J Intell Manuf (2011) 22:367–378
DOI 10.1007/s10845-009-0295-5

An efficient multiobjective genetic algorithm for mixed-model assembly line balancing problem considering demand ratio-based cycle time

Wenqiang Zhang · **Mitsuo Gen**

Received: 5 June 2008 / Accepted: 20 June 2009 / Published online: 24 July 2009

Abstract Mixed-model assembly lines (mALs) are becoming more and more important by producing different models of the same product on an assembly line. How to calculate the cycle time based on demand of different models also making problem more difficult. According to different work experiences and skill level, the processing time of a given task and the operating costs such as wages differ among workers. Appointing the proper worker to the proper station and assigning the suitable task to the suitable station in order to decrease the cycle time, increase the line efficiency, and reduce the total cost make the problem more complex. This paper proposes a new concept for calculating the cycle time based on demand ratio of each model and another one for calculating the human resource cost. A generalized Pareto-based scale-independent fitness function genetic algorithm (gp-siffGA) is described for solving mixed-model assembly lines balancing (mALB) problems to minimize the cycle time, the variation of workload and the total cost under the constraint of precedence relationships at the same time. The gp-siffGA uses Pareto dominance relationship to solve the problems without using relative preferences of multiple objectives. Comparisons with existing multiobjective genetic algorithms demonstrate that our approach efficiently solves mALB problems.

Keywords Mixed-model assembly line balancing · Worker allocation · Multiobjective genetic algorithm · Demand ratio-based cycle time · Pareto dominance relationship

W. Zhang (✉) · M. Gen
Graduated School of Information, Production and Systems, Waseda University, 2-7 Hibikino, Wakamatsu-ku, Kitakyushu City, Fukuoka 808-0135, Japan
e-mail: zhangwq@ruri.waseda.jp

Introduction

With the increasing and varied demands of customers at recent market, manufacturers are forced to produce different models of the same product on an assembly line. In mixed-model assembly lines (mALs), according to different work experiences, the skill level of a worker in a given task differs greatly among workers (Weng et al. 2006). Usually there are a few skilled workers who can deal with the tasks on a high skill level. So the efficiency of the line can be improved by reducing the task times. Furthermore, operating the line causes short-term operating costs such as wages, material, set-up, inventory and incompletion costs. So, the higher the skill level is, the more the total cost will be. How to allocate proper worker to proper station and assign the suitable task to suitable station in order to decrease the cycle time, increase the line efficiency and reduce the total cost are also a complex problem in mixed-model assembly line balancing (mALB) problems.

The problem of designing and balancing of assembly lines has been extensively examined in the literature and a number of review studies have been published, including Tasan and Tunali (2008), Levitin et al. (2006), Rekiek et al. (2002), Becker and Scholl (2006), and Scholl and Becker (2006). Both exact and heuristic procedures—and more recently, meta-heuristic procedures—have been developed to solve this problem.

Pinto et al. (1983) considered assembly line balancing problem where several alternatives are available in the choice of processing at each station. They described a method of simultaneously considering both the choice of manufacturing alternatives and the assignment of tasks to stations so as to minimize the total costs (labor and fixed) over the expected life of the production line. Graves and Holmes (1988) suggested an algorithm for assignment of activities and equipment to assembly line stations, satisfying the annual production rate. The objective of their work is to minimize the total cost that is composed of fixed equipment usage and tooling costs, variable equipment usage and set-up costs.

As set out in Becker and Scholl (2006), the assignment of task to machine problem is equivalent to the problem of assigning workers whose task performance speeds are different. Akagi et al. (1983), Wilson (1986), and Lutz et al. (1994) assumed that the performance speed of all persons who are manufacturing the same task is equal and that the time necessary to finish a task depends on the number of workers assigned to a station.

Hopp et al. (2004) set out a case in which workers can vary in speed and are benchmarked by defining the speed factor of each worker relative to a "standard worker"; moreover, they assumed that a worker's speed factor applies uniformly across all tasks. However, it is based on a line that is already designed and balanced, and the goal is to minimize the number of changes of workers from stations with surplus capacity to stations with a heavier workload, to help out temporarily.

Zhang et al. (2008) proposed a random key-based approach to deal with single ALB problem with worker allocation considering the worker experiences and skill level. They used an average distance method between different Pareto set obtained by random key-based GA and priority key-based GA to measure the performance.

Multiobjective genetic algorithms (moGAs) have been recognized to be well-suited for solving ALB problems because their abilities to exploit and explore multiple solutions in parallel and to find a widespread set of nondominated solutions in a single run (Deb 2001). Several moGAs based on Pareto dominance relationship are proposed to solve multiobjective optimization (moOP) problems directly, and present more promising results than single-objective optimization techniques theoretically and empirically (Chen and Ho 2005). By making use of Pareto dominance relationship, moGAs are capable of performing the fitness assignment of multiple objectives without using relative preferences of multiple objectives. Thus, all the objective functions can be optimized simultaneously. As a result, moGA seems to be an alternative approach to solving production planning problems on the assumption that no prior domain knowledge is available.

But the previous researchers do not considers that the cycle time need associated with the demand ratio of each model in mALB problem. The average cycle time of different models can not reflect that demand ratio of different models impact on cycle time. So, the demand-based cycle time is become more important.

The purpose of this paper is to resolve the above problems and the organization of paper is shown as follows: In the next section, we formulate the mathematical model for mALB with worker allocation (mALB-wa) problem. We then adopt generalized Pareto-based scale-independent fitness function genetic algorithm (gp-siffGA) to solve this problem. Next numerical experiments for various scales of ALB problems are used to demonstrate the effectiveness of the proposed approach. Finally, the conclusion and future research work are given in the last section.

