Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 2785278, 13 pages
https://doi.org/10.1155/2020/2785278

Research Article

An Optimization Model for Assembly Line Balancing Problem with Uncertain Cycle Time

Yong Cao,[1] Yuan Li,[1] Qinghua Liu,[2] and Jie Zhang[1]

[1]*School of Mechanical Engineering, Northwestern Polytechnical University, Xi'an 710048, China*
[2]*AVIC Shaanxi Aircraft Industry (Group) Company, Ltd., Hanzhong, Shaanxi 723200, China*

Correspondence should be addressed to Yuan Li; liyuannwpu@126.com

Received 27 November 2019; Revised 4 May 2020; Accepted 1 June 2020; Published 25 June 2020

Academic Editor: Pietro Bia

With the drastic change in the market, the assembly line is susceptible to some uncertainties. This study introduces the uncertain cycle time to the assembly line balancing problem (ALBP) and explores its impact. Firstly, we improve the traditional precedence graph to express the precedence, spatial, and incompatible constraints between assembly tasks, which makes ALBP more realistic. Secondly, we establish the assembly line balancing model under an uncertain cycle time, which is defined as an interval whose size can be adjusted according to the level of uncertainty. The objective of the model was to minimize the number of stations and the cycle time. Thirdly, we integrate the operator's skill level into the model, and a multipopulation genetic algorithm is used to solve it. The method proposed in this study is verified by several test problems of different sizes. The results show that when the cycle time is uncertain, the proposed method can be used to obtain more reasonable results.

1. Introduction

Assembly lines are widely used in mass production, and they determine a number of indicators, such as production efficiency, production cost, and enterprise efficiency. To ensure the performance and productivity of assembly plants, there exists a well-known decision problem called the assembly line balancing problem (ALBP). The scientific solution to this problem has been a concern of researchers since 1965. ALBP allocates all assembly tasks to a certain number of stations while optimizing one or more objectives (e.g., number of workstations or cycle time) without violating certain technological, operational, and spatial constraints [1, 2]. Because of the wide usage of assembly lines, many survey papers appear in the literature [2–6]. ALBP can generally be divided into two classes: the simple assembly line balancing problem (SALBP) and the generalized assembly line balancing problem (GALBP) [1]. SALBP assumes that stations are arranged in a straight line, each assembly task has a defined time, and the assembly line produces just one product. There are four types of SALBP. SALBP-I aims to minimize the number of stations for a given cycle time of the assembly line. SALBP-II aims to minimize the cycle time for a given number of stations. SALBP-E aims to maximize the efficiency of the assembly line, where neither the number of stations nor the cycle time is given. In SALBP-F, the number of stations and the cycle time are given, and the problem is to evaluate whether the assembly line is feasible. GALBP includes all of the other problems, such as multiproduct hybrid assembly lines, U-shaped assembly lines, parallel workstations, and stochastic task durations. GALBP covers a wider area than SALBP. We focus on SALBP-I because it has been widely used in practice to minimize the number of stations for a given cycle time. A guide for the class of SALBP-1 can be seen in the literature [7].

With the diversification, differentiation, and intensification of the market, assembly lines face many uncertainties, such as task time, demand, and machine failure [8–10]. These will easily disrupt the normal operation of an assembly line, cause failure of a production plan, and reduce customer satisfaction. To this end, the study of ALBP in an uncertain environment is of great significance to manufacturing companies. Assembly lines can have deterministic or stochastic cycle times [11]. The cycle time is deterministic in traditional SALBP-I [1, 12], and there is a significant gap between research and real-world applications. Researchers and industrial practitioners are working to solve SALBP-I as more realistic assembly line settings and features are considered. This study deals with SALBP-I by considering an uncertain cycle time as an important characteristic of production systems.

Variation in demand has a significant impact on assembly performance. Demand for a new product can be uncertain; hence, the quantity to be produced is unknown, and the cycle time of assembly lines will be uncertain. In real-life assembly, it is easy to establish a relationship between the quantity to be produced and the cycle time of the assembly line. Therefore, to cope with uncertain demand, we must integrate an uncertain cycle time into traditional ALBP. Uncertainty in an assembly line can be represented using either a probability or a fuzzy set theory [13–15]. Probability theory enables the calculation of the chance of an event's occurrence, and it requires information from observations associated with the event. Fuzzy set theory focuses on imprecise phenomena based on fuzzy mathematics. To encode the uncertainty, it must choose reasonable membership functions according to the knowledge of the event. Observations or knowledge of an event may be rare in practice, especially in the initial stages of product development. For example, the cycle time may be just an interval in the early design phase, and only its bounds can be predicted. An optimization model for ALBP with an interval cycle time can cope with this situation.

This study proposes two mathematical models. SALBP-I with certain cycle time aims to minimize the number of stations. SALBP-I with interval cycle time adds an additional objective to traditional SALBP-I. After that, SALBP-I with interval cycle time may be a variant of SALBP-E, which aims to optimize both the number of stations and the cycle time. The two models are implemented with a multipopulation genetic algorithm. The remainder of this study is organized as follows: Section 2 reviews the literature of ALBP with uncertainties. Section 3 analyzes and represents the constraints in SALBP. Mathematical models under certain and uncertain cycle times are presented in Section 4. Section 5 discusses implementation of the models. Section 6 presents computational results and their analysis. Concluding remarks and future work directions are given in Section 7.

2. Literature Review

Uncertainties affect the evaluation of a company's production capacity, which will influence its decisions. Hence, to consider the uncertainty of the production environment is of great significance to establish a reliable assembly line. Researchers have studied the influence of uncertainties on ALBP.

Mirzaei et al. [15] studied objective functions to minimize the number of stations, total tool purchasing cost, and cycle time. The cost of each tool and the task times are assumed to be trapezoidal fuzzy numbers to reflect their uncertainty. Hazr and Dolgui [16] integrated the interval uncertainty of operation times into ALBP and proposed a decomposition-based exact algorithm with enhancement strategies to solve large-scale ALBP. The objective is to minimize the cycle time. Gurevsky et al. [17] tackled the balancing problem for straight assembly lines where task times are not known exactly or are given by intervals. The objective is to assign tasks so as to minimize the number of workstations. A breadth-first search procedure finds the best and worst solution. Liu et al. [18] introduced the fuzzy knowledge method to ALBP to address the problem that the distribution of task time is unknown in advance. An approximate mixed-integer second-order cone programming (MI-SOCP) model was proposed to solve the problem. Li et al. [19] proposed a multiobjective optimization model and neighborhood search algorithm for ALBP considering the uncertainty of task time. The objectives are to maximize assembly line reliability and minimize cycle time. This method is suitable for situations of uncertain task times or small data. Li et al. [20] used uncertainty theory and introduced the uncertain task time as a constraint in ALBP-I. The problem was solved using a newly developed restart neighborhood search method. Dong et al. [21] established the assembly line balancing model considering interval task time and spatial constraints. The study compromised between line efficiency and equipment cost. A biobjective chance-constrained mixed 0-1 programming model was developed to simultaneously minimize the cycle time and equipment cost. Li et al. [22] proposed an assembly line balancing model considering uncertain production capacity, aiming to make the cycle time as close as possible to the normal value. The objective was to satisfy the demand in each possible scenario with the minimum labor cost for both normal shifts and overtime work. A heuristic algorithm was developed to quickly find a feasible solution. Chica et al. [23] proposed an approach to evaluate the robustness of a hybrid assembly line in terms of uncertainty of demand. Samouei et al. [24] explored ALBP considering uncertain requirements for semiautomated assembly lines, aiming to find the task allocation with the lowest configuration cost. Pereira et al. [25] proposed a robust formulation to handle uncertain operation times. A branch-and-bound algorithm was proposed to solve the model. The objective included minimizing the number of stations and the cost. It was found that an assembly line can be protected from moderate levels of uncertainty at the expense of small quantities of additional resources. Fathi et al. [26] considered stochastic task times and zoning constraints and integrated the minimization of the maximum of stations' mean time and variance into ALBP. A mixed-integer programming (MIP) model solved ALBP. Zhang et al. [27] considered uncertain task times in type-II mixed-model ALBP. A robust optimization model was formulated to hedge against uncertainty, and a hybrid genetic algorithm was proposed to solve the problem. Fisel et al. [28] considered the potential need for adaptation in ALBP and used mixed-integer linear optimization to solve the problem. The study focused on production cost, as well as flexibility and changeability of the system. Babazadeh et al. [29, 30] proposed a biobjective fuzzy mixed-integer linear programming model to represent uncertainty in task-processing times, considering the minimization of the number of stations and the cycle time. They proposed an enhanced NSGA-II algorithm with a new repairing mechanism to solve the model [31, 32].

