

Balancing mixed-model assembly lines with sequence-dependent tasks via hybrid genetic algorithm

Qihua Tang¹ · Yanli Liang¹ · Liping Zhang¹ ·
Christodoulos A. Floudas^{2,3} · Xiaojun Cao⁴

Received: 22 March 2014 / Accepted: 30 May 2015 / Published online: 10 June 2015
© Springer Science+Business Media New York 2015

Abstract Close connections existing among sequence-dependent tasks should be emphasized while assembling products within automotive or electronic industries. This paper addresses the mixed-model assembly line balancing problem with sequence-dependent tasks with two objectives, the minimization of cycle time and workload variance. A hybrid genetic algorithm with novel logic strings was proposed to address the problem. First, both the sequence-dependent connections and precedence relations are integrated into the combined precedence graph so as to transform the original problem into the single-model assembly line balancing problem and to decrease the computational complexity. Second, three heuristic factors are hybridized into the process of initialization with the purpose of improving the quality of initial solution population. Third, considering sequence-dependent tasks, logic strings are designed to ensure the feasibility of chromosomes during two-point crossover and insertion mutation operations. Computational studies have demonstrated that the proposed algorithm can solve problems to near-optimality and even optimality with less computational effort.

Keywords Mixed-model assembly line balancing · Sequence-dependent tasks · Combined precedence graph · Hybrid genetic algorithm · Elite preservation strategy

Supported by Natural Science Foundation of China under Grant Nos. 50875190 and 51275366.

✉ Qihua Tang
tangqihua@wust.edu.cn

✉ Christodoulos A. Floudas
floudas@tamu.edu

¹ Industrial Engineering Department, Wuhan University of Science and Technology, Wuhan 430081, Hubei, China

² Texas A&M Energy Institute, Texas A&M University, College Station, TX 77843, USA

³ Artie McFerrin Department of Chemical Engineering, Texas A&M University, College Station, TX 77843, USA

⁴ Technique Center of Dongfeng Peugeot Citroen Automobile Company, Wuhan 430056, Hubei, China

1 Introduction

An assembly line is a flow-line production system, which is typical in high quantity production of standardized commodities and low volume production of customized products. Originally, assembly lines targeted at mass production of single products. With the diversified needs of the customers, the number of product models began to increase. For example, in the automotive and electronic industries, the customers hope to buy individualized products according to their own requirements and affordability, which impels manufacturers to extend the capability of existing assembly lines so as to produce various products concurrently. In such a context, mixed-model assembly lines arise. It specializes in assembling a variety of product models with similar characteristics at the same time.

The single-model assembly line balancing problem (SALB-P) was first proposed and formulated as a mathematical problem by Salveson [1]. Subsequently, additional product models are introduced, which leads to the mixed-model assembly line balancing problem (MMALB-P). MMALB-P needs to make decisions about how to assign all the given tasks to workstations under the constraints of precedence relations. The MMALB-P is generally classified into two categories: Type-I, the number of workstations is minimized for a given cycle time; Type-II, the number of workstations is fixed and the objective is to minimize cycle time. Type-II is also NP-hard [2–4], but more popular in contemporary enterprises because it can increase productivity without purchasing new machines or expanding facilities within existing assembly lines. Hence, Type-II is studied in this paper.

Unlike SALB-P, MMALB-P is much more complicated since it necessitates additional considerations about interactions among product models. As addressed by several researches in the last few decades [4–7], the task common to several models should be allocated to exactly one workstation by utilizing the joint precedence graph or combined precedence graph [8–12]. As for joint precedence graphs, Boysen [12] proposed a modified approach based on option mix forecasts, which allows for reliable model mix forecasts when product variety is high. This option-based approach is a better choice in practice in terms of computational effort and outcome.

