Abstract code: 015-0215

Meta Heuristics to Minimize Line Stoppage Time in Mixed-Model Sequencing Problem

Takayoshi Tamura[*1], Tej S. Dhakar[*2], Katsuhisa Ohno[*3], Taiji Okumura[*1]

[*1]*Nagoya Institute of Technology, Nagoya 466-855, Japan*

+81(52)735-5390, tamura.takayoshi@nitech.ac.jp

[*2]*Southern New Hampshire University, Manchester, NH 03106*

(603)644-3106, t.dhakar@snhu.edu

[*3]*Aichi Institute of Technology, Toyota 470-0392, Japan*

+81(565)48-8121, ohno@aitech.ac.jp

POMS 21st Annual Conference

Vancouver, Canada

May 7 to May 10, 2010

Meta Heuristics to Minimize Line Stoppage Time in Mixed-Model Sequencing Problem

Takayoshi Tamura[*1], Tej S. Dhakar[*2], Katsuhisa Ohno[*3], Taiji Okumura[*1]

[*1]*Nagoya Institute of Technology, Nagoya 466-855, Japan*

[*2]*Southern New Hampshire University, Manchester, NH 03106*

[*3]*Aichi Institute of Technology, Toyota 470-0392, Japan*

Mixed-model assembly line is utilized to assemble many product variants on a single line in automobile and other industries. In JIT production system, the autonomation (Jidoka) allows workers to stop a conveyor line whenever they fail to complete their assembly operation within a predetermined process time. Given this situation, it becomes important to determine the production sequence to minimize the total conveyor stoppage time in the mixed-model assembly line. In this research, we propose an explicit and complete formulation of the production sequencing problem for the mixed-model line that aims to minimize the total line stoppage time. Meta heuristics based on Simulated annealing (SA), Tabu search (TS) and Genetic algorithm (GA) are developed. An algorithm to evaluate the total line stoppage time based on the formulation is proposed to save computation time compared to simulation.

Keywords: Mixed-model assembly line, production sequencing, line stoppage time minimization, utility work minimization, and meta heuristics

1. Introduction

Mixed-model assembly line is utilized to assemble many product variants on a single line in automobile and other industries [1]. In JIT production system, the autonomation (Jidoka) allows workers to stop a conveyor line whenever they fail to complete their assembly operation within a predetermined process time. Given this situation, it becomes important to determine the production sequence to minimize the total conveyor stoppage time in the mixed-model assembly line [1]-[4].

The production sequencing problem in the mixed- model assembly line has become an important practical research area in order to enhance the productivity of the assembly line. Most of the times, minimizing the utility work or the work load fluctuations are selected as the objectives (see [1][5]-[12]) instead of minimizing the line stoppage time. Evaluating the line stoppage time for a given production sequence consumes much more time than evaluating the utility work or the work load fluctuations.

Next section presents the formulation of the problem by Tamura, et al. [13]. Section 3 discusses the impact of objective functions such as to minimize the utility work or to level the work load on the minimization of the line stoppage time. The development of efficient algorithm to evaluate the line stoppage time using the exact formulation of the problem when a production sequence is given, is given in Section 4. The next section presents meta heuristic algorithms based on Simulated annealing (SA), Genetic algorithm (GA), and Tabu search (TA). Their efficiencies are compared with each other as well as Goal chasing method (see [1]) using numerical examples.

2. Problem formulation

2.1 Assumptions

The following assumptions are made in the research presented in this paper.

(1) A worker is assigned to each station.

(2) A conveyor line with a given constant speed is considered.

(3) Each product is moved continuously by the conveyor and each worker assembles the product moving along the conveyor in his/her given working area.

(4) Distance between two successive products on the conveyor (i.e. cycle time) is a given constant.

(5) If a task on a product at a station is not completed until the end of the station, line stoppage occurs at the end of the station until the task is completed.

(6) During the line stoppage, each worker continues a task at his/her station.

(7) Service discipline at each station is first-come-first served (FCFS). No interruption of a task is permitted.

(8) Assembly time of a task is deterministic and known.

(9) Working areas of two successive stations do not overlap and a worker is not permitted to cross the boundary between his station and the adjacent stations.

(10) A walk from a product to the next product takes a given time which is constant for each work station and is independent of the line stoppage time.

2.2 Notation

The following notation is used in the problem formulation.

M = Number of stations,

K = Number of products to be sequenced,

L_m = Length (of working area) of station m,

D = Distance between two successive products,

w_m = Time required to walk from a product to the next product on the line (walk time to the next product),

v_c = Conveyor speed,

v_w = Walk speed of worker,

$i(k)$ = k th product in a production sequence,

$r_{k,m}$ = Arrival time of product $i(k)$ at station m,

$b_{k,m}$ = Process start time of product $i(k)$ at station m,

$f_{k,m}$ = Process completion time of product $i(k)$ at station m,

$s_{k,m}$ = Line stoppage time occurred for product $i(k)$ at station m,

$t_{k,m}$ = Process time (i.e. assembly time) of product $i(k)$ at station m,

$a_{i,m}$ = Process time of product i at station m,

$y_{i,k}$ = Binary variable representing a production sequence, which equals one when product i occupies the k th position in a production sequence, otherwise equals zero.

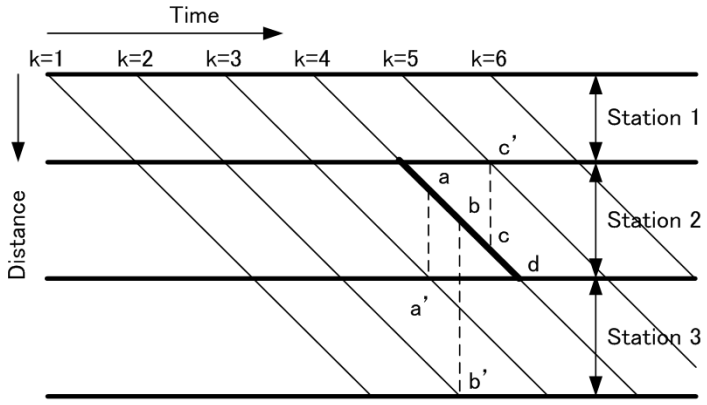

Figure 1: An example of positions where line stoppages may occur when product $i(4)$ moves within station 2.

2.3 Problem formulation

To formulate the problem explicitly the following two types of sets are defined [13].

Line stoppages of $i(k)$ within station m which are caused by other products occur when they are at station ends of the following set $Q_{k,m}$:

$$Q_{k,m} = \{(p,q) \mid (k-1)D + \sum_{j=1}^{m} L_j > (p-1)D + \sum_{j=1}^{q} L_j > (k-1)D + \sum_{j=1}^{m-1} L_j\}$$

$$\cup \{(p,q) \mid (k-1)D + \sum_{j=1}^{m} L_j = (p-1)D + \sum_{j=1}^{q} L_j, k > p\} \quad \text{for all } k,m \qquad (1)$$

The second term in equation (1)

$$\{(p,q) \mid (k-1)D + \sum_{j=1}^{m} L_j = (p-1)D + \sum_{j=1}^{q} L_j, k > p\}$$

is to clarify which product causes a line stoppage when two or more products come across boundaries of stations at the same time. For $(p,q) \in Q_{k,m}$, if the p^{th} product $i(p)$ causes a stoppage at the end of station q, product $i(k)$ is stopped at a position in station m. The set is used to compute the arrival time of product $i(k)$ at the next station based on the previous station.