Mathematical formulation

The mALB-wa problem concerns with the assignment of the proper worker to the proper station and assigning the suitable task to the suitable station in order to minimize the cycle time, minimize the variation of workload and minimize the total cost under the constraint of precedence relationships. In this study, we consider the mALB-wa problem subject to the following assumptions:

- A1. The assembly line is capable of producing more than one type of product simultaneously, not in batches.
- A2. The assembly of each model requires performing a set of tasks which are connected by precedence relations.
- A3. A subset of tasks is common to all models; the precedence graphs of all models can be combined into a non-cyclical joint precedence graph.
- A4. Tasks, which are common to several models, are performed by the same station but they may have different processing times (i.e., zero processing times indicate that the task is not required for the model).
- A5. The demands for all models during the planning period are fixed and known.
- A6. A worker is assigned to one station, and will only process the tasks which are assigned to that station.
- A7. Worker processing time for each task is known and deterministic.
- A8. Task processing time differs among workers because of their varying work experience and he/she can process all the tasks.
- A9. Workers have different worker cost according to the different skill level. The worker cost is the same for each task in each model.
- A10. A worker can process all the tasks for each model, and his/her work experience differs among tasks.

The notation used in this section can be summarized as follows:

Indices

j, k:	indices of task ($j, k = 1, 2 \ldots, n$)
i:	index of station ($i = 1, 2 \ldots, m$)
w:	index of worker ($w = 1, 2, \ldots, m$)
l:	index of model ($l = 1, 2, \ldots, p$)

Decision variables

$$x_{jil} = \begin{cases} 1, & \text{if task } j \text{ of model } l \text{ is assigned on station } i \\ 0, & \text{otherwise} \end{cases}$$

$$y_{iw} = \begin{cases} 1, & \text{if worker } w \text{ is working on station } i \\ 0, & \text{otherwise} \end{cases}$$

Parameters

n: number of tasks
m: number of stations, number of workers
p: number of models
d_{jw}: worker cost of worker w process task j
d_T: total worker cost
r_l: expected demand ratio of model l during the planning period
t_{jlw}: processing time of task j for model l processed by worker w
t_{jw}: processing time of task j for combined model processed by worker w

$$t_{jw} = \sum_{l=1}^{p} r_l t_{jlw} \tag{1}$$

$c_{T'}$: cycle time for combination of models
c_{Tl}: cycle time for model l

$$c_{Tl} = \max_{1 \le i \le m} \left\{ \sum_{j=1}^{n} \sum_{w=1}^{m} t_{jlw} x_{jil} y_{iw} \right\} \tag{2}$$

$\overline{c_T}$: average cycle time for models

$$\overline{c_T} = \frac{1}{p} \sum_{l=1}^{p} c_{Tl} \tag{3}$$

Suc(j): set of direct successors of task j
Pre(j): set of direct predecessors of task j
S_i: set of tasks assigned to station i
$t_l(S_i)$: processing time at station i for model l

$$t_l(S_i) = \sum_{j=1}^{n} \sum_{w=1}^{m} t_{jw} x_{jil} y_{iw}, \quad \forall i \tag{4}$$

u_{il}: utilization of the station S_i for model l

$$u_{il} = t_l(S_i)/c_{T'} \tag{5}$$

$\overline{u_l}$: average utilization of all stations for model l

$$\overline{u_l} = \frac{1}{m} \sum_{i=1}^{m} u_{il} \tag{6}$$

Mathematical model

$$\min \quad c_T = \sum_{l=1}^{p} \left\{ r_l \max_{1 \le i \le m} \left\{ \sum_{j=1}^{n} \sum_{w=1}^{m} t_{jlw} x_{jil} y_{iw} \right\} \right\} \tag{7}$$

$$\min \quad v = \sum_{l=1}^{p} \left\{ r_l \sqrt{\frac{1}{m} \sum_{i=1}^{m} (u_{il} - \overline{u_l})^2} \right\} \tag{8}$$

$$\min \quad d_T = \sum_{i=1}^{m} \sum_{j \in S_i} \sum_{w=1}^{m} d_{jw} y_{iw} \tag{9}$$

$$\text{s.t.} \sum_{i=1}^{m} i x_{jil} \ge \sum_{i=1}^{m} i x_{kil}, \quad \forall k \in \text{Pre}(j), \quad \forall j, l \tag{10}$$

$$\sum_{i=1}^{m} x_{jil} = 1, \quad \forall j, l \tag{11}$$

$$\sum_{w=1}^{m} y_{iw} = 1, \quad \forall i \tag{12}$$

$$\sum_{i=1}^{m} y_{iw} = 1, \quad \forall w \tag{13}$$

$$x_{jil} \in \{0, 1\}, \quad \forall j, i, l \tag{14}$$

$$y_{iw} \in \{0, 1\}, \quad \forall i, w \tag{15}$$

The first objective (7) of the model is to minimize the cycle time of the assembly line based on demand ratio of each model. The second objective (8) is to minimize the variation of workload. The third objective (9) is to minimize the total worker cost. Inequity (10) states that all predecessor of task j must be assign to a station, which is in front of or the same as the station that task j is assigned in. Equation 11 ensures that task j must be assigned to only one station. Equation 12 ensures that only one worker can be allocated to station i. Equation 13 ensures that worker w can be allocated to only one station. Constrains (14) and (15) represent the usual integrity restrictions.

There are three definitions for cycle time: Equation 2 is cycle time of model l, Eq. 3 is average cycle time for models and Eq. 7 is demand ratio-based cycle time. What is different between these three cycle times. The Fig. 1 shows the comparison between three cycle time.

If the assembly line just produce one model such as model 1 (model 2), the cycle time of model 1: 50 time units (model 2: 60 time units) can be used as a reference. Average cycle time: 55 time units can be used when the assembly line processes two models. However, using average cycle time as a

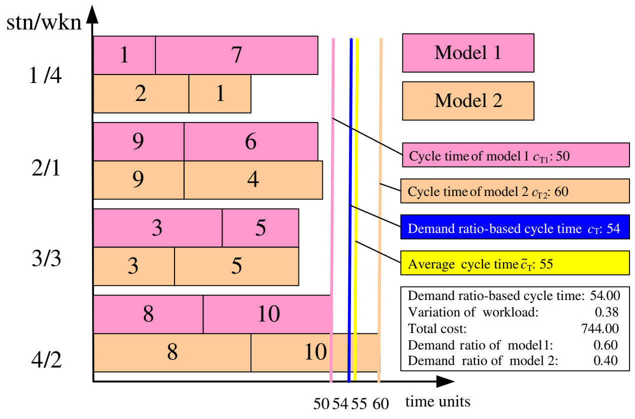


Fig. 1 The comparison of demand ratio-based cycle time and average cycle time

reference when the demand ratio of models is different can increase the idle time and decrease the line efficiency. The demand ratio-based cycle time: 54 time units is calculated based on the demand ratio of models (demand ratio of model 1: 0.60, demand ratio of model 2: 0.40). If the demand ratio of model 1 is bigger than model 2, the cycle time will close to the cycle time of model 1. On the contrary, the cycle time will toward cycle time of model 2. If the demand ratio of models is same, the cycle time is the average time of two models. So, the demand ratio-based cycle time can reflect the relationship between the demand ratio of models and cycle time.

