Int j simul model 18 (2019) 1, 175-186
ISSN 1726-4529 Original scientific paper

AN IMPROVED GENETIC SIMULATED ANNEALING ALGORITHM FOR STOCHASTIC TWO-SIDED ASSEMBLY LINE BALANCING PROBLEM

Yang, M. S.; Ba, L.[#]; Liu, Y.; Zheng, H. Y.; Yan, J. T.; Gao, X. Q. & Xiao, J. M.
School of Mechanical and Precision Instrument Engineering, Xi'an University of Technology, 710048 Xi'an, China
E-Mail: yangmingshun@xaut.edu.cn, xautbali@163.com ([#] Corresponding author)

Abstract
Considering the random makespans of assembly jobs, this paper develops a mathematical model of stochastic two-sided assembly line balance problem (STALBP) and designs an improved genetic simulated annealing algorithm (IGSAA) based on priority coding. Firstly, a coding method was proposed based on priority value. Then, the job assembly sequence was derived from the chromosome coding sequence, and the job allocation positions were determined, forming a specific allocation plan. After that, the author designed the corresponding decoding method. To enhance the local search ability of the genetic algorithm (GA), the simulated annealing operation was introduced after the mutation, reversing the individuals in the temporary child population. Next, the superiority of the IGSAA was verified by a set of standard examples, and an actual loader assembly line with normally distributed job makespans was balanced by the proposed algorithm. The research findings provide a valuable reference for the balancing of assembly lines.
(Received, processed and accepted by the Chinese Representative Office.)

Key Words: Stochastic Two-Sided Assembly Line Balance Problem (STALBP), Improved Genetic Simulated Annealing Algorithm (IGSAA), Makespan, Assembly Job, Previous Job

1. INTRODUCTION

The two-sided assembly line is a line where jobs on the same job can be performed in parallel at both sides of the line. It is widely used in the assembly of large and complex mechanical products, such as automobiles, trucks and loaders [1]. Compared with the single-sided assembly line, the two-sided assembly line enjoys a short length, a low material handling cost, efficient operator movements, and a limited cost of tools and fixtures [2]. The balanced production of the two-sided assembly line calls for a balancing technology that minimizes the number of required positions, reduces the initial machine investment, equalize the load of each machine, and control the uneven idleness.

In recent years, the two-sided assembly line balancing problem (TALBP) has received wide attention. The makespan of each assembly job is generally assumed as a constant [3]. In actual production, however, the makespan often varies with the random factors (e.g. operation skill, fatigue degree and tool performance), especially when manual operation is involved on the assembly line. Hence, the assembly line designed under the said assumption may face the risk that the assembly job cannot be completed within the production tact in actual production. Below is a brief review of the studies on the TALBP.

Li et al. [1] designed an improved iterative greedy algorithm for the TALBP, and verified its advancement by several standard examples. Targeting the TALBP with uncertain job time, Tang et al. [3] combined the teaching algorithm and variable neighbourhood search into a hybrid algorithm, established a mathematical model based on the hybrid algorithm that minimizes the number of paired machines and the total number of machines. Li et al. [4] reviewed the heuristic solutions to the TALBP, and compared the recent heuristic algorithms using several standard examples. To minimize the number of paired machines and the total

number of machines, Kucukkoc et al. [5] developed a mathematical model considering the underground machine TALBP, designed an improved ant colony algorithm (ACA). Lei and Guo [6] put forward a variable neighbourhood search algorithm for the second type of TALBP. Kucukkoc and Zhang [7] created an ACA based on flexible agent for parallel TALBP, and proved the algorithm as effective and advanced by different scale examples. Tapkan et al. [8] considered walking distances into a parallel TALBP. An improved bee algorithm was designed for solving the problem. Kucukkoc and Zhang [9] solved the type-E parallel TALBP by an improved ACA. Yuan et al. [10] designed a hybrid bee mating optimization algorithm for the TALBP. In light of tact time and priority constraints, Tuncel and Aydin [11] set up a mathematical model for the TALBP, designed a teaching algorithm. Purnomo and Wee [12] built a mathematical model of dual-target TALBP with the goal of maximizing productivity and balancing load, and designed a harmony search algorithm.

Besides, metaheuristic algorithms are widely applied and deeply researched for some other production scheduling problems, such as flow-shop scheduling problem [13], job-shop scheduling problem [14], flexible job-shop scheduling problem [15, 16] and so on.

In view of the above, this paper constructs a stochastic TALBP, denoted as the STALBP, based on the deterministic TALBP, considering the random makespans of assembly jobs, and presents an improved genetic simulated annealing algorithm (IGSAA) on priority value coding. Then, an example of loader assembly line balancing problem with normal distribution of job makespans was analysed, and a specific balance plan was given, which verifies the effectiveness of the model and algorithm.

2. PROBLEM DESCRIPTION

The STALBP in this paper is defined as follows: Under the given production tact, it is assumed that the makespans of the assembly jobs are independent of each other, in line with the normal distribution and subjected to the constraints of inseparability, priority relationship and operational orientation. In addition, all jobs should be allocated as evenly as possible to the machines on both sides of the assembly line, to minimize the number of positions, maximize the efficiency and minimize the smoothness index.