In order to compute the arrival time of the next product at station m based on the arrival time of the previous product, another set denoted by $R_{k,m}$ is defined, which is the set of line stoppage positions during the time interval between arrival time $r_{k,m}$ of product $i(k)$ and arrival time $r_{k+1,m}$ of the next product at station m. The set is given by the following equations:

$$R_{k,1} = \{(p,q) \mid kD > (p-1)D + \sum_{j=1}^{q} L_j > (k-1)D\} \quad \text{for all } k \qquad (2)_1$$

$$R_{k,m} = \{(p,q) \mid kD + \sum_{j=1}^{m-1} L_j > (p-1)D + \sum_{j=1}^{q} L_j > (k-1)D + \sum_{j=1}^{m-1} L_j\}$$

$$\cup \{(p,q) \mid kD + \sum_{j=1}^{m-1} L_j = (p-1)D + \sum_{j=1}^{q} L_j, k > p\} \quad \text{for all } m \geq 2, k \qquad (2)_2$$

Consider positions at which line stoppages occur in Figure 1, where the horizontal and vertical axes are time and distance, respectively. Each oblique line in the figure corresponds to a product and each point on the oblique line shows the possible position of the product where a line stoppage occurs. Since these oblique lines are parallel and have the same interval D in distance (or D/v_c in terms of time), as Figure 1 shows, the line stoppage may occur at positions such as a, b, c and d while the product $i(4)$ is within station 2, if a line stoppage

does occur. In Figure 1, for example set $Q_{4,2}$ corresponds to a set of positions a, b and c and is given by $Q_{4,2} = R_{4,2} = \{(3,2), (2,3), (5,1)\}$.

Using sets $Q_{k,m}$ and $R_{k,m}$, the problem to minimize the total line stoppage time LS in the mixed model assembly line is formulated by the equations (3)-(13), where equation (3) is the objective function and equations (4)-(13) are constraints of the problem.

$$\text{Minimize} \quad LS = \sum_{k=1}^{K} \sum_{m=1}^{M} s_{k,m} \tag{3}$$

subject to

$$\sum_{i=1}^{K} y_{i,k} = 1 \qquad \text{for all} \quad k \tag{4}$$

$$\sum_{k=1}^{K} y_{i,k} = 1 \qquad \text{for all} \quad i \tag{5}$$

$$b_{k,m} \geq r_{k,m} \qquad \text{for all} \quad k,m \tag{6}$$

$$f_{k,m} = b_{k,m} + t_{k,m} \qquad \text{for all} \quad k,m \tag{7}$$

$$t_{k,m} = \sum_{i=1}^{K} a_{i,m} y_{i,k} \qquad \text{for all} \quad k,m \tag{8}$$

$$r_{k,m+1} \geq f_{k,m} \qquad \text{for all} \quad k,m \tag{9}$$

$$b_{k+1,m} \geq f_{k,m} + w_m \qquad \text{for all} \quad k,m \tag{10}$$

$$r_{k,m+1} = r_{k,m} + \frac{L_m}{v_c} + \sum_{(p,q) \in Q_{k,m}} s_{p,q} + s_{k,m} \qquad \text{for all} \quad k,m \tag{11}$$

$$r_{k+1,m} = r_{k,m} + \frac{D}{v_c} + \sum_{(p,q) \in R_{k,m}} s_{p,q} \qquad \text{for all} \quad k,m \tag{12}$$

$$r_{k,m}, b_{k,m}, f_{k,m}, s_{k,m} \geq 0, \quad y_{i,k} \in \{0,1\} \qquad \text{for all} \quad i,k,m \tag{13}$$

Equations (4) and (5) ensure that each product i is assigned to a different position k in a production sequence. Equation (6) represents that assembly task of product $i(k)$ at station m starts after arriving at the station. Based on the start time, completion time of a task for product $i(k)$ at station m is computed by equation (7). Equation (9) ensures that product $i(k)$ goes to the next station after completing the task assigned to station m, where

$r_{k,M+1}$ represents the time when product $i(k)$ comes off the assembly line. Equation (10) ensures that the processing of the next product $i(k+1)$ at station m starts after completion of product $i(k)$ at the station. Equation (11) is the time spent by product $i(k)$ at station m and equation (12) ensures that two successive products must maintain a constant time interval D/v_c even if line stoppages occur.

It can be shown that this formulation is necessary and sufficient (see Tamura, et al. [13]).

3. Are Alternative Goals Sufficient to Minimize Line Stoppage Time?

3.1 Utility work minimization and work load smoothing

In the mixed-model assembly line sequencing problem, utility work minimization and work load smoothing as well as minimization of line stoppage time are used as objective functions in order to enhance the efficiency of the assembly line. The utility work minimization [5]-[9] and work load fluctuation minimization [10]-[12] objectives are relatively easy to evaluate than line stoppage time which requires a simulation for a given production sequence. Also since the problem formulation for the utility work minimization or work load smoothing is simpler, it is possible to develop relatively simple heuristics for them. For the line stoppage time minimization problem, exact formulation of the problem has been considerably more difficult [4][6][13].

The purpose of this section is to discuss the effectiveness of utility work minimization and work load smoothing as objective functions for the mixed-model sequencing problem as compared to the line stoppage time objective function. This comparison will be achieved by numerical experiments.

The work load fluctuation denoted by WLD is defined as the deviation of work load of each product at each station. In the paper the value of WLD is determined by the following equation [1]:

$$WLD = \sum_{k=1}^{K} \sum_{m=1}^{M} \left(\frac{k}{K} T_m - \sum_{j=1}^{k} a_{i(j),m} \right)^2 \tag{14}$$

where

$$T_m = \sum_{i=1}^{K} a_{i,m} \tag{15}$$

When worker's walk time w_m to go back to the next product is ignored, the utility work denoted by UW [5][7] for a given production sequence is computed by

$$UW = \sum_{k=1}^{K} \sum_{m=1}^{M} \max\left\{0, \hat{b}_{i(k),m} + a_{i(k),m} - L_m / v_c\right\} \tag{16}$$

where

$$\hat{b}_{i(k),m} = \max\left\{0, e_{i(k-1),m} - D / v_c\right\} \tag{17}$$

$$e_{i(k),m} = \min\left\{\hat{b}_{i(k),m} + a_{i(k),m}, L_m / v_c\right\} \tag{18}$$

3.2 Two types of data sets used in numerical experiments

For the experiments reported in the paper, two types of assembly time data were generated using two types of normal random numbers as follows:

(1) Type I

Step 1. Generate an average assembly time $\bar{a}_i$ for each product i using the normal distribution $N(\mu, \mu\sigma_P)$.

Step 2. Generate assembly time $a_{i,m}$ for each product i at station m using the normal distribution $N(\bar{a}_i, \bar{a}_i\sigma_S)$.

(2) Type II

Step 1. Generate an average assembly time $\bar{b}_m$ for each station m using the normal distribution $N(\mu, \mu\sigma_S)$.

Step 2. Generate assembly time $a_{i,m}$ for each product i at each station m using the normal distribution $N(\bar{b}_m, \bar{b}_m\sigma_P)$.

For both type I and type II, the expected value of $a_{i,m}$ equals μ and the coefficient of variation (denoted by σ) is computed as follows:

$$\sigma^2 = \sigma_P^2 + \sigma_S^2 + \sigma_P^2\sigma_S^2 \tag{19}$$

However, the assembly time $a_{i,m}$ of type I differs from $a_{i,m}$ of type II in terms of variance. The $a_{i,m}$ of type I has a larger variance among products, while $a_{i,m}$ of type II has a larger variance among stations. For type I $a_{i,m}$ data, the variation among products and variation among stations are given by the following equations respectively, based on simple stochastic analysis:

$$E[(\frac{1}{M}\sum_m a_{i,m} - \mu)^2] = V_P(a_{i,m}) = \mu^2\sigma_P^2 + \frac{1}{M}\mu^2\sigma_S^2 + \frac{1}{M}\mu^2\sigma_P^2\sigma_S^2 \tag{20}$$

$$E[(\frac{1}{K}\sum_i a_{i,m} - \mu)^2] = V_S(a_{i,m}) = \frac{1}{K}\mu^2\sigma_P^2 + \frac{1}{K}\mu^2\sigma_S^2 + \frac{1}{K}\mu^2\sigma_P^2\sigma_S^2 \tag{21}$$

Table 1 shows an example of assembly time $a_{i,m}$ of type I generated by setting $\mu = 450$ and $\sigma_P = \sigma_S = 0.1$. Then, $\sigma = 0.142$, $V_P(a_{i,m}) = 2025$, and $V_S(a_{i,m}) = 581$.

