Using Multi-objective Genetic Algorithm with Fuzzy Logic Controller for Assembly Line Balancing Problem with Worker Allocation

Wenqiang Zhang*, Lin Lin and Mitsuo Gen
Graduate School of Information, Production and Systems, Waseda University
2-7 Hibikino, Wakamatsu-ku, Kitakyushu, Fukuoka, 808-0135, Japan

Received 25-27 February 2008; Select for review from ILM 2008; Accepted 16 May 2008

ABSTRACT

Assembly lines are typical flow oriented production systems, which are of great importance in the industrial production of high quantity standardized commodities. The assembly line balancing (ALB) problems deal with the assignment of various tasks to an ordered sequence of workstations, while optimizing one or more objectives without violating restrictions imposed on the line.

In this study, we formulated the mathematical model for ALB problem with worker allocation. Later we proposed a random key-based genetic algorithm (rkGA) to solve this problem. Moreover, we presented an auto-tuning strategy, which adopts fuzzy logic controller (FLC) to tune the probabilities of the genetic operators. Finally, in order to evaluate the effectiveness of the proposed approach, we made several computational experiments using various ALB problems. The results indicated that the proposed approach improved quality of solutions and enhanced rate of convergence more than the other existing GA approaches.

Keywords: assembly line balancing (ALB), multi-objective genetic algorithm (moGA), random key-based genetic algorithm (rkGA), fuzzy logic controller (FLC), worker allocation.

1. INTRODUCTION

In the modern industry, due to variety in design, there exist various products in a factory and the product mixes change greatly. On the other hand, workers in a factory are composed of regular members and contract-base employees, so that replacement of workers may occur once or twice a year. The fresh workers are assigned to worker stations after basic training, such as how to operate some machine or how to perform basic simple work tasks. After that, they have to improve their operations skill level during on-the-job training. Therefore, according to different work experiences, the skill level of a worker in a given task differs greatly among workers (Weng et al., 2006). Usually in a factory, there are only a few skilled workers who can deal with all kinds of tasks on a high skill level. Therefore, in the real world, for each worker there always exist tasks that he/she has not mastered. Under this production environment, the assembly line should be often rebalanced, in order to improve efficiency of the line. Consequently, the assembly line balancing problem with worker allocation (ALB-wa) actually includes two sub-problems: the task assignment problem and the worker allocation problem.
The problem of design 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 (2007), 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. Most of the papers assume that resources are homogeneous; hence, the duration of tasks does not depend on the stations to which they are assigned and any task can be carried out at any station. Working with heterogeneous resources, in terms of times and/or costs, involves solving a two-fold assignment problem: resources must be assigned to stations, while tasks are assigned simultaneously to those same stations. This case is often referred to as the assembly line design 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 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 equipment selection problem is equivalent to a problem of selecting workers whose task performance speeds are different. In any event, in these works—see, for example, Akagi et al. (1983), Wilson (1986) and Lutz et al. (1994)—it is normally 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 (on a linear basis with the number of workers in some papers and on a non-linear basis in others); in other cases, it is a question of deciding which workers should work on a given shift and which should not.

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 (similar to the industrial case examined here). 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.

In this study, we have proposed a random key-based genetic algorithm (rkGA) approach to solve the ALB-wa problem. The paper is organized as follows. In section 2, we formulated the mathematical model for ALB-wa problem. In section 3, we proposed an rkGA to solve this problem. Moreover, we presented auto-tuning strategy that it adopts fuzzy logic controller (FLC) to tune the probabilities of the genetic operators depending on the change of the average fitness of parents and offspring which is occurred at each generation. In section 4, numerical experiments for various scales of assembly line balancing (ALB) problems have been used to demonstrate the effectiveness of the proposed approach. Finally, the conclusion and future research work are given in the last section.