The line efficiency (*LE*) and the smoothness index (*SI*) are two common indicators of assembly line performance. The former equals to the ratio of the total makespan of all assembly jobs to the total time of the enabled machine, while the latter describes the relative smoothness of the assembly line. The value of the *SI* is negatively correlated with the load difference between machines and positively with the degree of balance. In addition to the *LE* and the *SI*, the following parameters were defined before establishing the mathematical model of the STALBP:

- n – the total number of assembly jobs;
- T – the set of assembly jobs, $T = \{1, 2, \ldots, n\}$;
- NM – the number of positions required for the assembly line;
- NM_0 – the initial number of positions;
- LE_0 – the initial line efficiency;
- SI_0 – the initial smoothness index;
- P – the set of positions, $P = \{1, 2, \ldots, NM\}$;
- $Tsum$ – the total makespan of all jobs;
- SL_k – the operation duration of the last job at the K^{th} machine (machine load);
- SL_{max} – the maximum machine load;
- v_1 – the position weight;
- v_2 – the line efficiency weight;
- v_3 – the smoothing index weight;

CT – the production tact;
Pr_i – the set of previous jobs for job i;
t_i – the makespan of job i;
t_i^f – the makespan of job i on machine;
α – the completion rate, i.e. the probability of a job to be assembled within the tact time.

Owing to the random makespans, the makespan of job i (t_i), the makespan of job i on machine (t_i^f) and the tact constraint in the STALBP should be determined differently from those in the deterministic TALBP.

(1) The makespan of job i (t_i)

The previous research has shown that the makespans of the assembly jobs are independent of each other, and usually in line with the normal distribution, i.e. $t_i \sim N(\mu_i, {\sigma_i}^2)$ [17]. Note that μ_i and σ_i are the mean and standard deviations of t_i, respectively.

(2) The makespan of job i on machine (${t_i}^f$)

Since the makespans of the assembly jobs are distributed normally, the makespans of the jobs on machine must also obey the normal distribution. The makespans of the jobs in a stochastic two-sided assembly line are illustrated in Fig. 1, where jobs a, b and c are independent of each other and in line with the normal distribution, i.e. $t_a \sim \mathrm{N}(\mu_a, {\sigma_a}^2)$, $t_b \sim \mathrm{N}(\mu_b, {\sigma_b}^2)$ and $t_c \sim \mathrm{N}(\mu_c, {\sigma_c}^2)$. Assuming that job b is a previous job of job c, jobs a(L), b(R) and c(L) were allocated to both sides of the assembly line. Then, the makespan of job a on machine can be expressed as ${t_a}^f = t_a$, and that of job b on machine as ${t_b}^f = t_b$. Under the priority relationship between the jobs, the makespan of job c on machine can be derived as ${t_c}^f = {t_a}^f + t_{c,idle} + t_c = {t_b}^f + t_c$, where $t_{c,idle}$ is the invalid waiting time before the start of job c (the shaded part in Fig. 1). According to the additivity of normal distributions (Lévy–Cramér theorem), we have ${t_c}^f \sim \mathrm{N}(\mu_b + \mu_c,\ {\sigma_b}^2 + {\sigma_c}^2)$.

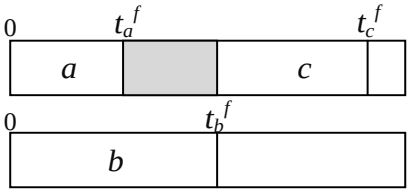

Figure 1: The makespans of three jobs in a stochastic two-sided assembly line.

(3) The tact constraint

When the makespans of the assembly jobs obey the normal distribution, the machine load is a random variable subjected to the normal distribution. In theory, there is no guarantee that the job on machine can be assembled within a fixed production tact, no matter how large the production tact. There is a certain probability for the machine load to exceed the production tact. The probability for a job to complete the assembly within the tact time is known as the completion rate, denoted as α ($0.5 < \alpha < 1$).

3. MODEL ESTABLISHMENT

Based on the above description, the mathematical model of our STALBP can be established as follows.

The line efficiency:

$$LE = \frac{TSum}{NS \cdot \mathrm{CT}} = \frac{\sum_{i=1}^{n} \mu_i + \Phi^{-1}(\alpha)\sqrt{\sum_{i=1}^{n} {\sigma_i}^2}}{NS \cdot \mathrm{CT}} \quad (1)$$

The smoothness index:

$$SI = \sqrt{\frac{\sum_{k=1}^{NS} (SL_k - SL_{\max})^2}{NS}} \tag{2}$$

The objective function (minimal smoothness index and maximal line efficiency):

$$\min f = v_1 \frac{NM}{NM_0} + v_2 \frac{LE_0}{LE} + v_3 \frac{SI}{SI_0} \tag{3}$$

The STALBP is subjected to the same constraints as the basic TALBP, except the tact constraint. The difference in tact constraint is attributed to the random makespans in the STALBP. The tact constraint of the STALBP is shown in Eq. (3) above, which means that the probability for all jobs on machine to be completed within the production tact should not be less than the completion rate *a*:

$$P\{t_i^f \le \mathrm{CT}\} \ge \alpha, \ \forall i \in T \tag{4}$$

Since the makespan of job *i* on machine $t_i^f \sim N(\mu_i^f, \sigma_i^{f^2})$, the normal distribution can be transformed into the standard normal distribution:

$$P\{t_i^f \le \mathrm{CT}\} = P\{\frac{t_i^f - \mu_i^f}{\sqrt{\sigma_i^{f^2}}} \le \frac{\mathrm{CT} - \mu_i^f}{\sqrt{\sigma_i^{f^2}}}\} = \Phi(\frac{\mathrm{CT} - \mu_i^f}{\sqrt{\sigma_i^{f^2}}}) \ge \alpha \tag{5}$$