Table 1. Data of assembly times

Item	Station					
	1	2	3	4	5	6
1	455	481	544	449	374	400
	366	403	404	390	375	354
3	428	394	410	461	472	376
4	526	506	482	530	548	550
5	510	521	534	435	527	576
6	421	434	404	367	442	419
7	483	541	506	577	521	484

3.3 Numerical experiment A

The discussion whether WLD and UW are effective or not for reducing the line stoppage time can be made directly by enumerating every production schedule and depicting a scatter graph of relationship between UW (or WLD) and LS for all production sequences for examples of small size.

A simulation experiment was executed using data given in Table 1, where $K = 7$ (7!=5040), $M = 6$, and $\mu = 450$ seconds. Other data were set to $v_c = 0.6$ m/min (1 cm/sec), $D = 5$ m ($D/v_c = 500$ seconds), and $L_m = 5.14$ m ($L_m/v_c = 514$ seconds). The station length L_m was set to $\mu(1+\sigma)v_c$, where $\mu\sigma v_c$ (or $\mu\sigma$) is called an allowance of station length in the paper.

Figure 2 and Figure 3 are scatter graphs for the utility work and the work load fluctuation, respectively, which are depicted using 5,040 production schedules. From two figures, we neither see correlation between utility work UW and line stoppage time LS, nor between work load fluctuation WLD and LS. Hence, minimizing UW or WLD would minimize LS only if we are lucky.

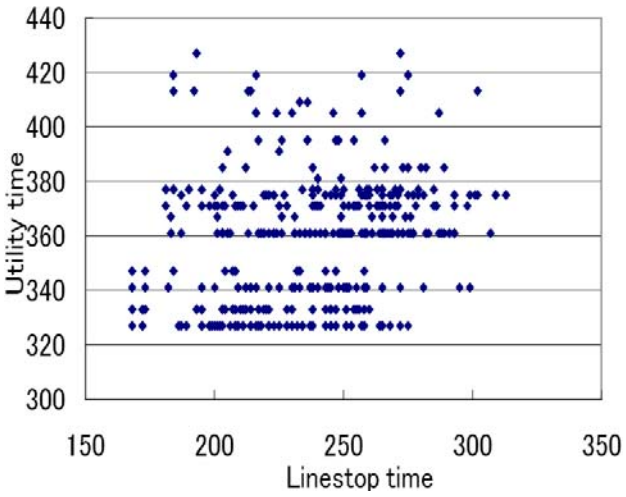

Figure 2. Scatter graph between *UW* and *LS* when $L_m = \mu(1+\sigma)v_c$.

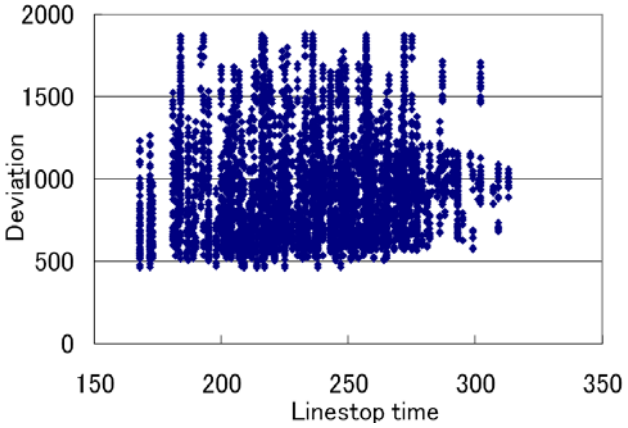

Figure 3. Scatter graph between *WLD* (represented by 'Deviation') and *LS* when $L_m = \mu(1+\sigma)v_c$.

Scatter graphs for UW and WLD using the same data except $L_m = 5.78\text{m}$ $(= \mu(1+2\sigma)v_c)$ are given in Figure 4 and Figure 5. Figure 4 shows a strong correlation between UW and LS. From Figure 2 and Figure 4 we conclude that the UW can be used as an alternative objective function of the LS when the allowance of station length is sufficiently long. However, Figure 5 scarcely shows a correlation between WLD and LS, even if the allowance of station length is long. In this example, the minimal line stoppage time (equal to zero) is achieved at the minimal value of WLD. This is a chance occurrence.

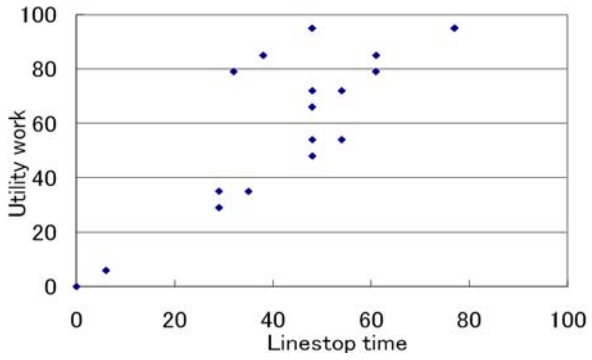

Figure 4. Scatter graph between *UW* and *LS* when $L_m = \mu(1+2\sigma)v_c$.

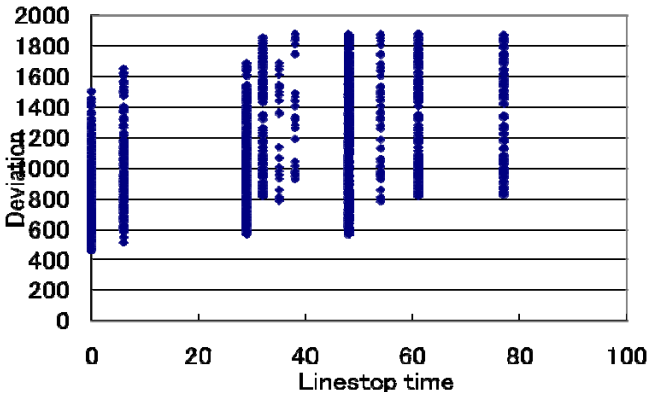

Figure 5. Scatter graph between *WLD* (represented by 'Deviation') and *LS* when $L_m = \mu(1+2\sigma)v_c$.

From many numerical examples, where all production sequences were enumerated and their *UW* and *WLD* plotted on scatter graphs for each example, we can conclude as follows:

(1) Line stoppage time *LS* varies widely among production sequences, and hence it is important to determine a production sequence that minimizes *LS* in order to enhance production efficiency.

(2) Utility work *UW* may be used as an alternative objective function of *LS* when the allowance of station length is large, i.e. when the minimal line stoppage time is short. Nevertheless, the dependence on the allowance of station length restricts the use of UW in place of *LS*.

(3) Work load fluctuation *WLD* may not be appropriate to use as an alternative objective function of *LS* even if the allowance of station length is large.

In case of assembly lines for cars, smoothing of the part usage rate is also important [17]-[19]. The objective function includes work load leveling as well as minimizing stock level of parts allocated beside the assembly line. In the car sequencing problem, assembly time $a_{i,m}$ for product i at station m depends on the product structure, i.e. $a_{i,m}$ becomes

a larger value when an optional part is attached to product i at station m [18]. For the product sequencing problem with the assembly time structure same as the car sequencing problem, numerical experiments show results similar to conclusions (1) to (3) mentioned above.