2. MATHEMATICAL FORMULATION

The ALB-wa problem concerns with the assignment of the tasks to stations and the allocation of the available workers for each station in order to minimize the cycle time under the constraint of precedence relationships. In this study, we consider the ALB-wa problem subject to the following assumptions:

A1. The precedence constraints among assembly tasks are known and constant.
A2. A worker is assigned to one station, and only processes the tasks assigned to that station.
A3. A task cannot be split amongst two or more stations.
A4. The processing time of worker for each task is known.
A5. Material handling, loading and unloading times, set-up and tool changing times are negligible, or are included in the processing times.
A6. Task processing time differs among workers because of workers’ differences in work experience.
A7. A worker can process all the tasks, and his/her work experience differs among tasks.
A8. Stations are located along a conveyor belt according to increased station index.

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

Indices

\( j \) : index of task \((j, k = 1, ..., n)\)
\( i \) : index of station \((i = 1, ..., m)\)
\( w \) : index of worker \((w = 1, 2, ..., m)\)

Parameters
\( n \) : number of tasks
\( m \) : total number of stations/workers
\( c_T \) : cycle time of the assembly line
\( t_{ps} \) : processing time of \( j \)th task for \( w \)th worker
\( \text{Pre}(j) \) : set of direct predecessors of task \( j \)
\( S_i \) : set of tasks assigned to station \( i \)
\( t(S_i) \) : processing time at station \( i \)

\[ t(S_i) = \sum_{j=1}^{n} \sum_{w=1}^{m} t_{ps} x_{ij} y_{iw}, \forall i \]
utilization of the station $S_i$

$$u_i = \frac{1}{\max_{1 \leq r \leq m} t(S_r)}$$

$u$: average utilization of all stations

$$u = \frac{1}{m} \sum_{i=1}^{m} u_i$$

### Decision Variables

$$x_{ij} = \begin{cases} 
1, & \text{if task } j \text{ is assigned to station } i \\
0, & \text{otherwise} 
\end{cases}$$

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

### Mathematical Model

Minimize:

$$c_T = \max_{1 \leq i \leq m} \left\{ \sum_{j=1}^{n} \sum_{w=1}^{m} t_{ijw} x_{ij} y_{iw} \right\}$$

(1)

Minimize:

$$v = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (u_i - u)^2}$$

(2)

Subject to:

$$\sum_{i=1}^{m} i x_{ij} \geq \sum_{i=1}^{m} i x_{ik}, \quad \forall k \in \text{Pre}(j), \forall j$$

(3)

$$\sum_{j=1}^{n} x_{ij} = 1, \quad \forall j$$

(4)

$$\sum_{w=1}^{m} y_{iw} = 1, \quad \forall j$$

(5)

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

(6)

$$x_{ij} \in \{0, 1\} \quad \forall i, j$$

(7)

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

(8)

The first objective (1) of the model is to minimize the cycle time of the assembly line. The second objective (2) is to minimize the variation of workloads. Inequity (3) states that all predecessors of task $j$ must be assigned to a station, which is in front of or the same as the station that task $j$ is assigned in. Equation (4) ensures that task $j$ must be assigned to only one station. Equation (5) ensures that only one worker can be allocated to station $i$. Equation (6) ensures that worker $w$ can be allocated to only one station.

### 3. RANDOM KEY-BASED GA APPROACH

In this section, we proposed rkGA approach using random key-based encoding method. In the proposed approach, a combination of the arithmetical crossover and partial mapped crossover (PMX), swap mutation and immigration operator were used as genetic operations, and the adaptive weight approach (AWA) was used to determine fitness value.