Sorting out Eq. (5), the tact constraint of the STALBP can be obtained as:

$$\mu_i^f + \Phi^{-1}(\alpha)\sqrt{\sigma_i^{f^2}} \le CT, \forall i \in T \tag{6}$$

In addition, all assembly jobs must be allocated in a manner that each job is allocated to the left or right side of a position (the inseparability constraint):

$$\sum_{p \in P} \sum_{d \in D} x_{ipd} = 1, \forall i \in T \tag{7}$$

A job cannot be allocated before the allocation of all its previous jobs (the priority sequence constraint):

$$\sum_{q \in P} \sum_{w \in D} q x_{hqw} \le \sum_{p \in P} \sum_{d \in D} p x_{ipd}, \forall i \in V, h \in Pr_i \tag{8}$$

A job must be allocated to the left or right side of the assembly line (the operation orientation constraint):

$$\sum_{p \in P} x_{ip0} = 1, \ \forall i \in TL \tag{9}$$

$$\sum_{p \in P} x_{ip1} = 1, \ \forall i \in TR \tag{10}$$

4. THE IGSAA

This section combines the genetic algorithm (GA) and the simulated annealing algorithm (SAA) into the IGSAA to solve the STALBP based on priority value coding.

4.1 Encoding and decoding

(1) Encoding

Let *n* be the total number of assembly jobs, and *a*, *b*, and *c* (*a* + *b* + *c* = *n*) be the jobs subjected to the operation orientation constraints of left side (*L*), right side (*R*) and either side (*E*), respectively. Then, *n* random unequal integers were generated from the interval [1, *n*], and arranged in a row to form a chromosome. The element *W*(*i*) at position *i* of the

chromosome stands for the allocation priority of the i^{th} assembly job. Taking the P16 as an example, a feasible chromosome coding plan is shown in Fig. 2 below.

1	15	11	16	5	10	6	2	8	9	7	3	4	13	12	14

Figure 2: An example of chromosome coding.

(2) Decoding

The decoding is a two-phase operation.

Phase 1: Deriving the job assembly sequence

Step 1. Enter the chromosomal sequence *List*, with an empty set *indiJob* to store the job assembly sequence corresponding to the chromosome. Hence, the set of current jobs $CS = T$, with T being the set of all assembly jobs.

Step 2. Check that no previous or subsequent job in the candidate set *CS* is stored in the set of allocable jobs *FS*.

Step 3. Find the priority value corresponding to each job in the *FS* in the chromosome list, and denote the job with the highest priority as *seleJob*.

Step 4. Add the job *seleJob* to the set *indiJob*, and delete the *seleJob* from the candidate set *CS*.

Step 5. Judge whether the set *CS* is empty; if yes, terminate the program; otherwise, return to Step 2.

Fig. 3 presents the assembly sequence obtained from chromosome coding sequence in Fig. 2 according to the above steps.

2	5	1	4	3	6	7	10	9	13	16	12	8	11	14	15

Figure 3: An example of the assembly sequence.

Phase 2: Preparing the allocation plan

There are four common priority rules for the job allocation: (1) the rule of minimum makespan: the job with shorter makespan should be prioritized; (2) the rule of minimum job encoding: the job with smaller serial number should be prioritized; (3) the rule of maximum number of adjacent subsequent jobs: the job with more subsequent jobs should be prioritized; (4) the rule of random selection: the jobs should be selected randomly before allocation. To overcome the shortcomings of each rule, the four priority rules were adopted respectively to output four feasible balance plans, and the one with the best balancing effect was adopted.
The workflow is illustrated in Fig. 4.

4.2 Basic flow of the IGSAA

(1) Population initialization

The integers were selected randomly from the interval [1, n] to form a chromosome sequence of nonrepetitive random integers. This process was repeated N (P) times, producing N (P) chromosomes, where N (P) is the population scale.

(2) Fitness calculation

The objective function was taken as the fitness function. The fitness of the i^{th} individual in the population can be computed as:

$$f(i) = v_1 \frac{NM}{NM_0} + v_2 \frac{LE_0}{LE} + v_3 \frac{SI}{SI_0} \tag{11}$$

(3) Selection

The gambling method was adopted to select individuals for the mating library at a probability negatively correlated with the fitness.