The gp-siffGA approach

Many moGAs differ mainly in the fitness assignment strategy which is known as an important issue in solving moOPs (Gen and Cheng 2000). The gp-siffGA uses a generalized pareto-based scale-independent fitness function (gp-siff) considering the quantitative fitness values in Pareto space for both dominated and nondominated individuals (Ho et al. 2004). The gp-siff makes the best use of Pareto dominance relationship to evaluate individuals using a single measure of performance.

In this section, we adopted gp-siffGA approach using random key-based encoding method. In this approach, a combination of the arithmetical crossover operator (for task priority vector), weight mapped crossover (for worker allocation vector), swap mutation operator, immigration operator and a binary tournament selection operator were used as genetic operations, and the fitness assignment strategy was used to determine fitness value.

The overall structure of adopted gp-siffGA is given as follows:

```
Procedure : gp-siffGA for mALB-wa problem
Input: Data set of mALB-wa problem,
   GA parameters ($popSize$, $maxGen$, $p_C$, $p_M$, $p_I$)
Output: Pareto optimal solutions $E$
begin
   $t \leftarrow 0$;
   initialize $P(t)$ by random key-based encoding;
   calculate objectives $c_T(P)$, $v(P)$, $d_T(P)$ by
   random key-based decoding routine;
   create Pareto $E(P)$;
   calculate fitness $eval(P)$ by gp-siff approach;
   while (not terminating condition) do
      create $C(t)$ from $P(t)$ by arithmetical crossover
         and weight mapping crossover routine;
      create $C(t)$ from $P(t)$ by swap mutation routine;
      create C($t$) from P($t$) by immigration routine;
      calculate objectives $c_T(C)$, $v(C)$, $d_T(C)$ by
         random key-based decoding routine;
      calculate $eval(P, C)$ by gp-siff routine;
      update Pareto $E(P, C)$;
      select $P(t+1)$ from $P(t)$ and $C(t)$ by binary
         tournament selection routine;
      $t \leftarrow t + 1$;
   end
   output Pareto optimal solutions $E(P, C)$
end
```

Genetic representation

In ALB problem, encoding a solution into a chromosome is a key issue for GAs. For this issue, we used the random key-based genetic representation, which was proposed by Bean (1994).

In this study, in order to increase the quality of solution and convergence speed, we used a real number string coding, gp-siffGA to solve the mALB-wa problem. The detailed encoding and decoding process of a chromosome consists of five phases:

Phase 1: Combining all the models

Step 1.1: Combining all the models into a combined precedence graph
Step 1.2: Calculating the processing times for the combined model

Phase 2: Creating a task sequence

Step 2.1: Generating a random key-based chromosome by encoding method
Step 2.2: Creating a task sequence by decoding method

Phase 3: Assigning tasks to each station

Step 3.1: Calculating the lower bound and the upper bound cycle time
Step 3.2: Finding an optimal cycle time by bisection searching

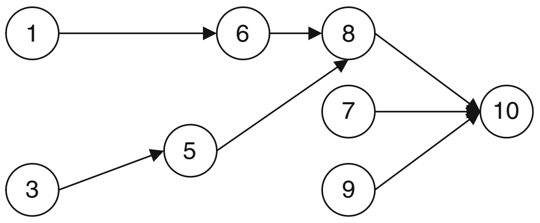


Fig. 2 A precedence graph of model 1

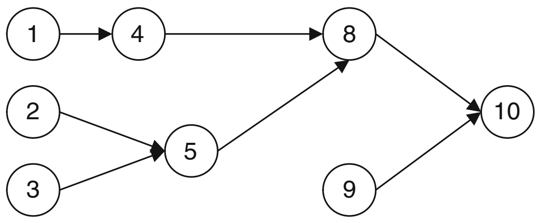


Fig. 3 A precedence graph of model 2

Table 1 Data set of model 1

Task j	Predecessor Pre(j)	Task time unit			
		w_1	w_2	w_3	w_4
1	Ø	17	23	17	13
2	–	0	0	0	0
3	Ø	22	15	27	25
4	–	0	0	0	0
5	{3}	21	25	16	32
6	{1}	28	18	20	21
7	Ø	42	28	23	34
8	{6}	17	23	40	25
9	Ø	19	18	17	34
10	{7,8,9}	16	27	35	26

Step 3.3: Dividing the task sequence

Phase 4: Assigning worker to each station

Step 4.1: Encoding for worker assignment by randomly assigning the worker to each station
Step 4.2: Assigning the worker into each station by breakpoint decoding method

Phase 5: Designing a Schedule

Step 5.1: Creating a schedule for the assembly line
Step 5.2: Drawing a Gantt chart for this schedule

The chromosome includes four parts: task priority vector (v_1), worker allocation vector (v_2), task sequence vector (v_3), breakpoint vector (v_4). The task priority vector (v_1) used a real number to identify the priority level for one task generated by random key-base encoding method. The higher priority value of task is, the earlier the task start. the locus of each gene of worker allocation vector (v_2) represent station id and gene are presented as integer number, which is worker id assigned to this station. After applying random key decoding method, the task sequence vector (v_3) is generated. The breakpoint vector (v_4) is used to separate the task sequence into machine.

Phase 1: Combining all the models

Step 1.1: Combining all the models into a combined precedence graph.