The overall structure of adopted random key-based GA is given as follows:

```
procedure: The rkGA for ALB-wa problem
input: Data set of ALB-wa problem, GA parameters (popSize, maxGen, pM, pC, pI)
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 adaptive-weight 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;
        update Pareto $E(P, C)$;
        calculate $eval(P, C)$ by adaptive-weight fitness assignment routine;
        adjust $p_B(t), p_C(t)$ and $p_I(t)$ by fuzzy logic controller;
        select $P(t+1)$ from $P(t)$ and $C(t)$ by mixed roulette wheel selection routine;
        $t \leftarrow t + 1$;
    end
    output Pareto optimal solutions $E(P, C)$
end
```

#### 3.1 Genetic Representation

In ALB problem, encoding a solution into a chromosome is a key issue for GAs. For this issue, we modified the priority-based genetic algorithm (priGA), which was proposed by Gen and Cheng (1997) to solve shortest path problems.

In this study, in order to increase the quality of solution and convergence speed, we used a random key-based genetic algorithm (rkGA) approach with a real
number string coding, to solve the ALB-wa problem. In
rkGA, for obtaining a permutation from a chromosome,
the genes are treated as random keys. The rkGA is a
powerful method to represent permutations, particularly,
because there is no infeasibility problem to deal with
traditional crossover operators produces only feasible
offspring. Moreover, relative and absolute ordering in-
formation can be preserved after recombination.

The detailed encoding and decoding process of a
chromosome consists of three phases:

**Phase 1: Encoding of a random key-based chromosome.**

**Phase 2: Creating a task sequence.**

**Phase 3: Assigning tasks to stations.**

As an illustrative example, an assembly line having
10 tasks, 4 workstations and 4 worker are used. The pre-
cedence graph and the processing time of worker are
shown in Fig. 1 and Table 1.

The chromosome is composed of two parts (see
Fig. 2): the first part is the task priority vector and the
second part is worker allocation vector.

**Phase 1: Encoding of a random key-based chromosome**

In this study, the task priority vector is presented as
a random key-based encoding, i.e., the locus of each gene
represent task ID and gene are presented as real number.
The locus of each gene in worker allocation vector repre-
sent station ID and gene are presented as worker ID. An
example of generated random key-based chromosome is
shown in Fig. 2.

**Phase 2: Creating a task sequence**

Creating a task sequence procedure is same as
priority-based decoding procedure. At the beginning,
we try to find a task for the first station. Task 1, 2, 3, 7, 9
are eligible for the position, which can be easily fixed
according to adjacent relation among tasks. From the
task priority vector, we can get priorities of them are
5.31, 9.14, 7.59, 8.13, 2.72 respectively. The task 2
has the highest priority and is put into the station 1.
Then we delete task 2 off the precedence graph. Be-
cause node 7 has the largest priority value, it is put into
the station 1. From the worker allocation vector, we
allocate worker 2 to station 1. By the same manner, we
can encode the chromosome easily. The generated task
sequence vector is shown in Fig. 3.

**Phase 3: Assigning tasks to stations**

The random-key based decoding procedure consists
of four main steps:

**Step 1:** Generate the task sequence vector from the
chromosome.

**Step 2:** 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.

\[
c_{LB} = \frac{1}{m} \sum_{j=1}^{n} \min_{1 \leq w \leq m} \{ t_{jw} \}
\]

\[
c_{UB} = \frac{1}{m} \sum_{j=1}^{n} \max_{1 \leq w \leq m} \{ t_{jw} \} + \max_{1 \leq j \leq n} \{ t_{jw} \}
\]

**Step 3:** Find out optimal cycle time by bisection search-
ing. The procedure of bisection searching is shown in Fig. 4.

**Step 4:** Partition the task sequence into \( m \) parts with the
optimal cycle time based on the worker allocation vector.

![Fig. 1. A precedence graph for example](image)

![Fig. 2. The chromosome of the example](image)

![Fig. 3. The task sequence of the example](image)
Genetic operators mimic the process of heredity of genes to create new offspring at each generation. Using the different genetic operators has very large influence on GA performance. Therefore, it is important to examine different genetic operators. Considering the characteristic of rkGA, we used arithmetical crossover operator and immigration operator.