As for MMALB-P with sequence-dependency, several contributions can be found, in which production sequence-dependency, resulting from differences among product models, is involved. The number of sequence-dependent setups was optimized by Giard and Jeunet [13] through a cost function. Under the constraint of setup times, Ozturk et al. [14] and Yolmeh and Kianfar [15] balanced and scheduled flexible mixed-model assembly lines. Due to facility constraints, tasks belonging to different car types should be processed in one workstation, and thus the operation of process precision is dependent on the production sequence [16]. Mosadegha and Zandieh [17] solved the balancing and sequencing problems, where processing time of each task was station-dependent, so that production sequence had a great influence on the workload of given workstations and line efficiency. Hamta et al. [18] proposed a new solution method based on the combination of particle swarm optimization algorithm with variable neighborhood search to solve a single-model assembly line balancing problem with three objectives. There, the task time depends on worker(s) (or machine(s)) learning for the same or similar activities and sequence-dependent setup times existing between tasks. In rare cases, task sequence-dependency within one product model is considered. In 2006, sequence-dependent assembly line balancing problem of single model was defined by Scholl et al. [19], where two tasks interact and the task times are influenced due to strict technological requirements. Note that, the MMALB-P with the sequence-dependent tasks has not been studied very well. For the sake of industrial applications, this paper focuses on the MMALB-P with sequence-dependent connections.

In the past several decades, many approaches such as exact, heuristic algorithm and meta-heuristic algorithm have been developed to tackle the mixed-model assembly line balancing problem. Bukchin and Rabinowitch [4] developed an optimal solution procedure based on a backtracking branch-and-bound algorithm to minimize the sum of costs about stations and task duplication. Hadi [20] presented a binary goal programming model. Helgeson and Birnie [21] developed a heuristic approach known as the ranked positional weighted technique (RPWT) which provides near-optimal solutions indeed to reduce computational times. Bock [22] proposed distributed search procedures by integrating specific aspects relevant to the automotive industry. To improve the combinatorial optimization capability in discrete search-space, meta-heuristics were studied in the past decade. Vilarinho and Simaria [6] proposed a two-stage procedure based on simulated annealing, employing different goals in each stage, to minimize the number of workstations first for a given cycle time and to balance workloads between and within workstations then. Simaria and Vilarinho [23] presented an iterative genetic algorithm-based procedure for solving MMALB-P with parallel workstations, to maximize the production rate of the line under the pre-determined number of operators. A generalized Pareto-based scale-independent fitness function genetic algorithm was proposed by Zhang and Gen [24], where three objectives are introduced: to minimize cycle time, workload variation and the sum of costs. Manavizadeh et al. [25] introduced a multi-objective genetic algorithm to solve a mixed-model assembly line problem, considering cycle time and the number of stations simultaneously. A comparison was carried out between six multi-objective evolutionary algorithms so as to determine the best method to solve the specific problem.

Hybrid algorithms represent a significant attempt for solving NP-hard problems in an effective, reliable and accurate way without any human intervention [26], and are introduced in this study. Akpinar and Bayhan [27] presented a novel hybrid genetic algorithm by integrating a genetic algorithm with three different heuristics, (1) the RPWT from Helgeson and Birnie [21], (2) the Kilbridge and Wester heuristic [28], and (iii) phase-I of Moodie and Young method [29]. Noorul [11] also proposed a hybrid genetic algorithm approach in which initial solutions were obtained using a modified ranked positional weight method. These approaches reduce both search space and search time effectively.

In this study, a hybrid genetic algorithm is proposed to solve MMALB-P with sequence-dependent tasks. This paper is organized as follows: sequence-dependent tasks and their industrial applications are explained in detail in Sect. 2. The mathematical model of MMALB-P with sequence-dependent tasks is presented in Sect. 2.3. The hybrid genetic algorithm is presented in Sect. 3. The experimental results are discussed and analyzed in Sect. 4. The conclusions are provided in Sect. 5.

2 Problem formulation

During balancing workloads along assembly lines, there exist certain connections among sequence-dependent tasks. Such connections have not yet been studied in the literature.

2.1 Sequence-dependent tasks

Close connections, except precedence relations, exist among tasks during the assembly of products. These connections originate from two phases of operation management as follows.