3.4 Numerical experiment B

For large size problems with a number of products, we can evaluate the impact of minimizing the utility work or the work load fluctuation on the line stoppage time minimization. In order to discuss the impact, numerical experiments were conducted. Data were set as follows: $K = 50$, $M = 10$, $v_c = 1$ cm/sec, $\mu = 450$ seconds, $D = 500$ seconds, $w_m = 4$ seconds, and

(1) $\sigma_I \in \{0.1, 0.3\}$,

(2) $\sigma_S \in \{0.1, 0.3\}$,

(3) $L_m \in \{ \mu(1+\sigma)v_c , \mu(1+2\sigma)v_c \}$.

Where the value of σ is computed by Equation (19) and assembly time $a_{i,m}$ is generated by using the type I procedure.

In order to obtain a near-optimal production sequence for each of objective functions LS, UW, and WLD, a simple SA (Simulated annealing) algorithm is applied (see [7][15]). The process of the SA algorithm is as follows:

SA algorithm

Step 1: Initialization: Set iterations $Itr \coloneqq 0$ and $I_T \coloneqq 0$. Set temperature $\tau \coloneqq \tau_0$, where notation τ_0 represents the initial temperature. Generate an initial sequence π and evaluate the objective function value $F(\pi)$.

Step 2: Set $Itr \coloneqq Itr + 1$ and $I_T \coloneqq I_T + 1$. If $Itr > I_0$ then set $\tau \coloneqq \alpha \times \tau$ and $Itr \coloneqq 0$.

Step 3: If $\tau < \tau_f$ or $I_T > I_{\max}$, then terminate the algorithm.

Step 4: Generate a neighborhood of π which is denoted by π' and compute $F(\pi')$.

Step 5: Compute $\Delta F = F(\pi') - F(\pi)$. If $\Delta F < 0$ or $\lambda < \exp(-\Delta F / \tau)$ for a random number λ generated uniformly, update by setting $\pi := \pi'$, $F(\pi) := F(\pi')$, and $Itr := 0$. Go back Step 2.

A neighborhood π' is generated by the following steps: (1) a position (denoted by k) in the current production sequence π is selected randomly, and (2) $i(k)$ and $i(k+1)$ are exchanged if $k < K$, otherwise $i(k)$ and $i(k-1)$ are exchanged. Parameters predetermined by additional experiments are set to $\tau_0 = 2000$, $I_0 = 100$, and $\alpha = 0.95$. The termination conditions are set to $\tau_f = 0.1$ and $I_{\max} = 16000$.

Experimental results are given in Figure 6 through Figure 8. Figure 6 and Figure 8 show improvement rates of line stoppage time *LS* by minimizing each of objective functions *LS*, *UW*, and *WLD*. Figure 6 is obtained for $L_m = \mu(1+\sigma)v_c$ and Figure 8 for $L_m = \mu(1+2\sigma)v_c$. The improvement rate is defined by $100 \times (LS_{init} - LS_{best}) \div LS_{init}$ (%), where

LS_{init} : An initial value of *LS* which is evaluated for an initial sequence. This value is used for all algorithms.

LS_{best} : *LS* value evaluated for the best production sequence of each objective function obtained by SA algorithm.

In the experiment, two data sets are generated. Figure 6 through 8 show the average values of results obtained for these two data sets. From Figure 8, we can see that the minimization of *UW* sometimes achieves values of *LS* close to the ones obtained by the algorithm to minimize *LS* when the allowance of station length is larger, i.e. $L_m = \mu(1+2\sigma)v_c$. However even when L_m is set to $\mu(1+2\sigma)v_c$, minimization of *UW* may result in a worse *LS*, and hence the minimization of *UW* is not reliable, i.e. unstable to use as an alternative objective function for *LS* minimization.

The minimization of *WLD* results in fairly inferior result for every level of (σ_P, σ_S) even when the allowance of station length is relatively large, although the minimization of

WLD does do well on the work load smoothing as shown in Figure 7. It is easy to show that the minimization of *WLD* becomes effective as an objective function to minimize *LS* when the production line consists of only one station.

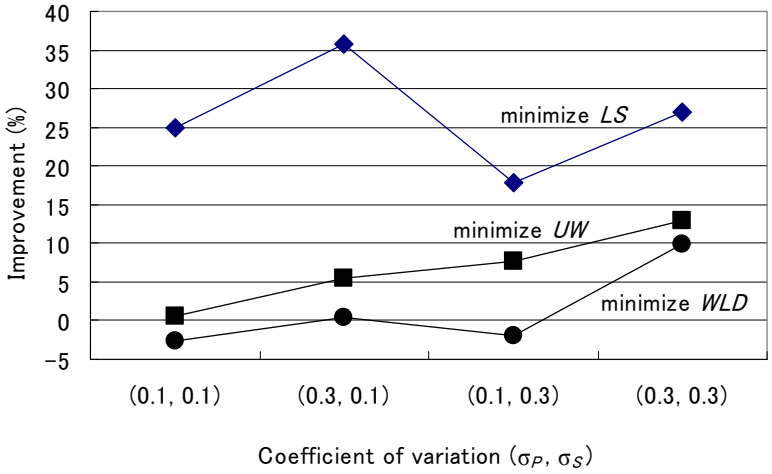

Figure 6. Improvement rate of line stoppage time *LS* by minimizing each of the objective functions *LS*, *UW*, and *WLD* when $L_m = \mu(1+\sigma)v_c$.

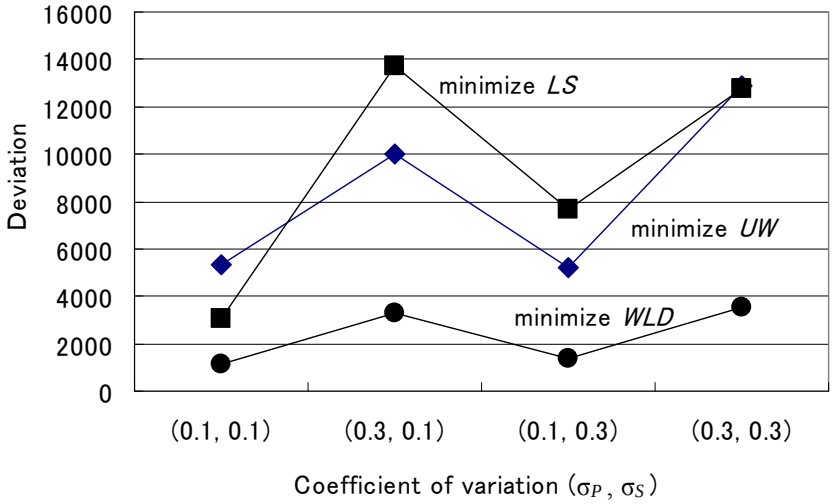

Figure 7. Work load fluctuation *WLD* for the best production sequence obtained by minimizing each of objective functions *LS*, *UW*, and *WLD* when $L_m = \mu(1+\sigma)v_c$.

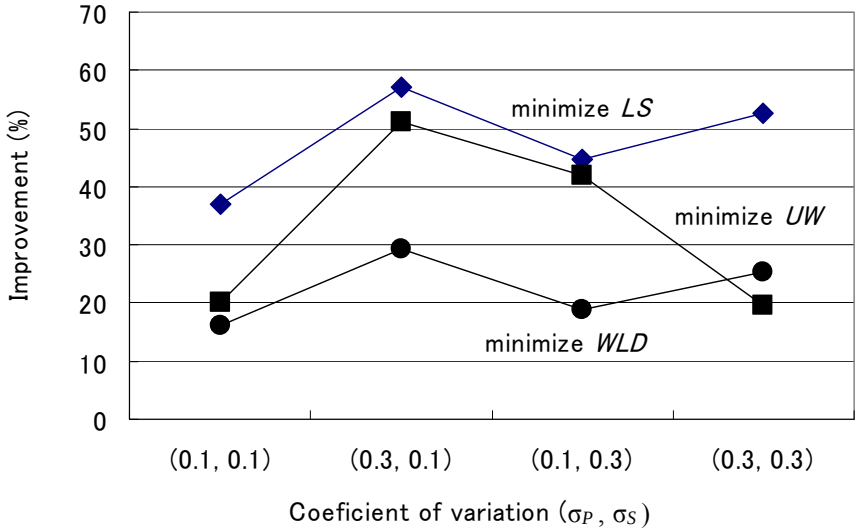

Figure 8. Improvement rate of line stoppage time *LS* by minimizing each of the objective functions *LS*, *UW*, and *WLD* when $L_m = \mu(1+2\sigma)v_c$.

