# Genetic Algorithms for Buffer Size and Work Stations Capacity in Serial-Parallel Production Lines

Abu Qudeiri Jaber[1], Rizauddin Ramli[2] and Yamamoto Hidehiko[3]
Intelligent Manufacturing Systems Laboratory, Gifu University
1-1 Yanagido, Gifu Shi, 501-1193, Japan
{[1]k3812203, [2]k3812205}@edu.gifu-u.ac.jp, [3]yam-h@gifu-u.ac.jp

## Abstract

Recently, many production lines that have complicated structures such as parallel, reworks, feed-forward, etc. are widely used in high volume industries. Among them, the serial-parallel production line (S-PPL) is one of the more common production styles in many modern industries. One of the methods used for studying the S-PPL design is through genetic algorithms (GA). One of the important jobs to use GA is how to express a chromosome. In this paper, we attempt to find the nearest optimal design of an S-PPL that will maximize production efficiency by optimizing the following 3 decision variables: buffer size between each pair of work stations, machine numbers in each of the work stations; and, machine types. In order to do this we present a new GA-simulation based method to find the nearest optimal design for our proposed S-PPL. For efficient use of GA, our GA methodology is based on a technique that is called gene family arrangement method (GFAM) which arranges the genes inside individuals. An application example shows that after a number of operations based on the proposed simulator, the nearest optimal design of S-PPL can be found.

**Keywords**: Serial-parallel production line, buffer size, Genetic algorithms, Throughput evaluation.

## 1. Introduction

Production lines that have complicated structures such as parallel, reworks, feed-forward, etc. are widely used in high volume industries [1, 2]. Among them, the serial-parallel production line (S-PPL) [1, 3] is one of the more common production styles in many modern industries. S-PPL is mainly used to increase the capacity of one work station that has a lower speed than other work stations by reducing the variation of material flow speed through the overall production line. Furthermore, S-PPL also reduces the effect of machine failure during processing time. Despite many methodologies developed to study S-PPL, several researchers have described the optimization of production lines using various optimization methods, such as functional approximation and evaluation [4], knowledge-based methods [5], simulated annealing [6], heuristics algorithm [7], dynamic programming method [8], and other search methods. One of the methods used for studying the buffer size in production lines is genetic algorithms (GA) [9, 10].

Almost all researchers assumed that the machine numbers are fixed and only concentrate on finding the buffer size. In this paper, we attempt to find the nearest optimal design of an S-PPL that will maximize production efficiency by optimizing the following 3 decision variables: buffer size between each pair of work stations, machine numbers in each of the work stations; and, machine types. In order to do this we present a new GA-simulation based method to find the nearest optimal design for our proposed S-PPL. One of the important taskes in using GA is how to express a chromosome. For the efficient use of GA, our GA methodology is based on a technique that is called gene family arrangement method (GFAM) which arranges the genes inside individuals.

In evaluating the S-PPL, each work station that consists of multiple parallel machines is combined into one equivalent single machine that turns the S-PPL into a serial production line. Then, the serial production line includes equivalent machines which can be approximated by using a well known decomposition approach.

## 2. Problem Description

Consider a series-parallel production line with $K$ work stations ($S_1$, $S_2$, … , $S_K$) and $K-1$ buffers. Each work station $i$ ($i = 1, 2, \ldots, K$) in the S-PPL can contain several multiple parallel machines as shown in Figure 1.

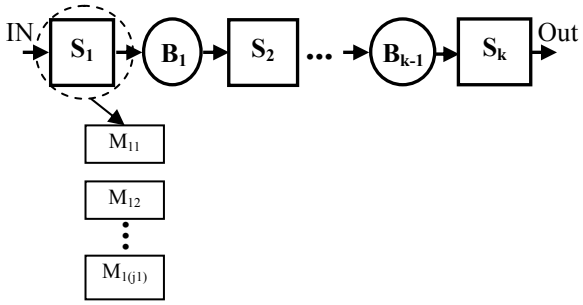


Figure1: S-PPL Model

Before formulating the problem, notations are introduced as follows.

| | |
|---|---|
| $P$ | the total production rate |
| $B$ | $\{B_i\}$ buffer size between stations $i$ and $i+1$ |
| $S$ | maximum capacity of buffer |
| C | $\{C_i\}$ number of parallel machines in work station $i$ |
| $T$ | $\{T_{ij}\}$ type of machine $j$ that is located in |

station $i$

$M^{a}_{group}$ the available types of machines,

$M^{a}_{group} = \{M^{a}_{1}, M^{a}_{2}, \ldots, M^{a}_{l}\}$.

