World Applied Sciences Journal 6 (2): 168-181, 2009
ISSN 1818-4952

An Efficient Method to Solve a Mixed-model Assembly Line Sequencing Problem Considering a Sub-line

S.M.J. Mirzapour Al-e-Hashem and M.B. Aryanezhad

Department of Industrial Engineering, Iran University of Science and Technology,
P.C.16846113114, Tehran, Iran

Abstract: To meet the diversification of consumer's preferences, mixed-model assembly lines were installed in many manufacturing plants. In some of them, a large variation exists in assembly times among different product types. The large variation reduces production efficiency and may cause a line stoppage. These variations can be reduced by installing a bypass sub-line which processes a portion of assembly operations of products with relatively longer assembly times. In spite of its significance, sequencing problem on bypass sub-line rarely has been discussed in the literature. This paper deals with a sequencing problem with a bypass sub-line (MALSP-B) with the goals of leveling the part usage rates and reducing line stoppages. The former objective is taken from the literature and the later is introduced in this paper. Also some unsynchronized situations which are disregarded in previous studies are considered. Finally a hybrid algorithm based on Genetic Algorithm (GA) and event based procedure is developed to solve the problem. Efficiency of the proposed algorithm is demonstrated through comparing with optimal solutions are obtained from an exhaustive enumeration method.

Key words: Sequencing . mixed model assembly line . bypass sub-line . hybrid genetic algorithm . event based procedure . enumeration

INTRODUCTION

Just-in-time (JIT) is a management philosophy that uses a set of integrated activities to achieve manufacturing flexibility with minimal inventories [1]. One manufacturing problem that is often associated with JIT practices is sequencing mixed models on assembly lines. Mixed model assembly lines are widely used in manufacturing industries to meet diversified demand of consumers without the need for large product inventories. Sequencing products to be assembled at the mixed model assembly line is recognized as an important work for improving its performance. Note that this problem assumes that mixed model assembly line balancing problem is solved before. In other words; first Mixed Model Assembly Line Balancing Problem (MALBP) is solved then being aware of the number of stations and also configuration of operation assignment, the problem is determining the sequence of all products in a way that some manager's objectives are met. This problem is called in the literature as mixed Model Assembly Line Sequencing Problem (MALSP) which differs from MALBP. Therefore in the following the literature of sequencing problem on mixed model assembly line is reviewed.

The sequencing may vary, depending on the goals of a company. Mainly the following five goals have been discussed in the literature:

- To keep a constant usage rate of every part used in the assembly line [2-10].
- To level the loads (assembly times) at each work station on the assembly line [11, 12].
- To keep the constant rate of feeding products into the assembly line [13].
- To minimize the total conveyor stoppage time [14-16].
- To minimize the number of required setups [1, 17].

Boysen *et al*. [18] presented a survey study and classified this model from several points of view. Rahimi-Vahed and Mirzaei first, presented a multi objective model for mixed model assembly line sequencing problem then solved it by a new frog-leaping algorithm [19]. Kim and Jeong [20] presented a new model for mixed model assembly line sequencing problem to minimize unfinished works. Rahimi-Vahed *et al*. developed a multi objective mixed model assembly line sequencing problem and solved it via scatter search [21]. Ding compared two weighted

Corresponding Author: Dr. S.M.J. Mirzapour Al-e-Hashem, Unit 6, No.7, Bahar Azadi Alley, Abolfazl st., Marzdaran Blvd., Tehran, Iran, PC: 1461743361

approaches for sequencing mixed model assembly line with multiple objectives; leveling workloads for stations on the line and keeping a constant rate of usage for every part used on the line [22].

When some different types of products are manufactured at the mixed-model assembly line and assembly times are significantly different among these product types, the production efficiency usually reduces due to the line stoppage [16, 23]. The variation in the assembly times can be reduced by installing a bypass sub-line. If we could classify all products into two categories in a way that the first category have small production time and the other have large production time; installing a bypass sub-line would seem necessary. The bypass sub-line is set up adjacent to a main assembly line and processes a portion of assembly operations of products with relatively longer assembly times [2, 12]. Even if installing the bypass sub-line the problem of sequencing the products remains to be solved. The sequencing problem in a mixed model assembly line with a bypass sub-line (MALSP-B) for levelling the part usage rate and work loads was considered by [12].

The purposes of this paper are to consider a sequencing problem that levels the part usage rates and minimizes the line stoppage rate for mixed model assembly line with a bypass sub-line. A hybrid meta heuristic algorithm is developed to solve the problem. In this algorithm, a GA is combined with an event based procedure to gain advantageous of that to consider Bypass sub-line, unsynchronized situations and Line stoppages which are disregarded in the literature. To demonstrate the efficiency of the proposed algorithm, the result of solving test problems, via our proposed algorithm, is compared with the optimal solution which is obtained via an exhaustive enumeration method.

Problem description (MALSP-B): Consider an assembly line which consists of a main line with k_1 stations and a bypass sub-line with k_2 stations. It is assumed that all stations are balanced before (Mixed Model assembly line Balancing Problem). All products are classified into the following two categories: products which are processed at the bypass sub-line (type2) and products which are processed at the main line (type1). But products of the same type don't use necessarily the same parts. Also if they use the same parts, their consumption rates are not necessarily the same.

This typically assembly line is shown in Fig. 1.

Assembly rout of product type1 are $m_1, m_2, \ldots, m_H, m_{H+1}, \ldots, m_{k1}$, respectively. That is; a product type1 moves directly to station $m_{(H+1)}$ (which is named junction station), after completion of its operation at H_{th} station of the main line (m_H).

Assembly route of product type2 are $m_1, m_2, \ldots, m_H, b_1, b_2, \ldots, b_{k2}, m_{H+1}, \ldots m_{k1}$ respectively. That is; a product type2 moves to the first station of the bypass sub-line (b_1) after completion of its operation at H_{th} station of the main line (m_H). After this product is processed at the last station of the bypass sub-line (b_{k2}), it returns to the main line at station $m_{(H+1)}$.

The production sequence at the bypass sub-line is obviously a partial sequence of the production sequence at the first H stations of the main line. It means that product sequence in the bypass sub-line is the same as the product sequence at the first H station of the main line from which products type1 are subtracted. And production sequence at the first H station of the main line differs from the case at the remaining stations of the main line. Moreover, Cycle times of the main line (c_1) and bypass sub-line (c_2) can be different.

The question is determining the sequencing of all products in a way that some objectives of the managers are met.

