

## **Development of Production Simulator for Buffer Size Decisions in Complex Production Systems Using Genetic Algorithm\***

Jaber ABU QUDEIRI<sup>\*\*</sup>, Hidehiko YAMAMOTO<sup>\*\*</sup>,  
Rizauddin RAMLI<sup>\*\*</sup> and Khalid R. Al-MOMANI<sup>\*\*\*</sup>

<sup>\*\*</sup>Intelligent Manufacturing Systems Laboratory, Gifu University

1-1 Yanagido, Gifu Shi, 501-1193, Japan

<sup>\*\*\*</sup> King Abdullah University Hospital, Jordan

k3812203@edu.gifu-u.ac.jp

### **Abstract**

Deciding on buffer size for production lines has gained more and more importance because of growing complexity of the production lines and production costs. Many researches have developed numerous techniques for gaining the optimal buffer size which can aid in achieving maximum production rate. One of the search methods that can be used for studying buffer size in production lines is a genetic algorithm (GA). In this paper, we propose a production simulator (PS) to determine optimal or near optimal buffer size and, at the same time, to minimize the number of generations required to reach this optimal buffer size for the complex production systems (CPS). Our PS consists of a GA and a discrete event. For the most efficient use of the GA we introduce a new encoding method for GA called a Multi-Vector Encoding Method (MVEM).

**Key words:** Complex Production System, Multi Vector Encoding Method, Buffer Size, Genetic Algorithm, Production Simulator.

### **1. Introduction**

Deciding on buffer size for production lines has gained more and more importance because of the growing complexity of the production lines and production costs. Many articles and researches related to buffer size have been published<sup>(1), (2), (3), (4)</sup>. 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 is a model or approach that can be used to evaluate and measure production line performance. The buffer size is optimized via various techniques, such as functional approximation and evaluation<sup>(5)</sup>, knowledge based methods<sup>(6)</sup>, simulated annealing<sup>(7)</sup>, dynamic programming method<sup>(8)</sup>, and other search methods. One of the search methods that can be used for studying the buffer size in production lines is a genetic algorithm (GA)<sup>(9) (10) (11)</sup>. 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 first and most important step in GA is encoding. The conventional GA operation is generally based on an individual with a linear gene arrangement, which represents the genes as a single line (vector) inside the individual. However, for some complex production systems (CPS), it is difficult to use a linear arrangement of genes in individuals to find a

buffer size. This is because the CPS buffers are arranged in more than one line, and it is impossible to express such an arrangement with conventional gene arrangement methods. Another technique for solving this problem is called the matrix encoding method, which is based on a matrix arrangement for the genes in the individual by dividing the CPS into cells and encoding the buffers in its matched cell in the matrix.

In this paper, we propose a production simulator (PS) to find the optimal or near optimal buffer size of CPS, while at the same time, to minimize the number of generations to reach this optimal buffer size for the CPS. Our PS consists of a GA and a discrete event. For some complex problems, the conventional type of crossover and mutation needs a lot of generations to reach the best answer (individual). This paper introduces a new encoding method for GA called a Multi-Vector Encoding Method (MVEM). MVEM can achieve efficient use of the GA. Before going through the proposed MVEM, a typical structure of CPS is explained in § 2.

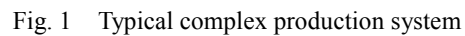
## 2. Typical Structure of CPS and Model Assumptions

Production lines with other complex structures, such as parallel, rework, feed-forward, etc., are widely used in high volume industries. A typical structure of a CPS is shown in Figure 1, where the rectangles represent the machine tools and circles represent the buffers. The main operation line is typically the principle production line which is used to manufacture products and is shared by most of the products produced by the CPS. The main production line terminates at the end of the CPS. In addition to the main production line, the CPS comprises part or all of the following production lines.

- Parallel lines: parallel lines are a splitting of a main production line into two or more parallel lines, merging into a main line again. Parallel lines are often used to increase production capacity or product varieties in many productions.
- Rework paths: rework paths are often included into the CPS for reprocessing or multi-processing the jobs in order to attain higher quality. In the rework path, rejected parts are repaired, and after remedying their faults, the parts will join the main production line again for re-processing once rework is completed.
- Feed-forward lines: feed-forward lines run parallel to the main production line and are terminated by inserting the manufactured product into the main production line.
- Feeder lines: Feeder lines, also called assembly lines, are used to bring two or more parts together to form a single part.