Analysis of the literature indicates that while uncertainties in the production environment have been considered, problems remain to be solved. Research has focused primarily on the uncertainty of task times, and uncertainty of cycle times must still be discussed. This uncertainty can occur in all phases of the design, use, and evaluation of an assembly line. The cycle time may be an interval predicted in the design phase, or it may vary with orders and delivery dates. We introduce an interval of the cycle time to ALBP. An efficient algorithm to solve ALBP is of broad concern in industrial and academic circles. The scale of ALBP is generally large; hence, an efficient solution is required. We introduce a multipopulation genetic algorithm to the assembly line balancing problem.

3. Preprocessing of Assembly Tasks

3.1. Representation of Constraints between Assembly Tasks. The assembly processes of a product are decomposed into several assembly tasks, each comprising a series of actions that cannot or need not be divided further. The essence of ALBP is to assign a series of assembly tasks to a certain number of stations, while satisfying precedence constraints and specified cycle times. Precedence constraints are the most basic and important type of constraints in ALBP, reflecting the order of assembly tasks. Intuitively, assembly tasks and precedence constraints can be represented by a directed graph, called a precedence graph [33, 34]. In Figure 1, the circle represents the assembly task, and the straight line with an arrow indicates the precedence constraints. For products with complex structures (such as airplanes and cars), precedence constraints cannot fully express the relationships between assembly tasks. Spatial constraints and incompatible constraints between assembly tasks also must be considered. We introduce spatial constraints and incompatible constraints on the basis of the traditional precedence graph as follows.

There are two meanings of spatial constraints. Multiple associated assembly tasks must be completed on the same station, and there is a maximum number of assembly tasks that can be allocated to one station. In actual production, assembly tasks that have cooperative relationships or are of the same kind are usually assigned to the same station to improve efficiency. At the same time, the number of assembly tasks that can be assigned to one station is limited to ensure that each operator has sufficient space. In Figure 1, the ellipse represents the spatial constraint, and the assembly tasks within it must be assigned to the same station. The figure does not show the maximum number of tasks that can be allocated to a station; this will be quantified in the assembly line balancing model.

Incompatible constraints mean that some assembly tasks cannot be assigned to the same station. For example, some tasks to measure quality usually cannot be placed in the same location as other tasks. In Figure 1, assembly tasks within dashed circles of the same color cannot be assigned to the same station.

3.2. Topological Sorting for Precedence Graph. Assembly tasks are arranged in a linear sequence on an actual assembly line, and ALBP-I is a reasonable division of assembly tasks. To guarantee the precedence constraints between assembly tasks, a topological sorting algorithm preprocesses them in the precedence graph to form a topological order.

The precedence graph is a directed acyclic graph. Nodes represent assembly tasks, and directed edges represent the sequences between them. The precedence graph is represented as follows:

$$\mathrm{PG} = \left\{V, E, f_{\text{spatial}}, f_{\text{incompatible}}\right\}, \tag{1}$$

where V and E represent the sets of vertices and edges, respectively. If there is a directed edge $e = \langle a, b\rangle$, then task a is the immediate predecessor of task b. If there is a directed path $\langle a, \cdots, b\rangle \in V$, then task a is the indirect predecessor of task b. f_{spatial} represents spatial constraints. If $f_{\text{spatial}}(V_s) = 1$, then there is a spatial constraint for the assembly tasks in the vertex set V_s; otherwise, there is not. $f_{\text{incompatible}}$ represents incompatible constraints. If $f_{\text{incompatible}}(V_c) = 1$, then the assembly task in vertex set V_c satisfies the incompatible constraint; otherwise, there is no such constraint; $V_s, V_c \in V$.

Through the topological sorting algorithm, a linear sequence $v_1, v_2, \cdots, v_n$ can be obtained, where n is the number of vertices in the precedence graph. This sequence is called the topological order (TO), and it reflects the sequence of assembly tasks. For example, if task v_1 is the immediate or indirect predecessor of task v_2, then task v_1 will be in front of task v_2 in the TO; that is, task v_1 will be assigned to the station before v_2. It should be noted that if the vertices in the vertex set V_s satisfy the spatial constraint, then V_s will be considered as a whole in topological sorting. The topological sorting algorithm constantly searches for vertices without predecessors. A vertex with no predecessor will be inserted in the TO and removed from the precedence graph. The algorithm is completed when all of the vertices are in the TO. If the TO of all of the vertices is not available, then the assembly processes represented by the precedence graph are infeasible. According to [35], the procedure of the topological sorting algorithm is presented in Algorithm 1.

The advantage of the topological sorting algorithm is to maximize the efficiency of task assignment. It can generate multiple topological orders. Ten topological orders of Figure 1, in which the precedence constraints between tasks can always be satisfied, are presented in Table 1 . Therefore, the use of these topological orders may improve the efficiency of ALBP.

4. Mathematical Modeling

Traditional ALBP-I aims to minimize the number of stations and corresponding task assignments and meet the fixed cycle time. It must be changed when the cycle time is uncertain. In the assembly line design phase, a company's forecast order

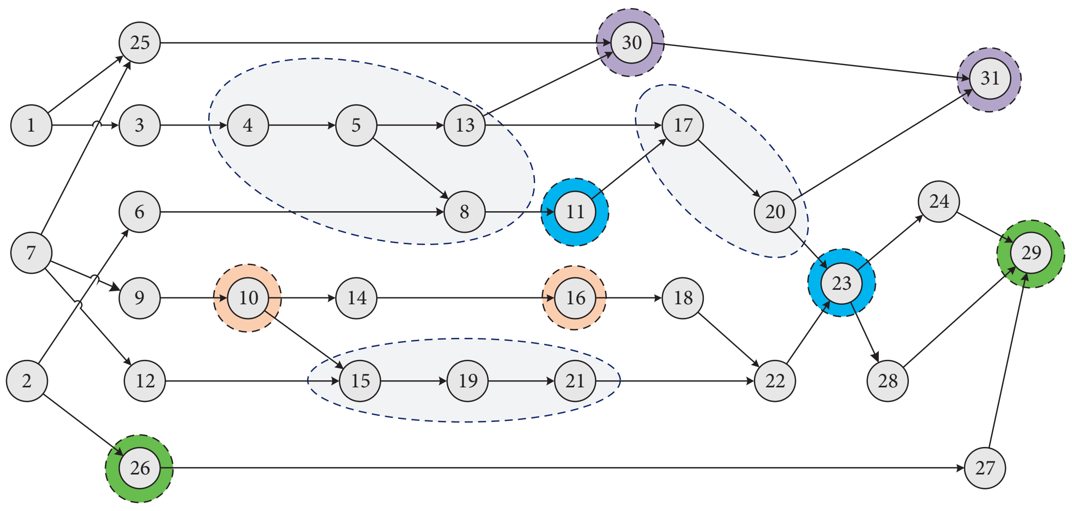

Figure 1: Precedence graph.

```
Procedure: Topological Sorting
Input: Precedence graph; Let $n$ be the number of vertices; Let $T = \varnothing$ be the TO of the precedence graph.
Output: A nonempty TO ($T$).
  For $i = 1$ to $n$ do
    Begin
      If every vertex has a predecessor
        Then the precedence graph is infeasible stop;
      Else pick a vertex $v_i$ which has no predecessors;
      $T = T \cup v_i$;
      Delete $v_i$ and all edges leading out of $v_i$ from precedence graph
    End
  End
```

Algorithm 1: Pseudocode of topological sorting.

Table 1: Ten topological orders of Figure 1.