### 3.2 Genetic Operators

3.2.1 Crossover operator

Crossover is the main genetic operator. It operates on two parents (chromosomes) at a time and generates offspring by combining both chromosomes’ features. In ALB problems, crossover plays the role of exchanging each partial tasks of two chosen parents in such a manner that the offspring produced by the crossover represents.

In this study, arithmetical crossover is used for task priority vector and partial mapped crossover (PMX) is used for worker allocation vector.

#### Arithmetical Crossover (for task priority vector)

The basic concept of this kind of operator is borrowed from the convex set theory. Generally, the weighted average of two vectors $v_i(i)$ and $v_k(i)$ of $j$th chromosome and $k$th chromosome, is calculated as follows:

$$\lambda_1 v_i(i) + \lambda_2 v_k(i)$$

if the multipliers are restricted as

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

The weighted form is known as convex combination. If the nonnegative condition on the multipliers is dropped, the combination is known as an affine combination. Finally, if the multipliers are simply required to
be in real space $E$, the combination is known as a linear combination. Similarly, arithmetic operators are defined as the combination of two chromosomes $v_j$ and $v_k$ as follows:

\[ v_j'(i) = \lambda_1 v_j(i) + \lambda_2 v_k(i), \quad \forall i \]  \hspace{1cm} (12)  

\[ v_k'(i) = \lambda_1 v_j(i) + \lambda_2 v_k(i), \quad \forall i \]  \hspace{1cm} (13)  

In this study, $\lambda_1$ and $\lambda_2$ are generated by randomly. The example of arithmetical crossover for task priority is shown in Fig. 7.

**Partial Mapped Crossover (for worker allocation vector)**

The PMX procedure consists of four steps:

**Step 1:** Selecting two positions by randomly.

**Step 2:** Exchange two substrings.

**Step 3:** Determining the mapping relationship.

**Step 4:** Legalizing offspring.

The reader may refer to Gen and Cheng (1997) for more information.

### 3.2.2 Mutation operator

Mutation is a background operator which produces spontaneous random changes in various chromosomes. Several mutation operators have been proposed for the rkGA representation, such as swap mutation, inversion mutation, and insertion mutation, and so on. In this study, we use swap mutation for generating various offspring. Swap mutation selects two positions at random and then swaps the gene on these positions.

### 3.2.3 Immigration operator

The trade-off between exploration and exploitation in serial GAs for function optimization is a fundamental issue. To search effectively and efficiently, a GA must maintain a balance between these two opposing forces.

Michael 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).

#### 3.2.4 Selection operator

Selection (reproduction) operator is intended to improve the average quality of the population by giving the high quality chromosomes a better chance to get copied into the next generation. In this study, roulette wheel selection (RWS) is used to process selection operator. RWS is to determine selection probability or survival probability for each chromosome proportional to the fitness value. Then a model of roulette wheel can be made displaying these probabilities. The selection process is based on spinning the wheel the number of times equal to population size, each time selection a single chromosome for the new population. The wheel features the selection method as a stochastic sampling procedure.

### 3.3 Evaluation Function

In this research, two optimization criteria are employed to calculate the fitness value.

- Minimization of the cycle time of the assembly line
- Minimization of the variation of workload

Gen and Cheng (2000) proposed an Adaptive Weight Approach (AWA) which utilizes some useful information from the current population to readjust weights to obtain a search pressure toward a positive ideal point.