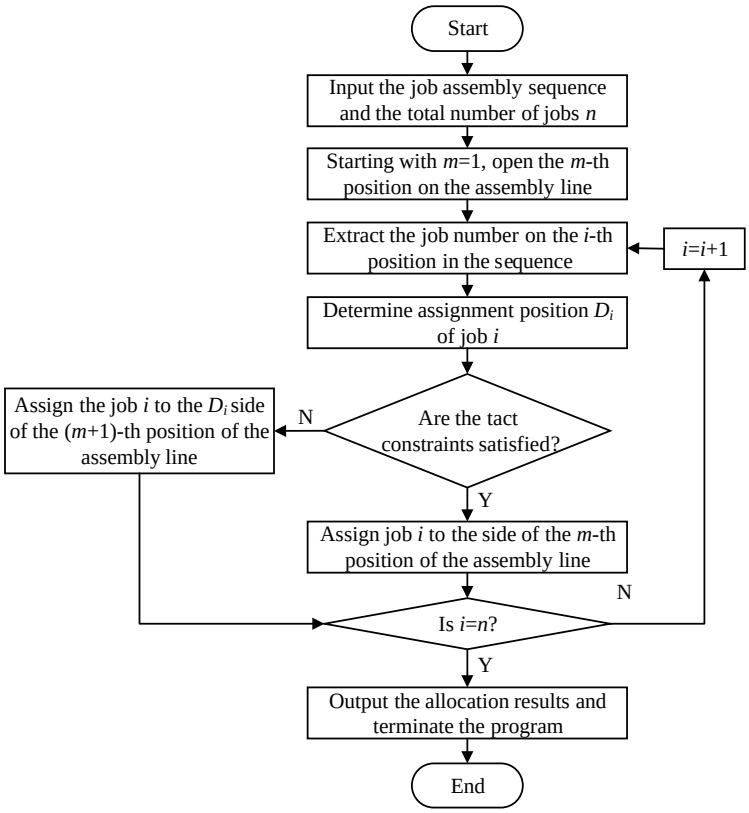

Figure 4: Workflow of decoding process.

(4) Crossover

Based on the priority value encoding, the priority relationship between the assembly jobs need no consideration during the crossover and mutation of the chromosomes. Here, the precedence operation crossover (POX) is employed for the crossover operation.

Assuming the parent chromosomes F_1 and F_2 were cross-operated, and 4 position numbers were randomly generated as numbers are [3, 9, 10, 14]. The resulting child chromosomes O_1 and O_2 are shown in Fig. 5 below.

F_1	1	15	11	16	5	10	6	2	8	9	7	3	4	13	12	14
F_2	5	10	16	3	7	6	12	11	13	2	14	15	8	9	1	4
O_1	1	15	11	16	5	10	6	2	13	8	7	3	4	9	12	14
O_2	5	10	16	3	7	6	12	11	8	2	14	15	9	13	1	4

Figure 5: An example of the POX.

(5) Mutation

Assuming the two random positions were [4, 9]. An example is shown in Fig. 6.

Before mutation	1	15	11	16	5	10	6	2	13	8	7	3	4	9	12	14
After mutation	1	15	11	13	5	10	6	2	16	8	7	3	4	9	12	14

Figure 6: An example of the mutation operation.

(6) Simulated annealing operation

To enhance the local search ability of the GA, a simulated annealing operation was introduced after the mutation operation. Assuming the two random positions are $P_1 = 10$ and $P_2 = 15$, the result of the reversal operation is shown in Fig. 7.

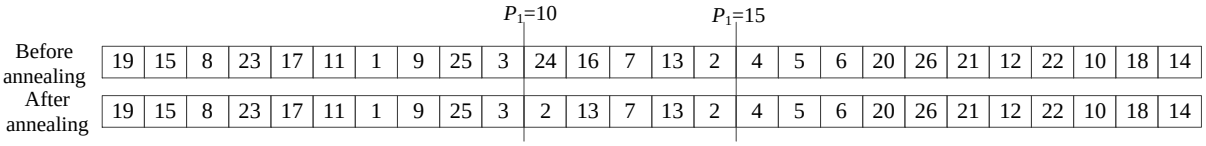

Figure 7: An example of simulated annealing operation.

(7) The overall process of the algorithm

The workflow of the IGSAA is described in Fig. 8 below.

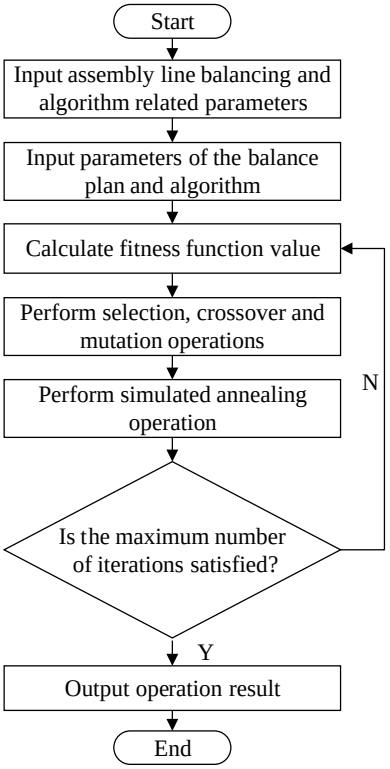

Figure 8: Workflow of the IGSAA.

5. EXAMPLE VERIFICATION

For the standard example of P9, P12, P16, P24 deterministic TALBP, it is known that the processing time of assembly jobs is a constant value T_i. On this basis, the author constructed an STALBP example with the job makespan $t_i \sim N(\mu_i, {\sigma_i}^2)$, and the mean processing time μ_i of job i equal to T_i. To observe the effect of our algorithm in different variance regions, the variance was divided into low variance and high variance. The value of low variance ${\sigma_i}^1$ is a random number between $(0, (T_i/4)^2)$, and that of high variance ${\sigma_i}^2$ is a random number between $(0, (T_i/2)^2)$. The proposed IGSAA was programmed by MATLAB R2012b. To facilitate the comparison experiment, the variance values were generated indirectly by 'rand('seed',1)' of MATLAB, where the low variance is rand $(1, n)\cdot(T_i/4)^2$ and the high variance is equally available. Under different production tacts, the completion rate α equaled 0.9, 0.95 and 0.975 ($\Phi^{-1}(\alpha)$ was 1.28, 1.645 and 1.96). Then, the SAA and the proposed IGSAA were adopted to solve the above problem.