No.	Topological orders
1	7, 2, 26, 6, 27, 1, 9, 10, 14, 3, 25, 4, 12, 15, 19, 5, 21, 13, 8, 11, 16, 30, 18, 17, 20, 22, 23, 28, 31, 24, 29
2	2, 26, 7, 12, 6, 9, 10, 15, 1, 27, 25, 19, 14, 21, 3, 4, 5, 13, 8, 30, 11, 16, 18, 17, 22, 20, 31, 23, 28, 24, 29
3	1, 7, 3, 25, 12, 9, 10, 4, 2, 5, 14, 13, 15, 6, 30, 19, 8, 16, 11, 26, 17, 27, 18, 20, 31, 21, 22, 23, 28, 24, 29
4	7, 9, 12, 1, 25, 2, 10, 6, 26, 15, 14, 19, 27, 21, 3, 4, 5, 13, 8, 16, 11, 18, 30, 17, 22, 20, 23, 31, 24, 28, 29
5	2, 26, 27, 7, 12, 1, 25, 9, 3, 10, 6, 14, 15, 19, 21, 4, 5, 13, 30, 8, 16, 18, 11, 17, 20, 31, 22, 23, 28, 24, 29
6	1, 7, 9, 3, 2, 4, 5, 26, 12, 10, 15, 27, 14, 19, 6, 8, 11, 21, 13, 25, 17, 16, 18, 22, 20, 30, 23, 28, 24, 29, 31
7	7, 12, 1, 3, 4, 2, 9, 5, 6, 10, 25, 15, 26, 19, 27, 14, 13, 30, 21, 8, 11, 17, 20, 31, 16, 18, 22, 23, 24, 28, 29
8	1, 3, 4, 5, 2, 6, 26, 8, 11, 27, 7, 9, 10, 14, 12, 15, 25, 13, 17, 16, 19, 20, 18, 21, 30, 31, 22, 23, 24, 28, 29
9	1, 2, 6, 26, 27, 3, 7, 25, 4, 9, 5, 10, 13, 14, 30, 12, 8, 15, 11, 17, 20, 31, 16, 19, 18, 21, 22, 23, 28, 24, 29
10	2, 1, 3, 7, 25, 9, 10, 26, 12, 15, 27, 4, 14, 5, 19, 13, 6, 21, 30, 8, 16, 11, 17, 20, 31, 18, 22, 23, 24, 28, 29

quantity may be in an interval; hence, it is not fixed, and the cycle time of the assembly line is also in an interval. In this case, ALBP-I must be expanded to what we call EALBP-I. The mathematical models of ALBP-I and EALBP-I are established below, and their differences and similarities are discussed.

4.1. Notations.

The symbols used in ALBP-I and EALBP-I are defined in Table 2.

4.2. Modeling under Certain Cycle Time.

ALBP-I assumes that the cycle time is determined to minimize the number of stations; that is, minimal resources, such as equipment and operators, are required. The operator's skill level is an important factor affecting the cycle time of an assembly line, and it can be dynamically adjusted and configured during operation. Therefore, in assembly line design, there is a need to combine the task assignment with the operator's skill level to obtain the optimal solution [36]. The mathematical model of ALBP-I is as follows:

Table 2: Notation used in mathematical models.

Symbol	Definition
Index	
i, j	Task index
k	Station index
n	Number of tasks
p	Operator's skill level; 1, 2, and 3, respectively, indicate low, medium, and high skill
Parameter	
C	Cycle time
m	Number of stations
$C_{\max}$	Maximum cycle time
$C_{\min}$	Minimum cycle time
t_{ip}	Time required to complete task i when operator's skill level is p
P_{ij}	Precedence constraints between tasks i and j. $P_{ij} = 1$ indicates precedence constraints; otherwise, there is no precedence constraint.
SC_{ij}^{+}	Spatial constraint between tasks i and j. $SC_{ij}^{+} = 1$ indicates tasks must be assigned to the same station; otherwise, this is not required.
SC_{ij}^{-}	Incompatible constraint between tasks i and j. $SC_{ij}^{-} = 1$ indicates the tasks cannot be assigned to the same station; otherwise, they can.
R_p	Number of operators with skill level p
$O_{\max}$	Maximum number of tasks that can be accommodated on a station
Variable	
x_{ik}	Binary variable. $x_{ik} = 1$ indicates that task i is assigned to station k.
r_{ip}	Binary variable. $r_{ip} = 1$ indicates that operators with skill level p are assigned to task i.
y_k	Binary variable. $y_k = 1$ indicates that station k is enabled.

$$\min \sum_{k=1}^{m} y_k, \tag{2}$$

subject to

$$\sum_{i=1}^{n} x_{ik} r_{ip} t_{ip} \le C y_k, \quad \forall k = 1, 2, \cdots, m, \ \forall p = 1, 2, 3, \tag{3}$$

$$\sum_{k=1}^{m} x_{ik} = 1, \quad \forall i = 1, 2, \cdots, n, \tag{4}$$

$$\sum_{k=1}^{m} k \times x_{ik} \le \sum_{k=1}^{m} k \times x_{jk}, \quad \forall P_{ij} = 1, \ \forall i = 1, 2, \cdots, n, \quad \forall j = 1, 2, \cdots, n, \tag{5}$$

$$y_{k+1} \le y_k, \quad \forall k = 1, 2, \cdots, m, \tag{6}$$

$$x_{ik} = x_{jk}, \quad \forall (i, j) \in SC_{ij}^{+}, \ \forall k = 1, 2, \cdots, m, \tag{7}$$

$$x_{ik} + x_{jk} \le 1, \quad \forall (i, j) \in SC_{ij}^{-}, \ \forall k = 1, 2, \cdots, m, \tag{8}$$

$$\sum_{i=1}^{n} x_{ik} \le O_{\max}, \quad \forall k = 1, 2, \cdots, m, \tag{9}$$

$$\sum_{i=1}^{n} r_{ip} \le R_p, \quad \forall p = 1, 2, \tag{10}$$

$$x_{ik} \in \{0, 1\}, \quad \forall i = 1, 2, \cdots, n, \ \forall k = 1, 2, \cdots, m, \tag{11}$$

$$r_{ip} \in \{0, 1\}, \quad \forall i = 1, 2, \cdots, n, \ \forall p = 1, 2, 3, \tag{12}$$

$$y_k \in \{0, 1\}, \quad \forall k = 1, 2, \cdots, m. \tag{13}$$

Constraint (3) indicates that the total task time of each station does not exceed the cycle time C. Constraint (4) indicates that each task can only be assigned to one station. Constraint (5) represents the precedence constraints between tasks. Constraint (6) indicates that each station must be continuously enabled; that is, if station k is enabled, then station $k-1$ must be enabled; otherwise, there will be a discontinuity in the station index. Constraint (7) indicates that a set of tasks need to be assigned to the same station. Constraint (8) indicates that there are incompatible constraints between tasks. Constraint (9) indicates the maximum number of tasks that can be assigned to each station. Constraint (10) indicates that the number of operators assigned to a station should be less than the total number of available operators. Constraints (11)–(13) indicate that x_{ik}, r_{kp}, and y_k are binary variables.

4.3. Modeling under Uncertain Cycle Time.

4.3. Modeling under Uncertain Cycle Time. A dynamic market shows dynamic changes in the demand for products; that is, order quantities and delivery dates may be adjusted as the market changes. These differences require assembly lines to have different cycle times. To gain a competitive advantage, a company must respond promptly to these changes. To build a new assembly line to meet a new cycle time is costly. Therefore, such changes must be considered when designing an assembly line.

Companies usually evaluate order quantities and forecast maxima and minima when they construct an assembly line. In this paper, the order quantity interval is converted to a cycle time interval, which is introduced in ALBP-I. Assume that C^0 is the nominal cycle time and D is the maximum change in the cycle time. The cycle time interval can be obtained as follows:

$$C \in \left[C^0 - D, C^0 + D\right]. \tag{14}$$

The level of uncertainty α is introduced, and the following two constraints are obtained:

$$C \le C^0 + \alpha D, \tag{15}$$

$$C \ge C^0 - \alpha D, \tag{16}$$

where $\alpha \in (0, 1]$. In the design phase of an assembly line, decision makers can choose a reasonable value of α to cope with uncertainty in the cycle time.

Our goal is to find the most scientific number of stations and task assignments during the design phase of an assembly line, so that a company can best cope with the uncertainty of the cycle time. Let $C_{\max} = C^0 - \alpha D$ and $C_{\min} = C^0 + \alpha D$. We add the following constraints to the mathematical model of ALBP-I:

$$C_{\min} - C \leq 0, \quad (17)$$

$$C - C_{\max} \leq 0. \quad (18)$$

The objective must then be adjusted accordingly. If (2) is used as the objective, then the calculation will always satisfy the maximum cycle time. Therefore, a new objective is proposed:

$$\min \sum_{k=1}^{m} y_k \text{ and } \min \frac{C}{C_{\min}}. \quad (19)$$

A weighted summation method is used to convert (19) to a single objective:

$$\min\left(w_1 \times \sum_{k=1}^{m} y_k + w_2 \times \frac{C}{C_{\min}}\right), \quad (20)$$

where w_1 and w_2 are weight coefficients. According to (19), w_2 should be greater than w_1, which reflects the influence of the cycle time interval on the objective.

5. Implementation of Multipopulation Genetic Algorithm

Individual properties in the population tend to be consistent after several iterations of the genetic algorithm, which causes convergence to a local solution. We propose a multipopulation genetic algorithm (MGA) for ALBP to improve the genetic algorithm.