A description of the notations of machine tools and buffers introduced to the decomposed lines are as follows:

$$\begin{aligned}
 \text{Line 1} &: M_1^1, \dots, M_{N1}^1, B_1^1, \dots, B_{N1}^1 \\
 \text{Line 2} &: M_1^2, \dots, M_{N2}^2, B_1^2, \dots, B_{N2}^2 \\
 \text{Line 3} &: M_a, M_1^3, \dots, M_{N3-1}^3, B_1^3, \dots, B_{N3}^3 \\
 \text{Line 4} &: M_b, M_1^4, \dots, M_{N4-1}^4, M_b, B_b, B_1^4, \dots, B_{N4-1}^4, B_s \\
 \text{Line 5} &: M_1^5, \dots, M_{N5}^5, M_c, B_m, B_1^5, \dots, B_{N5-1}^5, B_c \\
 \text{Line 6} &: M_c, M_1^6, \dots, M_{N6}^6, B_d, B_1^6, \dots, B_{N6-1}^6 \\
 \text{Line 7} &: M_d, M_1^7, \dots, M_{N7-1}^7, M_e, B_e, B_1^7, \dots, B_{N7-1}^7 \\
 \text{Line 8} &: M_e, M_1^8, \dots, M_{N8-1}^8, B_1^8, \dots, B_{N8-1}^8 \\
 \text{Line 9} &: M_1^9, \dots, M_{N9}^9, B_1^9, \dots, B_{N9+1}^9 \\
 \text{Line 10} &: M_1^{10}, \dots, M_{N10-1}^{10}, M_f, B_f, B_1^{10}, \dots, B_{N10-1}^{10} \\
 \text{Line 11} &: M_f, M_1^{11}, \dots, M_{N11-1}^{11}, B_1^{11}, \dots, B_{N11+1}^{11} \\
 \text{Line } 11+i &: M_1^{11+i}, \dots, M_{N(11+i)}^{11+i}, B_1^{11+i}, \dots, B_{N(11+i)-1}^{11+i}
 \end{aligned}$$



420

Where

$M_j^i$	Machine tool $j$ in line $i$ .
$N_i$	Number of machine tools in line $i$ .
$M_w$ , $M_b$ and $M_e$	Feeder, rework path and feed-forward merged machine tools, respectively.
$M_c$ and $M_d$	Feed-forward and rework path split machine tools, respectively.
$M_f$	Scrap machine tool.
$B_s$ and $B_m$	Parallel split and merge buffers, respectively.
$k$	The number of parallel lines

Each pair of machine tools in the CPS is separated by a buffer, with no buffer in front of the first machine tool in the main production line, and the feeder line and products are assumed to be available at all times in front of the bay of the main production line and feeder line. Also there is no buffer next to the last machine tool in the CPS.

The following assumptions are introduced for the CPS model used in this paper:

1. Each machine tool in the complex production system has two states: up and down. When the machine tool is down, no production takes place. When up, the machine tools in the main production line, feeder line, feed-forward line and rework path are capable for production.
2. The capacity of each buffer  $j$  for each Line  $i$ ,  $i=1, \dots, k+1$ ,  $j=1, \dots, N_i-1$  is  $B_j^i$ ,  $0 \leq B_j^i < \infty$ .
3. A machine tool is starved at time  $t$  if the upstream buffer is empty at time  $t$ . Machine tools  $M_1^1$  and  $M_2^2$  are never starved. In addition, each of the machine tools that merge two or more lines is starved if all buffers in front of the bay of the merged machine tool are empty. It is assumed that machine tool  $M_b$  always takes parts from the rework path first if it is not empty. The other merge machine tools take parts without any priority.
4. A machine tool is blocked at time  $t$  if the downstream buffer is full at time  $t$ . Machine tool  $M_{N8}^8$  is never blocked. The split machine tool  $M_d$  is blocked by the main production line if it produces a good part and  $B_1^7$  is full, and blocked by the rework path if it produces a defective part and  $B_1^{10}$  is full.
5. A good part is sent to  $B_1^7$  if it is not full and a defective part is sent to  $B_1^{10}$  if it is not full. A part is defective with probability  $\alpha$ ,  $0 \leq \alpha < 1$ . A severe defective part is scrapped with probability  $\beta$  at machine tool  $M_1^{11}$ ,  $0 \leq \beta < 1$ .
6. At machine tool  $M_e$ , a part has a probability  $\gamma$ ,  $0 \leq \gamma < 1$  of being sent to the feed-forward line.
7. The first machine tool in each parallel line has equal probability of taking a part from buffer  $B_s$  if it is not blocked, and the last machine tool in each parallel line has equal probability of sending a part to buffer  $B_m$  if it is not starved.