Let the weights of the objective function be $v_1 = 1$, $v_2 = 0.5$ and $v_3 = 0.3$. Through several experiments, the SAA parameters were set as: initial temperature $Tb = 1000$, termination temperature $To = 0.001$ and cooling coefficient $\beta = 0.9$; the IGSAA parameters were set as: the maximum number of iterations $N = 150$, population size $N(P) = 20$, crossover probability $Pc = 0.8$, mutation probability $Pm = 0.2$, start time of simulated annealing $Tb = 100$, termination

time To = 10, and cooling coefficient β = 0.6. The two algorithms were run 20 times respectively, and the optimal solution in the 20 operations was taken as the final solution of each algorithm. The number of positions corresponding to the final solution was recorded, together with the required number of machines *NS*, line efficiency *LE* and smoothing index *SI*. Tables I and II compare the results of the two algorithms under low variance and high variance, respectively.

Table I: Results comparison at low variance.

Problem	Tact	Φ-1 (α)	SAA			IGSA		
			NM [*NS*]	*LE* (%)	*SI*	*NM* [*NS*]	*LE* (%)	*SI*
P9	5	1.28	3 [5]	72.1950	0.6709	3 [5]	72.1950	0.4579
		1.645	3 [5]	73.3912	0.7438	3 [5]	73.3912	0.4736
		1.96	3 [5]	74.4235	0.9896	3 [5]	74.4235	0.4910
	6	1.28	2 [4]	75.2031	0.2443	2 [4]	75.2031	0.2443
		1.645	2 [4]	76.4492	0.6793	2 [4]	76.4492	0.3140
		1.96	2 [4]	77.5245	1.1430	2 [4]	77.5245	0.4064
P12	5	1.28	5 [8]	65.5138	1.5529	5 [7]	74.8729	0.6098
		1.645	5 [7]	75.8551	1.0654	5 [7]	75.8551	0.6395
		1.96	5 [7]	76.7027	1.1119	4 [7]	76.7027	0.9059
	6	1.28	4 [6]	72.7931	0.7829	3 [6]	72.7931	0.5625
		1.645	4 [6]	73.7480	1.2335	3 [5]	88.4976	0.3701
		1.96	4 [6]	74.5721	1.4693	3 [6]	74.5721	1.1157
	7	1.28	4 [6]	74.5721	1.4693	3 [5]	74.8729	0.6031
		1.645	3 [6]	63.2126	1.3417	3 [5]	75.8551	0.5979
		1.96	3 [6]	63.9189	0.9891	3 [5]	76.7027	0.5552
	8	1.28	3 [5]	65.5138	1.0661	2 [4]	81.8922	0.6214
		1.645	3 [4]	82.9665	0.9604	2 [4]	82.9665	1.2668
		1.96	3 [5]	67.1149	3.2743	2 [4]	83.8936	0.6817
P16	16	1.28	5 [8]	67.0536	4.8578	5 [7]	76.6327	1.1434
		1.645	5 [8]	67.9066	3.2013	5 [7]	77.6075	1.0786
		1.96	5 [9]	61.0157	4.6413	5 [7]	78.4488	2.0965
	19	1.28	4 [7]	64.5328	2.8286	3 [6]	75.2883	2.3275
		1.645	4 [7]	65.3537	3.4613	3 [6]	76.2460	3.0664
		1.96	4 [7]	66.0621	4.7003	3 [6]	77.0725	2.5056
	22	1.28	3 [6]	65.0217	5.5999	3 [5]	78.0261	0.7011
		1.645	3 [6]	65.8488	3.4271	3 [5]	79.0186	2.0254
		1.96	4 [6]	66.5626	4.8011	3 [5]	79.8751	2.0244
P24	20	1.28	7 [11]	66.4577	5.4492	5 [9]	81.2261	2.0369
		1.645	6 [11]	67.2622	4.7255	6 [10]	73.9884	2.5863
		1.96	7 [12]	62.2935	5.6107	6 [11]	67.9565	3.9264
	25	1.28	5 [9]	64.9809	6.3784	4 [8]	73.1035	2.3248
		1.645	5 [9]	65.7675	3.6840	4 [8]	73.9884	1.9707
		1.96	5 [9]	66.4464	5.4374	4 [8]	74.7522	1.7721
	30	1.28	4 [7]	69.6224	4.3649	3 [6]	81.2261	2.3737
		1.645	4 [7]	70.4652	4.2135	3 [6]	82.2094	1.0187
		1.96	4 [8]	62.2935	5.8297	4 [7]	71.1926	2.1585
	35	1.28	3 [6]	69.6224	3.7397	3 [5]	83.5468	2.2721
		1.645	3 [6]	70.4652	4.9449	3 [6]	70.4652	1.9254
		1.96	3 [6]	71.1926	4.3644	3 [5]	85.4311	2.3341
	40	1.28	3 [6]	60.9196	7.0163	2 [4]	91.3793	1.7637
		1.645	3 [6]	61.6570	6.3073	3 [5]	73.9884	2.4020
		1.96	3 [6]	62.2935	5.0915	3 [5]	74.7522	1.9534

Table II: Results comparison at high variance.