Figure 2 shows the flowchart of the two-population genetic algorithm, whose basic idea is to replace the single population of the genetic algorithm with two subpopulations which evolve independently and then make an individual exchange according to rules until the optimal solution is found. The multipopulation genetic algorithm is characterized by faster convergence while maintaining population diversity. Through the extension of Figure 2, the flowchart of multipopulation genetic algorithm can be easily obtained. We introduce the main steps to solve EALBP-I using the multipopulation genetic algorithm:

Step 1: Set up parameters. *SubP* is the number of subpopulations, *PS* is the number of individuals in each subpopulation, *PM* is the probability of mutation, *PC* is the probability of crossover, and *MR* is the migration rate.

Step 2: Generate initial population. The chromosome structure has three layers. The first is the assembly task, with length *n*. The second is the cycle time, with code length 1. The third is the operator's skill level for each assembly task, with length *n*. To ensure that the generated initial population always satisfies the precedence constraint, the topological sorting algorithm proposed in Section 3.2 is used to generate the initial population.

Step 3: Calculate fitness values. The calculation of fitness for MGA is performed according to (19), with fitness function:

$$F_i = \frac{1}{f_i}, \quad (21)$$

where F_i is the fitness value of the i^{th} individual and f_i is the value of the objective function calculated by the i^{th} individual.

Step 4: Selection operation. We use roulette to select individuals; i.e., an individual's probability of selection is determined by the ratio of the individual's fitness to the total fitness of all individuals.

Step 5: Crossover operation. A two-point crossover method is adopted. Using the Buxey example [12], as illustrated in Figure 3(a), two crossover points are randomly generated, as defined by Cross_Point_1 and Cross_Point_2. The two parent chromosomes are divided into three parts: head, body, and tail. The gene segment, Body_1, in parent chromosome 1 can be found in parent chromosome 2. These genes are sorted by the gene sequence in parent chromosome 2. A new gene segment, Body_1_new, is generated, as illustrated in Figure 3(b), to replace Body_1 in parent chromosome 1. A new chromosome, child 1, is formed, as shown in Figure 3(c). By the same method, we can obtain another new chromosome, child 2, as illustrated in Figure 3(c). This crossover operation has the advantage that all of the tasks meet the sequence constraints. The child chromosomes are evaluated after crossover. If the fitness of a child is improved, then it is retained, and otherwise it is abandoned.

Step 6: Mutation operation. We use a single-point mutation method; that is, a code in the chromosome is randomly selected, and the mutation of the selected code is performed according to a certain probability.

Step 7: Population migration. This is performed according to a randomly generated migration rate. When the generated migration rate exceeds a given value, population migration is performed.

6. Results and Analysis

We used basic data from https://assembly-line-balancing.de/ to verify the validity of the proposed methods. The basic data were adapted according to the needs of this paper. Cases with task numbers 11, 21, 29, 35, 83, and 111 were selected to verify the adaptability of the proposed methods. For each problem, the operator's skill levels (low, medium, high) were considered. The higher the skill level, the shorter the task time. The task time in the basic data was set to that of operators with high skill. This was multiplied by a factor of 1.2 or 1.1 to obtain the task time of operators with low and medium skill, respectively. At the same time, the number of operators with low skill, medium skill, and high skill was set to be 50% of the number of tasks, respectively. $O_{\max} = 15$. Referring to [37–39], MGA had the following initial

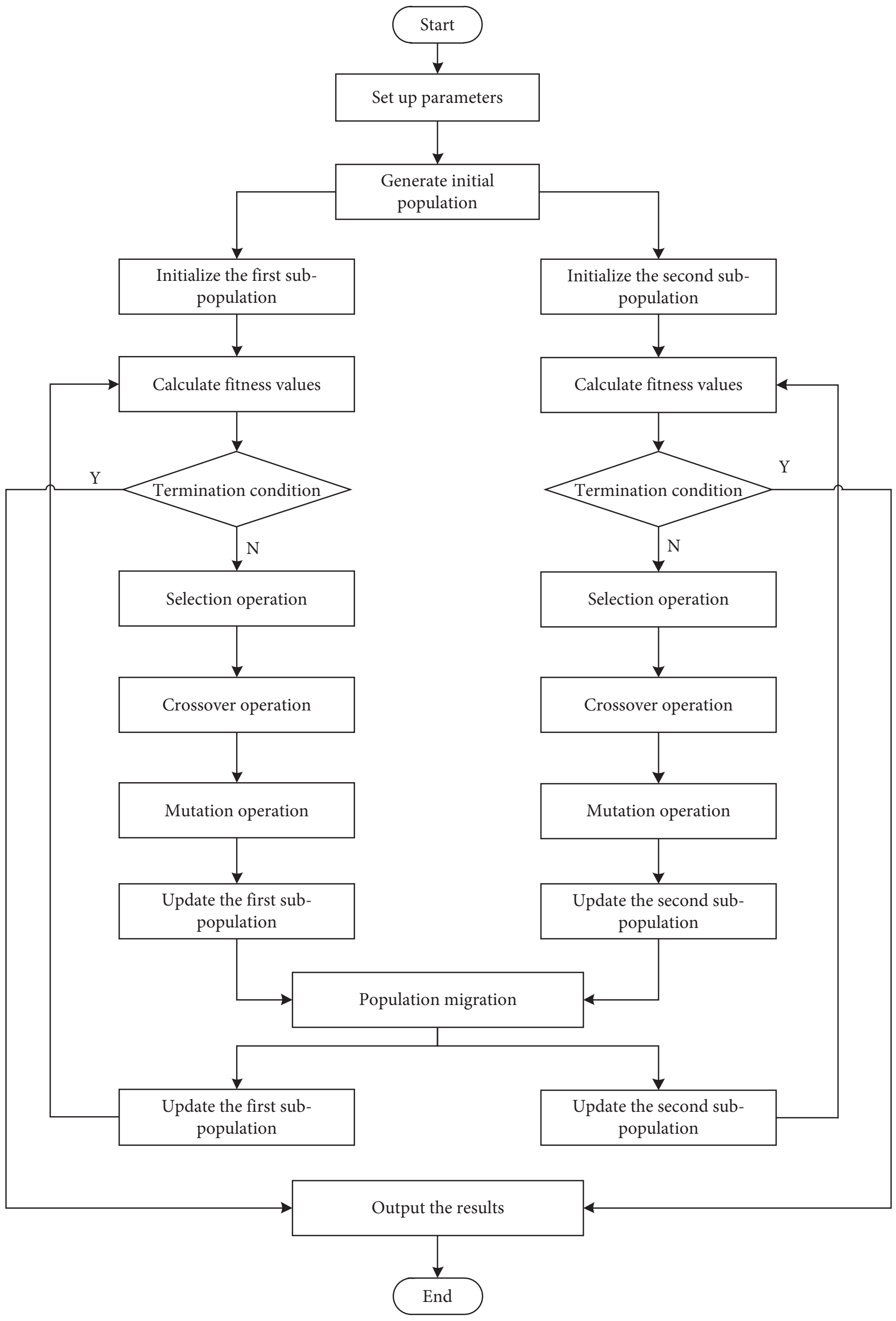

Figure 2: Flowchart of multipopulation genetic algorithm.

configuration: $SubP = 5$, $PS = 50$, $PM = 0.1$, $PC = 0.8$, and $MR = 0.1$.