Table 2 Data set of model 2

Task j	Predecessor Pre(j)	Task time unit			
		w_1	w_2	w_3	w_4
1	Ø	18	22	19	13
2	Ø	21	22	16	20
3	Ø	12	25	17	15
4	{1}	29	21	19	16
5	{2,3}	31	25	26	22
6	–	0	0	0	0
7	–	0	0	0	0
8	{4}	27	33	40	25
9	Ø	19	13	17	34
10	{8,9}	26	27	35	16

Table 3 Data set of cost

Task j	Cost unit			
	w_1	w_2	w_3	w_4
1	83	78	81	87
2	79	78	84	80
3	88	75	73	85
4	71	79	81	84
5	69	75	74	78
6	72	82	80	79
7	58	72	77	66
8	73	67	60	75
9	81	87	83	66
10	74	73	65	74

As an illustrative example, a mALB having ten tasks, four stations, four workers and two models are used. The precedence graph and the processing time of worker for model 1 are shown in Fig. 2 and Table 1.
The precedence graph and the processing time of worker for model 2 are shown in Fig. 3 and Table 2.
The data set of cost is shown in Table 3.
We combine the two models into a combined precedence graph as shown in Fig. 4.

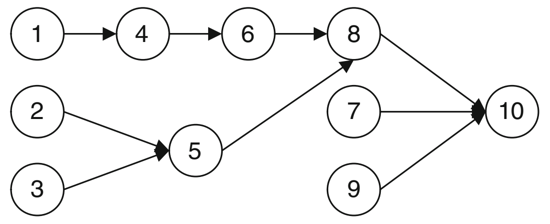


Fig. 4 A precedence graph of combined model

Table 4 Data set of combined model

Task j	Predecessor Pre(j)	Task time unit			
		w_1	w_2	w_3	w_4
1	Ø	17.4	22.6	17.8	13.0
2	Ø	8.4	8.8	6.4	8.0
3	Ø	18.0	19.0	23.0	21.0
4	{1}	11.6	8.4	7.6	6.4
5	{2,3}	25.0	25.0	20.0	28.0
6	{4}	16.8	10.8	12.0	12.6
7	Ø	25.2	16.8	13.8	20.4
8	{6}	21.0	27.0	40.0	25.0
9	Ø	19.0	16.0	17.0	34.0
10	{7,8,9}	20.0	27.0	35.0	22.0

Locus: task id	1	2	3	4	5	6	7	8	9	10
Task priority (v_1)	3.31	7.14	6.59	2.24	8.67	5.38	4.13	9.87	10.72	1.91

Locus: station id	1	2	3	4
Worker allocation (v_2)	3	4	1	2

Fig. 5 The chromosome of the example

Step 1.2: Calculating the processing times for the combined model.
The expected demand ratio of two models during the planning period is set to 0.6 and 0.4, respectively. The calculated processing time by Eq. 1 for combined model is shown in Table 4.

Phase 2: Creating a task sequence

Step 2.1: Generating a random key-based chromosome by encoding method.
The chromosome is composed of two parts (see Fig. 5); the first part is the task priority vector and the second part is worker allocation vector. A random method are used to generate the two vectors.
Step 2.2: Creating a task sequence by decoding method.
At the beginning, we try to find a task for the first station. Task 1, 2, 3, 7, 9 are eligible for the position. From the task priority vector, The task 9 has the highest priority and is put into the task sequence. Then we delete task 9 from precedence graph. By the same manner, we can encode the chromosome easily.
The generated task sequence vector is shown in Fig. 6.

Locus: gene id	1	2	3	4	5	6	7	8	9	10
Task sequence (v_3)	9	2	3	5	7	1	4	6	8	10

Fig. 6 The task sequence of the example

Locus: gene id	1	2	3	4
Breakpoint vector (v_4)	1	4	6	9

Fig. 7 The breakpoint vector

Phase 3: Assigning tasks to each station

Step 3.1: Calculatling the lower bound and the upper bound cycle time.
Calculate the lower bound (c_{LB}) and the upper bound (c_{UB}) of the cycle time for the solution represented by the task sequence vector and worker allocation vector.

$$C_{\mathrm{LB}} = \frac{1}{m}\sum_{j=1}^{n} \min_{1\le w\le m} \{t_{jw}\},$$

$$C_{\mathrm{UB}} = \frac{1}{m}\sum_{j=1}^{n} \max_{1\le w\le m} \{t_{jw}\} + \max_{\substack{1\le j\le n\\ 1\le w\le m}} \{t_{jw}\} \tag{16}$$

Step 3.2: Finding an optimal cycle time by bisection searching.
Step 3.3: Dividing the task sequence.
Partition the task sequence into m parts with the optimal cycle time based on the worker allocation vector.
After applying the decoding procedure to the example, the breakpoint vector in Fig. 7 is obtained.

Phase 4: Assigning worker to each station

Step 4.1: Encoding for worker assignment by randomly assigning the worker to each station.
In this phase, we code the worker assignment vector (v_2) by randomly as shown in Fig. 4, which indicates the worker assigned for each station. The locus of worker allocation vector represents the station ID.
Step 4.2: Assigning the worker into each station by breakpoint decoding method.
In this phase, this worker assignment vector (v_2) will be decoded into a solution with the task sequence vector (v_3), simultaneously by the breakpoint decoding procedure.
Based on the breakpoint vector, the task sequence vector is separated into four parts, all of which respectively correlated with one station. According to worker allocation