Often a goal in sequencing problem on the mixed model assembly line is keeping the usage of parts for the different products being produced as level as possible. Constant use of parts allows easier implementation of a JIT manufacturing environment due to lower variations in production quantities and

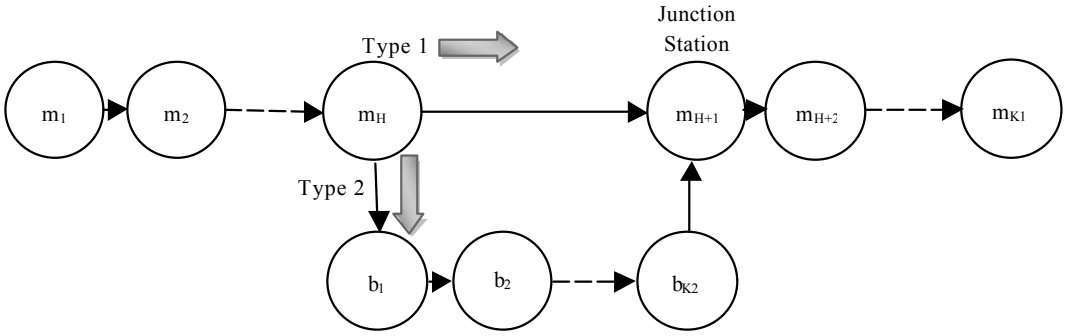


Fig. 1: Assembly line with a bypass sub-line

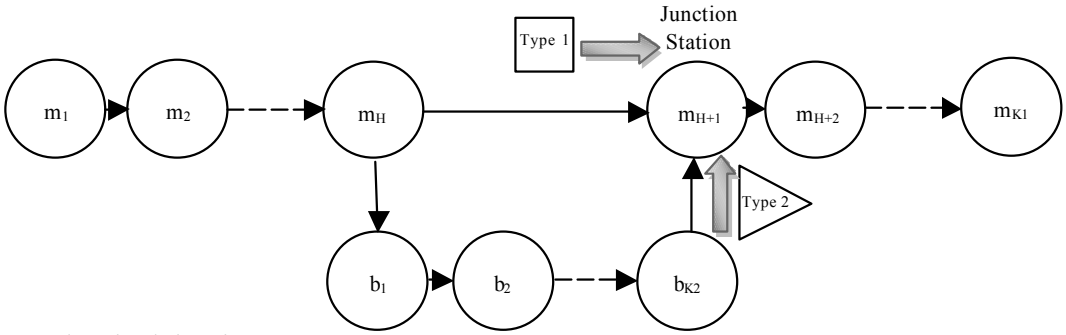


Fig. 2: Unsynchronized situation

work in process inventories. This objective function has been widely studied first by [2] and more recently by [3-5, 10, 12, 15, 23-26].

The other goal, introduced in this paper, is minimizing total waiting time at the stations due to occurrence of unsynchronized situations which are described in the following section. Since these two objective functions are not in the same dimension, we convert them to cost terms. Now these two terms are merged and formulated as a non linear integer programming.

Unsynchronized situations: Unsynchronized situation, occurred in junction station (m_H), is a situation in which two products of different types (one of type1 and the other of type2) reach to junction station simultaneously. Note that in previous papers theses situations are not considered and they are eliminated through assuming two buffers which were located right before and after the bypass sub-line. Considering these situations bring about some difficulties in using mathematical modeling approaches. Because these situations are rule based and some rule based approaches like event-based procedure are needed.

Generally in unsynchronized situation if two products (one of type1 and the other of type2) reach to junction station concurrently, then the right of way is given to the product with higher priority, else if they have the same priority, the right of way is given to product type1 (because the importance of main line).

It is clear that when a product type1 leaves station m_H and at the same time a product type2 leaves station bk2, an unsynchronized situation is occurred since it leads to either main line stoppage (MS) or sub-line stoppage (SS) (Fig. 2).

The rules can be used here depend on the nature of the assembly line, products priorities and maybe their types. Therefore, they can be different form one problem to another. Anyway in this paper some reasonable rules are considered and the problem is solved according to these hypothetical rules. All unsynchronized situations rules considered in this paper are as follows:

Situation#1: A product type1 leaves station m_H and a product type2 leaves station bk2 simultaneously. In this case, priority is usually given to main line. Therefore bypass line has to be stopped which is named as Sub-line Stoppage (SS).

Situation#2: A product type1 leaves station m_H and a product type2 ready to leave station b_{k2} which has been waiting in this station during the last cycle time and due to occurrence of the situation#1. In this case priority is given to sub-line, therefore main line has to be stopped which is names as Main line Stoppage (MS).

Situation#3: A product type1 ready to leave station m_H which has been waiting in this station during the last cycle time due to occurrence of the situation#2 and the product type2 leaves station b_{k2}. In this case, priority is given to main line. Therefore a SS is occurred.

In other situations, priority is determined, based on FIFO (First in first out) rule.

Notation

a_{ij}: The number of part j,j = (1,…,m) is required to assemble one unit of product i, i = (1,…,n).

d_i: Demand for product i.

$D = \sum_{i=1}^{n} d_i$: Total number of units for all product (total demand) also represents number of positions in sequence.

c_m: c_1 and c_2 denote cycle times at the main line and bypass sub-line respectively.

k_m: k_1 and k_2 denote number of stations at the main line and bypass sub-line respectively.

$r_j = \sum_{i=1}^{n} \frac{a_{ij} \cdot d_i}{D}$: Average consumption rate of part *j*.

x_{ik}: Total number of units of product *i* produced over k stages, k = 1,…,D.

$v_{jk} = \sum_{i=1}^{n} a_{ij} \cdot x_{ik}$: The quantity of part *j* required during *k* stages.

$$Q_{su} = \begin{cases} 1 \text{ ,if station s is stopped at stage u} \\ 0 \text{ ,otherwise} \end{cases}$$

Where, u is the counter of event in procedure and s is the number of stations, $s = \{m_1, m_2,\ldots, m_H, b_1, b_2,\ldots, b_{k2}, m_{(H+1)}, m_{(H+2)}\ldots, m_{k1}\}$

RT_{su}: Is the remaining time to complete current process at station s and at stage u.

Objective functions: The first objective of the model is leveling part usage rate and formulated as follows:

$$\text{Minf}_1 = \sum_{j=1}^{m}\sum_{k=1}^{D}\left|v_{jk} - k \times r_j\right| \quad (1)$$

St:

$$\sum_{i=1}^{n} x_{ik} = k \quad \forall k \quad (2)$$

$$\begin{aligned} &x_{i(k+1)} \geq x_{ik}, \forall i,k \\ &x_{ik} \text{integer}, \quad \forall i,k \end{aligned} \quad (3)$$

Where, $\sum_{j=1}^{m}\sum_{k=1}^{D}\left|v_{jk} - k \times r_j\right|$ is the cumulative deviation of real consumed parts from its average consumptions. The first constraint (Equation 2) ensures that in stage k, k products is produced which is equal to total number of units of product from various models. The second constraint (Equation 3) ensures that total number of units of product i produced over (k+1) stages is not lower than that of in previous stage.