Phase 1: Each job is divided into one or more tasks. Any task should be the smallest, indivisible and rational work element of the job, so as to increase the efficiency of the

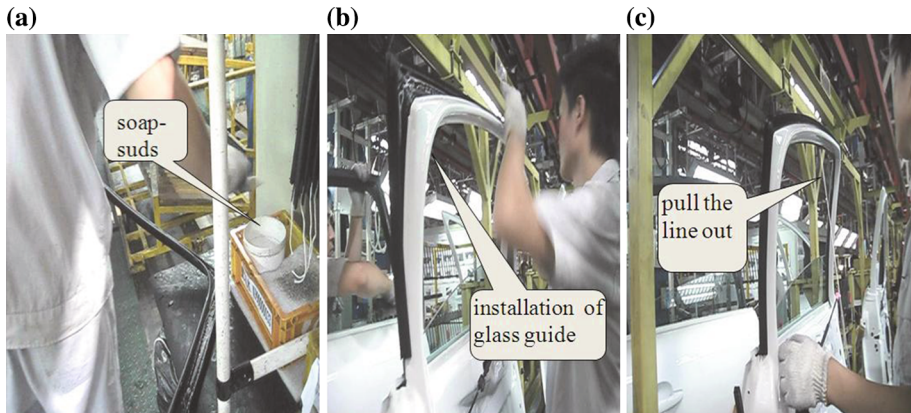


Fig. 1 The installation process of glass guide. **a** Coat soap-suds. **b** The installation of glass guide. **c** Pull the line out

assembly line, and to reduce the operation errors such as missing some minor tasks. After task division, the precedence relations among tasks are redefined and drawn into the precedence graph. In this way, each task can be performed complying with the assembly process strictly. This phase will be done by the process planning department.

Phase 2: Based on given task divisions and precedence relations, the objective is to balance the workload along the assembly lines. This is performed by the manufacturing workshop.

Due to the task division, certain connections appear in Phase 2. This is denoted as sequence-dependent relations in practice, and demonstrated by the following three examples.

Example 1 The glass guide is installed with three tasks as displayed in Fig. 1. Task 1 in Fig. 1a is to coat soap-suds as lubricant to facilitate the installation of glass guide. Task 2 in Fig. 1b is to install the glass guide immediately. Task 3 in Fig. 1c means to pull the line out immediately. If task 2 is not performed immediately after task 1, the lubricant will be dry and useless. Similarly, if task 3 is not performed immediately after task 2, other workers may touch the soap-suds and thus stain the door. In brief, tasks 1, 2 and 3 should be performed consecutively in one station, although such constraints have not been defined in the precedence graph.

Example 2 The fastening clamp plays an important role in fixing the glass on the door. The installation of the fastening clamp on the glass must be performed before installing the glass on the door. This relation has been represented in the precedence constraint. However, during the assembly, if two tasks as shown in Fig. 2, installing the fastening clamp on the glass and glass on the door, are completed on different workstations, then storage spaces along the assembly line will double, and the transportation times and costs for transferring glasses from one workstation to another will increase. Thus, these two tasks should be performed on the same workstation. Moreover, since the installation of fastening clamp on the glass is very simple, workers can complete it when they fetch the glass. Thus, these two tasks are defined as sequence-dependent tasks in practice. Such sequence-dependent connection between these two tasks should be considered beyond the precedence relation, to ensure that these two tasks can be performed consecutively, to reduce the walking time of workers on the production line, and to shorten the subsidiary operation times.