Problem	Tact	Φ-1 (α)	SAA			IGSA		
			NM [NS]	*LE* (%)	*SI*	*NM* [NS]	*LE* (%)	*SI*
P9	5	1.28	3 [6]	63.6585	1.2618	3 [6]	63.6585	0.7761
		1.645	3 [6]	65.6523	1.2571	3 [6]	65.6523	0.9139
		1.96	4 [7]	57.7483	1.4543	3 [6]	67.3730	1.0314
	6	1.28	3 [4]	79.5732	0.5539	2 [4]	79.5732	0.3185
		1.645	3 [5]	65.6523	0.5962	2 [4]	82.0654	0.2203
		1.96	3 [5]	67.3730	0.9555	3 [5]	67.3730	0.6589
P12	5	1.28	5 [8]	68.5280	1.1288	5 [8]	68.5280	0.9849
		1.645	6 [9]	62.4417	1.4172	5 [9]	62.4417	0.8956
		1.96	7 [10]	57.3843	1.3839	5 [9]	63.7603	0.9218
	6	1.28	4 [8]	57.1067	2.1256	3 [6]	76.1422	1.0998
		1.645	4 [7]	66.9018	1.3728	4 [6]	78.0521	0.6239
		1.96	5 [8]	59.7753	1.8559	4 [8]	59.7753	1.8319
	7	1.28	3 [6]	65.2648	0.9462	3 [5]	78.3177	0.4881
		1.645	4 [6]	66.9018	1.0970	3 [5]	80.2822	0.4675
		1.96	3 [6]	68.3146	1.6851	3 [6]	68.3146	1.3429
	8	1.28	3 [5]	68.5280	1.7514	3 [5]	68.5280	0.3286
		1.645	3 [5]	70.2469	1.3558	3 [5]	70.2469	0.4675
		1.96	3 [5]	71.7304	1.6576	3 [5]	71.7304	0.5704
P16	16	1.28	7 [9]	62.2620	2.6352	5 [8]	70.0448	2.2864
		1.645	6 [11]	52.1823	3.4147	5 [9]	63.7784	2.4547
		1.96	6 [10]	58.5783	3.1036	5 [10]	58.5783	2.9161
	19	1.28	4 [7]	67.4115	4.1030	3 [6]	78.6468	2.5207
		1.645	5 [8]	60.4216	4.9202	5 [7]	69.0533	2.3746
		1.96	5 [8]	61.6613	4.1383	5 [7]	70.4701	2.7139
	22	1.28	4 [6]	67.9222	2.3809	3 [5]	81.5066	2.5216
		1.645	4 [7]	59.6369	2.8968	3 [6]	69.5764	2.5256
		1.96	4 [7]	60.8605	4.6117	3 [6]	71.0040	2.6829
P24	20	1.28	7 [12]	63.5058	4.7088	7 [12]	63.5058	4.0034
		1.645	8 [14]	55.6978	5.0180	7 [13]	59.9822	4.4538
		1.96	9 [14]	56.7889	4.5117	7 [14]	56.7889	4.6433
	25	1.28	5 [9]	67.7395	2.6783	4 [8]	76.2070	2.1961
		1.645	6 [10]	62.3815	7.4206	5 [8]	77.9769	1.5082
		1.96	6 [10]	63.6035	5.3406	5 [9]	70.6706	2.5264
	30	1.28	4 [7]	72.5781	3.6236	4 [7]	72.5781	1.7014
		1.645	5 [9]	57.7607	5.9936	4 [7]	74.2637	3.3319
		1.96	5 [10]	53.0029	4.8347	4 [7]	75.7185	2.8567
	35	1.28	3 [6]	72.5781	9.3346	3 [6]	72.5781	2.0458
		1.645	4 [7]	63.6546	6.4171	3 [6]	74.2637	2.4050
		1.96	4 [8]	56.7889	8.66512	3 [6]	75.7185	2.8449
	40	1.28	3 [6]	63.5058	3.8053	3 [5]	76.2070	1.3028
		1.645	3 [6]	64.9808	4.1463	3 [5]	77.9769	2.3187
		1.96	3 [6]	66.2537	4.7140	3 [5]	79.5044	2.3128

Table I lists the results of 42 low variance experiments on the two algorithms. Table II lists the results of 42 high variance experiments on the two algorithms. In summary, the IGSAA is more likely to obtain fewer positions, higher efficiency and smaller smoothness index than the SAA, indicating that it outperformed the latter both in low and high variances.

6. EMPIRICAL ANALYSIS

This section applies the proposed model and algorithm to the balancing of a loader assembly line in an enterprise, which is expected to produce $H = 34{,}500$ loaders per year. The work schedule of the enterprise has $N = 3$ shifts, each of which lasts $T = 8$ hours, and the number of working days is $D = 280$. Processing time for completing the i^{th} ($i \le n$) assembly job is represented by symbol t_i. Assuming that the makespans of the jobs on the assembly line obey the normal distribution, the mean processing time μ_i of job i was equal to t_i, the variance σ_i^2 was selected as a random number in $(0, (t_i/25)^2)$, and the completion rate α was equal to 0.9.

The job priority relationship is presented as Fig. 9 below.