The second objective function which is introduced in this paper aims to minimize total waiting time at all stations during the production and it is equivalent to line stoppage reduction including SS and MS and formulated as follows:

$$\text{Minf}_2 = \sum_{u=0}^{U}\sum_{s=1}^{S} Q_{su} \times RT_{su} \quad (4)$$

Problem solving: Regarding to the nature of this assembly line, production sequence which enters the first station of the line (input sequence) is completely different from that of exits from the last station (output sequence). What is required to calculate the first objective function is the output sequence. on the other hand to calculate the second objective function, it's required to know total waiting time between the entrance of the first product of sequence into the first station of the assembly line and completion of last process of the last product at the last station of the assembly line. To know these details it is necessary to feed the products step by step into the assembly line according to a predetermined sequence and record assembly line condition in each step till the last product of sequence leaves the last station of the assembly line. To simulate assembly line, including the main line and bypass sub-line, an event based procedure is proposed. Stage or event here means the movement of a product from a station to another. But there are several products in the line so stage is the movement of the next product which its process is completed and also its next station is empty. It is noteworthy due to the unsynchronized situation rules which are defined to move products at the junction station $m_{(H+1)}$, every input sequence as vector X convert to a unique output sequence as vector Y. It's obvious that by changing these rules, output sequence which results from an input sequence will change consequently. In every stage of procedure, movement possibility of products along assembly line is checked. The products for which movement are possible, make progress. In the next stage, movement possibility is again checked and this cycle is repeated until the end of procedure.

Movement possibility of a typical product in station s and at stage u is checked according to two important facts; first, the process of this typical product should be completed in station s. second, the next station to which this product moves, should be vacant. Note that, there is maybe a product, for which process in station s is completed but the next station is not being vacant. This situation which is called here "station stoppage" leads the product (in spite of its process completion) to stop at station s till the next station will be vacant.

When time passes, the first product meets the movement possibility conditions is the next event would be occurred in our procedure.

Time Intervals (TI) between sequential stages are calculated according to the remaining time of running process in each station and also station stoppages:

$$TI_u = \min_s \left\{RT_{s(u-1)} \mid RT_{s(u-1)} > 0, Q_{s(u-1)} = 0\right\} \forall s \quad (5)$$

Where, RT_{su} is the remaining time of process of the product in station s at stage u. Therefore $RT_{su}>0$ means that process of the product at station s is not completed yet and therefore this station is not vacant at stage u. Also Q_{su} is a binary variable that indicates whether station s is stopped at stage u ($Q_{su} = 1$) or not ($Q_{su} = 0$). Therefore $Q_{su} = 0$ means that station s is not stopped at stage u.

In other words, Time interval (TI) is the time between two sequential events (stages u and u-1) and

that is equal to shortest time from the end of previous event takes to complete the process of a product in a station.

After the procedure is completed, output sequence is calculated and being aware of that, the first objective function can be obtained using formulation (1). On the other hand according to the records of the waiting times in stations during the procedure steps, the second objective function can be calculated too using formulation (2). This way, the objectives functions (1), (2) can be calculated for every input sequence which is fed to the assembly line. These two objectives are merged into a single objective with multiplying them to their unit costs:

$$\text{Min}Z = w_1 \cdot f_1 + w_2 \cdot f_2 \qquad (6)$$

Where, w_1 and w_2 are the unit cost of deviation from average part usage and the unit cost of line stoppages, respectively.

Event based procedure: Step$_0$: initialize $g = 0$ (counter of output) and $u = 0$ (counter of stage)

Step$_1$: Take an input sequence such as vector X

Step$_2$: Feed the first product of sequence into the first station (m_1)

Step$_3$: At the stations which a product enters them at the stage u; set $Q_{su} = 0$ and calculate the remaining times (RT_{su}), using the following equation for them.

$$\text{if} \begin{cases} s \in \text{subline} \Rightarrow RT_{su} = c_2 \\ s \in \text{mainline} \Rightarrow RT_{su} = c_1 \end{cases}$$

Where, $s = \{m_1, m_2, \ldots, m_H, b_1, b_2, \ldots, b_{k2}, m_{(H+1)}, m_{(H+2)} \ldots, m_{kl}\}$ and c_1, c_2 denote cycle times at the main line and bypass sub-line respectively.

Step$_4$: set $u = u + 1$

Step$_5$: Calculate value of TI_u using equation (5)

Step$_6$: Update RT_{su} using equation (7) and repeat the following sup-steps for each station (starting from last station)

$$RT_{su} = RT_{s(u-1)} - TI_u \qquad (7)$$

- ☑ If $RT_{su} \neq 0$ then set $Q_{su} = 1$
- ☑ If $RT_{su} = 0$ then
  - If $s = m_{kl}$ then:
    - Remove the product from station m_{kl} and put it in the output sequence (vector Y),
    - $g = g + 1$
  - Else if $Q_{(s+1)u} = 1$ then set $Q_{su} = 1$ else
    - if $s = b_{k2}$ and a product type$_1$ exists at station m_H for which $RT_{su} = 0$ then according to unsynchronized situation rules select the product with higher priority (one of the product exists at station b_{k2} or m_H) and move it to the station $m_{(H+1)}$ and set $Q_u = 0$ for the selected station and set $Q_u = 1$ for the other one (station b_{k2} or m_H).
    - If $s = m_1$ then move the product exists at station s to the next station (m_2) and feed the next product of sequence (vector X) into the first station.
    - If $s = m_H$ and a product type$_2$ exists at station s then move the product exists at this station to the next station (b_1).
    - Else if $s \neq m_H$ then move the product exists at station s to the next station ($s + 1$).

Step$_7$: If $g = D$ stop the algorithm, otherwise go to step$_3$.

Exhaustive enumeration method: In order to obtain the optimal solution for this problem, an exhaustive enumeration method (EE) is developed. Being aware of the discrete nature of the feasible solutions, EE generates all the feasible solutions of MALSP-B. For each solution which is generated from EE, event based procedure is run and its objective function is calculated, comparing the objective function values with previous ones leads us to optimal solution at the final step of EE. The outline of EE process used in this paper is summarized as follows:

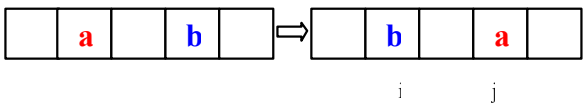
Step1: Let $n = 0$ (counter of feasible permutations) and $l = 0$ (counter of active permutations)

Step2: Generate a random feasible solution (permutation) and put it on the n_{th} array of the feasible solutions' matrix and let $n = n + 1$

Step3: Select the l_{th} permutation of the feasible solutions' matrix which is named "active" permutation.

