines 4, 5, 6, 7 and 11 have 10 machine tools for each. Line 10 has 5 machine tools. Lines 12, 13, 14, 15, 16 and 17 have 5 machine tools for each.
2. The parts sequences input into Line 1 and Line 2 are decided by using one to one method [20]; the parts varieties input into line 1 and into line 2 are 10 parts.
3. The machining time for each part (for the 20 parts) in each machine tool is chosen between 15-20 seconds.
4. Each machine tool stops 6 times an hour and stopping time is between 15 seconds.
5. Each machine tool stops for a quality check every 100

parts. Stopping time =15 seconds.

6. FPSS cycle time is 20 seconds.

7. Working time is 8 hours. The production ratio for each part is assumed to be between 50 and 100.

8. The maximum capacity of buffer size is 10.

9. A part is defective with probability $\alpha = 0.2$ at machine tool m_r . At machine tool m_f , a part has probability $\gamma = 0.5$ to be sent to the feed-forward line.

4.2 Results

The example adopts $k+11$ as 17. First, we describe the MEM gene expression matrix. The matrix of FPSS buffer size example has 20 columns and 17 rows. One of the individuals of the initial population was given in Figure 15. We simulated many trails to find the buffer size. After a number of GA generations based on the proposed MEM, the fitness reaches its maximum value. Figure 16 shows one of the best fitness curves. The fitness increases with the generations from approximately 75% to more than 88%, indicating that FPSS buffer size has improved. In other words, MEM utilizing GA expressions is useful for achieving the near optimal FPSS buffer size.

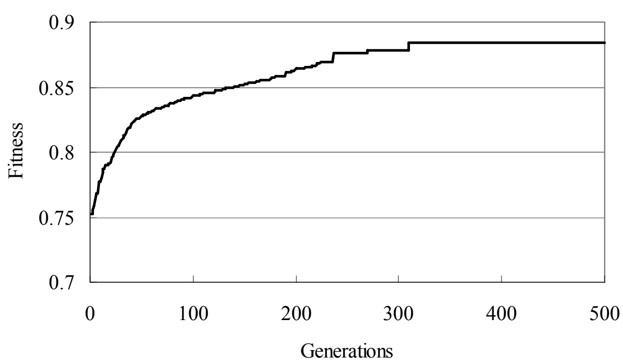

Fig. 16 Best Fitness Curve

$$
\begin{bmatrix}
10 & 4 & 5 & 1 & 1 & 8 & 7 & 2 & 10 & 1 & 5 & 10 & 3 & 5 & 9 & 4 & 8 & 8 & 1 & 3 \\
7 & 9 & 10 & 5 & 10 & 4 & 7 & 4 & 7 & 10 & 10 & 10 & 2 & 6 & 1 & 7 & 10 & 9 & 2 & 9 \\
1 & 7 & 6 & 2 & 5 & 4 & 4 & 2 & 9 & 3 & 7 & 3 & 4 & 8 & 3 & 0 & 0 & 0 & 0 & 0 \\
7 & 9 & 2 & 7 & 5 & 9 & 6 & 3 & 3 & 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
10 & 2 & 5 & 4 & 9 & 3 & 4 & 5 & 3 & 2 & 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
9 & 1 & 7 & 9 & 1 & 3 & 10 & 6 & 1 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 5 & 5 & 3 & 8 & 9 & 6 & 2 & 9 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 7 & 6 & 8 & 10 & 3 & 2 & 3 & 2 & 3 & 5 & 8 & 7 & 6 & 6 & 9 & 8 & 4 & 9 & 3 \\
16 & 10 & 4 & 2 & 9 & 7 & 4 & 10 & 3 & 6 & 6 & 5 & 2 & 5 & 10 & 1 & 3 & 3 & 5 & 5 \\
5 & 3 & 5 & 5 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
8 & 5 & 4 & 5 & 9 & 10 & 6 & 8 & 4 & 4 & 9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
5 & 7 & 1 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
6 & 10 & 6 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 2 & 7 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 4 & 10 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
7 & 3 & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 6 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
$$

Fig.15 One of the initial population individual

One of the acquired FPSS buffer size was given in Table 1.

Table 1: Acquired buffer size

	$B_j^{i\,*}$																				
i \ *j*	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	
1	1	3	4	7	9	10	2	6	9	4	1	8	10	1	1	4	10	2	3	8	
2	2	7	8	6	3	8	1	9	1	5	6	1	9	6	1	5	10	9	5	9	
3	10	3	10	2	2	4	3	8	2	10	3	4	8	2	6						
4	2	5	7	8	6	1	10	7	8	7											
5	2	2	10	10	8	7	7	8	7	10	9										
6	9	1	2	7	6	9	10	3	6	7											
7	9	10	7	6	4	1	2	8	9	9											
8	6	1	3	4	2	2	3	1	9	4	2	2	8	10	9	2	4	6	4	2	10
9	1	8	3	8	5	6	5	2	2	1	1	3	8	5	9	5	10	5	2	1	5
10	4	2	9	2	8																
11	6	5	3	1	9	1	9	3	4	5	3										
12	5	1	6	7																	
13	9	3	2	10																	
14	2	6	8	7																	
15	2	5	2	10																	
16	5	7	9	2																	
17	9	9	1	1																	

* the buffer size *j* located in line *i*

5. Conclusions

This study described a proposal for a model FPSS. We used this FPSS model to study buffer size decision-making. In addition to the main production line, our FPSS model contained 4 sub-lines, i.e., parallel lines, a rework path, a feed-forward line and a feeder line. We also proposed MEM as a new GA expression method to carry out the GA. By applying crossover and mutation techniques, we could determine the genes corresponding to FPSS buffer size.