### 3. Genetic Algorithm

A GA paradigm has been proposed to solve a wide range of problems. GA has been successfully applied to optimization problems in diverse fields. It differs from other search techniques which depend on a natural genetic evaluation process. GA starts with an initial set of solutions randomly selected and called a *population*. A suitable encoding for each solution in the population is used to allow computation of *fitness*. The solution set in the population, called a *chromosome* or *individual*, represents a solution to the optimization problem. Each individual contains a number of *genes*. The individuals in the initial population are *evaluated* to measure its *fitness*. To create the next population, new individuals are formed by either merging two individuals from the current population using a *crossover* operator or modifying an individual solution using a *mutation* operator. Based on the individuals' fitness, the individuals to be included in the next population are then probabilistically selected from the set of individuals in the current population. The iteration, called a *generation* will continue until fitness reaches its maximum value. The best overall solution becomes the candidate solution to the problem.



This paper tries to determine optimal or near optimal buffer size for the CPS. The arrangement of genes in individuals is different from that obtained using the conventional arrangement methods. The operations for our GA are also different. The characteristics of the GA are described in §§ 3-1 ~ 3-5.

### 3.1 Encoding

One of the important tasks in using GA is how to express a chromosome. The encoding process is often the most difficult aspect of solving a problem using GA. When applying them to a specific problem it is often hard to find an appropriate representation of the solution that will be easy to use in the crossover process. The traditional ways to represent a solution for production system are liner encoding method and matrix encoding method.

The linear encoding method is the simplest case of encoding and it's used in many articles<sup>(8), (12), (13), (14)</sup>. In the linear encoding method, the individual is represented as a single line and we are dealing with this line as a chromosome. The individual can be represented as follows:

$$[G_1 \ G_2 \ G_3 \ \dots \ G_{N-1} \ G_N]$$

where  $N$  is the number of genes in each individual.

The Second traditional way to encode the production system is the matrix encoding method. In the matrix encoding method<sup>(13), (15), (16)</sup>, the CPS can be represented as a matrix containing a number of columns and rows depending on the architecture of the CPS. In this method crossover and mutation are applied for the column of genes instead of the single gene in the linear encoding method. The genes in this encoding method are arranged as  $M \times N$  matrix just like the machine tools arrangement. Consider the CPS shown in Figure 3, where the individual can be expressed as shown in Figure 4. This research adopts each buffer size in front of the bay of each machine tool as the gene.