Step4: Consider the exchange between the elements i and j in the active permutation. ($i, j \in \{0, D\}, i \neq j$)

For example:



- if this new permutation (created from exchange) is different from n-1 previous permutations (located

- in the feasible solutions' matrix), then put it on the n_{th} array of the feasible solutions' matrix and let $n = n + 1$, consider another duplex exchange in the active permutation and repeat this sub step
- if all possible duplex exchanges are considered, deactivate current active permutation and go to *step 5*

Step5: If $n = \frac{D!}{\prod_{i=1}^{N}(d_i)!}$, stop; all of the feasible solutions are gathered in the feasible solutions' matrix. Otherwise let $l = l + 1$ and go to *step 3*

Note that EE just produces feasible solutions. Combining this algorithm and proposed procedure leads to optimal solution which is shown schematically in Fig. 3.

Combinatorial complexity: Finding production sequences with desirable levels of both number of products and their demands, is NP-hard as pointed out by McMullen and Tarasewich [1]. Total number of sequences (feasible solutions) for a mixed-model sequencing problem having *n* products can be computed using the general formula to compute the number of permutations of a multi-set as follows:

$$\text{Totalsequences} = \frac{D!}{\prod_{i=1}^{n}(d_i)!}$$

As mentioned before, to obtain the optimal solution an exhaustive enumeration method is used. As the problem increases in size, the number of feasible solutions increases in an exponential fashion, thereby attainment of optimal solutions becomes impractical for large scale problems. Problems with a large number of possible solutions usually cannot be solved to optimality within a reasonable amount of time.

As a typical example in a problem with four products and three unit of demand for each one, the number of possible input sequences is equal to:

$$\text{Totalsequences} = \frac{(4\times 3)!}{(3!)^4} = 369600$$

Genetic algorithm: The GA is a stochastic search technique. It can explore the solution space by using the concept taken from natural genetics and evolution theory. In classical Genetic algorithm the chromosomes of the population are evaluated according to a predefined fitness function, which is usually equal to a straightforward and easy calculated objective function but in our proposed GA the objective function of each chromosome is calculated during the event base procedure which is described in previous section. In each generation of GA, first, population is selected from the mating pool according to the selection strategy, then for every chromosome of this population the event based procedure is run to compute the fitness value of that chromosome. The outline of the genetic search process used in this paper is summarized as follows:

1. Randomly generate an initial population of chromosomes with a population size P.
2. Run event based procedure to evaluate each chromosome in the population according to the objective function.
3. Apply the Monte Carlo selection technique to select parent chromosomes from the current population. This is used for choosing randomly the parents for crossover and mutation. In this procedure first for every chromosome a random number (between 0 and 1) generated, then chromosomes for which this number is lower than the mutation probability are selected for mutation,

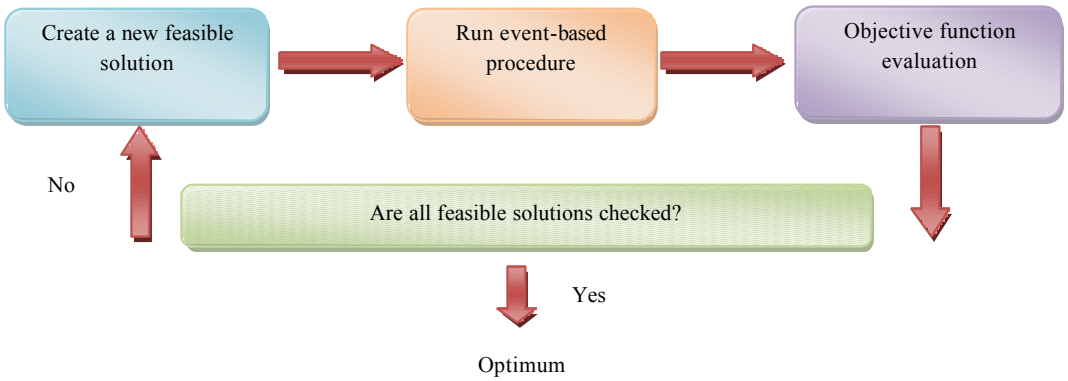


Fig. 3: Flowchart of exhaustive enumeration method

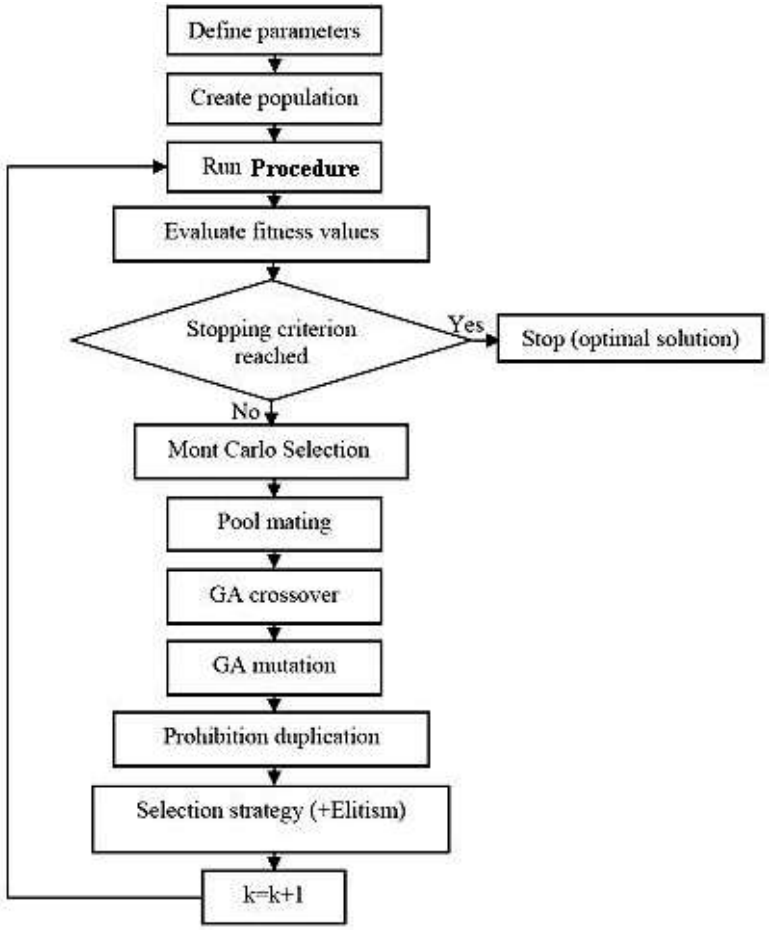


Fig. 4: Flowchart of GA optimization procedure

this procedure is repeated in order to select chromosomes for crossover. Note that for a typical chromosome both of mutation and crossover operators can be applied.