The level of uncertainty α was assumed to vary within a certain range. The α level determined the cycle time interval. The cycle time was assumed to change within $C^0(1 \pm \alpha)$. The number of stations at different α levels was calculated, as displayed in Table 3. Table 2 provides four types of results, namely, robust, regular, best, and worst. It should be noted that the robust results considered the uncertainty in cycle time, expressed as an interval. The regular results considered the normal cycle time. The best results were calculated when the cycle time reached $C^0 + C^0\alpha$. The worst result was calculated when the cycle time reached $C^0 - C^0\alpha$. As can be seen in Table 3, when the number of tasks was 11, 21, or 29, the number of stations in a robust situation was consistent with that in a regular situation for different α levels. When the number of tasks was 35, the number of stations in a robust situation was slightly larger than that in a regular

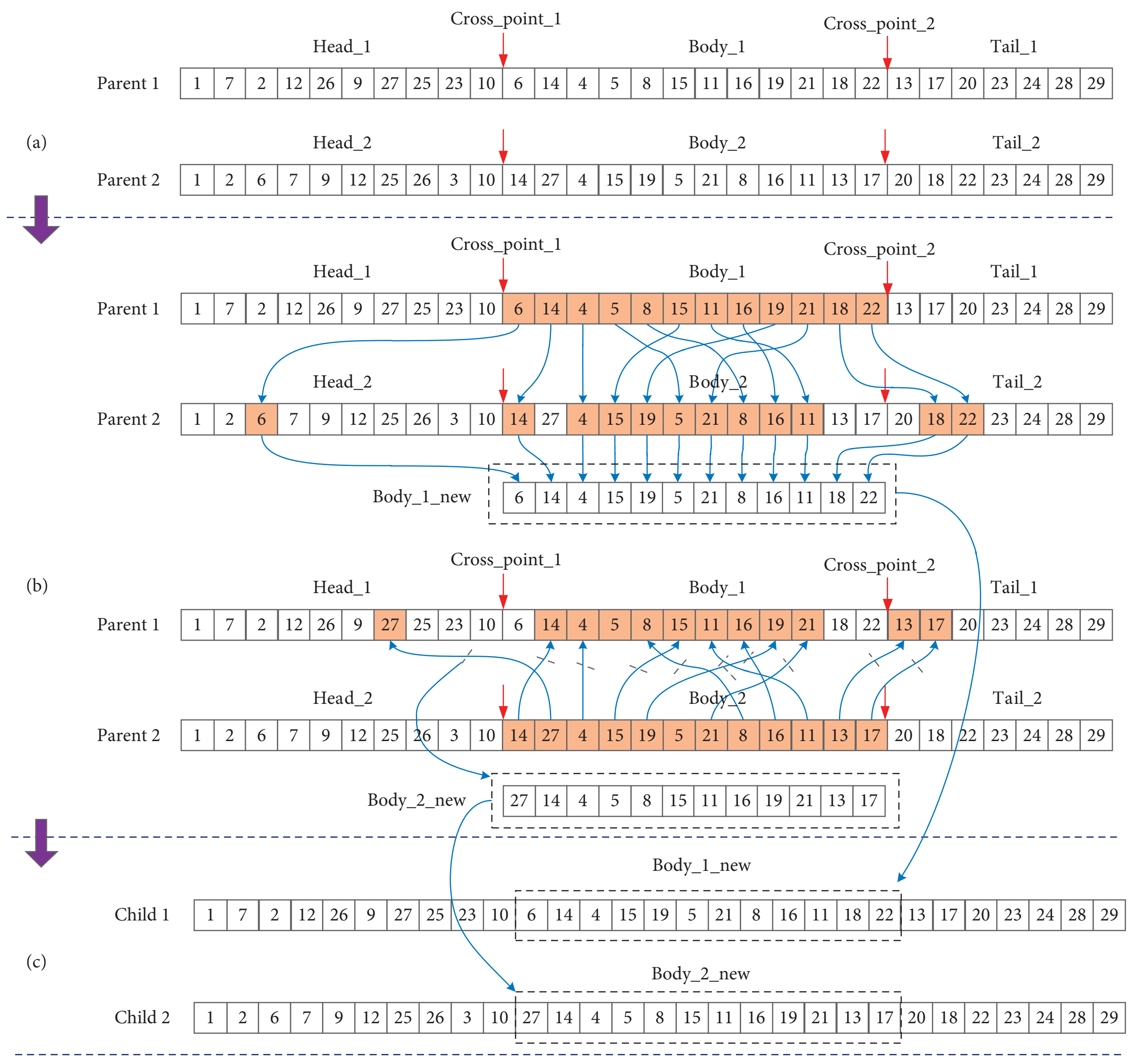
(a)
Cross_point_1
Cross_point_2
Head_1
Body_1
Tail_1
Parent 1: 1 7 2 12 26 9 27 25 23 10 6 14 4 5 8 15 11 16 19 21 18 22 13 17 20 23 24 28 29
Head_2
Body_2
Tail_2
Parent 2: 1 2 6 7 9 12 25 26 3 10 14 27 4 15 19 5 21 8 16 11 13 17 20 18 22 23 24 28 29
(b)
Body_1_new: 6 14 4 15 19 5 21 8 16 11 18 22
Body_2_new: 27 14 4 5 8 15 11 16 19 21 13 17
(c)
Child 1: 1 7 2 12 26 9 27 25 23 10 6 14 4 15 19 5 21 8 16 11 18 22 13 17 20 23 24 28 29
Child 2: 1 2 6 7 9 12 25 26 3 10 27 14 4 5 8 15 11 16 19 21 13 17 20 18 22 23 24 28 29

FIGURE 3: Example of crossover: (a) selection of crossover point, (b) crossover operation and (c) results.

situation for different α levels. When the number of tasks was 83 or 111, as α increased, the number of stations in a robust situation was equal to or larger than that in a regular situation. That is, the lower bound of the cycle time interval had an increasing influence on the number of stations as the number of tasks and α increased. As is easily seen from Table 3, as α increased, the number of stations in the best situation gradually decreased, which was due to the gradual increase of $C_{\max}$. With the increase of α, the number of stations in the worst situation increased, which was due to the gradual decrease of $C_{\min}$. However, when $\alpha \geq 0.4$, the number of stations in the worst situation may be unsolvable.

After introducing the level of uncertainty α, the cycle time becomes a variable that can be changed. Different numbers of stations, as well as different task and operator assignments, will output different cycle times. The cycle time at different α levels is calculated for each situation in Table 3. As shown in Table 4, the cycle time in the regular situation is basically consistent with C^0, because the influence of uncertainty is not considered. For the six examples in Table 4, the cycle time of the robust situation mostly decreases as α increases. Hence, as α increases, the cycle time calculated by EALBP-I is closer to $C^0 - C^0\alpha$, as shown in Figures 4–7. For the best situation, the cycle time will increase with α, which is caused by the increase of $C^0 + C^0\alpha$. For the worst situation, the cycle time will decrease as α increases, because of the decrease of $C^0 - C^0\alpha$. However, when $\alpha \geq 0.4$, the cycle time in the worst situation may be unsolvable, which is consistent with the results in Table 3.

Table 3: Number of stations for different problems and α levels.

Number of tasks	α	C^0	Number of stations			
			Robust	Regular	Best	Worst
	0.1	10	6	6	5	6
	0.2	10	6	6	5	8
11	0.3	10	6	6	4	8
	0.4	10	6	6	4	—
	0.5	10	6	6	4	—
	0.1	20	6	6	6	7
	0.2	20	6	6	5	8
21	0.3	20	6	6	5	9
	0.4	20	6	6	4	—
	0.5	20	7	6	4	—
	0.1	40	10	10	9	11
	0.2	40	10	10	8	12
29	0.3	40	10	10	8	13
	0.4	40	10	10	7	—
	0.5	40	10	10	7	—
	0.1	50	12	11	10	13
	0.2	50	12	11	9	—
35	0.3	50	12	11	9	—
	0.4	50	12	11	8	—
	0.5	50	12	11	8	—
	0.1	8000	11	11	10	13
	0.2	8000	12	11	9	14
83	0.3	8000	13	11	9	15
	0.4	8000	14	11	8	—
	0.5	8000	14	11	8	—
	0.1	25000	7	7	7	8
	0.2	25000	7	7	6	9
111	0.3	25000	8	7	6	10
	0.4	25000	9	7	5	12
	0.5	25000	9	7	5	14

Table 4: Cycle times for different problems and α levels.

Number of tasks	α	Cycle times			
		Robust	Regular	Best	Worst
	0.1	9	9.8	11	9
	0.2	9	9.8	11.5	7.7
11	0.3	9	9.8	12.8	7
	0.4	9	9.8	13.9	—
	0.5	9.3	9.8	14.4	—
	0.1	18.6	19.9	22	18
	0.2	19.3	19.9	24	16
21	0.3	19	19.9	25.8	14
	0.4	18.5	19.9	27.9	—
	0.5	16.1	19.9	30	—
	0.1	37	40	44	36
	0.2	36.8	40	47.4	32
29	0.3	36.8	40	52	27.9
	0.4	36.3	40	55.2	—
	0.5	36.4	40	59.6	—
	0.1	47.4	50	54.7	45
	0.2	46.2	50	59.9	—
35	0.3	45	50	65	—
	0.4	45.7	50	70	—
	0.5	45.4	50	74.8	—
	0.1	7456.2	7991.6	8774.7	7184.1
	0.2	7091	7991.6	9566	6362
83	0.3	6632.1	7991.6	10291.7	5571
	0.4	6146.9	7991.6	11116.6	—
	0.5	6219.3	7991.6	11762.8	—
	0.1	23462.7	24894.6	26791.1	22125
	0.2	23813.5	24894.6	29963.5	19954.6
111	0.3	20398.4	24894.6	32481.0	17428.3
	0.4	19043.3	24894.6	34676.8	14975.8
	0.5	18572.7	24894.6	37297.8	12320.7

In EALBP-I, the number of stations and the cycle time are both variables. Therefore, EALBP-I is actually a variant of SALBP-E. To compare the robust situation with the other three situations, production efficiency E is introduced. This is the reciprocal of the product of the number of stations and the cycle time; i.e., $E = 1/(mC)$ [40]. An important goal of assembly line design is to maximize productivity. According to Tables 3 and 4, the production efficiency is obtained for the six cases, as shown in Figures 8–13.