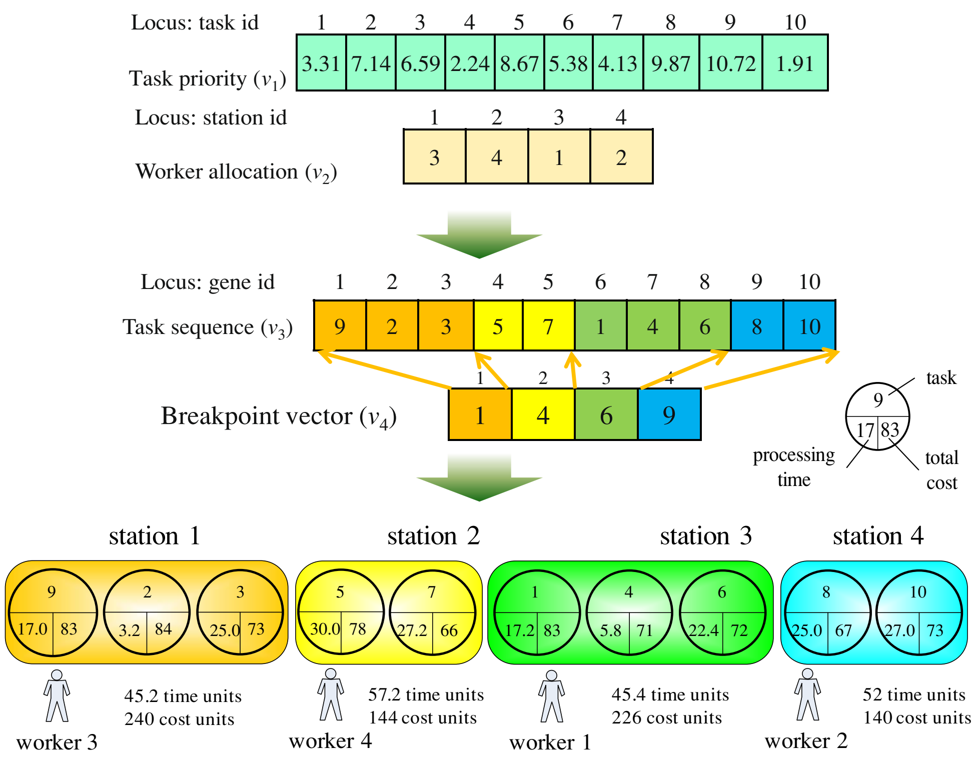
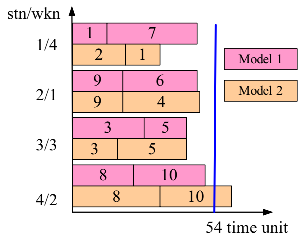


Fig. 8 The phenotype of the example chromosome



Cycle Time:	54.00
Variation of Workload:	0.38
Total Cost:	744.00
Demand ratio of Model 1:	0.60
Demand ratio of Model 2:	0.40

Fig. 9 The balancing chart of the best solution

vector, we assign the works to each station. Fig. 8 illustrates the phenotype of the example chromosome.

Phase 5: Designing a Schedule

Step 5.1: Creating a schedule for the assembly line.
Using the chromosomes, the schedule can be constructed as follows:
Schedule of model 1 $S = \{(j, S_i, w, t_{jlw})\}$
S={(2, S_1, 4, 0-0), (1, S_1, 4, 0-13), (7, S_1, 4, 13-47), (9, S_2, 1, 47-66),
(4, S_2, 1, 66-66), (6, S_2, 1, 66-94), (3, S_3, 3, 94-121), (5, S_3, 3, 121-137),
(8, S_4, 2, 137-160), (10, S_4, 2, 160-187)}
Schedule of model 2 S={(j, S_i, w, t_{jlw})}
S={(2, S_1, 4, 0-20), (1, S_1, 4, 20-33), (7, S_1, 4, 33-33), (9, S_2, 1, 33-52),
(4, S_2, 1, 52-81), (6, S_2, 1, 81-81), (3, S_3, 3, 81-98), (5, S_3, 3, 98-124),
(8, S_4, 2, 124-157), (10, S_4, 2, 157-184)}
Step 5.2: Drawing a Gantt chart for this schedule.

We can draw a Gantt chart according to the above schedule, but, it isn't optimized schedule. So we used a preference method based on weighted factor to select one solution from Pareto set as an example to draw a Gantt chart.
In the real-world, the assembly line is not just for producing one unit of the product. It should produce several units. Figures 9 and 10 show the Gantt charts for one unit and three units of product, respectively.

Genetic operators

Genetic operators mimic the process of heredity of genes to create new offspring at each generation. Considering the characteristic of random key-based genetic representation, we used arithmetical crossover operator (for worker allocation vector), weight mapped crossover (for task priority vector), swap mutation operator, immigration operator and a binary tournament selection operator.

Crossover operator

In this study, arithmetical crossover was used for task priority vector and weight mapped crossover (WMX) was used for worker allocation vector.

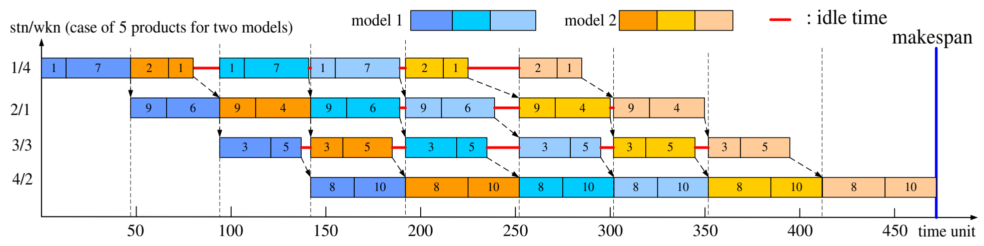


Fig. 10 The Gantt chart of the best compromise solution (five products for two models)

Arithmetical Crossover (for task priority vector)

The basic concept of this kind of operator is borrowed from the convex set theory (Gen and Cheng 1997). Generally, the weighted average of two vectors $v_j(i)$ and $v_k(i)$ of jth chromosome and kth chromosome, is calculated as follows:

$$\lambda_1 v_j(i) + \lambda_2 v_k(i) \tag{17}$$

if the multipliers are restricted as

$$\lambda_1 + \lambda_2 = 1, \lambda_1 > 0 \text{ and } \lambda_2 > 0 \tag{18}$$

Arithmetic operators are defined as the combination of two chromosomes v_j and v_k as follows:

$$v'_j(i) = \lambda_1 v_j(i) + \lambda_2 v_k(i), \quad \forall i \tag{19}$$

$$v'_k(i) = \lambda_1 v_k(i) + \lambda_2 v_j(i), \quad \forall i \tag{20}$$

Weight Mapped Crossover (WMX) (for worker allocation vector)

Weight Mapped Crossover (WMX) can be viewed as an extension of one-point crossover for permutation representation. As one-point crossover, (1) two chromosomes (parents) will be to choose a random cut-point, (2) generate the offspring by using segment of own parent to the left of the one-cut point, (3) and then, remapping the right segment that base on the weight of other parent of right segment (Gen et al. 2008).

Mutation operator

Several mutation operators have been proposed for the GA representation, such as swap mutation, inversion mutation, and insertion mutation, and so on. In this study, we use swap mutation for generating various offspring for task priority vector and worker allocation vector. Swap mutation selects two positions at random and then swaps the gene on these positions. We define notation p_M as mutation probability.