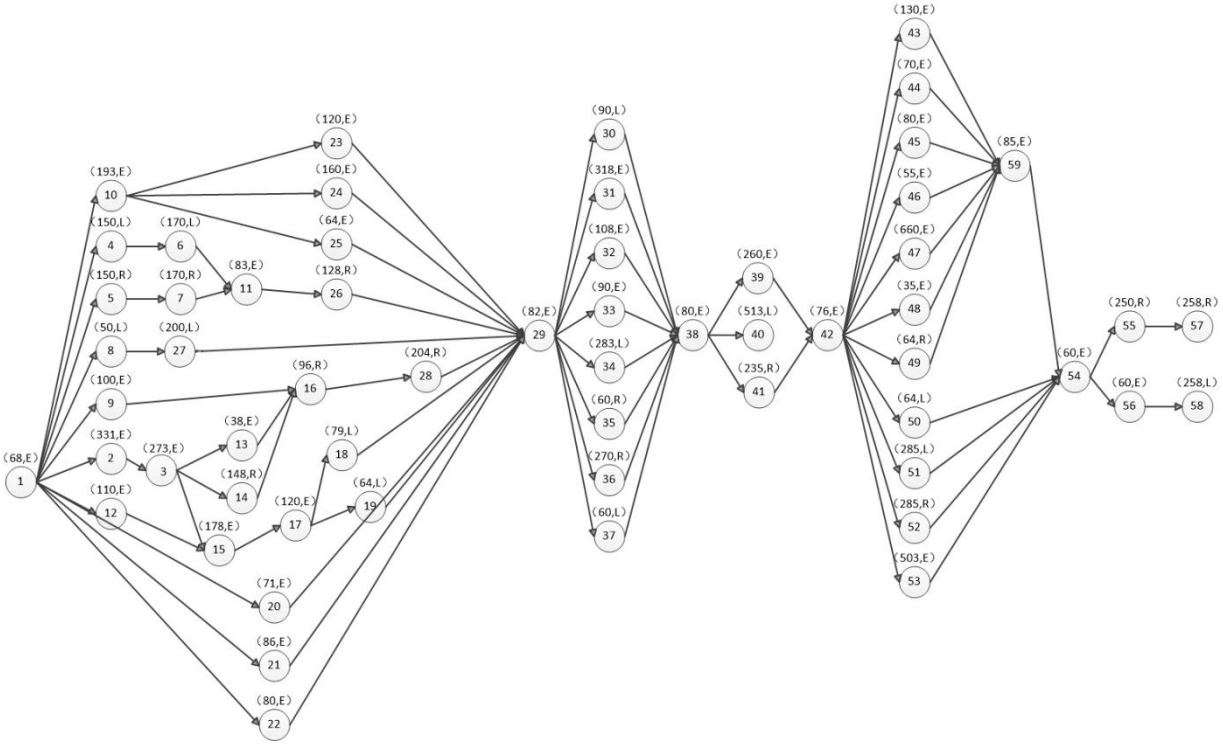

Figure 9: The priority sequence diagram.

The IGSAA and the SAA were both introduced to solve the above example. The objective function weights were set to $v_1 = 1$, $v_2 = 0.5$ and $v_3 = 0.1$ of the objective function. Through multiple experiments, the algorithm parameters were set as: population size $N(P) = 20$, the maximum number of iterations $N = 200$, crossover probability $Pc = 0.8$, mutation probability $Pm = 0.2$, initial temperature $Tb = 100$, termination temperature $To = 10$, and cooling coefficient $\beta = 0.7$. The algorithm was run 20 times, and the best solution was selected as the final solution. The balance plans of the IGSAA and the SAA are displayed in Table III.

According to results from Table III, the IGSAA plan had a shorter, more efficient and more balanced assembly line than the SAA plan. Thus, the IGSAA plan was determined as the final and optimal balance plan.

7. CONCLUSIONS

(1) Considering the random makespans of assembly jobs in two-sided assembly line balance problem, this paper develops a mathematical model to optimize the position number, line efficiency and smoothness index of the STALBP.

Table III: The allocation of jobs for each machine.

SAA			IGSA		
Position	Work station	Assigned operation element	Position	Work station	Assigned operation element
1	1	1, 20, 3	1	1	1, 22, 20, 24, 9
	2	2, 12		2	21, 10, 12
2	3	13, 21, 4, 6	2	3	23, 4, 3
	4	15, 17, 5, 7		4	2
3	5	10, 25, 9, 24, 18, 19	3	5	15, 18, 8, 27
	6	11, 26, 22, 23, 14, 16		6	14, 17, 13, 16, 28
4	7	8, 27, 33, 30, 37	4	7	19, 6, 25
	8	28, 29, 35, 32		8	5, 7, 11, 26, 29, 35
5	9	31, 34	5	9	37, 32, 33, 30, 34
	10	36, 38		10	36, 31
6	11	-	6	11	40, 42
	12	39, 41, 42		12	38, 41, 39
7	13	51	7	13	53, 45, 50
	14	47		14	47
8	15	48	8	15	46, 51
	16	45, 49, 53		16	49, 48, 43, 44, 59, 52
9	17	40, 44, 46	9	17	54, 56, 58
	18	43, 52		18	55, 57
10	19	50, 56, 58			
	20	59, 54, 55, 57			

(2) In addition, the author proposed an IGSAA based on priority value coding and details the design of each module of the algorithm.

(3) The advantages of the IGSAA were verified by standard examples. Then, the algorithm was applied to solve an actual example, yielding a final balance plan.