The efficiency in the robust situation is generally the highest. As shown in Figures 9 and 13, when the number of tasks is 21 and 111, the efficiency of the robust situation is the highest. In Figures 8, 10, and 12, only when $\alpha = 0.3$, the efficiency of the robust situation is less than that in one of the other situations. In Figure 11, when $\alpha \geq 0.3$, the efficiency of the robust situation is higher than that in the other three situations.

We take the cases with 21 tasks to validate the effectiveness of MGA. It is assumed that $\alpha = 0.1$. The maximum number of iterations is set to 400. The weight coefficients in expression (20) are $w_1 = 0.2$ and $w_2 = 1$. The general GA [41] and MGA in this study are used to solve EALBP-I. The results are presented in Table 5. The number of stations is the same. However, the maximum assembly time calculated by MGA is 18.6 s, which is less than 19.0 s, as calculated by GA. The convergence curves of MGA and GA are presented in Figure 14. These indicate that MGA converges faster than GA.

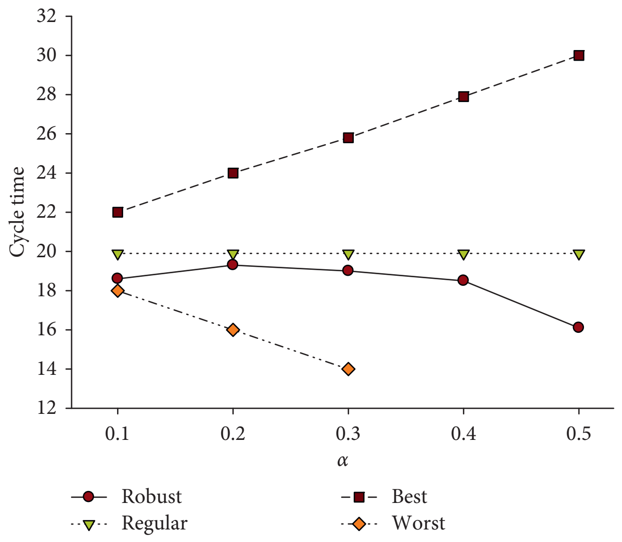

Figure 4: Cycle times for problem with 21 tasks.

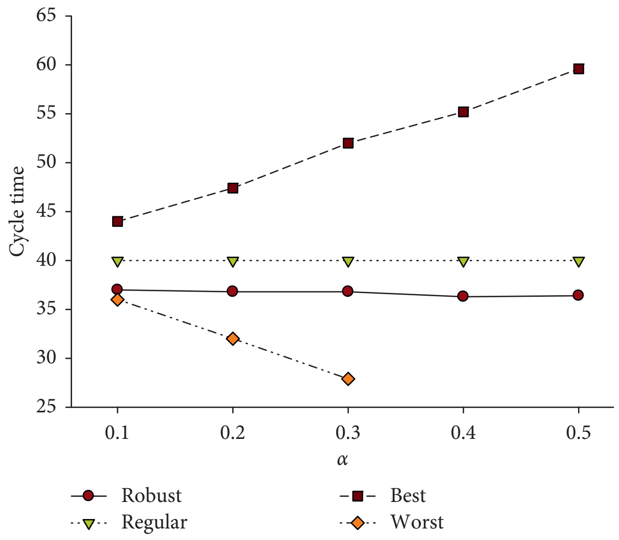

FIGURE 5: Cycle times for problem with 29 tasks.

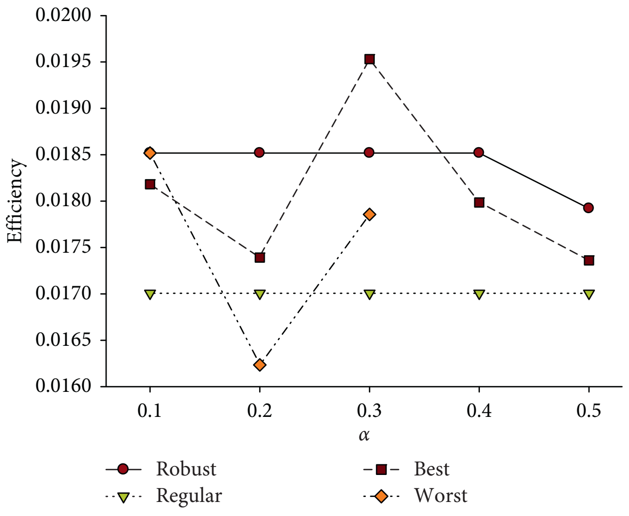

FIGURE 8: Efficiency for problem with 11 tasks.

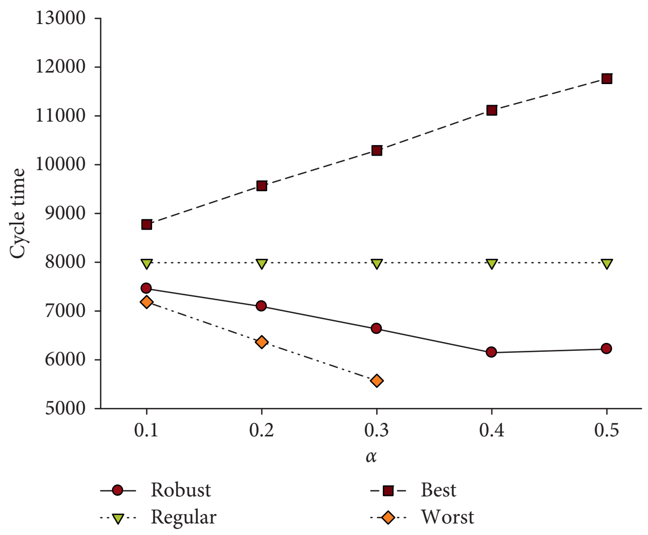

FIGURE 6: Cycle times for problem with 83 tasks.

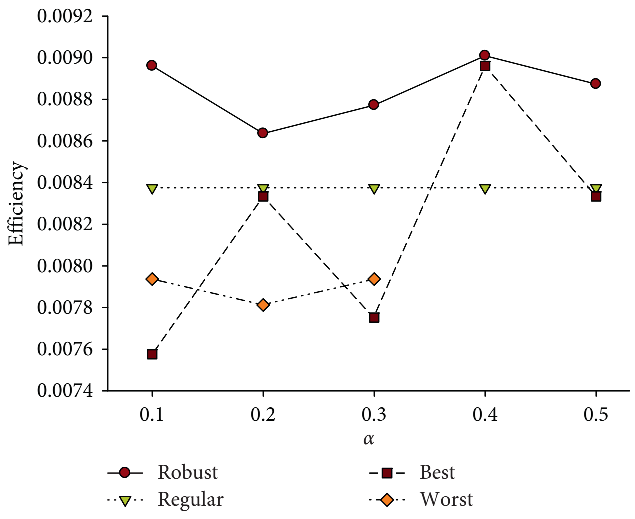

FIGURE 9: Efficiency for problem with 21 tasks.

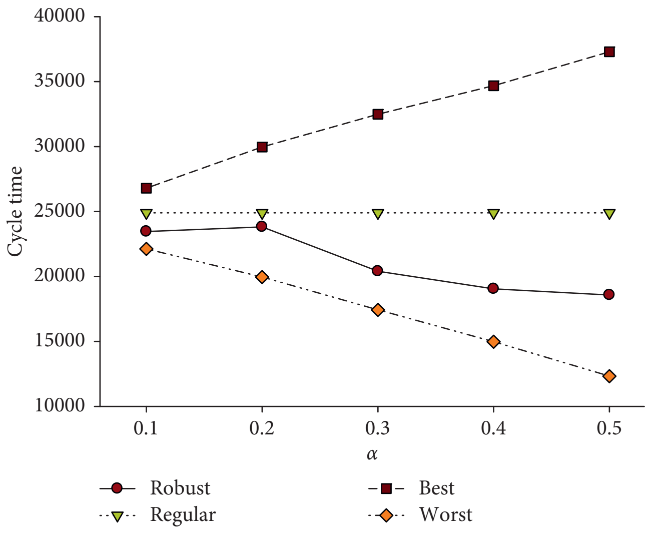

FIGURE 7: Cycle times for problem with 111 tasks.