For the minimization case, we need to transform the original problem into its equivalent maximization problem and then use AWA method.

\[
\max \{ z_1 = f_1(v_k) = \frac{1}{c_T}, \quad z_2 = f_2(v_k) = \frac{1}{v} \} \]  \hspace{1cm} (14)

For the solutions at each generation, $z_q^{\text{max}}$ and $z_q^{\text{min}}$ are the maximal and minimal values for the $q$th objective as defined by the following equations:

\[
z_q^{\text{max}} = \max \{ f_q(v_k), \quad k = 1, 2, ..., \text{popSize} \}, \quad q = 1, 2 \]  \hspace{1cm} (15)

\[
z_q^{\text{min}} = \min \{ f_q(v_k), \quad k = 1, 2, ..., \text{popSize} \}, \quad q = 1, 2 \]  \hspace{1cm} (16)

The adaptive weights are calculated as
where \( w_q = \frac{1}{x_i^\text{max} - x_i^\text{min}}, \) \( q = 1, 2 \)

The weighted-sum objective function for a given chromosome is then given by the following equation:

\[
eval(v_k) = \sum_{q=1}^k w_q(f_q(v_k) - z_q^\text{min}), \quad k = 1, 2, ..., \text{popSize}
\]

### 3.4 Hybridization with Fuzzy Logic Control (FLC)

In our implementation of FLC, we automatically regulated the probability of crossover and mutation during the evolutionary process. Let \( \Delta f(t) \) is the different of average fitness function between the \( t \)th and \( (t-1) \)th generation. In this study, the inputs of FLC are \( \Delta f(t) \) and \( \Delta f(t+1) \). The outputs of the FLC are \( \Delta c(t) \) (the change of crossover probability). The membership functions of fuzzy all input and output linguistic variables are illustrated in Fig. 8.

Based on a number of experimental data and domain expert opinion, the input values are respectively normalized into integer values in the range \([-4.0, 4.0]\) according to their corresponding maximum/minimum values. The control action value for crossover, mutation and immigration operations is determined by using a look-up table as given in Table 2.

Then, the changes on crossover ratios are determined as follows:

![Membership functions of \( \Delta f(t) \) and \( \Delta f(t+1) \)](a)

![Membership functions of \( \Delta c(t) \)](b)

Where NR – Negative larger, NL – Negative large, NM – Negative medium, NS – Negative small, ZE – Zero, PS – Positive small, PM – Positive medium, PL – Positive large, PR – Positive larger

**Fig. 8. Membership functions of \( \Delta f(t) \) and \( \Delta c(t) \)**

**Table 2. Control action for crossover ratios**

<table>
<thead>
<tr>
<th>( z(i,j) )</th>
<th>-4</th>
<th>-3</th>
<th>-2</th>
<th>-1</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>-4</td>
<td>-4</td>
<td>-3</td>
<td>-3</td>
<td>-2</td>
<td>-2</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>-3</td>
<td>-3</td>
<td>-3</td>
<td>-2</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

**Locus: task**

<table>
<thead>
<tr>
<th>Task priority (( v_t ))</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.62</td>
<td>8.28</td>
<td>7.18</td>
<td>6.47</td>
<td>5.82</td>
<td>5.58</td>
<td>2.13</td>
<td>1.87</td>
<td>1.07</td>
<td>1.52</td>
<td></td>
</tr>
</tbody>
</table>

**Locus: station**

<table>
<thead>
<tr>
<th>Worker allocation (( v_w ))</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 9. The chromosome of the best compromise solution**

\[
\Delta c(t) = 0.02 \times z(i,j) \text{ where } i,j \in \{-4, -3, -2, -1, 0, 1, 2, 3, 4\}.
\]

The value of crossover ratio, mutation ratio and immigration ratio for the next generation is calculated as follows:

\[
p_C(t) = p_C(t-1) + \Delta c(t)
\]

\[
p_M(t) = 0.8 \times (1 - p_C(t))
\]

\[
p_I(t) = 1 - p_C(t) - p_M(t)
\]

### 4. EXPERIMENTS AND DISCUSSION

In order to illustrate the applicability of the proposed method, first we solved the example problem in Fig. 1. The best compromise solution is shown in Fig. 9. According to the results, we make a schedule for the problem as follows:

\[
S = \{(t_1, w_1: 0-17), (t_2, w_1: 17-38),
(t_3, w_1: 38-50), (t_4, w_2: 50-71),
(t_5, w_3: 102-128), (t_6, w_2: 71-89),
(t_7, w_3: 128-151), (t_8, w_4: 151-176),
(t_9, w_2: 89-102), (t_{10}, w_4: 176-202)\}
\]

The balancing chart of the best compromise solution is shown in Fig. 10 and the Gantt chart of best compromise solution for the schedule is shown in Fig. 11.