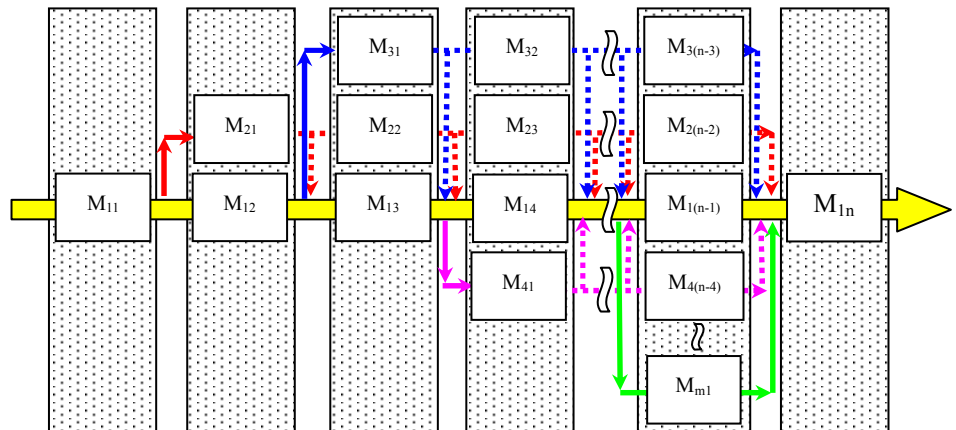


Fig. 3 CPS Example

$$\begin{bmatrix} G_{11} & G_{12} & G_{13} & \dots & G_{1(\hat{n}-1)} & G_{1\hat{n}} \\ G_{21} & G_{22} & G_{23} & \dots & G_{2(\hat{n}-1)} & 0 \\ 0 & G_{32} & G_{33} & \dots & G_{3(\hat{n}-1)} & 0 \\ 0 & 0 & G_{43} & \dots & G_{4(\hat{n}-1)} & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & G_{m(\hat{n}-1)} & 0 \end{bmatrix}$$

Fig. 4 Expressed Individual By Matrix Encoding

In the case we are studying here, namely CPS, it is difficult to use conventional arrangement methods for genes in individuals to find a buffer size. This is because the CPS buffers are arranged in more than one line, and it is impossible to express such an arrangement with conventional gene arrangement methods. In order to solve this problem, we introduce MVEM. MVEM expresses the individuals with multi vectors; each vector represents one part of the production line buffers. According to this arrangement method, the buffer size arrangement for the typical complex production system shown in Figure 1 can be defined as given in Eq. (1).

$$\text{Buffer Size} = \left\{ \begin{bmatrix} B_1^1 & B_2^1 & \dots & B_{N_1-2}^1 & B_{N_1-1}^1 \\ B_1^2 & B_2^2 & \dots & B_{N_2-2}^2 & B_{N_2-1}^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ B_1^{k+11} & B_2^{k+11} & \dots & B_{N_{(k+11)}-2}^{k+11} & B_{N_{(k+11)}-1}^{k+11} \end{bmatrix} \right\} \quad (1)$$

where  $B_j^i$  is the size of buffer  $j$  in line  $i$ .

Each buffer size in the CPS represents a gene, thus the individuals used in our GA can be coded by replacing buffer sizes  $B_j^i$  in Eq. (1) with genes  $G_j^i$  in each line  $i$ ,  $i = 1, \dots, 11 + k$ .  $\forall$  buffer  $j, j = 1, 2, \dots, N_i - 1$ , the individual is defined by Eq. (2).

$$\text{Individual} = \left\{ \begin{array}{ll} V_1 : & \text{For line } 1 \\ V_2 : & \text{For line } 2 \\ & \vdots \\ V_n : & \text{For line } n \end{array} \right\} \quad (2)$$

which gives:

$$\text{individual} = \left\{ \begin{bmatrix} G_1^1 & G_2^1 & \dots & G_{N_1-2}^1 & G_{N_1-1}^1 \\ G_1^2 & G_2^2 & \dots & G_{N_2-2}^2 & G_{N_2-1}^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ G_1^{k+11} & G_2^{k+11} & \dots & G_{N_{(k+11)}-2}^{k+11} & G_{N_{(k+11)}-1}^{k+11} \end{bmatrix} \right\} \quad (3)$$

The number of vectors of the individual is not limited, which means that any CPS with any number of production lines can be dealt with.

### 3.2 Initial Population

The initial population is randomly selected. The initial population contains  $N$  number of individuals. Each individual expresses genes as shown in Eq (3). Each gene in the individuals can be selected as follows.

$$G_j^i = \text{random} \left( 0 \rightarrow B_{\max} \right) \quad (4)$$

$$\forall i = 1 \dots k + 11 \text{ and } j = 1, \dots, N_{(k+11)} - 1$$

Where

$B_{\max}$  Maximum buffer size

$N$  Number of individuals in the Population.

### 3.3 Crossover

The arrangement of genes in individuals using MVEM is different from that which is obtained using a conventional arrangement method. The crossover operations for our GA are also different. The main difference between MVEM crossover and conventional methods crossover is that the crossover operations in MVEM is applied to each vector in the individual, independently, using a different crossover point for each vector. The crossover of our GA is carried out using the following steps. Figure 5 shows the crossover graphically.