4. Development of Algorithms Using Meta Heuristics

In the development of algorithms using meta heuristics, previous research has concentrated on obtaining a more precise solution. However, we concentrate here on developing an efficient procedure to evaluate the line stoppage time using the problem formulation given in Section 2. The evaluation of the objective function to minimize the line stoppage time relies on simulation in the traditional algorithms [3][4]. The CPU time for algorithms using meta heuristics is here consumed mainly in evaluating the objective function, which restricts the use of the algorithms to find an optimal or a near-optimal sequence for a real assembly line.

4.1 Line stoppage time evaluation procedure

(1) To arrange each pair (k,m) in ascending order of arrival time $r_{k,m}$

We first arrange each pair (k,m) in ascending order of arrival time $r_{k,m}$. Note that the order does not change when a line stoppage occurs. The order represented by notation $A(\ell)$ is determined by the following steps:

Step 1: Compute $r_{k,m}$ for all (k,m) by setting line stoppage time $s_{k,m}$ to zero in equations (11) and (12), where $r_{1,1}=0$. Set $r_{k,m}$ to $r_{k,m}^{(0)}$ for every (k,m).

Step 2: Set $B \coloneqq \{(k,m) \mid K \geq k \geq 1, 1 \leq m \leq M+1\}$ and $\ell = 1$.

Step 3: $A(\ell) \coloneqq (k,m) = \min_{k'} \{(k',m') \mid (k',m') = \text{index} \min_{(a,b)\in B} r_{a,b}^{(0)}\}$

Step 4: Update B by $B \coloneqq B / A(\ell)$. If $B = \phi$, terminate the algorithm.

Step 5: $\ell \coloneqq \ell + 1$. Go back *Step* 3.

In Step 3, term $\min_{k'} \{(k',m') \mid \cdots\}$ corresponds to the second term of $Q_{k,m}$ in equation (1).

(2) A new procedure to evaluate the total line stoppage time *LS*

When a production sequence is given, the total line stoppage time *LS* is evaluated by the following steps, where $LS(\ell)$ represents the total line stoppage time yielded until $r_{k,m}$, where $A(\ell) = (k,m)$:

Step 1: Initialization: $\ell = 1$, $LS(1) = 0$, $r_{1,1} = r_{1,1}^{(0)}$, $b_{1,1} = r_{1,1}$, and $f_{1,1} = b_{1,1} + t_{1,1}$.

Step 2: $\ell \coloneqq \ell + 1$. If $\ell > K(M+1)$, then the algorithm.

Step 3: If $(k,1) \in A(\ell)$, then

$r_{k,1} = r_{k,1}^{(0)} + LS(\ell - 1)$,

$b_{k,1} = \max\{f_{k-1,1} + w_1, r_{k,1}\}$,

$f_{k,1} = b_{k,1} + t_{k,1}$, and

$LS(\ell) = LS(\ell - 1)$. Go back to *Step* 2.

Step 4: If $m \geq 2$ for $(k,m) \in A(\ell)$, compute the following values:

$$s_{k,m-1} = \max\{0,\ f_{k,m-1} - r_{k,m-1}^{(0)} - \frac{L_{m-1}}{v_c} - LS(\ell - 1)\},$$

$LS(\ell) = LS(\ell - 1) + s_{k,m-1}$,

$r_{k,m} = r_{k,m}^{(0)} + LS(\ell)$,

$b_{k,m} = \max\{f_{k-1,m} + w_m, r_{k,m}\}$, and

$f_{k,m} = b_{k,m} + t_{k,m}$. Go back to *step* 2.

The first equation in *Step* 4 can be derived from Equations (9) and (11). It is easy to show that the algorithm satisfies all constraints of the problem formulated in Section 2.3.

(3) A simulation procedure to evaluate the total line stoppage time *LS*

The total line stoppage time *LS* is traditionally evaluated by simulation. Therefore, it is necessary to define a simulation process in order to compare the efficiency of the proposed *LS* evaluation algorithm with a traditional simulation process. The simulation time elapses by the following four events A to D:

Even A: Time when a task at station *m* completes,

Event B: Time when a product reaches the end of station *m*,

Event C: Time when a worker at station *m* reaches the next product or the start position of the station, and

Event D: Time when a product is loaded onto the assembly line.

The simulation steps can be sketched roughly for a given production sequence as follows:

Step 1: Set the initial event at the first station.

Step 2: Find an event with the first event time. If the event is empty, then terminate the simulation. Otherwise the current time is skipped to the time when the event occurs. Select a step from *Steps* 3 to 6 depending on the event and go to that step. It is supposed that the event occurs at station *m*.

Step 3: Completion of a task at station *m* (event A)

1) Set event C for station *m*, which is a worker's event walking back to the next product.
2) If the assembly line stops and the stoppage is caused by station *m*, then make the assembly line resume work.

3) If there is another task to be executed currently at the end of a station, stop the assembly line and update $r_{p,q}$ for all products on the line by adding the line stoppage time.

Step 4: A product reaches the end of station *m* (event B)

1) When for a product the assembly task at station *m* has been completed, the following steps are executed; enter the next station $m+1$ and set event B at station $m+1$. Then if the worker at station $m+1$ is idle, start the assembly task there and set event A at station $m+1$.
2) When for a product the assembly task at station *m* is not completed, make the assembly line stop and update $r_{p,q}$ for all products on the line.

Step 5: Worker at station *m* to reaches the next product or the start position of the station (event C)

1) If the worker reaches the next product, start the task and set event A.
2) If the next product does not arrive at the station, set the worker to idle.

Step 6: A product is loaded onto the assembly line (event D)

1) If the worker at station 1 is idle, start the task and set event A at station 1 and event B at the end of station 1.
2) Set the new event D.

4.2 Determining the computational complexity to evaluate the line stoppage time for a given production sequence

(1) The proposed procedure to evaluate the total line stoppage time *LS*

In the procedure given in (1) of Section 4.1 $A(\ell)$ is set up once in a meta heuristic algorithm such as SA, TS and GA. We therefore estimate a computation order for the second procedure given in (2) of Section 4.1. In one iteration of the procedure values of $s_{k,m}$, $r_{k,m}$, $b_{k,m}$, $f_{k,m}$,

and $LS(\ell)$ for each pair (k,m) are computed. Therefore, the computational complexity becomes $O(KM)$.

(2) The simulation method to evaluate the total line stoppage time *LS*

The computational complexity in Step 2 of the simulation becomes $O(M)$ for each event finding. Since the number of events in the complete iteration is $O(KM)$, the computational complexity in Step 2 is given by $O(KM^2)$. The computational complexity in Steps 3 to 6 is totally $O(KM)$ although the time is longer than the event finding for each event.

(3) Comparison of two procedures by a numerical experiment

In order to compare the computation time of the proposed procedure to the time for simulation, a numerical experiment is conducted. Data is set as follows: $L_m = 5.34$ m, $D = 5$ m (500 seconds), $v_c = 1$ cm/sec, $w_m = 4$ seconds, $K \in \{10, 30, 50\}$, and $M \in \{4, 7, 10\}$. A set of assembly times is generated for each K and M using Type I mentioned in Section 3.2, where $\mu = 450$, $\sigma_P = 0.2$, and $\sigma_S = 0.1$. For each numerical example 10,000 production sequences are generated by exchanging two randomly selected adjacent products.