Later, in order to evaluate the performance of the proposed method, a large set of problems were used. Since in the literature, no benchmark data sets exist for ALB-wa, we collected 8 representative precedence graphs
from web, which are widely used in the ALB problem literature as shown in Table 3 (Scholl, 1993). From each precedence graph, 4 different ALB-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. This measure indicates the expected quality of achievable solutions and complexity of the problem. For each problem, the number of workstation is equal to the number of workers, and each task can be processed on any worker. The task time data are generated at random, while two statistical dependences are maintained: (1) Statistical dependence of task times on the task type; (2) Statistical dependence of task times on the worker type.

The parameters are set following:

Population size, \( \text{popSize} = 100 \)
Stopping condition, \( \text{maxGen} = 10000 \)
Crossover probability, \( p_C = 0.7 \)
Mutation probability, \( p_M = 0.24 \)
Immigration probability, \( p_I = 0.06 \)

### Table 3. The computational results for ALB-wa problem

<table>
<thead>
<tr>
<th>No.</th>
<th>No. of Tasks</th>
<th>No. of Stations</th>
<th>WEST Ratio</th>
<th>priGA</th>
<th>rkGA with FLC</th>
<th>Variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25</td>
<td>3</td>
<td>8.33</td>
<td>510</td>
<td>503</td>
<td>0.017</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>6.25</td>
<td>330</td>
<td>218</td>
<td>213</td>
<td>0.031</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>4.17</td>
<td>131</td>
<td>374</td>
<td>356</td>
<td>0.032</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>2.78</td>
<td>250</td>
<td>566</td>
<td>560</td>
<td>0.012</td>
</tr>
<tr>
<td>5</td>
<td>35</td>
<td>8.75</td>
<td>456</td>
<td>332</td>
<td>328</td>
<td>0.022</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>7.00</td>
<td>263</td>
<td>185</td>
<td>171</td>
<td>0.042</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>5.00</td>
<td>263</td>
<td>185</td>
<td>171</td>
<td>0.042</td>
</tr>
<tr>
<td>8</td>
<td>12</td>
<td>2.92</td>
<td>502</td>
<td>502</td>
<td>490</td>
<td>0.022</td>
</tr>
<tr>
<td>9</td>
<td>53</td>
<td>10.60</td>
<td>313</td>
<td>313</td>
<td>298</td>
<td>0.015</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>7.57</td>
<td>402</td>
<td>402</td>
<td>392</td>
<td>0.023</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>5.30</td>
<td>269</td>
<td>269</td>
<td>248</td>
<td>0.032</td>
</tr>
<tr>
<td>12</td>
<td>14</td>
<td>3.79</td>
<td>235</td>
<td>235</td>
<td>229</td>
<td>0.043</td>
</tr>
<tr>
<td>13</td>
<td>70</td>
<td>10.00</td>
<td>198</td>
<td>198</td>
<td>177</td>
<td>0.051</td>
</tr>
<tr>
<td>14</td>
<td>10</td>
<td>7.00</td>
<td>529</td>
<td>529</td>
<td>509</td>
<td>0.021</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>5.00</td>
<td>392</td>
<td>392</td>
<td>371</td>
<td>0.023</td>
</tr>
<tr>
<td>16</td>
<td>19</td>
<td>3.68</td>
<td>248</td>
<td>248</td>
<td>224</td>
<td>0.043</td>
</tr>
<tr>
<td>17</td>
<td>89</td>
<td>11.13</td>
<td>224</td>
<td>224</td>
<td>210</td>
<td>0.043</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>7.42</td>