**Step1:** Two individuals are selected from the current population according to their fitness.

**Step2:**  $L$  number of crossover points are randomly selected (number of crossover points,  $L$ , equals the number of vectors in the individuals, which equals the number of lines in CPS). The crossover points,  $CP_1, CP_2, \dots, CP_L$  are selected as follows:

$$CP_{line i} = random[1 \rightarrow N_j - 2],$$

where  $j$  refers to the position between genes  $G_j^{line i}$  and  $G_{j+1}^{line i}$ .

**Step3:** All elements in each vector before the crossover points  $G_{cpi}$ ,  $i = 1, 2, \dots, L$  are saved for each of the two individuals selected in step1.

**Step4:** All elements in each vector after the crossover points  $G_{cpi}$ ,  $i = 1, 2, \dots, L$  are saved for each of the two individuals selected in step1.

**Step5:** The elements before crossover points for each vector are copied from one individual and the elements after crossover points for each vector are copied from the second.

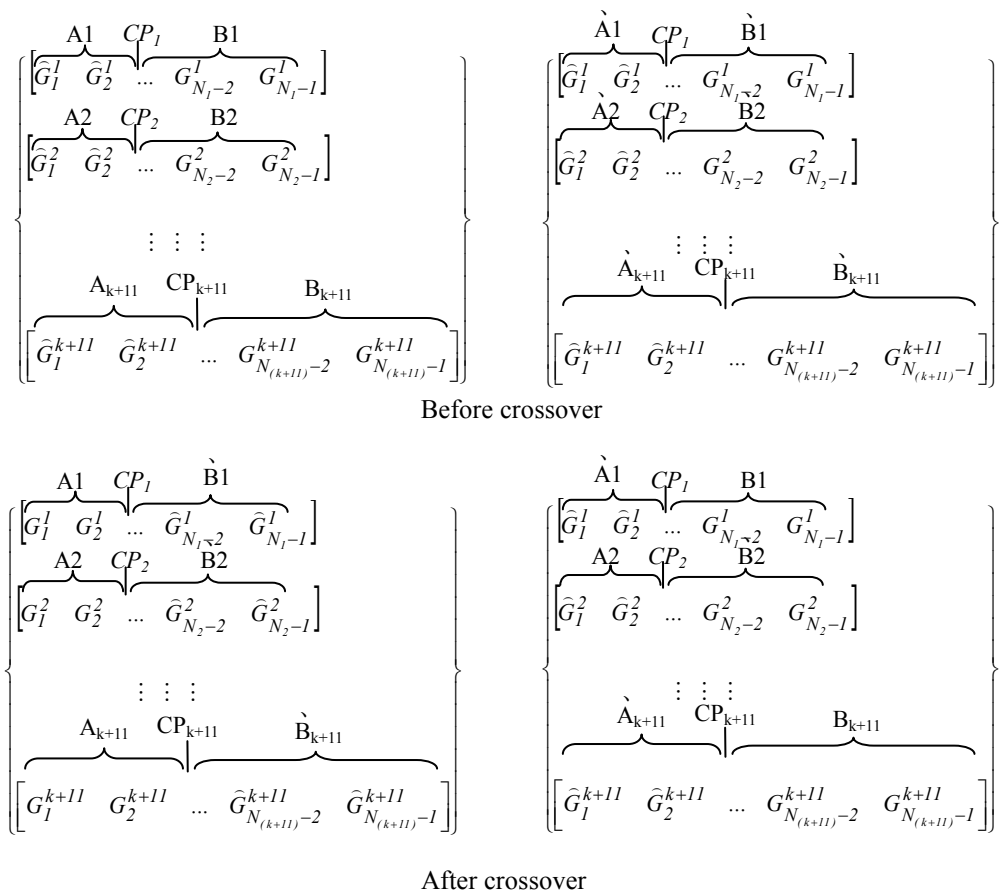


Fig. 5 Individuals before and after crossover

### 3.4 Mutation

The mutation of our GA is different because the gene expression adopts MVEM. The characteristic of the mutation is to change the value of one gene for a vector in the individual. The mutation is carried out using the following steps.