The problem can be formulated mathematically as follows:

Maximize $P(B\,C,T)$ (1)

Subjected to $1 \le B \le S$ (2)

$T_{ij} \in \{M^{a}_{1}, M^{a}_{2}, \ldots, M^{a}_{l}\}$ (3)

## 3. Throughput Evaluation of the S-PPL

In order to evaluate the S-PPL, the multiple parallel machines in each work station are combined into one equivalent single machine. In this way the S-PPL can be converted into a serial production line where each machine in the serial line is equivalent to one set of multiple parallel machines. Then, the serial production line includes equivalent machines which can be approximated by using a well known decomposition approach [11].

### 3.1. Replacing Each Work Station by an equivalent machine

Let us consider a work station $i$ in the S-PPL model with machines $M_{ij}$, $j= 1, \ldots, j1$ as shown in Figure 1. Assume that the uptime and the downtime of the aggregated machines $M_{ij}$ are randomly variable and distributed exponentially with parameters $P_i$ and $R_i$. Then, the parameters $P_e$ and $R_e$ for the equivalent machine can be calculated as follows.

$$P_e = S\,\frac{P_{11}\,R_{11}\sum_{i=2}^{K}(P_{i1}+R_{i1})}{\sum_{i=1}^{K}S_i R_{i1}\sum_{i=1,\,j\neq i}^{K}(P_{j1}+R_{j1})}, \quad (4)$$

$$R_e = S\,\frac{P_{11}\,R_{11}\sum_{i=2}^{K}(P_{i1}+R_{i1})}{\sum_{i=1}^{K}S_i P_{i1}\sum_{i=1,\,j\neq i}^{K}(P_{j1}+R_{j1})} \quad (5)$$

Where $S = \sum_{i=1}^{K} S_i$

### 3.2. Serial Line Aggregation

Previously, no closed form expression for performance of a serial line with more than two (non-identical) machines has been available. Therefore, many approximation approaches have been used to evaluate the production line based on aggregation and decomposition. In this paper, we introduce the aggregation procedure proposed by Li [11], which modifies the machine downtime parameter to accommodate starving and blocking information. This aggregation procedure is a good approximation and results in good accuracy. Consider a serial line with machines ($M^{e}_{1}, M^{e}_{2}, \ldots, M^{e}_{K}$) and buffer size $B_1$ to $B_{K-1}$. The first two machines aggregate into a single machine $M^{f}_{2}$ with downtime parameter $r^{f}_{2}$ and the uptime parameter $p^{f}_{2}$ is defined as follows.

$$r^{f}_{2} = r_2[1 - Q(p_1, r_1, p_2, r_2, N_1)], \quad (6)$$

$$p^{f}_{2} = p_2 + r_2 Q(p_1, r_1, p_2, r_2, N_1) \quad (7)$$

Where $Q(p_1, r_1, p_2, r_2, N_1)$ is the probability that machine $M^{e}_{2}$ is starved and is defined by [12] as follows:

$$Q(p_1, r_1, p_2, r_2, N) = \begin{cases} \dfrac{(1-e_1)(1-\Phi)}{1-\Phi e^{-\beta N}+r_2}, & \text{if } \dfrac{p_1}{r_1} \neq \dfrac{p_2}{r_2} \\ \dfrac{p_1(pp_1)(rr_1)}{(p_1+r_1)[(pp_1)(rr_1)+p_2 r_1(pp_1+rr_1)N]}, & \text{if } \dfrac{p_1}{r_1} = \dfrac{p_2}{r_2} \end{cases} \quad (8)$$

$$e_i = \frac{r_i}{p_i + r_i}, i = 1,2 \quad , \quad \Phi = \frac{e_1(1-e_2)}{e_2(1-e_1)} \quad , \quad pp_i = p_i + p_{i+1}$$

$$rr_i = r_i + r_{i+1} \text{ and } \beta = \frac{(p_1+p_2+r_1+r_2)(p_1 r_2 - p_2 r_1)}{(p_1+p_2)(r_1+r_2)}.$$

Next, $M^{f}_{2}$ is aggregated with $M^{e}_{3}$ to result in $M^{f}_{3}$, and so on until all $K$ machines are aggregated in a single one, $M^{f}_{n}$. This constitutes forward aggregation. Then, in backward aggregation, the last machine, $M^{e}_{K}$, is aggregated with $M^{e}_{K-1}$ to result in $M^{b}_{n-1}$ and so on until all machines are aggregated in a single machine, $M^{b}_{1}$. The procedure is repeated until convergence is satisfied. By following the aggregation procedure, the production rate can be approximated as,

$$PR(p_1, r_1, \ldots, p_n, r_n, N_1, \ldots, N_{n-1}) = \frac{r^{f}_{N}}{p^{f}_{N}+r^{f}_{N}} = \frac{r^{b}_{1}}{p^{b}_{1}+r^{b}_{1}} \quad (9)$$