We used our developed FPSS model to determine some FPSS buffer sizes. The FPSS had 180 machine tools. The parts varieties included 20 parts which were input into FPSS, and the FPSS was operated under specified conditions. As a result, FPSS throughput reached its maximum value after 1300 generations. In other words, the GA based on the proposed expression MEM is useful for achieving near optimal FPSS buffer size. The results of the study can be used to improve production efficiency, and production engineers can use these results when making decisions regarding buffer size.

Manuscript received January 5, 2007
Manuscript revised January 25, 2007

References

[1] Altiparmak, F., Bulgak, A. A., Optimization of Buffer Sizes in Assembly Systems Using Intelligent Techniques. *In Winter Simulation Conference* (2002), pp. 1157-1162, CA., USA.

[2] Bulgak, A. A., Diwan, P. D. and Inozu, B., Buffer Size Optimization in Asynchronous Assembly Systems Using Genetic Algorithms. *Computers and Industrial Engineering*, Vol. 28, No. 2 (1995), pp. 309-322.

[3] Hillier, F. S., So, K. C. and Boling, R. W., Notes: Toward Characterizing the Optimal Allocation of Storage Space in Production Line Systems with Variable Processing Times. *Management Science*, Vol. 39 No. 1 (1993), pp. 126-133.

[4] Gershwin, S. and Schor, J., Efficient Algorithms for Buffer Space Allocation. *In International Workshop on Performance Evaluation and Optimization of Production Lines* (1997), pp. 217-228, Samos, Greece, University of the Aegean, Department of Mathematics.

[5] Enginarlar, E., Li, J., Meerkov, S. M. and Zhang, R. Q., Buffer Capacity for Accommodating Machine Downtime in Serial Production Lines. *International Journal of Production Research*, Vol. 40, No. 3 (2002), pp. 601-624.

[6] Vouros, G.A., Papadopoulos, H. T., Buffer Allocation in unreliable production lines using a knowledge based system, *Computers and Operations Research,* Vol. 25, No. 12. (1998), pp. 1055-1067.

[7] Spinellis, D. D. and Papadopoulos, C. T., A simulated annealing approach for buffer allocation in reliable

production lines, *Ann. Oper. Res*, Vol. 93 (2000), pp. 373–384

[8] Jafari, M. A. and Shanthikumar, J. G., Determination of optimal buffer storage capacities and optimal allocation in multistage automatic transfer lines, *IIE Transactions*, Vol. 21, No. 2 (1989), pp. 130-135..

[9] Goldberg, D. E., *Genetic Algorithms: In Search of Optimization & Machine Learning* (1989), Addison-Wesley.

[10] Lawrence, D., *Handbook of Genetic Algorithms* (1991), Van Nostrand Reinhold, New York.

[11] Askin, R.G. and Standrige, C.R. (1993). Modeling and Analysis of Manufacturing Systems. John Wily, New York.

[12] Papadopoulos, H .T., Heavey, C. and Browne, J. (1993). Queueing Theory in Manufacturing Systems Analysis and Design, Chapman&Hall, London.

[13] Buzacott, J .A. and Shanthikumar, J. G. (1993). Stochastic Models of Manufacturing Systems. Prentice Hall, New Jersey.

[14] Perros, H.(1994). Queueing Network with blocking. Oxford University Press.

[15] Gershwin, S. B. (1994). Manufacturing Systems Engineering, Prentice Hall, New Jersey.

[16] Altiok, T. (1997). Performance Analysis of Manufacturing Systems, Springer-Verlag, NewYork.

[17] Dallery, Y. and Gershwin, S. B. (1992). Manufacturing Flow Line Systems: A Review of Models and Analytical Results, Queueing Systems Theory and Applications. 12(1-2): 3–94.

[18] Papadopoulos, H.T. and Heavey, C. (1996). Queueing Theory in Manufacturing Systems Analysis and Design: A Classification of Models for Production and Transfer Lines, European journal of operational Research, 92(1): 1–27.

[19] Bussmann, S. and Schild, K. (2001). An Agent-based Approach to the Control of Flexible Production Systems. *8th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2001)*, pp. 169 – 174, France.

[20] Yamamoto, H (2000). One-by One Parts Input Method by off-line Production Simulator System with GA. *European Journal of Automation, Hermes Science Publication*, pp. 1173 – 1186.

[21] Yamamoto, H., Marui, E. and Abu Qudeiri, J. (2003). Development of New FTL Simulator for Buffer Size Decision. IEEE International Symposium on Computational CIRA03", Kobe, Japan, pp1358-1363.

Jaber Abu Qudeiri received the B.S. degree in Mechanical Engineering from University of Jordan in 1992 and his M.S. of Mechanical and Systems Engineering from Gifu University in 2005. He is a PhD student of Gifu University. His major fields of study are genetic algorithm, production system simulation, CAD/CAM and production systems layout design.

Hidehiko Yamamoto is a Professor in the Department of Human and Information System Engineering, Gifu University, Japan. He received his B. Eng. and M. Eng from Nagoya Institute of Technology, Japan. After graduating, he joined Toyota Industries Corporation and developed many production lines for automotive parts. While working for Toyota, he received his PhD from Nagoya Institute of Technology, Japan. His research interests include production engineering, intelligent system, automation and knowledge learning.

Rizauddin Bin Ramli was born in Malaysia (1973). He received his B. Eng from Kyoto University of Japan in 1997 and his M. Sc of Mechanical and Systems Engineering from Gifu University in 2005. He is a PhD student of Gifu University. His major fields of study are genetic algorithm, production system simulation and robotics.