**Step1:** One individual is randomly selected from the current population.

**Step2:** Mutation places are randomly selected, (number of mutation places equals the number of vectors in the individual, one mutation place for each vector). The mutation places,  $MP_i$ ,  $i=1, 2, \dots, L$  are selected as follows

$$MP_i = \text{random} [1 \rightarrow N_j - 1], j \text{ here refers to line number.}$$

**Step3:** The genes in the selected places are replaced by a new value; the new value is randomly selected from  $(1 \sim B_{max})$ . Figure 6 shows the mutation graphically, and assumes that the mutation places for the first, second and last vectors ( $MP_1$ ,  $MP_2$  and  $MP_{k+11}$ ) are selected randomly as 2,  $N_2-2$  and  $i$  respectively,

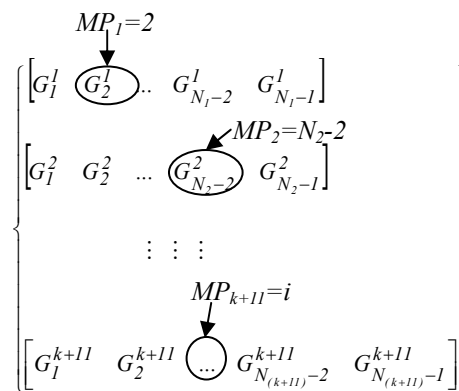


Fig. 6 Mutation

### 3.5 CPS Performance Evaluation

To measure the production efficiency of this system, the PS must find the fitness ( $F$ ) for each individual, which is defined by the following relations:

$$F = \frac{\text{Total production amounts}}{\text{Theoretical production amounts}} \quad (5)$$

The total production amounts are the final number of products at the end of the simulation (real time operation).

Where:

The total production amounts are the final number of products at the end of the simulation (real time operation), and theoretical number of products is given by Eq. (6):

$$\text{Theo. } P = \frac{T}{T_{mc}} \quad (6)$$

Where:

Theo.  $P$                       Theoretical number of products.  
 $T$                               Working time.  
 $T_{mc}$                           Machine tool cycle time.

substituting Eq. (6) in (5) we have:

$$F = \frac{P}{T / T_{mc}} \quad (7)$$

Where  $P$  is the number of products (Real time operation).



#### 4. Production Simulator

To find the buffer size in front of the bay of machine tools in the CPS that maximizes the production efficiency of the CPS, we propose a production simulator (PS). The developed PS consists of a GA and a discrete event as shown in Figure 7. The PS searches the buffer size in front of the bay of each machine tool in the CPS by repeating discrete event and utilizing the GA. A program, using the C++ language, is used to establish the PS. When the simulation is started, the PS reads the specified production system characteristics, such as the number of machine tools, machine tool stops, machining time for each product in each machine tool, production ratio, number of defective products, etc. for the desired CPS. Based on this input, the PS searches the buffer sizes that maximize the throughput of the CPS.

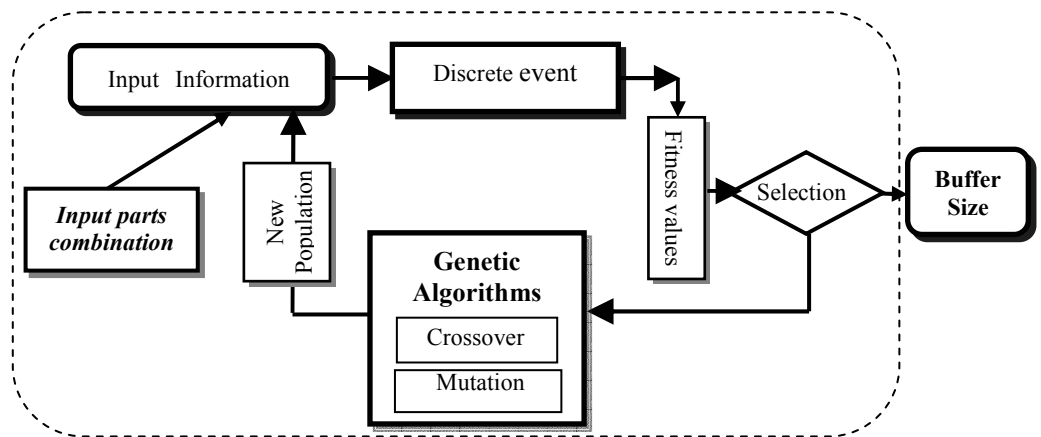


Fig. 7 Genetic algorithm and simulation evaluation

Before presenting the simulator algorithm, the following notations are introduced first:

- N Number of individuals in the population
- s Number of individuals that select on elitist strategy
- $C_r$  Crossover rate
- $M_r$  Mutation rate
- $F(i)$  Fitness of individual  $i$

The simulator algorithm is presented below.