Table 2 and Table 3 show the CPU time for each K and M obtained by the proposed procedure and simulation, respectively. Figure 9 shows a relationship between the number of stations and the CPU time, where $K = 50$. From Table 2 and Table 3, we estimate a regression curve for each procedure, which are given by

$$CPU_{proposed}(K,M) = 0.0654 \times K \times M \tag{22}$$

$$CPU_{simulation}(K,M) = 0.1388 \times K \times M + 0.0075 \times K \times M^2 \tag{23}$$

As we can find from tables, for $K = 30$ and 50, the CPU time using the simulation is around twice that of the proposed procedure when $M = 4$ and becomes more than triple when $M = 10$.

Table 2. CPU times (in second) of 10,000 sequences using the proposed procedure.

Number of items		10	30	50
Number of stations	4	0	9	15
	7	4	14	23
	10	7	19	32

Table 3. Computation times (in second) of 10,000 sequences using the simulation.

Number of items		10	30	50
Number of stations	4	6	20	34
	7	12	40	67
	10	20	63	107

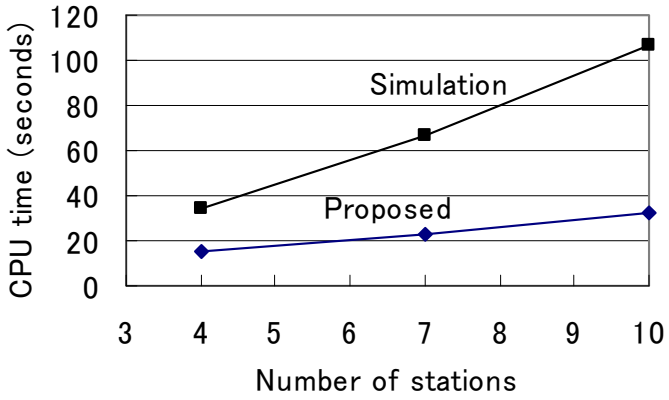

Figure 9. CPU times corresponding to the number of stations, where $K = 50$.

4.3 Development of algorithms using meta heuristics

This section develops three algorithms using SA, TS and GA [7][15], respectively, as well as Goal chasing (GC) [1] and compares their performance based on numerical experiments. The SA, TS, and GA developed here are basic ones.

(1) SA algorithm

The algorithm mentioned in Section 3.4 is used.

(2) TS algorithm

In this research, a probabilistic TS (see [15]) is used to avoid that the algorithm converges to a local optimal solution. Notation used in TS algorithm is defined as follows:

Itr : Index of iterations,

$I_{\max}$: Maximum number of iterations to terminate the algorithm,

N : Number of neighborhoods generated at each iteration,

TLL : Tabu list length,

$F_{\min}$: Current best objective value.

1) Neighborhood and tabu list

In this paper, a neighborhood of the current solution is defined in the same way as in SA. The best sequence becomes the next solution, which is selected from N neighborhoods generated randomly. Tabu list memorizes locations where changed units were placed in the previous sequences. This information is held during TLL iterations in the tabu list.

2) TS algorithm

TS algorithm consists of the following steps:

Step 1: Set $Itr := 1$. Initialize the tabu list as an empty set. Generate an initial solution π and evaluate $F(\pi)$. Set $F_{\min} := F(\pi)$.

Step 2: Update $Itr := Itr + 1$. If $Itr > I_{\max}$, then terminate the algorithm.

Step 3: Generate N neighborhoods randomly for the current solution π while taking account of the tabu list together with the aspiration strategy. For each of the N neighborhoods, evaluate $F(\pi')$.

Step 4: Update π by the best sequence of the N neighborhoods.

Step 5: Update the tabu list.

Step 6: If $F_{\min} > F(\pi)$, update $F_{\min} := F(\pi)$. Go back to *Step* 2.

(3) GA algorithm

1) Crossover

In order to avoid creation of off-springs that represent invalid solutions of the problem, we use an order-based crossover proposed by Dewdney [22]. In the order-based crossover, each gene in a sequence is represented by its relative position based on a basic sequence. Consider an example of this representation as shown in Figure 10. Let ABCDE be the basic sequence for a set of five units. Then, a sequence ACBED has the genotype representation 12121, since gene C has the second position in the remaining sub-sequence BCDE, gene B the first position in sub-sequence BDE and so on.

In this research, 2-point crossover is applied, in which two parents represented by the genotype are randomly selected and two random cut points are generated. The genes in the first and third parts of a chromosome divided by the two cut points are copied to the corresponding child and the genes between the two cut points are copied to the other child.

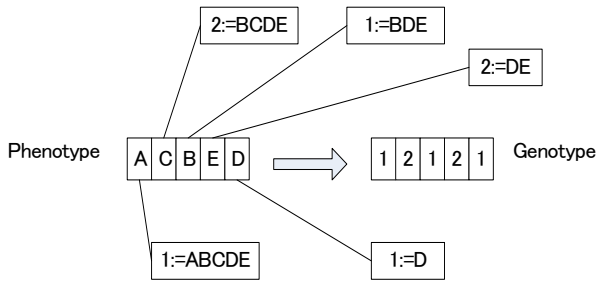

Figure 10. An example of order-based representation for a sequence.

2) Mutation

At most one gene is selected in a chromosome with mutation probability θ. If a gene is selected, it and its adjacent gene are exchanged.

3) Selection

Roulette strategy for selection and the elite strategy for preserving the best E individuals are utilized.

4) Fitness

Fitness value for sequence π at generation g (denoted by $fit_g(\pi)$) is computed by the following equation:

$$fit_g(\pi) = \max_{a \in G(g)} F(a) - F(\pi) + 1 \tag{24}$$

where $G(g)$ is a set of individuals at generation g. The number one is appended in this equation in order to allow for the possibility that the worst solution in $G(g)$ will be selected.

5) GA algorithm

GA algorithm consists of the following steps:

Step 1: Set $g := 1$. Generate an initial set of individuals whose size is N. Set it as $G(1)$.

Step 2: Evaluate $F(\pi)$ for all $\pi \in G(g)$, and then compute fitness value $fit_g(\pi)$ for each $\pi \in G(g)$.

Step 3: Copy the best E individuals in $G(g)$ to $G(g+1)$ (elite strategy).

Step 4: Select $(N-E)/2$ pairs from $G(g)$ for crossover operation.

Step 5: Execute 2-point crossover according to roulette strategy using fitness value $fit_g(\pi)$ in order to generate $N-E$ children.

Step 6: Apply mutation operation with probability θ. Note that mutation is not applied to individuals preserved by the elite strategy.

Step 7: Set $g := g+1$. If $g > g_{\max}$ then terminate the algorithm. Otherwise go back to *Step* 2.

(4) GC algorithm

In GC algorithm, when a partial sequence $\{i(1), i(2), \cdots, i(k-1)\}$ is given, product $i(k)$ which minimizes WLD_k defined by

$$WLD_k = \sum_{m=1}^{M} \left(\frac{k}{K} T_m - \sum_{j=1}^{k-1} a_{i(j),m} - a_{i(k),m} \right)^2 \quad (25)$$

is determined as the next product to be launched. In this research, this basic algorithm is used.

4.3 Data for numerical experiments

(1) Experimental data sets

Experimental conditions and data for the assembly line are set as follows:

We set $K = 50$ items, $M = 10$ stations, $v_c = 1$ cm/sec, $D = 5$ m, $w_m = 0$, and $\mu = 450$ seconds. Three types of assembly time $a_{i,m}$ are generated, which are Type I, Type II, and a type similar to the assembly time used in the car sequencing problem. Three sets of assembly times are generated using three different random number series for each combination of parameters $\sigma_P \in \{0.1, 0.3\}$, $\sigma_S \in \{0.1, 0.3\}$, and $L_m \in \{ \mu(1+\sigma), \mu(1+2\sigma) \}$.