## 4. Gene Family Arrangement Method

One of the important tasks in using GA is how to express a chromosome. As we described above, this research solves three different decision variables. To represent these variables, we propose GFAM as a new arrangement method which arranges the genes in each individual. Furthermore, GFAM adopts two groups of genes in each individual. The first group represents the buffer size and is located in the even positions of the individual ($G_2$, $G_4$, … , $G_{2k-2}$, $G_{2k}$), where $k$ is the number of work stations. The second group represents the number of machines in each work station and it is located in the odd positions of the individual ($G_1$, $G_3$, … , $G_{2k-3}$, $G_{2k-1}$). Each of the items in the odd group includes a family of genes which represent the machine types in each work station; each family of genes is coded as follows.

$$G_i = \begin{bmatrix} G_{i1} \\ \vdots \\ G_{ij} \end{bmatrix} \quad \forall\, i = 1,\ 3\ \ldots 2k-3,\ 2k-1 \quad (10)$$

Where $j$ is the number of parallel machines in work station $i$. The number of items in the odd and even groups is not limited, which means that any production line with any number of work stations that include any number of machines and any buffer size between each pair of work stations can be dealt with.

## 5. Genetic Algorithms for the Optimal Design of S-PPL

GA is a global optimization technique used for various optimization problems [9, 10]. In this paper, we present the determination of a near optimal design for an S-PPL. Since the developed GFAM is different from conventional individual expressions, the operational procedure for our GA is also different. The characteristics of the GA are described in sections 5-1 and 5-2.

### 5.1. Crossover

In this research, the encoding method to express each individual using GFAM is different from that obtained using conventional encoding methods. The crossover operations for our GA system operate by using two processes. The first crossover is similar to the conventional crossover method, i.e., the genes after the crossover point are swapped between two individuals. On the other hand, the second crossover method swaps the genes between two families for each of the individuals that results from the first process.

The crossover operations are generated by using the following steps:
**Step 1:** Randomly select two individuals from the current population.
**Step 2:** Randomly select a crossover point and swap the genes after that crossover point.
**Step 3:** Randomly select two items (families), *I1* and *I2* from the odd group in the first individual generated from step 2.
**Step 4:** Randomly select a family crossover point.
**Step 5:** Randomly select the number of genes after the family crossover point to be included in the crossover, $N^F$. $N^F < I1$ and $N^F < I2$.
**Step 6:** Replace the genes after the family crossover point one by one. If one of the two families reaches its last gene, continue the replacement from the first gene in the family.
**Step 7:** Repeat steps 3-7 for the second individual generated from step 2.
Figure 2 shows the crossover process graphically.

### 5.2. Mutation

The mutation of our GA system is different from the traditional mutation operator because the gene expression adopts GFAM. The mutation is carried out by using the following steps:

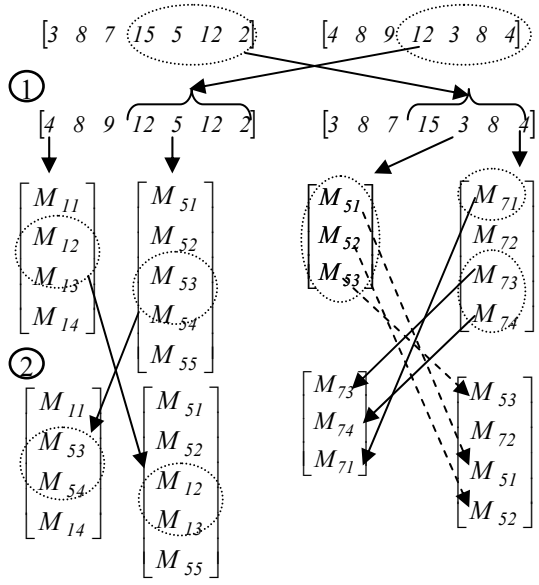


Figure 2: Crossover process

**Step 1:** Select one individual randomly from the current population.
**Step 2:** Randomly select one of the genes of a single gene type (first group).
**Step 3:** Change the value of the selected gene to a new value, which can also be selected randomly between *(1-S)*.
**Step 4:** Randomly select one gene family (Second group).
**Step 5:** Change the value of the selected gene to a new value, which can also be selected randomly between (1- the number of the selected family members).
**Step 6:** Randomly select one gene family, again.
**Step 7:** Randomly select one member of the selected gene family.
**Step 8:** Change the type of the selected family member to a new value, which can also be selected randomly.

### 5.3. Implementation of the GA

Before describing our implementation of the genetic algorithm, the following notations are defined.