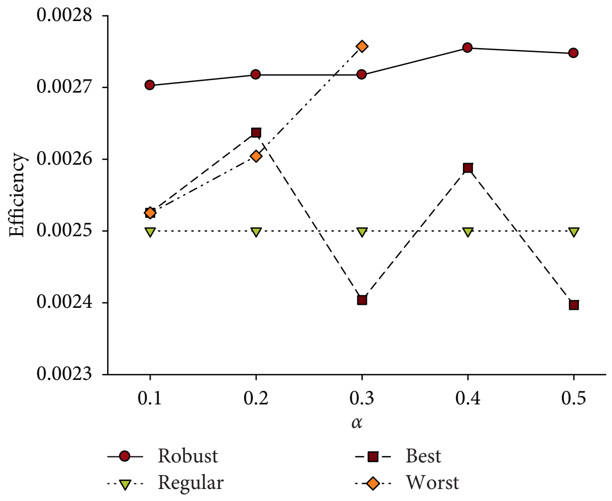

FIGURE 10: Efficiency for problem with 29 tasks.

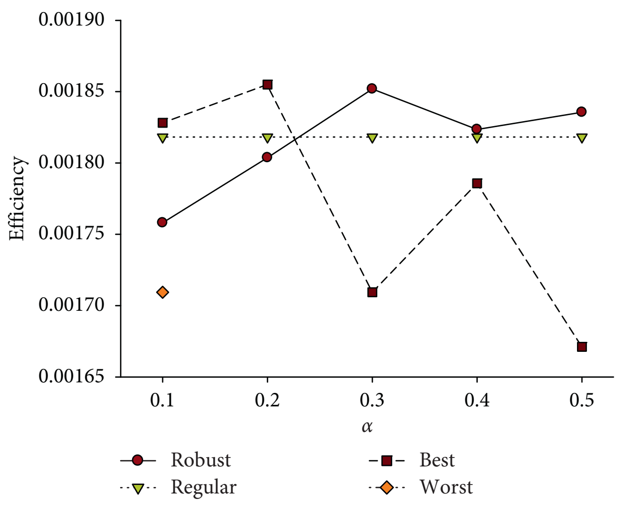

Figure 11: Efficiency for problem with 35 tasks.

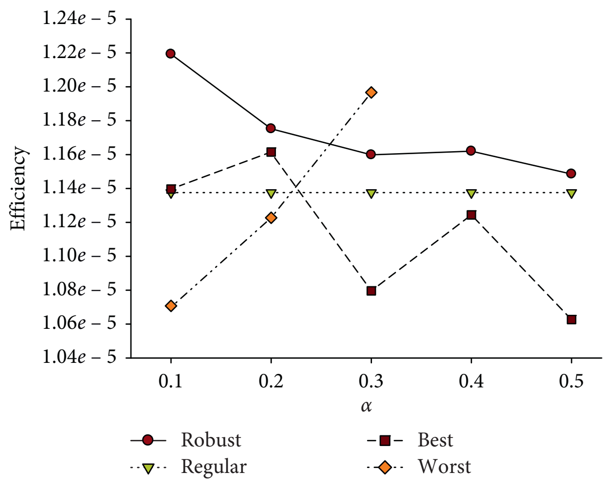

Figure 12: Efficiency for problem with 83 tasks.

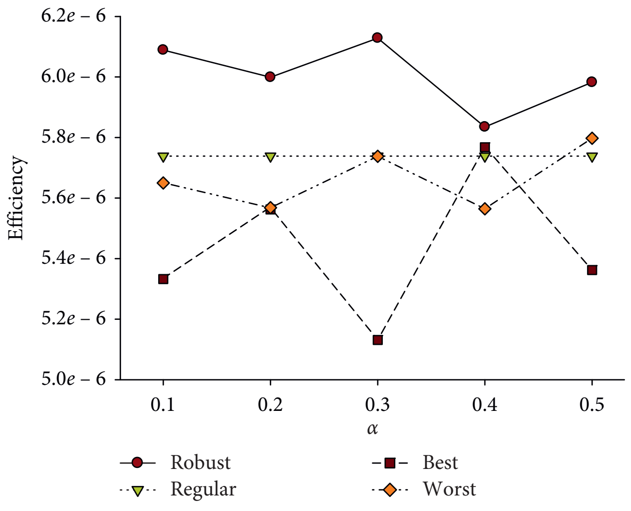

Figure 13: Efficiency for problem with 111 tasks.

Table 5: Assembly balancing result of MGA versus general GA.

MGA			GA		
Station no.	Set of tasks	Total assembly time (s)	Station no.	Set of tasks	Total assembly time (s)
1	1, 3, 4	18.5	1	1, 2, 3	18.2
2	5, 7	17.8	2	4, 5, 6	18.9
3	14, 2, 6, 21	18.1	3	21, 7, 14	19.0
4	8, 9, 12, 11, 10	18.6	4	8, 9, 12, 10, 11	18.9
5	15, 16, 13, 18	18.6	5	13, 15, 18, 16	19.0
6	17, 20, 19	18.6	6	17, 20, 19	18.4

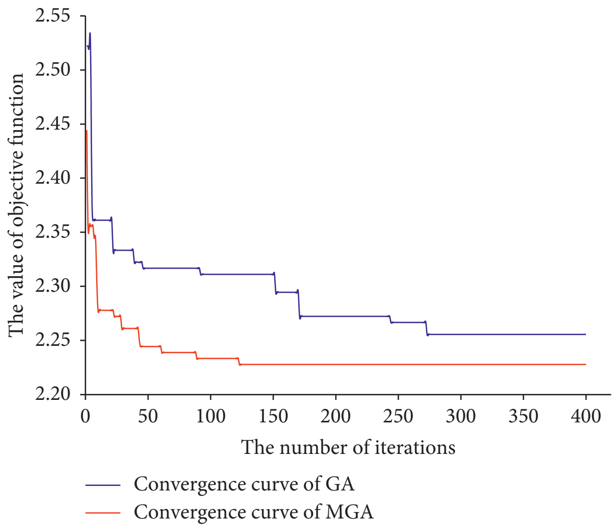

Figure 14: Convergence curves of MGA versus general GA.

7. Conclusions

With fierce market competition, the assembly line is susceptible to some uncertainties, the influence of which on assembly line balancing is of concern to manufacturers. In this study, an uncertain cycle time is introduced to ALBP. Spatial and incompatible constraints are added based on the traditional precedence graph. Spatial constraints are suitable to situations where multiple assembly tasks must be assigned to the same station. Incompatible constraints are suitable to situations where two assembly tasks must be assigned to different stations. The introduction of these constraints makes ALBP more suitable to engineering practice. Two mathematical models of ALBP are established. In the first, the cycle time is fixed, and the objective is to minimize the number of stations. In the second, the cycle time is in a changeable interval, and the objective is to minimize both the number of stations and the cycle time. The level of uncertainty α is introduced to adjust the range of the cycle time interval. In addition, operator skill is defined by three levels, and the selection of skill level is integrated into ALBP. Finally, a multipopulation genetic algorithm is used to solve our model. Six cases are analyzed in the study, each considering robust situation, regular situation, best situation, and worst situation. The research results show that the cycle time interval has a significant influence on the assembly line balancing problem. The current model is suitable to the situation of a new assembly line for a new product, in which case observations or knowledge of the cycle time may be limited. Designers can use an interval to represent the cycle time. In the future, uncertain cycle and task times will be considered in ALBP.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Shaanxi Key Research and Development Program (Grant no. 2019ZDLGY02-01).

References

[1] N. Boysen, M. Fliedner, and A. Scholl, "A classification of assembly line balancing problems," *European Journal of Operational Research*, vol. 183, no. 2, pp. 674–693, 2007.

[2] O. Battaïa and A. Dolgui, "A taxonomy of line balancing problems and their solutionapproaches," *International Journal of Production Economics*, vol. 142, no. 2, pp. 259–277, 2013.

[3] M. Eghtesadifard, M. Khalifeh, and M. Khorram, "A systematic review of research themes and hot topics in assembly line balancing through the web of science within 1990–2017," *Computers & Industrial Engineering*, vol. 139, no. 1, 2020.

[4] Z. Li, I. Kucukkoc, and J. M. Nilakantan, "Comprehensive review and evaluation of heuristics and meta-heuristics for two-sided assembly line balancing problem," *Computers & Operations Research*, vol. 84, no. 8, pp. 146–161, 2017.