Immigration operator

Moed et al. (1991) proposed an immigration operator which, for certain types of functions, allows increased exploration while maintaining nearly the same level of exploitation for the given population size. The algorithm is modified to the following four steps: (1) include immigration routine, in each generation, (2) generate and (3) evaluate *popSize*·p_I random members, and (4) replace the *popSize*·p_I worst members of the population with the *popSize*·p_I random members (p_I, called the immigration probability).

Selection operator

Genetic algorithms use a selection mechanism to select individuals from the population. In this paper, the selection operator of gp-siffGA uses a binary tournament selection which works as follows. Choose two individuals randomly from the population and copy the better individual into the intermediate population.

Evaluation function

For the evaluation of individuals, the Pareto dominance relationship and generalized Pareto-based scale-independent fitness function method were used.

Pareto dominance relationship

Assume all the objective functions f_i $(i = 1, 2, 3)$ are to be minimized. Mathematically, moOPs can be represented as the following vector mathematical programming problems:

$$\min f(S)=\{f_1(S)=c_T(S),\ f_2(S)=v(S),\ f_3(S)=d_T(S)\} \tag{21}$$

where S denotes a solution and $f_i(S)$ is generally a non-linear objective function.

When the following inequalities hold between two solutions S_1 and S_2, S_2 is a nondominated solution and is said to dominate S_1 $(S_2 \succ S_1)$:

$$\forall i : f_i(S_1) \geq f_i(S_2) \text{ and } \exists\ j : f_j(S_1) > f_j(S_2) \tag{22}$$

Table 5 The information for three benchmark problems

Problem no.	No. of tasks	No. of stations	WEST ratio	No. of models	Demand ratio of each model
1	35	4	8.75	3	0.4, 0.4, 0.2
2		5	7.00	3	0.4, 0.4, 0.2
3		7	5.00	3	0.4, 0.4, 0.2
4		12	2.92	3	0.4, 0.4, 0.2
5	70	7	10.00	5	0.2,0.2, 0.1, 0.1, 0.4
6		10	7.00	5	0.2, 0.2, 0.1, 0.1,0.4
7		14	5.00	5	0.2, 0.2, 0.1, 0.1, 0.4
8		19	3.68	5	0.2, 0.2, 0.1, 0.1, 0.4
9	148	10	14.80	7	0.05, 0.05, 0.3, 0.2, 0.1, 0.1, 0.2
10		14	10.57	7	0.05, 0.05, 0.3, 0.2, 0.1, 0.1, 0.2
11		21	7.05	7	0.05, 0.05, 0.3, 0.2, 0.1, 0.1, 0.2
12		29	5.10	7	0.05, 0.05, 0.3, 0.2, 0.1, 0.1, 0.2

When the following inequality hold between two solutions S_1 and S_2, S_2 is said to weakly dominate S_1 ($S_2 \succeq S_1$):

$$\forall i : f_i(S_1) \geq f_i(S_2) \tag{23}$$

A feasible solution S^* is said to be a Pareto-optimal solution if and only if there does not exist a feasible solution S where S dominates S^*. The corresponding vector of Pareto-optimal solutions is called Pareto-optimal front.

Generalized Pareto-based scale-independent fitness function

The gp-siff can be as follows: Let the fitness value of an individual S_i be a tournament-like score obtained from all participant individuals by the following function:

$$eval(S_i) = p(S_i) - q(S_i) + c, i = 1, 2, \ldots, popSize \tag{24}$$

where

$$p(S_i) = \left\{ |S_j| \,\middle|\, \underset{S_i \neq S_j}{f_k(S_j) \geq f_k(S_i)}, \quad \forall\, k, j, \quad (k = 1, 2, \ldots, q, \;\; j = 1, 2, \ldots, popSize) \right\} \tag{25}$$

$$q(S_i) = \left\{ |S_j| \,\middle|\, \underset{S_i \neq S_j}{f_k(S_i) \geq f_k(S_j)}, \quad \forall k, j, \quad (k = 1, 2, \ldots, q, \;\; j = 1, 2, \ldots, popSize) \right\} \tag{26}$$

where $p(S_i)$ is the number of individuals which can be dominated by the individual S_i, and $q(S_i)$ is the number of individuals which can dominate the individual S_i in the objective space. In this paper, c is the number of all participant individuals. Generally, a constant c can be optionally added in the fitness function to make fitness values nonnegative when the binary tournament selection is used. If a proportional-based selection like roulette wheel selection method is used, the c is necessary to make fitness value nonnegative.

The gp-siff uses a pure fitness assignment strategy, which differs from the traditional Pareto-ranking methods. The gp-siff can assign discriminative fitness values not only to nondominated individuals but also to dominated ones. In traditional Pareto-ranking methods, the individuals with same rank value have same fitness value. For example, for each individual with rank value 1, the fitness value is same: 1. But, in gp-siff, the fitness value is different according to the number of dominated by this individual because the c is constant and the $q(S_i)$ is 0. So, the more the fitness value is, the better the individual will be.

Experiments and discussion

In order to evaluate the performance of the proposed method, a large set of problems are used. We collect three representative precedence graphs from Scholl (1993), which are widely used in the ALB literature. These precedence graphs contain with 35, 70, and 148 tasks. From each precedence graph, four different mALB-wa problems are generated by using different WEST ratios: 3, 5, 7, 10 and 15. WEST ratio, as defined by Dar-EI (1973), measures the average number of activities per station. The details of this benchmark are shown in Table 5. For each problem, the number of workstation is equal to the number of workers, and each task can be processed on any worker. The worker cost data are generated according to work skill level (Weng et al. 2007).

All the simulation experiments were performed on Pentium 4 processor (2.8 GHz clock), the program was written in C# language. The adopted parameters of the gp-siffGA are listed as following:

Population size: $popSize = 500$;
Maximum generations: $maxGen = 1000$;
Crossover probability: $p_C = 0.60$;
Mutation probability: $p_M = 0.20$;
Immigration probability: $p_I = 0.06$;

We compared the results of proposed GA with random key-based Genetic Algorithm (rkGA) approach to examine its efficiency of the former. The rkGA used the same random key-based genetic representation and different genetic operators: arithmetical crossover operator (for worker allocation vector), weight mapped crossover (for task priority vector), swap mutation operator, Roulette Wheel Selection (RWS) operator. For evaluation function, the Adaptive Weight Approach (AWA) (Gen and Cheng 2000) was used for rkGA.