Step 1: Randomly generate the initial population with  $N$  individuals.

Step 2: Read the specified production system characteristics, such as a number of machine tools, machine tools stops, machining time for each product in each machine tool, production ratio, number of defective products, etc. in the desires CPS.

Step 3: Carry out the discrete simulator with the input information and calculate the fitness of each individual by using the discrete simulator results.

Step 4: Select  $s$  individuals that have high fitness based on elitist strategy and send the selected individuals to the next generation.

Step 5: Calculate selection probability using Eq. (8).

$$PR(i) = F(i)^2 / \sum_{i=1}^N F(i)^2 \quad (8)$$

Step 6: Select a pair of individuals by the selection probability based on fitness and carry out crossover operation.

Step 7: Select an individual based on fitness and carry out mutation operation.

Step 8: Apply the following rule:

*if*  $(C_r \times N + M_r \times N) = N - s$ ,

*then*, continue to step 9,

*else*, return to step 6.

Step 9: Regard N individuals acquired in steps 4-8 as the next generation population.

Step 10: Loop until fitness reaches its maximum value.

The relationships between the GA and the discrete event of PS are as follows: The initial individuals corresponding to a buffer size in front of each machine tool are randomly selected and are input into the discrete event. The discrete event simulates many hours' productions and finds production efficiency. After the GA operations in conjunction with the results of the discrete event, the next generation's individuals are generated. The new individuals are input into the discrete event again. In this way, the cycle including the GA system and the discrete event is repeated until a better fitness is found.

## 5. Numerical Experiments

We used our developed PS to determine CPS buffer sizes. The CPS we adopted in this example has 180 machine tools and the maximum buffer size in front of the bay of each machine tool is 10. The production conditions are as follows:

Referring to the CPS shown in Figure 1, the CPS we adopted has the following assumptions:

1- The number of machine tools for each line is given below.

Line 01: 20 machine tools

Line 02: 20 machine tools

Line 03: 15 machine tools

Line 04: 10 machine tools

Line 05: 10 machine tools

Line 06: 10 machine tools

Line 07: 10 machine tools

Line 08: 20 machine tools

Line 09: 20 machine tools

Line 10: 5 machine tools

Line 11: 10 machine tools

Lines 12, 13, 14, 15, 16 and 17 have 5 machine tools for each. Here we choose  $k = 5$ .

2- The parts sequences input into line 1 and line 2 are decided by using one to one method<sup>(17)</sup>, 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 15 sec.

5- Each machine tool stops for a quality check every 100 parts. Stopping time = 15 sec.

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

7- The maximum buffer capacity is 10.

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

9- A severe defective part is scrapped if the part circles for the fourth time in the rework path.

## 6. Results

Figure 8 shows the best fitness curves for the matrix encoding method and MVEM. In this curve, the fitness increases with the generations until it reaches its maximum value, it is clear that the linear encoding method is the least efficient. So, we used the matrix encoding method and MVEM while running a simulation to compare these two methods. Figure 8 shows the best fitness curves for the matrix encoding method and MVEM. Table 1 shows the buffer size of the CPS when the fitness reaches its maximum value.