(2) Set parameters for each algorithm

Parameters for each algorithm which are determined by preliminary experiments are as follows:

1) Parameters for SA

We set $\tau_0 = 2000$, $I_0 = 100$, and $\alpha = 0.95$. The termination conditions are set to $\tau_f = 0.1$ and $I_{\max} = 16000$.

2) Parameters for TS

$N = 20$, $TLL = 10$, and $I_{\max} = 800$ (i.e. the total iterations become 16000).

3) Parameters for GA

$N = 40$, crossover probability is 0.9, $E = 1$, $\theta = 0.09$, and $g_{\max} = 400$ (i.e. the total iterations become 16000).

4.4 Results of the numerical experiments

The results are summarized as follows:

(1) Type I data

For type I data, Figure 11 and Figure 12 show the improvement rate in terms of the total stoppage time. Figure 11 shows results for $L_m = \mu(1+\sigma)$ and Figure 12 for $L_m = \mu(1+2\sigma)$.

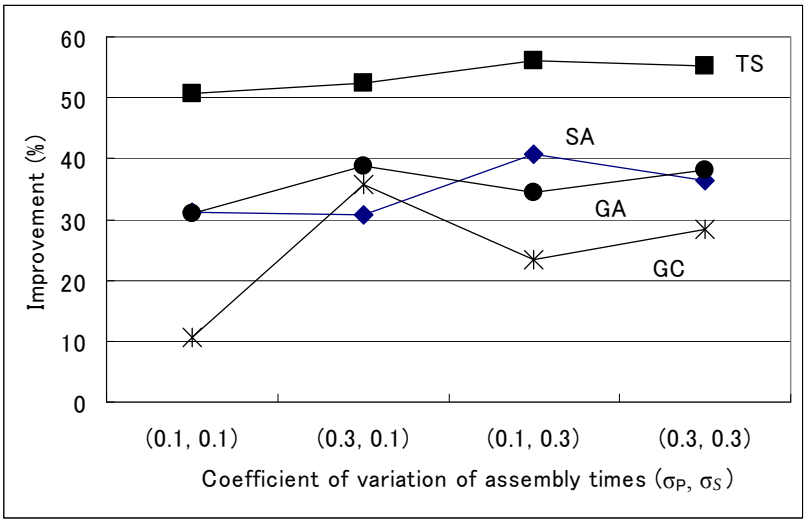

Figure 11. Improvement rate obtained by each algorithm when $L_m = \mu(1+\sigma)$.

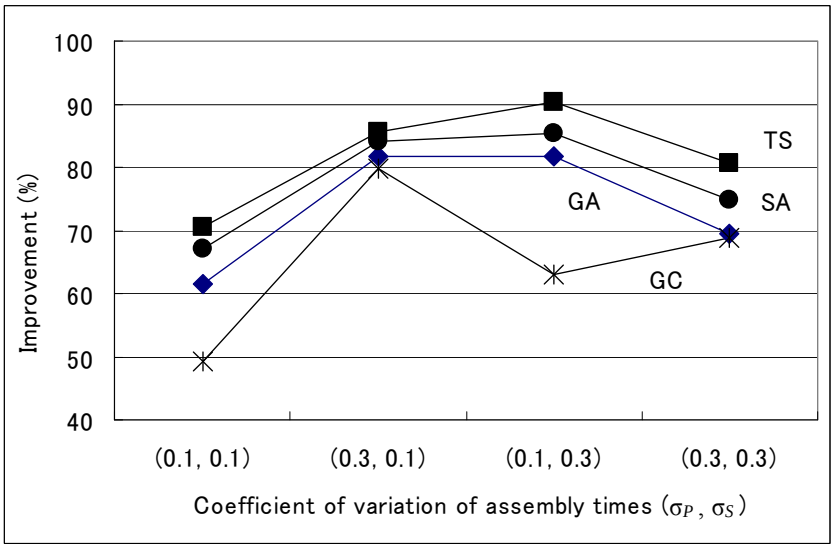

Figure 12. Improvement rate obtained by each algorithm when $L_m = \mu(1+2\sigma)$.

From these figures we conclude as follows:

1) When the allowance of station length is small (corresponding to $L_m = \mu(1+\sigma)$), the improvement rate is small. This can be seen from the definition of improvement rate which is represented as $100 \times (LS_{init} - LS_{best}) \div LS_{init}$ given in Section 3.4. When the

allowance of station length is large, value of LS_{best} becomes small (i.e. close to zero) and hence the improvement rate becomes large.

2) Line stoppage time minimization is fairly effective as an objective function to reduce the total line stoppage time.

3) TS algorithm is relatively more efficient than other algorithms.

4) GC algorithm is not a stable algorithm to obtain a good solution although it levels work load fluctuations well as shown in Figure 13.

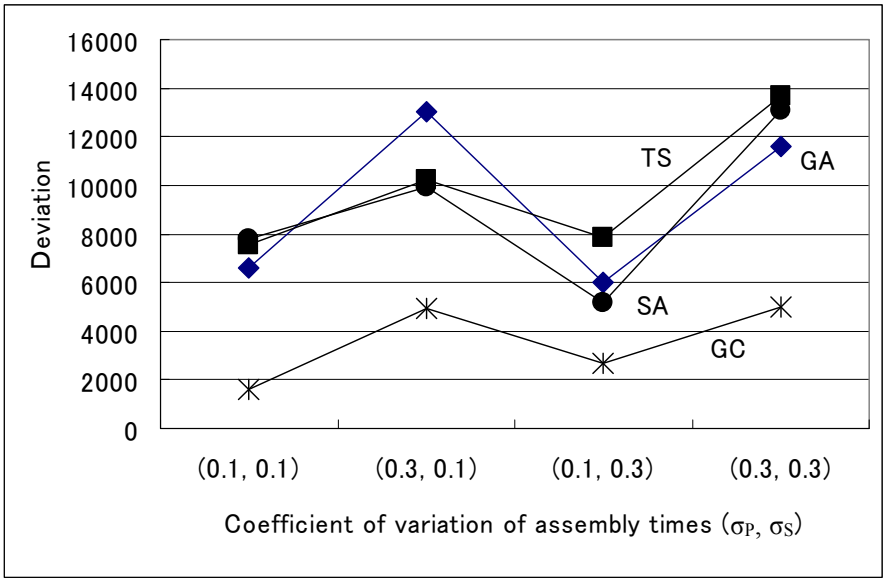

Figure 13. Work load fluctuations for the best solution obtained by applying each algorithm when $L_m = \mu(1+2\sigma)$.

(2) Type II data

Using Type II data, Figure 14 and 15 show improvement rates in terms of the total line stoppage time obtained by each algorithm when $L_m = \mu(1+\sigma)$ and $L_m = \mu(1+2\sigma)$, respectively. Since Type II data has a small bias in assembly times among products, improvement rates for the data become smaller than for Type I data especially when σ_S is set to 0.3 for both $L_m = \mu(1+\sigma)$ and $L_m = \mu(1+2\sigma)$.