In order to evaluate the efficiency of the different approaches, the coverage metric method proposed by Zitzler and Thiele (1999) was used. The coverage metric $C(A, B)$ of two solution sets A and B is the percent of the of individuals in B who are weakly dominated by individuals in A. The larger $C(A, B)$ is, the better performed by A than B.

$$C(A, B) = \frac{\left|\left[b \in B \middle| \left\{\exists a \in A : a \succeq b\right\}\right]\right|}{|B|} \tag{27}$$

The value $C(A, B) = 1$ means that all individuals in B are weakly dominated by A. On the contrary, $C(A, B) = 0$ denotes that none of individuals in B is weakly dominated by A. Because the C measure considers the weakly dominance relationship between two sets A and B, $C(A, B)$ is not necessarily equal to $1 - C(B, A)$.

The comparison results of two solution sets using the coverage metric are depicted using Box-and-whisker plots. Box-and-whisker plots are helpful in interpreting the distribution of data. We know that the median of a set of data separates the data into two equal parts. Data can be further separated into quartiles.

- The first quartile is the median of the lower part of the data.
- The second quartile is another name for the median of the entire set of data.
- The third quartile is the median of the upper part of the data.
- Quartiles separate the original set of data into four equal parts.

Each of these parts contains one-fourth of the data.

Figure 11 depicts the coverage metric of C(gp-siffGA, rkGA) and C(rkGA, gp-siffGA) from 30 runs. For each run, the solution set of two algorithms are compared using the coverage metric.

The first plot on the left of Fig. 11a describes the coverage metric of C(gp-siffGA, rkGA) for solving the small problem 35 tasks with 4 stations. From this plot, the lowest value of C(gp-siffGA, rkGA) is 0.26 and the highest value is 0.82, the value of median is 0.57. The first quarter of coverage metric data of C(gp-siffGA, rkGA) from 30 runs is distributed from 0.26 to 0.42, the half of data is located from 0.42 to 0.69 and the last quarter of data is located from 0.69 to 0.82.

On the contrary, the lowest value of C(rkGA, gp-siffGA) is 0.07 and the highest value is 0.71, the value of median is 0.40 as shown in the first plot on the left of Fig. 11b. The first quarter of coverage metric data of C(rkGA, gp-siffGA) from 30 runs is distributed from 0.07 to 0.30, the half of data is located from 0.30 to 0.54 and the last quarter of data is located from 0.54 to 0.71. So, the performance of gp-siffGA is better than rkGA for solving the small size problem with 35 tasks, 4 stations.

As the complexity of problems increases, Fig. 11 shows that more than 60–90% of the nondominated solutions obtained by rkGA are dominated by the nondominated solutions obtained by gp-siffGA in solving the problems 35 tasks with 5, 7, and 12 stations, 70 tasks with 7, 10, 14, and 19 stations, 148 tasks with 10, 14, 21, and 29 stations.

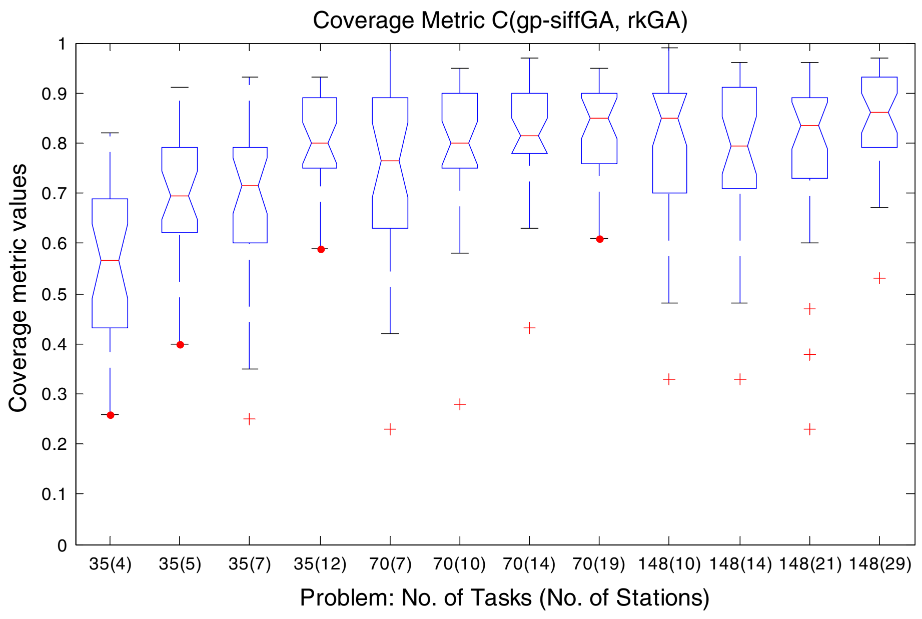
Conclusions

In this study, we concern with the assignment of the suitable task to the suitable station and the allocation the proper worker to the proper station in order to minimize the cycle time, minimize the variation of workload and minimize the total cost under the constraint of precedence relationships. This paper proposes a new concept for calculating the cycle time based on demand ratio of each model and another one for calculating the human resource cost. To solve above problem, firstly a random key-based representation method is used for task priority vector adapting the GA, while randomly generated method is adopted for worker allocation vector. Following, advanced genetic operators were adapted to the specific chromosome structure and the characteristics of the mALB-wa problem were used. Moreover, a generalized Pareto-based scale-independent fitness function genetic algorithm (gp-siffGA) was used to solve the mALB-wa problem without using relative preferences of multiple objectives. Finally, the performance of proposed method was validated through numerical experiments. Comparisons with existing multiobjective genetic algorithms demonstrate that our approach efficiently solves mALB problems.