4. Apply crossover and mutation operators to generate the offsprings based on the values of crossover and mutation probabilities (pc and pm, respectively) and put them on the pool mating. A similarity checking is used to prevent parents from duplicating offsprings.
5. Use elitism strategy to fix the potential best number of chromosomes by copying half of the best chromosomes of both parent and offspring population into the succeeding generation. Then rest of the population is brought randomly from the mating pool (including parents and offsprings). Total number of chromosomes is kept constant for computational economy and efficiency.
6. Check the pre-specified stopping criterion. If the stopping criterion is reached, the search process stops. Otherwise, proceed to the next generation and go to step 2.

The flow chart of the GA optimization procedure is shown in Fig. 4.

Chromosome encoding: The encoding of solutions to a problem should ensure that all possible solutions can be generated provided that chromosomes are generated at random. For combinatorial ordering problems, usually permutation encodings are used, because the order of items can be most naturally modeled in this way. Here, the order within the permutation is interpreted as a sequence while scanning it from left to right during the decoding. For our purpose, we used a permutation of all products involved in an instance of a sequencing problem. The order of products in the permutation serves as a sequence of that product which consecutively assembles a schedule for production. Note that all permutations represent feasible input sequence directly.

Crossover: Recombination plays a crucial role for the GA performance. According to the widely accepted design goal for crossover operators, information already

Parent permutation1	A	B	A	C	C	B	A
Parent permutation2	B	A	C	B	A	A	C
Select parent no. (1 or 2)	1	2	2	1	2	2	1
Offspring permutation	A	B	C	A	B	A	C

Fig. 5: Precedence preservative crossover

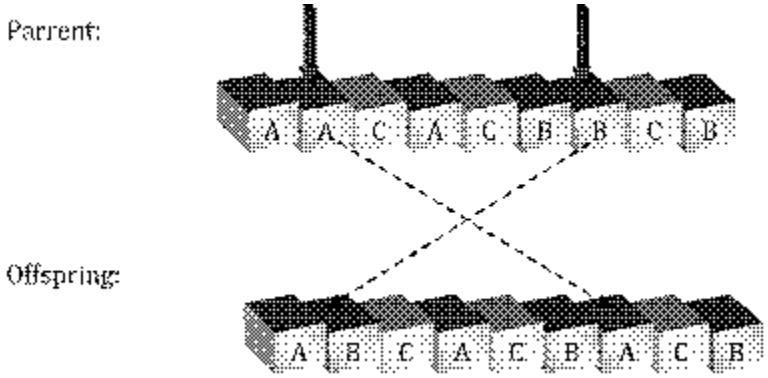


Fig. 6: Mutation crossover

existing in the parent solutions should be recombined without introducing new information. In the following we describe the Precedence Preservative Crossover (PPX) which is illustrated in Fig. 5 for a problem consisting of three products A, B and C with demands equal to 3, 2 and 2 respectively.

The operator works as follows: A vector of length,

$$D = \sum_{i=1}^{n} d_i$$

representing the number of products involved in the problem is randomly filled with elements of the set {1, 2}. This vector defines the order in which the products are successively drawn from parent 1 and parent 2. We can also consider the parent and offspring permutations as lists, for which the operators 'append' and 'delete' are defined. First we start by initializing an empty offspring. The leftmost product in one of the two parents is selected in accordance with the order of parents given in the vector. After a product is selected it is deleted in both parents. Finally the selected product is appended to the offspring. This step is repeated until both parents are empty and the offspring contains all products involved.

Mutation: This operator should alter a permutation only slightly. The rationale behind this idea is that a small modification in the permutation will result in a small deviation of its fitness value only. In this way the information newly introduced by a mutation has a reasonable chance to be selected and accordingly passed on into future generations. In our GA we alter a permutation by first picking (and deleting) a product before reinserting this product at a randomly chosen position of the permutation (Fig. 6).

Parameter setting: In terms of genetic algorithm performance, two measures are of interest: objective function performance and CPU time. Objective function performance implies a comparison of the best objective function obtained via genetic algorithm and the optimal solution obtained via exhaustive enumeration method. CPU time here means a comparison of required CPU time for the genetic algorithm and the required CPU time for exhaustive enumeration method. These algorithms were run on a system with Pentium IV processor at 2.6 GHz under Windows XP using 512 MB of RAM.

For tuning the GA for the mixed-model sequencing problems, extensive experiments were conducted with differing sets of parameters. At the end, the following set was found to be effective in terms of genetic algorithm performance. It should be noted that changing these parameters may result in different outcomes than those achieved in this research. The Mutation, crossover and Elitism Probabilities and Population Size were set to 0.2, 0.8, 0.5 and 30 respectively.

Population size parameter: In order to set a reasonable value for population size a sample problem

Table 1: Data for problem P_0

Problem	Product$_i$	Type	Demand	Number of solutions	Part$_i$ 1	2	3	4	5	6	7	8	9	10
P_0	1	Type$_2$	5	6.2×10^{14}	5	7	1	6	6	4	2	6	2	4
	2	Type$_1$	5		6	3	6	4	4	0	6	6	5	0
	3	Type$_2$	5		7	3	6	2	3	2	4	5	7	2
	4	Type$_1$	5		2	4	5	5	1	4	6	2	5	4
	5	Type$_2$	5		2	7	0	0	0	1	7	5	7	3

Table 2: Sensitivity analysis for population size in example P_0

Population size	Average objective	Best objective	Time to solve (s)	Number of runs
15	339.60	339.4	11	10
20	337.95	337.8	15	10
25	337.52	337.4	18	10
30*	335.42	335.4	22	10
40	336.03	336.4	29	10
50	336.11	336.8	37	10
100	336.26	335.8	80	10

Table 3: Sensitivity analysis for mutation and crossover probabilities

(Mutation, crossover)	Average objective	Best objective	Time to solve (s)	Number of runs
(0.1, 0.9)	340.60	339.2	23	10
(0.2, 0.8)*	338.95	337.2	24	10
(0.3, 0.7)	339.52	338.4	22	10
(0.4, 0.6)*	337.62	337.4	22	10
(0.5, 0.5)	341.23	341.2	22	10
(0.6, 0.4)	340.79	340.5	23	10
(0.7, 0.3)	342.82	342.3	23	10
(0.8, 0.2)	342.99	343.0	25	10
(0.9, 0.1)	345.05	344.6	25	10

(P_0) is considered. The proposed algorithm is run for this sample problem for different values of population size. Analyzing the results leads to near optimal value for this parameter. Note that when the population size increases, in each generation, more feasible solutions are generated and this makes the algorithm capable to search a bigger area with lesser iterations but at the same time causes the algorithm to spend more time because the need of more calculations.