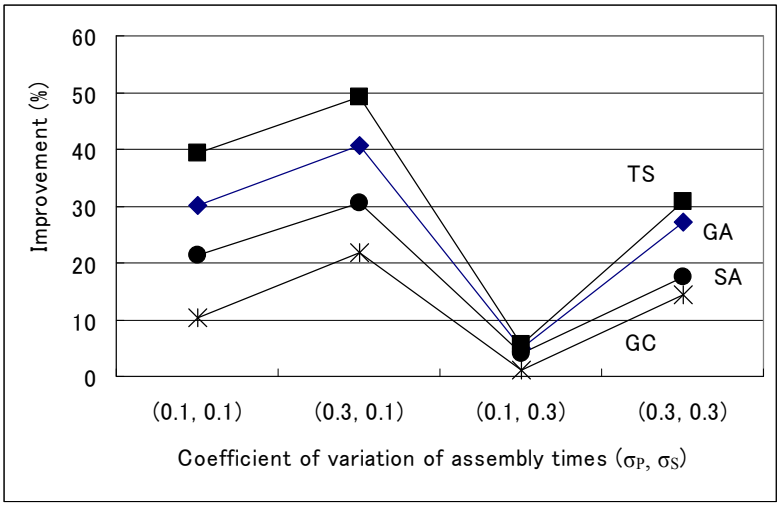

Figure 14. Improvement rate obtained by each algorithm when $L_m = \mu(1+\sigma)$.

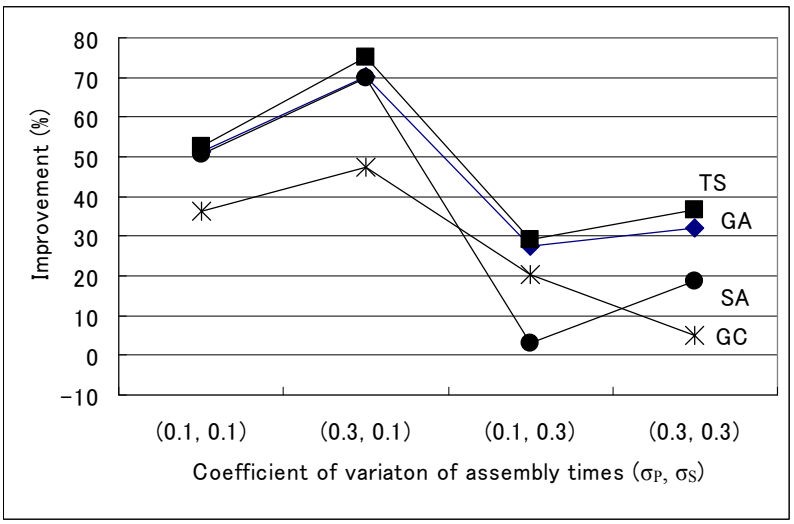

Figure 15. Improvement rate obtained by each algorithm when $L_m = \mu(1+2\sigma)$.

5. Conclusion

In this paper, we first discussed whether the minimization of utility work or of work load fluctuation is effective to reduce the total line stoppage time or not. This discussion was pursued by depicting scatter graphs between the two objective functions, first between the utility work and the line stoppage time and the second between the work load fluctuation and

the line stoppage time. It was found that the utility work correlates with the line stoppage time when the allowance of station length is large and the minimum value of the utility work is close to zero. However when the allowance of station length is small and when the line stoppage time is relatively large even for the optimal sequence, the utility work does not correlate with the line stoppage time and hence the utility work is not appropriate to use as an alternative objective function for the minimization of the line stoppage time. The use of the work load smoothing is not any more appropriate than minimizing the utility work for the minimization of line stoppage time except when the number of station is very small.

In the current research, we determined that it is most appropriate to use meta heuristics in order to minimize the total line stoppage time. In the meta heuristics, very large iterations are executed and at each iteration the line stoppage time is evaluated. Most of the CPU time of the algorithm is consumed in evaluating the line stoppage time. In this research, an efficient evaluation procedure has been proposed, which saves the CPU time compared to relying on simulation. For example, the CPU time for the proposed procedure was found to be less than one third of the CPU time for simulation for examples with 50 products and 10 stations.

The last part of the paper was dedicated to discussing the performance of the basic meta heuristics using three types of data. From numerical experiments, we concluded that (1) the line stoppage time varies widely depending on production sequence and hence it is important to minimize the line stoppage time, (2) Tabu search algorithm gives relatively better performance compared with Simulated annealing and Genetic algorithm. Goal chasing is inferior, although it achieves good work load leveling.

References

[1] Monden, Y.: *Toyota Production System (3rd Ed.)*. Engineering and Management Press, 1998.

[2] Xiaobo, Z. and Ohno, K. : Sequencing problem for a mixed-model assembly line in a JIT production system. Computers & Industrial Engineering, 1994, **27**, 71-74.

[3] Xiaobo, Z. and Ohno, K. : Algorithms for sequencing mixed models on an assembly line in a JIT production system. Computers & Industrial Engineering, 1997, **32**, 47-56.

[4] Xiaobo, Z., and Ohno, K.: Properties of a sequencing problem for a mixed model assembly line with conveyor stoppages, *European J. of Operational Research*, 2000, **124**, 560-570.

[5] Thomopoulos, N.T. : Line balancing-sequencing for mixed-model assembly. Management Science, 1967, **14**, B59-B75.

[6] Okamura, K. and Yamashina, H. : A heuristic algorithm for the assembly line model-mix sequencing problem to minimize the risk of stopping the conveyor. Int. J. of Production Research, 1979, **17**, 233-247.

[7] Scholl, A.: *Balancing and Sequencing of Assembly Lines,* Physica-Verlag, 1995.

[8] Ju Hyun, C., Kim, Y and Keun Kim, Y.: A genetic algorithm for multiple objective sequencing problems in mixed model assembly lines. Computers & Operations Research, 1998, 25-7&8, 675-690.

[9] Bolat, A. Savsar, M. and Al-Fawzan, M.A.: Algorithms for real-time scheduling of jobs on mixed model assembly lines. Advanced Engineering Informatics, 2007, 21-1, 85-99.

[10] Kubiak, W.: Minimizing Variation of production rates in just-in-time systems: A survey. *European Journal of Operational Research*, 1993, **66**, 259-271.

[11] Tamura, T., Long, H., and Ohno, K.: A sequencing problem to level part usage rates and work loads for a mixed-model assembly line with a bypass sub-line. *Int. Journal of Production Economics,* 1999, **60**&**61**, 557-564.

[12] Korkmazel, T., and Meral, S.: Bi-criteria sequencing methods for the mixed-model assembly line in just-in-time production systems. European Journal of Operational Research, 2001, 131, 188-207.

[13] Tamura, T., Dhakar, T., ICPR2009

[14] Boysen, N., Fiedner, M., and Scholl, A.: Sequencing mixed-model assembly lines: Survey, classification and model critique. *European J. of Operational Research*, 2008, **192**, 349-373.

[15] Burke, E.K., and Kendall, G. (Eds.): *Search Methodologies*, Springer, 2005.

[16] Celano, G, Costa, A., and Fichera, S.: A comparative analysis of sequencing heuristics for solving the Toyota Goal Chasing problem. *Robotics and Computer-Integrated Manufacturing*, 2004, **20**, 573- 581.

[17] Bard, J.F., Shtub, A., and Joshi, S.B.: Sequencing mixed-model assembly lines to level parts usage and minimize line length. *Int. J. of Production Research*, 1994, **32**, 2431-2454.

[18] Bautista, J., Companys, R., and Corominas, A.: Heuristics and exact algorithms for Monden problem. *European J. of Operational Research*, 1996, **88**, 101-113.

[19] Solnon, C., Cung, V.D., Nguyen, A., and Artigues, C.: The car sequencing problem: Overview of state-of-the-art methods and industrial case-study of the ROADEF'2005 challenge problem. *European J. of Operational Research*, 2008, **191**, 912-927.

[21] Yano, G.A., and Bolat, A.: Survey, development and application of algorithm for sequencing paced assembly lines, *J. of Manufacturing and Operations Management*, 1989, **2-3**, 172-198.

[22] Dewdney, A.K.: Exploring the field of genetic algorithm in a primordial computer sea full of flibs, *Scientific American*, 1985, **85-5,** 16-21.