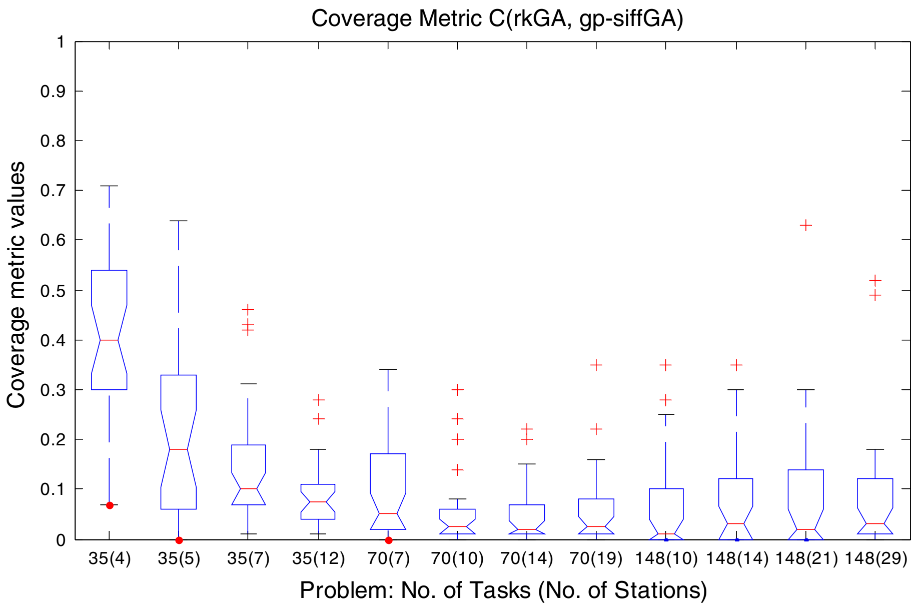
In the future, we are planning to consider the cases where more than one worker can be allocated to one station and the processing times are stochastic. Moreover, we are also planning to combine this algorithm with other evolutionary techniques.

Fig. 11 Box plots based on the coverage metric. **a** Coverage metric C(gp-siffGA, rkGA); **b** coverage metric C(rkGA, gp-siffGA)

(a) Coverage Metric C(gp-siffGA, rkGA)



(b) Coverage Metric C(rkGA, gp-siffGA)



Acknowledgments This work is partly supported by the Ministry of Education, Science and Culture, the Japanese Government: Grant-in-Aid for Scientific Research (No. 19700071, No. 17510138).

References

Akagi, F., Osaki, H., & Kikuchi, S. (1983). A method for assembly line balancing with more than one worker in each station. *International Journal of Production Research, 21*(5), 755–770.

Bean, J. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal On Computing, 6*(2), 154–160.

Becker, C., & Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research, 168*(3), 694–715.

Chen, J. H., & Ho, S. Y. (2005). A novel approach to production planning of flexible manufacturing systems using an efficient multi-objective genetic algorithm. *International Journal of Machine Tools and Manufacture, 45*(7–8), 949–957.

Dar-EI, E. (1973). MALB-a heuristic technique for balancing large single-model assembly lines. *AIIE Transactions, 34*, 343–356.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. Wiley Interscience Series in Systems and Optimization. New York: Wiley.

Gen, M., & Cheng, R. (1997). *Genetic algorithms and engineering design*. New York: Wiley.

Gen, M., & Cheng, R. (2000). *Genetic algorithms and engineering optimization*. New York: Wiley.

Gen, M., Cheng, R., & Lin, L. (2008). *Network models and optimization: Multiobjective genetic algorithm approach*. London: Springer.

Graves, S. C., & Holmes, C. R. (1988). Equipment selection and task assignment for multi-product assembly system design. *The International Journal of Flexible Manufacturing Systems, 1*, 31–50.

Ho, S. Y., Shu, L. S., & Chen, J. H. (2004). Intelligent evolutionary algorithms for large parameter optimization problems. *IEEE Transactions on Evolutionary Computation, 8*(6), 522–541.

Hopp, W., Tekin, E., & Van, M. P. (2004). Benefits of skill chaining in serial production lines with cross-trained workers. *Management Science, 50*(1), 83–98.

Levitin, G., Rubinovitz, J., & Shnits, B. (2006). A genetic algorithm for robotic assembly line balancing. *European Journal of Operational Research, 168*(3), 811–825.

Lutz, C., Davis, K. R., & Turner, C. F. (1994). Development of operator assignment schedules: A DSS approach. *Omega, 22*(1), 57–67.

Moed, M. C., Stewart, C. V., & Kelly, R. B. (1991). Reducing the search time of a steady state genetic algorithm using the immigration operator. *Proceedings of IEEE international conference tools for artificial intelligence* (pp. 500–501).

Pinto, P. A., Dannenbring, D. G., & Khumawala, B. M. (1983). Assembly line balancing with processing alternatives: An application. *Management Science, 29*(7), 817–830.

Rekiek, B., Dolgui, A., Delchambre, A., & Bratcu, A. (2002). State of art of optimization methods for assembly line design. *Annual Reviews in Control, 26*(2), 163–174.

Scholl, A. (1993). Data of assembly line balancing problems. *Schriften zur Quantitativen Betriebswirtschaftslehre* 16/93, Th Darmstadt.

Scholl, A., & Becker, C. (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research, 168*(3), 666–693.

Tasan, S. O., & Tunali, S. (2008). A review of the current applications of genetic algorithms in assembly line balancing. *Journal of Intelligent Manufacturing, 19*(1), 49–69.

Weng, J. H., Banno, M., & Onari, H. (2006). A study on line operation planning for sewing works. *Journal of Japan Industrial Management Association, 56*(6), 471–477.

Weng, J., Banno, M., Okubo, H., & Onari, H. (2007). An integrated algorithm for operation assignment and worker allocation in assembly lines. *Journal of Japan Industrial Management Association, 58*(5), 383–394.

Wilson, J. (1986). Formulation of a problem involving assembly lines with multiple manning of work stations. *International Journal of Production Research, 24*(1), 59–63.

Zhang, W., Lin, L., & Gen, M. (2008). A multiobjective genetic algorithm based approach to assembly line balancing problem with worker allocation. *Journal of Society of Plant Engineers Japan (SOPEJ), 19*(4), 61–72.

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strengthen Pareto approach. *IEEE Transaction on Evolutionary Computation, 4*(3), 257–271.