[5] Ö. Hazır, X. Delorme, and A. Dolgui, "A review of cost and profit oriented line design and balancing problems and solution approaches," *Annual Reviews in Control*, vol. 40, pp. 14–24, 2015.

[6] P. Sivasankaran and P. Shahabudeen, "Literature review of assembly line balancing problems," *The International Journal of Advanced Manufacturing Technology*, vol. 73, no. 9-12, pp. 1665–1694, 2014.

[7] T. Pape, "Heuristics and lower bounds for the simple assembly line balancing problem type 1: overview, computational tests and improvements," *European Journal of Operational Research*, vol. 240, no. 1, pp. 32–42, 2015.

[8] A. Diabat, E. Dehghani, and A. Jabbarzadeh, "Incorporating location and inventory decisions into a supply chain design problem with uncertain demands and lead times," *Journal of Manufacturing Systems*, vol. 43, no. 4, pp. 139–149, 2017.

[9] Ö Hazır and A. Dolgui, "A review on robust assembly line balancing approaches," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 987–991, 2019.

[10] M. Salehi, H. R. Maleki, and S. Niroomand, "A multi-objective assembly line balancing problem with worker's skill and qualification considerations in fuzzy environment," *Applied Intelligence*, vol. 48, no. 8, pp. 2137–2156, 2018.

[11] O. A. Arık, E. Köse, and J. Forrest, "Simple assembly line balancing problem of Type 1 with grey demand and grey task durations," *Grey Systems: Theory and Application*, vol. 9, no. 4, pp. 401–414, 2019.

[12] H. Y Zhang, "An immune genetic algorithm for simple assembly line balancing problem of type 1," *Assembly Automation*, vol. 39, no. 1, pp. 113–123, 2019.

[13] G. Tian, M. C. Zhou, and P. Li, "Disassembly sequence planning considering fuzzy component quality and varying operational cost," *IEEE Transactions on Automation Science and Engineering*, vol. 15, no. 2, pp. 748–760, 2017.

[14] G. Tian, N. Hao, M. C. Zhou et al., "Fuzzy grey Choquet integral for evaluation of multicriteria decision making problems with interactive and qualitative indices," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 15, 2019.

[15] N. Mirzaei, A. Mahmoodirad, and S. Niroomand, "An uncertain multi-objective assembly line balancing problem: a credibility-based fuzzy modeling approach," *International Journal of Fuzzy Systems*, vol. 21, no. 8, pp. 2392–2404, 2019.

[16] N. HazR and A. Dolgui, "Assembly line balancing under uncertainty Robust optimization models and exact solution method," *Computers & Industrial Engineering*, vol. 65, no. 2, pp. 261–267, 2013.

[17] E. Gurevsky, Ö. Hazır, O. Battaïa, and A. Dolgui, "Robust balancing of straight assembly lines with interval task times," *Journal of the Operational Research Society*, vol. 64, no. 11, pp. 1607–1613, 2013.

[18] M. Liu, X. Liu, F. Chu, F. Zheng, and C. Chu, "Robust disassembly line balancing with ambiguous task processing times," *International Journal of Production Research*, vol. 64, pp. 1–30, 2019.

[19] Y. Li, Y. Fu, X. Tang et al., "Optimizing the reliability and efficiency for an assembly line that considers uncertain task time attributes," *IEEE Access*, vol. 64, pp. 34121–34130, 2019.

[20] Y. Li, M. Wen, R. Kang, and Z. Yang, "Type-1 assembly line balancing considering uncertain task time," *Journal of Intelligent & Fuzzy Systems*, vol. 35, no. 2, pp. 2619–2631, 2018.

[21] J. Dong, L. Zhang, and T. Xiao, "A hybrid PSO/SA algorithm for bi-criteria stochastic line balancing with flexible task times and zoning constraints," *Journal of Intelligent Manufacturing*, vol. 29, no. 4, pp. 737–751, 2018.

[22] J. Li and J. Gao, "Balancing manual mixed-model assembly lines using overtime work in a demand variation environment," *International Journal of Production Research*, vol. 52, no. 12, pp. 3552–3567, 2014.

[23] M. Chica, J. Bautista, Ó. Cordón, and S. Damas, "A multiobjective model and evolutionary algorithms for robust time and space assembly line balancing under uncertain demand," *Omega*, vol. 58, no. 1, pp. 55–68, 2016.

[24] P. Samouei and J. Ashayeri, "Developing optimization & robust models for a mixed-model assembly line balancing problem with semi-automated operations," *Applied Mathematical Modelling*, vol. 72, no. 8, pp. 259–275, 2019.

[25] J. Pereira and E. Álvarez-Miranda, "An exact approach for the robust assembly line balancing problem," *Omega*, vol. 78, no. 7, pp. 85–98, 2018.

[26] M. Fathi, A. Nourmohammadi, A. H. C. Ng, and A. Syberfeldt, "An optimization model for balancing assembly lines with stochastic task times and zoning constraints," *IEEE Access*, vol. 7, pp. 32537–32550, 2019.

[27] J.-H. Zhang, A.-P. Li, and X.-M. Liu, "Hybrid genetic algorithm for a type-II robust mixed-model assembly line balancing problem with interval task times," *Advances in Manufacturing*, vol. 7, no. 2, pp. 117–132, 2019.

[28] J. Fisel, Y. Exner, N. Stricker, and G. Lanza, "Changeability and flexibility of assembly line balancing as a multi-objective optimization problem," *Journal of Manufacturing Systems*, vol. 53, no. 10, pp. 150–158, 2019.

[29] M. H. Alavidoost, H. Babazadeh, and S. T. Sayyari, "An interactive fuzzy programming approach for bi-objective straight and U-shaped assembly line balancing problem," *Applied Soft Computing*, vol. 40, no. 3, pp. 221–235, 2016.

[30] H. Babazadeh, M. H. Alavidoost, M. H. Fazel Zarandi, and S. T. Sayyari, "An enhanced NSGA-II algorithm for fuzzy bi-objective assembly line balancing problems," *Computers & Industrial Engineering*, vol. 123, no. 9, pp. 189–208, 2018.

[31] H. Babazadeh and A. Esfahanipour, "A novel multi period mean-VaR portfolio optimization model considering practical constraints and transaction cost," *Journal of Computational and Applied Mathematics*, vol. 361, no. 12, pp. 313–342, 2019.

[32] H. Babazadeh and N. Javadian, "A novel meta-heuristic approach to solve fuzzy multi-objective straight and U-shaped assembly line balancing problems," *Soft Computing*, vol. 23, no. 17, pp. 8217–8245, 2019.

[33] G. Tian, Y. Ren, Y. Feng, M. Zhou, H. Zhang, and J. Tan, "Modeling and planning for dual-objective selective disassembly using and or graph and discrete artificial bee colony," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2456–2468, 2019.

[34] D. Li, C. Zhang, G. Tian, X. Shao, and Z. Li, "Multiobjective program and hybrid imperialist competitive algorithm for the mixed-model two-sided assembly lines subject to multiple constraints," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 1, pp. 119–129, 2018.

[35] J.-f. Yu, W.-B. Tang, Y. Li, and J. Zhang, "Dimensional variation propagation modeling and analysis for single-station assembly based on multiple constraints graph," *Assembly Automation*, vol. 36, no. 3, pp. 308–317, 2016.

[36] Y. Li, H. Wang, and Z. Yang, "Type II assembly line balancing problem with multi-operators," *Neural Computing and Applications*, vol. 31, no. S1, pp. 347–357, 2019.

[37] D. Yilmaz, "A genetic algorithm approach for balancing two-sided assembly lines with setups," *Assembly Automation*, vol. 39, no. 5, pp. 827–839, 2019.

[38] M. D. Michela and G. Dini, "Optimizing ergonomics in assembly lines: a multi objective genetic algorithm," *Cirp Journal of Manufacturing Science and Technology*, vol. 27, pp. 31–45, 2019.

[39] F. Masood, "An improved genetic algorithm with variable neighborhood search to solve the assembly line balancing problem," *Engineering Computations*, vol. 37, no. 2, pp. 501–521, 2019.

[40] Y. Li, *Assembly line rebalancing with non-constant task time attribute*, Ph.D. dissertation, Rutgers University, New Brunswick, Germany, 2016.

[41] M. D. Michela and G. Dini, "Designing assembly lines with humans and collaborative robots: a genetic approach," *Cirp Annals-Manufacturing Technology*, vol. 68, no. 1, pp. 1–4, 2019.