In our model, the makespans are assumed to obey the normal distribution. Compared with the basic two-sided assembly line balance problem, our STALBP model is in line with the actual situation because the real-world assembly process is subjected to various interferences. The research findings provide a valuable reference for the balancing of assembly lines.

ACKNOWLEDGEMENTS

This research is supported by the National Natural Science Foundation of China (Grant No: 61402361, 60903124); Xi'an University of Technology Initial Foundation for the PhDs (Grant No: 102-451117013).

REFERENCES

[1] Li, Z.; Tang, Q.; Zhang, L. (2017). Two-sided assembly line balancing problem of type I: Improvements, a simple algorithm and a comprehensive study, *Computers & Operations Research*, Vol. 79, 78-93, doi:10.1016/j.cor.2016.10.006

[2] Bartholdi, J. J. (1993). Balancing two-sided assembly lines: a case study, *International Journal of Production Research*, Vol. 31, No. 10, 2447-2461, doi:10.1080/00207549308956868

[3] Tang, Q.; Li, Z.; Zhang, L.; Zhang, C. (2017). Balancing stochastic two-sided assembly line with multiple constraints using hybrid teaching-learning-based optimization algorithm, *Computers & Operations Research*, Vol. 82, 102-113, doi:10.1016/j.cor.2017.01.015

[4] Li, Z.; Kucukkoc, I.; Nilakantan, J. M. (2017). Comprehensive review and evaluation of heuristics and meta-heuristics for two-sided assembly line balancing problem, *Computers & Operations Research*, Vol. 84, 146-161, doi:10.1016/j.cor.2017.03.002

[5] Kucukkoc, I.; Li, Z.; Karaoglan, A. D.; Zhang, D. Z. (2018). Balancing of mixed-model two-sided assembly lines with underground workstations: A mathematical model and ant colony optimization algorithm, *International Journal of Production Economics*, Vol. 205, 228-243, doi:10.1016/j.ijpe.2018.08.009

[6] Lei, D.; Guo, X. (2016). Variable neighborhood search for the second type of two-sided assembly line balancing problem, *Computers & Operations Research*, Vol. 72, 183-188, doi:10.1016/j.cor.2016.03.003

[7] Kucukkoc, I.; Zhang, D. Z. (2016). Mixed-model parallel two-sided assembly line balancing problem: A flexible agent-based ant colony optimization approach, *Computers & Industrial Engineering*, Vol. 97, 58-72, doi:10.1016/j.cie.2016.04.001

[8] Tapkan, P.; Özbakır, L.; Baykasoğlu, A. (2016). Bee algorithms for parallel two-sided assembly line balancing problem with walking times, *Applied Soft Computing*, Vol. 39, 275-291, doi:10.1016/j.asoc.2015.11.017

[9] Kucukkoc, I.; Zhang, D. Z. (2015). Type-E parallel two-sided assembly line balancing problem: Mathematical model and ant colony optimisation based approach with optimised parameters, *Computers & Industrial Engineering*, Vol. 84, 56-69, doi:10.1016/j.cie.2014.12.037

[10] Yuan, B.; Zhang, C.; Shao, X.; Jiang, Z. (2015). An effective hybrid honey bee mating optimization algorithm for balancing mixed-model two-sided assembly lines, *Computers & Operations Research*, Vol. 53, 32-41, doi:10.1016/j.cor.2014.07.011

[11] Tuncel, G.; Aydin, D. (2014). Two-sided assembly line balancing using teaching-learning based optimization algorithm, *Computers & Industrial Engineering*, Vol. 74, 291-299, doi:10.1016/j.cie.2014.06.006

[12] Purnomo, H. D.; Wee, H.-M. (2014). Maximizing production rate and workload balancing in a two-sided assembly line using Harmony Search, *Computers & Industrial Engineering*, Vol. 76, 222-230, doi:10.1016/j.cie.2014.07.010

[13] Chen, W.; Hao, Y. F. (2018). Genetic algorithm-based design and simulation of manufacturing flow shop scheduling, *International Journal of Simulation Modelling*, Vol. 17, No. 4, 702-711, doi:10.2507/IJSIMM17(4)CO17

[14] Hu, H. X.; Lei, W. X.; Gao, X.; Zhang, Y. (2018). Job-shop scheduling problem based on improved cuckoo search algorithm, *International Journal of Simulation Modelling*, Vol. 17, No. 2, 337-346, doi:10.2507/IJSIMM17(2)CO8

[15] Yang, F.; Ye, C. M.; Shi, M. H. (2018). A hybrid grey cuckoo search algorithm for job-shop scheduling problems under fuzzy conditions, *Advances in Production Engineering & Management*, Vol. 13, No. 3, 254-266, doi:10.14743/apem2018.3.288

[16] Ma, D. Y.; He, C. H.; Wang, S. Q.; Han, X. M.; Shi, X. H. (2018). Solving fuzzy flexible job shop scheduling problem based on fuzzy satisfaction rate and differential evolution, *Advances in Production Engineering & Management*, Vol. 13, No. 1, 44-56, doi:10.14743/apem2018.1.272

[17] Özcan, U. (2010). Balancing stochastic two-sided assembly lines: A chance-constrained, piecewise-linear, mixed integer program and a simulated annealing algorithm, *European Journal of Operational Research*, Vol. 205, No. 1, 81-97, doi:10.1016/j.ejor.2009.11.033