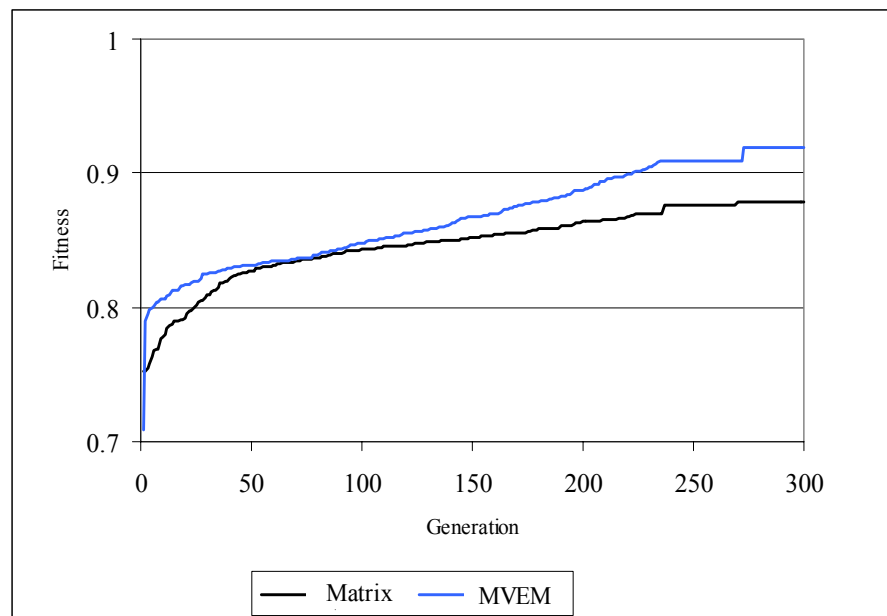


Fig. 8 Best fitness curves for the matrix encoding method and MVEM

**Table 1** Buffer size resulted by PS based on MVEM

Line	Buffer size*																				
L1	1	3	4	7	9	10	2	6	9	4	1	8	10	1	1	4	10	2	3	8	
L2	2	7	8	6	3	8	1	9	1	5	6	1	9	6	1	5	10	9	5	9	
L3	10	3	10	2	2	4	3	8	2	10	3	4	8	2	6						
L4	2	2	10	10	8	7	7	8	7	10											
L5	2	5	7	8	6	1	10	7	8	7	9										
L6	9	1	2	7	6	9	10	3	6	7											
L7	9	10	7	6	4	1	2	8	9	9											
L8	6	1	3	4	2	2	3	1	9	4	2	2	8	10	9	2	4	6	2		
L9	1	8	3	8	5	6	5	2	2	1	1	3	8	5	9	5	10	5	2	1	5
L10	4	2	9	2	8																
L11	6	5	3	1	9	1	9	3	4	5	3										
L12	5	1	6	7	L13			9	3	2	10	L14			2	6	8	7			
L15	2	5	2	10	L16			5	7	9	2	L17			9	9	1	1			

\* The buffer size of each line are arranged respectively

## 7. Conclusions

This study proposed a PS to decide on buffer size. Our PS was based on a GA and a discrete event and that could search for a better buffer size, and could aid in achieving the maximum production rate. This study has described a MVEM for a CPS. An MVEM is adopted for the gene arrangement in each individual. We applied our PS to some CPS examples. After a number of generations, the best size of the buffer of the CPS could be found. The results of the study can be used to improve the production plan, and production engineers can use these results when making decisions on a buffer size. It was found that using the MVEM led us to optimal buffer size with fewer of generations.

## 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) Forrest, S., Genetic Algorithms. *ACM Comput. Surv.*, Vol. 28, No. 1 (1996), pp. 77-83.
- (12) Emanuel Falkenauer. *Genetic Algorithms and Grouping Problems*. JohnWiley and Sons, 1998.
- (13) Homaifar Abdollah, Shanguchuan Guan, and Gunar E. Liepins. *Schema analysis of the traveling salesman problem using genetic algorithms*. *Complex Systems*, 6(2):183-217, 1992.
- (14) Wroblewski Jakub. *Theoretical foundations of order-based genetic algorithms*. *Fundamenta Informaticae*, 28(34):423-430, 1996.
- (15) Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 2nd edition, 1994.
- (16) Yamamoto, H., Abu Qudeiri J. and Marui, E., "Definition of FTL with bypass lines and its simulator for buffer size decision", *International journal for production and economics*, In Press, available online 16 April 2007.
- (17) Yamamoto, H. One-by One Parts Input Method by off-line Production Simulator System with GA. *European Journal of Automation, Hermes Science Publication*, pp. 1173 – 1186, 2000.