***Notations:***

| | |
|---|---|
| $PS$ | Population size. |
| $P_c$ | Crossover rate. |
| $P_m$ | Mutation rate. |
| $P_i$ | Selection probability of the individual $i$. |
| $F_i$ | Fitness of individual $i$. |
| $N$ | Number of individuals in the population. |
| $D_i$ | Individual $i$ in the population. |

The implementation of GA is presented below.
**Step1** [Initialization] Randomly generate an initial population.
**Step2** [Evaluation] Evaluate the fitness for each individual in the population.
**Step3** [Selection] Calculate the roulette selection probability $P_i$, $\forall i = 1, 2, \ldots, N$ by using equation (11).

$$P_i \leftarrow \sum_{j=1}^{i} (F_j \Big/ \sum_{k=1}^{N} F_k) \quad (11)$$

**Step4** [Create new population] For $i \le (P_c \times N)$, create $D_{1...i}$ using crossover operations. Set the other individuals using the roulette selection process.

**Step5** [Carried out mutation] Apply mutation operations on $(P_m \times N)$ individuals.

**Step6** [Keep fittest using elitist selection strategy] Randomly select one individual from the generated population. Replace the selected individual with the best individual in the current generation if it has not been selected through the roulette selection process.

**Step7** [Loop]: Loop until fitness reaches its maximum value.

## 6. Numerical Experiments

### 6.1. Simulation model

The simulation model of our S-PPL was developed using C++. The uptimes and the downtimes of the machines were all assumed to be randomly variable and distributed exponentially. The model was run until the fitness (throughput) attained maximum value. In each generation the parallel machines, each work station is replaced by an equivalent machine. Then, the throughputs of the serial line of the equivalent machines are evaluated. GA operations improve the throughput until the fitness becomes constant.

### 6.2. Results

We applied our algorithm to an example of an S-PPL with 10 work stations. The maximum allowable number of machines to be connected in parallel is 5. The algorithm was tested by performing many trials. Figure 3 shows the (fitness) throughput versus the number of generations.

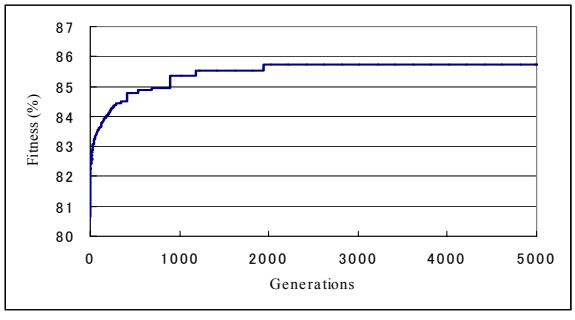


Figure 3: Best fitness curve

## 7. Conclusion

This study describes a new GA-simulation based method to find the nearest optimal design for a S-PPL. Instead of using buffer size as a single decision variable, this paper proposes an optimal design for the S-PPL by using three decision variables: buffer size between each pair of work stations, machine numbers inside the workstations, and machine types. The GA methodology is based on a new technique of gene expression that is called gene family arrangement method (GFAM) which arranges the genes inside individuals. We used the new GA-simulation based method to determine some S-PPL designs. After a number of generations, the nearest optimal S-PPL could be determined. The results of this study can be used to improve S-PPL design, and production engineers can use these results when they design a new S-PPL.

## References

[1] Dallery, Y. and Gershwin, S. B., "Manufacturing Flow Line Systems: A Review of Models and Analytical Results", *Queueing Systems theory and Applications*, *Special Issue on Queueing Models of Manufacturing Systems*, Vol 12, No. (1-2), pp. 3-94, 1992.

[2] Gershwin, S.B., "Manufacturing systems engineering ", *Prentice-Hall*, 1993

[3] Burman, M.H., "New Results in Flow Line Analysis", *Ph. D. Thesis, MIT, Cambridge MA,* 1995.

[4] 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, pp. 601-624, 2002.

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

[6] Spinellis, D., Papadopoulos, C.T., and MacGregor, J., "Large production line optimization using simulated annealing", *International Journal of Production Research*, Vol. 38, No.3, pp. 509–541, 2000.

[7] Papadopoulos, H.T. and Vidalis, M.I., "A heuristic algorithm for the buffer allocation in unreliable unbalanced production lines", *Computers & Industrial Engineering,* Vol. 41, No. 3, pp. 261–277, 2001.

[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, pp. 130-135, 1989.

[9] Forrest, S., "Genetic Algorithms", *ACM Computing Surveys*, Vol. 28, No. 1, pp. 77-83, 1996.

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

[11] Jingshan Li, "Overlapping Decomposition: A System-Theoretic Method for Modeling and Analysis of Complex Production Systems", *Technical Report*, General Motors Research & Development Center, 2003.

[12] Chiang S.Y., Kuo C.-T. and Meerkov S.M., "DT-Bottlenecks in Serial Production Line: Theory and Application", *IEEE Transactions on Robotics and Automation*, Vol. 16, pp. 567-580, 2000.