Problem P_0, is an assembly line with 5 products and 10 parts in which k_1, k_2, the number of stations located at main line and bypass sub-line, are equal to 10, 3, respectively also c_1, c_2, cycle times for main line and bypass sub-line, are equal to 4,2, respectively. The other common data are shown in Table 1.

A sensitivity analysis is done for population size parameter and the results are shown in Table 2. Note that algorithm is run 10 times for each value of population size and the best and average objective values are reported on the table.

As shown in Table 2, although the run time with population size 30, is greater than the case of the other population sizes such as 15, 20 and 25, but this difference is not considerable. Since the population size 30 makes the algorithm capable to converge to the optimal solution with sharper slope relative to other population sizes thus in this paper 30 is selected as population size. In the same way sensitivity analysis is repeated for other samples with various dimensions. As this expected the same results are obtained (Fig. 7).

Mutation and crossover parameters: In order to obtain acceptable values for mutation and crossover probabilities a sensitivity analysis has been done. Because of the interaction effects between mutation and crossover parameters, a simultaneous sensitivity analysis is done for them using problem P_0. The results are shown in Table 3.

As shown in Table 3, the best values for mutation and crossover probabilities are approximately equal to 0.4 and 0.6 respectively. Note that duplet (0.2, 0.8) is also considered as another candidate for mutation and crossover probabilities.

Computational experiments: Several test problems were used to evaluate the GA in terms of performance compared to optimality as well as performance in terms of CPU time. The experiments are implemented in two folds: first, for small-sized problems, the other for large-sized ones. They cover a diverse set of mixed-model sequencing problems, from the smallest problem with 13860 solutions to the largest one with 2.181×10^{135} possible solutions. In all examples, W_1 and W_2 are set to 1 Also cycle times for the main line and bypass sub-line are set to 4 and 2 minute, respectively.

Small sized problem sets: These problems are designed to validate the proposed GA.In these problems, $k_1 = 7$, $k_2 = 3$ and $H = 3$. And other common data are shown in Table 4. Computation results of

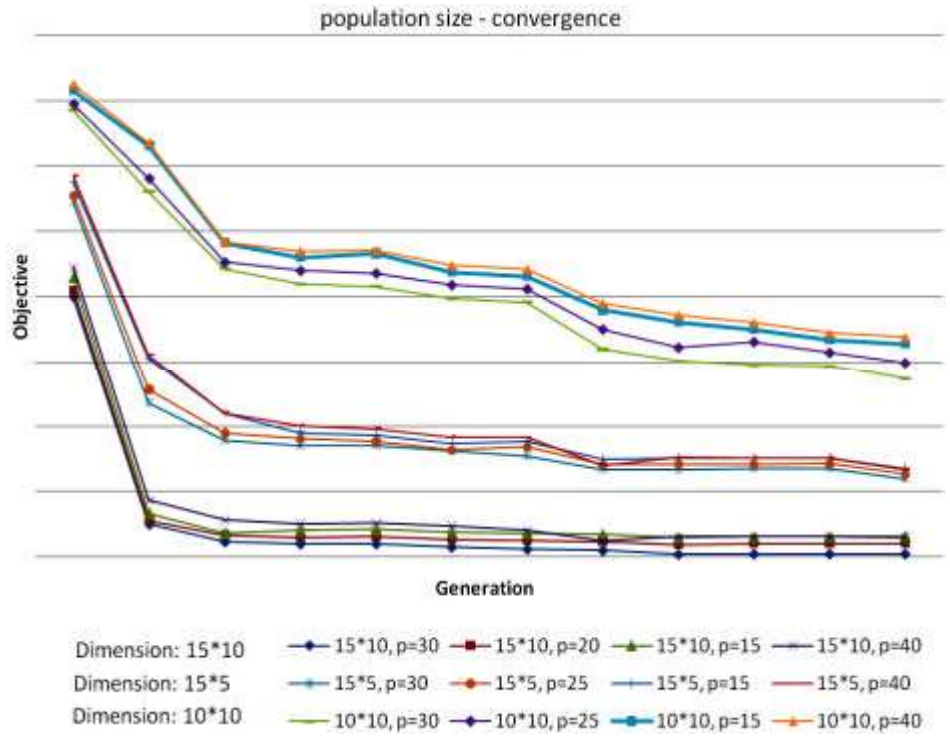


Fig. 7: Sensitivity analysis of population size for three problems with different dimensions

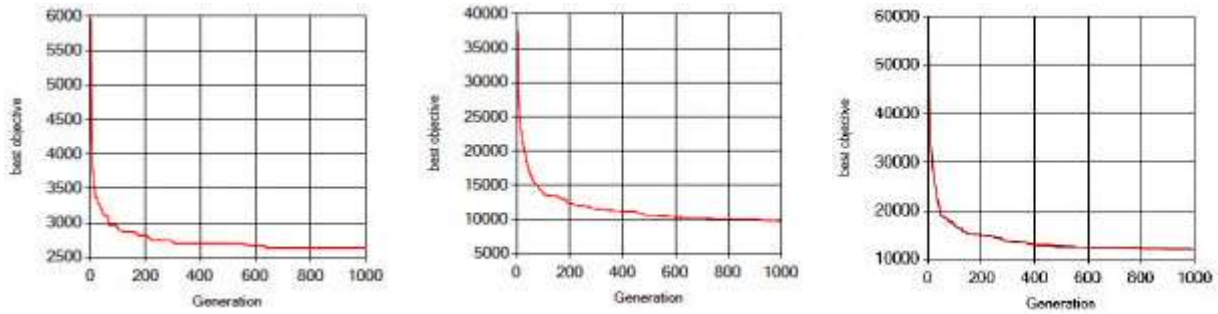


Fig. 8: Convergence of proposed GA in P_{15}, P_{16}, P_{17}

proposed GA against the optimum solution obtained via enumeration exhaustive method are presented in Table 5. The Results show that GA reaches to optimal solution in some small sized problems. In other cases converged to optimal solution in an efficient manner, with deviation at most 2.4% (see column Δ of Table 5). Note that, for all cases proposed GA is run 10 times so the best and average objective values for each case are reported in the table.

Large sized problem sets: These problems are designed to demonstrate efficiency of proposed GA in large scale problems. In this problem set, number of generation is set to 1000. And other common data are shown in Table 6.

A comparison of genetic algorithm results and complete enumeration is summarized in Table 7. This table shows that for large-sized problems, our proposed GA converges to optimal solution in an efficient manner; in contrast, enumeration exhaustive method can't solve the problem with a reasonable amount of time because the number of feasible solutions increases in an exponential manner and EE can not enumerate all of them less than three days. Because of that, the best solution of GA is compared whit the best solution of EE after 10 hours.