<td>210</td>
<td>210</td>
<td>196</td>
<td>0.043</td>
</tr>
<tr>
<td>19</td>
<td>16</td>
<td>5.56</td>
<td>196</td>
<td>196</td>
<td>182</td>
<td>0.043</td>
</tr>
<tr>
<td>20</td>
<td>21</td>
<td>4.24</td>
<td>182</td>
<td>182</td>
<td>168</td>
<td>0.043</td>
</tr>
<tr>
<td>21</td>
<td>111</td>
<td>12.33</td>
<td>168</td>
<td>168</td>
<td>154</td>
<td>0.043</td>
</tr>
<tr>
<td>22</td>
<td>13</td>
<td>8.54</td>
<td>154</td>
<td>154</td>
<td>140</td>
<td>0.043</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>6.53</td>
<td>140</td>
<td>140</td>
<td>126</td>
<td>0.043</td>
</tr>
<tr>
<td>24</td>
<td>22</td>
<td>5.05</td>
<td>126</td>
<td>126</td>
<td>112</td>
<td>0.043</td>
</tr>
<tr>
<td>25</td>
<td>148</td>
<td>14.80</td>
<td>112</td>
<td>112</td>
<td>98</td>
<td>0.043</td>
</tr>
<tr>
<td>26</td>
<td>14</td>
<td>10.57</td>
<td>98</td>
<td>98</td>
<td>84</td>
<td>0.043</td>
</tr>
<tr>
<td>27</td>
<td>21</td>
<td>7.05</td>
<td>84</td>
<td>84</td>
<td>70</td>
<td>0.043</td>
</tr>
<tr>
<td>28</td>
<td>29</td>
<td>5.10</td>
<td>70</td>
<td>70</td>
<td>56</td>
<td>0.043</td>
</tr>
<tr>
<td>29</td>
<td>297</td>
<td>15.63</td>
<td>56</td>
<td>56</td>
<td>42</td>
<td>0.043</td>
</tr>
<tr>
<td>30</td>
<td>29</td>
<td>10.24</td>
<td>42</td>
<td>42</td>
<td>28</td>
<td>0.043</td>
</tr>
<tr>
<td>31</td>
<td>38</td>
<td>7.82</td>
<td>28</td>
<td>28</td>
<td>24</td>
<td>0.043</td>
</tr>
<tr>
<td>32</td>
<td>50</td>
<td>5.94</td>
<td>24</td>
<td>24</td>
<td>20</td>
<td>0.043</td>
</tr>
</tbody>
</table>
The program was written in C language, and implemented on HP-d330 PC, with a 2.4G Intel Pentium-4 CPU, and 512 MB memory. For each problem, we ran the procedure for 20 times and compared the results of priGA and rkGA with FLC (see Table 3).

The changes in the crossover and mutation ratios are both shown in Fig. 12. The Pareto solution is illustrated in Fig. 13.

As shown in Table 3, the results indicated that the solution produced by rkGA with FLC is slightly better than priGA did, where both the variation of workload and cycle time are reduced.

The comparison of variation of workload between priGA and rkGA with FLC is shown in Fig. 14, and Fig. 15 shows that the comparison of cycle time between priGA and rkGA with FLC.

5. CONCLUSIONS

In this study, we extended the simple assembly line balancing problem to more complex assembly line balancing problem with worker allocation. To solve this problem, first a random key-based representation method adapting the GA was proposed. Following, advanced genetic operators adapted to the specific chromosome structure and the characteristics of the ALB-wa problem were used. Moreover, in order to strengthen the search ability, an effective Fuzzy Logic Controller (FLC) was also integrated under the framework of GA. Finally, the performance of proposed method was validated through numerical experiments. The results indicated that the proposed approach improved quality of solutions and enhanced rate of convergence more than the other existing GA approaches.

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.

REFERENCES


and task assignment for multi-product assembly system