Discussing the convergence of proposed GA, convergence graph for large sized problems 15, 16 and 17 are depicted in Fig. 8.

Table 4: Small-sized problems

Problem	Product$_i$	Type	Demand	Number of solutions	Part$_i$ 1	2	3	4	5	6	7	8	9	10
P_1	1	Type1	6	13860	4	2	1	3	5	2	2	0	6	2
	2	Type2	4		6	7	2	7	2	7	0	4	2	2
	3	Type1	2		1	4	3	7	2	6	7	6	0	0
P_2	1	Type1	7	25740	4	2	1	3	5	2	2	0	6	2
	2	Type2	4		6	7	2	7	2	7	0	4	2	2
	3	Type1	2		1	4	3	7	2	6	7	6	0	0
P_3	1	Type1	4	30240	6	4	3	5	7	4	4	2	0	4
	2	Type2	4		7	1	4	1	3	1	2	6	4	4
	3	Type1	4		3	5	5	0	4	0	1	0	2	2
P_4	1	Type1	6	60060	5	4	2	5	6	4	3	2	7	5
	2	Type2	4		1	3	5	2	5	3	3	7	5	5
	3	Type1	3		4	7	6	2	5	1	2	1	3	4
P_5	1	Type1	3	92400	7	5	0	1	6	6	4	2	7	2
	2	Type2	3		4	6	3	6	4	4	0	7	6	5
	3	Type1	2		0	0	3	6	2	3	3	4	5	7
	4	Type2	3		2	3	4	6	5	1	4	7	2	5
P_6	1	Type1	3	92400	5	7	4	4	1	4	2	7	5	4
	2	Type2	2		1	3	1	2	2	0	7	6	1	5
	3	Type1	2		0	0	3	6	0	3	4	5	0	1
	4	Type2	3		2	3	4	6	2	4	2	7	2	0
P_7	1	Type1	3	138600	4	0	7	6	5	4	4	4	4	2
	2	Type2	1		6	0	3	4	5	2	3	4	6	5
	3	Type1	4		2	3	4	6	5	0	7	6	5	2
	4	Type2	1		1	2	6	0	3	4	5	1	2	3
	5	Type1	2		2	3	4	6	5	2	3	4	6	5
P_8	1	Type1	3	277200	3	6	4	4	0	7	6	3	6	4
	2	Type2	2		3	6	2	3	3	4	5	3	6	2
	3	Type1	1		0	0	3	6	0	3	4	5	0	1
	4	Type2	2		2	3	4	6	2	4	2	7	2	0
	5	Type1	3		1	3	5	2	5	3	3	7	5	5
P_9	1	Type2	1	554400	2	3	3	4	5	2	3	3	4	5
	2	Type1	4		3	6	0	3	4	3	6	0	3	4
	3	Type2	1		2	3	1	3	5	2	5	3	3	7
	4	Type1	3		3	6	4	1	3	5	2	5	2	4
	5	Type2	3		7	5	0	1	6	6	4	2	7	7
P_{10}	1	Type1	1	831600	7	5	0	1	6	4	2	1	3	5
	2	Type2	3		2	3	3	7	5	0	1	6	6	4
	3	Type1	4		7	5	0	1	2	3	3	4	4	2
	4	Type2	2		4	6	3	6	3	6	0	3	6	7
	5	Type1	2		0	0	3	6	4	6	2	4	1	4
P_{11}	1	Type1	2	831600	4	2	3	4	2	1	3	5	2	2
	2	Type2	2		6	3	6	7	5	0	1	6	4	2
	3	Type1	1		1	4	6	4	6	3	6	4	6	7
	4	Type2	3		4	2	3	0	0	3	6	2	1	4
	5	Type1	2		6	3	6	7	5	0	1	6	4	2
	6	Type2	1		1	4	6	4	6	3	6	4	6	7
P_{12}	1	Type1	1	1663200	4	2	1	2	3	3	4	5	2	3
	2	Type2	2		6	7	2	3	6	0	3	4	3	6
	3	Type1	3		1	4	3	4	6	2	4	2	4	6
	4	Type2	1		6	7	5	0	1	6	6	7	5	0
	5	Type1	4		6	4	6	3	6	4	6	4	6	3
	6	Type2	1		3	0	0	3	6	2	3	0	0	3

Table 5: Results of small-sized problems

	Exhaustive enumeration		GA				
Problem	Optimum	CPU_{EE} (s)	Number of generations	Best objective	Average objective	CPU_{GA} (s)	Δ%*
P_1	147.90	703	100	147.90	147.90	1.0	0.00
P_2	152.06	2707	100	152.06	152.06	1.5	0.00
P_3	127.36	5458	100	127.36	127.51	2.5	0.00
P_4	113.79	17813	100	113.79	113.79	4.0	0.00
P_5	138.24	27538	100	138.24	139.04	5.0	0.00
P_6	158.32	37456	200	158.32	160.92	6.0	0.00
P_7	232.71	53368	250	233.50	238.20	7.0	0.34
P_8	353.32	72521	300	355.52	356.29	18.0	0.62
P_9	534.57	95047 (~1.1 days)	500	543.77	544.11	75.0	1.72
P_{10}	287.75	121067 (~1.4 days)	500	288.78	291.09	180.0	0.36
P_{11}	424.13	150691 (~1.7 days)	750	434.33	435.05	280.0	2.40
P_{12}	555.01	184023 (~2.1 days)	1000	560.01	562.21	350.0	0.90

Δ% =100×(GA's best objective-Optimum)/Optimum

Table 6: Large-sized problems

Problem	$Product_i$	Type	Demand	Number of solutions	Number of parts	Number of stations
P_{13}	1	Type1	5	4.658×10^{11}	10	10
	2	Type2	10			
	3	Type1	15			
P_{14}	1	Type1	10	1.036×10^{19}	15	15
	2	Type2	15			
	3	Type1	20			
P_{15}	1	Type1	10	6.689×10^{37}	30	30
	2	Type2	15			
	3	Type1	20			
	4	Type2	25			
P_{16}	1	Type1	20	6.770×10^{98}	40	30
	2	Type2	25			
	3	Type1	30			
	4	Type2	35			
	5	Type1	40			
P_{17}	1	Type1	40	2.181×10^{135}	40	30
	2	Type2	40			
	3	Type1	40			
	4	Type2	40			
	5	Type1	40			

Table 7: Results of large-sized problems

	Exhaustive Enumeration	GA			
Problem	Best objective after 10 hours	Best objective	Average objective	CPU_{GA} (s)	Number of generations
P_{13}	450.20	348.00	348.32	580	1000
P_{14}	970.45	685.83	690.99	890	1000
P_{15}	3125.33	2614.15	2629.01	1105	1000
P_{16}	13353.28	9010.67	9053.42	3466	1000
P_{17}	15475.19	11954.20	11995.20	5492	1000

The results show that proposed GA can converge to the optimal solution approximately in 1000 iterations even for large-sized problems.

CONCLUSION

When some different types of products are manufactured at the mixed-model assembly line and assembly times are significantly different among these product types, the production efficiency usually reduces due to the line stoppage. The variation in the assembly times can be reduced by installing a bypass sub-line. This paper investigates the mixed-model assembly line sequencing problem with a bypass sub-line. Problem is determining the sequence of all products in a way that manager's objectives are met. The objective function of the problem is a weighted sum of the two goals: leveling the part usage rates and reducing the line stoppage. The former objective is taken from the literature and the latter is introduced in this paper. Also some unsynchronized situations which are disregarded in previous studies are considered. Due to difficulties arise from these situations; we have to use rule-bases approaches like event-based procedure. Because of NP-hard nature of this problem, a meta-heuristic algorithm based on genetic algorithm and next event procedure is developed to solve the problem. Also a sensitivity analysis is done to obtain near the best values of proposed algorithm parameters. Since mathematical modeling approaches are not applicable in this specific problem, we use an exhaustive enumeration method (optimization algorithm) to calculate optimum solutions. Comparing these two algorithms indicates that proposed meta-heuristic dominate our optimization algorithm. To assess the GA efficiencies, two sets of numerical examples one for small-sized and the other for large-sized problems are presented. Comparison of the results indicates that proposed GA is able to find optimal solutions for small sized problems and near optimal solutions for large-sized ones, in a more reasonable time than optimization algorithm.

Mixed model assembly line sequencing problem with a bypass sub-line definitely doesn't come to an end and the path is still open for researches to develop other meta-heuristic algorithms and compare their results with this paper. Also studying on unsynchronized situations and using decision making techniques to determine optimal rules for them is a promising area for future researches.

REFERENCES

1. McMullen, P.R. and P. Tarasewich, 2005. A beam search heuristic method for mixed-model scheduling with setups. International Journal of Production Economics, 96: 273-283.
2. Monden, Y., 1993. Toyota production system, 2nd edition. Institute of Industrial Engineers, Norcross, GA.
3. Miltenburg, J. and G. Sinnamon, 1989. Scheduling mixed-model multi-level just-in-time production systems. International Journal Production Research, 27: 1487-1509.
4. Miltenburg, J. and G. Sinnamon, 1992. Algorithms for scheduling multi-level just-in-time production systems. IIE Transaction, 24: 121-130.
5. Miltenburg, J. and G. Sinnamon, 1995. Revisiting the mixed-model multi-level just-in-time scheduling problem. International Journal of Production Research, 33: 2049-2052.
6. Morabito, M.A. and M.E. Kraus, 1995. A note on 'scheduling mixed-model multi-level JIT production systems'. International Journal of Production Research, 33: 2061-2063.
7. Steiner, G. and J.S. Yeomans, 1996. Optimal level schedules in mixed-model, multi-level just in time assembly systems with pegging. European Journal of Operational Research, 95: 38-52.
8. Sumichrast, R.T. and E.R. Clayton, 1996. Evaluating sequences for paced, mixed-model assembly lines with JIT component fabrication. International Journal of Production Research, 34: 3125-3143.
9. Sumichrast, R.T. and R.S. Russell, 1990. Evaluating mixed-model assembly line sequencing heuristics for just-in-time production systems. Journal of Operations Management, 9: 371-389.
10. Sumichrast, R.T., R.S. Russell and BW. Taylor, 1992. A comparative analysis of sequencing procedures for mixed-model assembly lines in a just-in-time production system. International Journal of Production Research, 30: 199-214.
11. Yano, C.A. and R. Rachamadugu, 1991. Sequencing to minimize work overload in assembly lines with product options. Management Science, 37: 572-586.
12. Tamura, T., H. Long and K. Ohno, 1999. A sequencing problem to level part usage rates and work loads for a mixed-model assembly line with a bypass sub-line. International Journal of Production Research, 60-61: 557-564.
13. Kubiak, W., 1993. Minimizing variation of production rates in just-in-time systems: A survey. European Journal of Operational Research, 66: 259-271.
14. Xiaobo, Z. and K. Ohno, 1997. Algorithms for sequencing mixed models on an assembly line in a JIT production system. Computer and Industrial Engineering, 32: 47-56.
15. Xiaobo, Z., Z. Zhou and A. Asres, 1999. A not on Toyota's goal of sequencing mixed model on an assembly line, Computers and Industrial Engineering, 36: 57-65.

16. Tsai, L.H., 1995. Mixed-model sequencing to minimize utility work and the risk of conveyor stoppage. Management Science, 41 (3): 485-495.
17. Mansouri, S.A., 2005. A multi objective genetic algorithm for mixed model sequencing on JIT assembly line. European Journal of Operational Research, 167 (3): 696-716.
18. Boysen, N., M. Fliedner and A. Scholl, 2009. Sequencing mixed-model assembly lines: Survey, classification and model critique.European Journal of Operational Research, 192: 349-373.
19. Rahimi-Vahed, A. and A.H. Mirzaei, 2007. A hybrid multi-objective shuffled frog-leaping algorithm for a mixed-model assembly line sequencing problem. Computers and Industrial Engineering, 53: 642-666.
20. Kim, S. and B. Jeong, 2007. Product sequencing problem in Mixed-Model Assembly Line to minimize unfinished works. Computers and Industrial Engineering, 53: 206-214.
21. Rahimi-Vahed, A.R., M. Rabbani, R. Tavakkoli-Moghaddam, S.A. Torabi and F. Jolai, 2007. A multi-objective scatter search for a mixed-model assembly line sequencing problem. Advanced Engineering Informatics, 21: 85-99.
22. Ding, F.Y., J. Zhub and H. Sun, 2006. Comparing two weighted approaches for sequencing mixed-model assembly lines with multiple objectives. Int. J. Production Economics, 102: 108-131.
23. Xiaobo, Z. and K. Ohno, 1994. A sequencing problem for a mixed model assembly line in a JIT production system. Computer and Industrial Engineering, 27: 71-74.
24. Bolat, A., 1994. Sequencing jobs on an automobile assembly line: Objectives and procedures. International Journal of Production Research, 32: 1219-1236.
25. Bard, J., E. Dar-El and A. Shtub, 1992. An analytic framework for sequencing mixed model assembly lines. International Journal of Production Research, 30: 35-48.
26. Inman, R. and R. Bulfin, 1991. Sequencing JIT mixed-model assembly lines. Management Science, 37: 901-904.