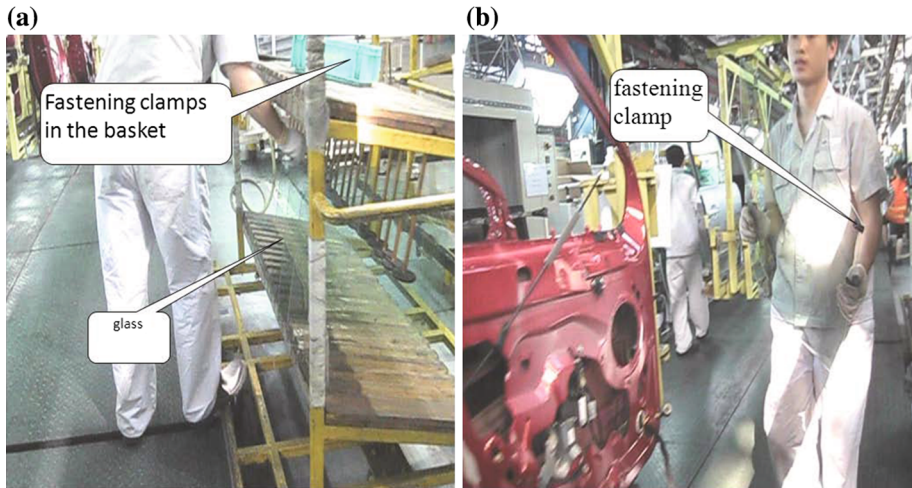


Fig. 2 The installation process of door glass. **a** The sub-installation of fastening clamp. **b** The installation of door glass

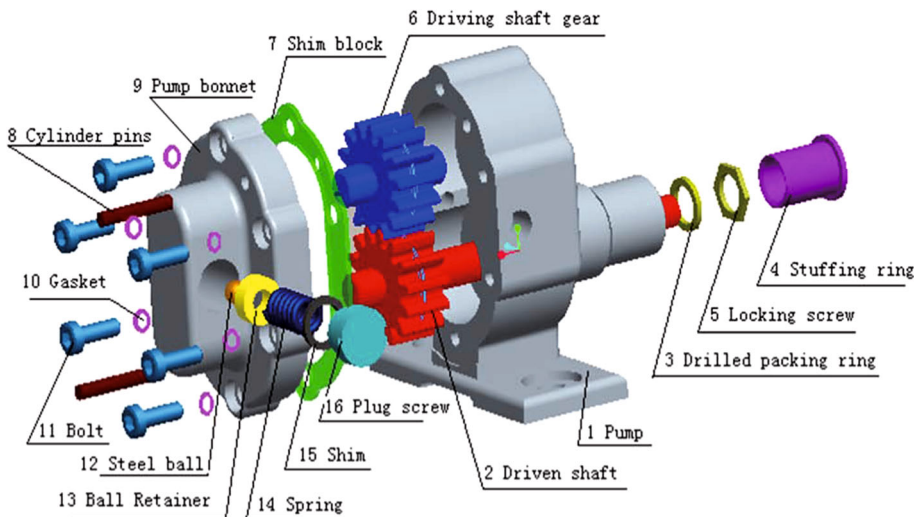


Fig. 3 Assembly body of gear pump

Example 3 In a real industrial operation, the gear pump of an automotive engine is assembled by 22 components. Sixteen of them will be left after the simplified treatment of screws, gaskets and pins. As shown in Fig. 3, six cylinder pins must be assembled immediately to prevent the pump bonnet from falling down. Such connections also exist in installing gasket and bolts, and setting up spring, shim and plug screw.

The above examples show that the sequence-dependent tasks take place frequently the practical assembly, and should be considered during balancing assembly lines.

Moreover, sequence-dependent tasks can be termed as tasks which should performed consecutively in the same workstation. Considering sequence-dependent tasks, the assembly

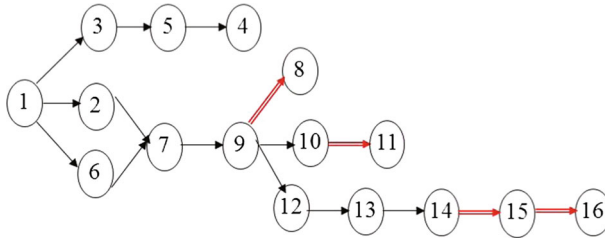


Fig. 4 Precedence graph with sequence-dependent tasks

procedure of the gear pump can be depicted as the precedence graph of Fig. 4, where circles represent tasks, arrows between tasks stand for precedence constraint and arrows with double lines highlight sequence-dependent relations between tasks.

If a sequence-dependent relation exists between tasks i and l with i ahead of l , the following principles must be held during assigning tasks among workstations:

- (1) Tasks i and l must be assigned to the same workstation;
- (2) Task l must be performed immediately after i , meaning that l has higher priority than other immediate following tasks of i ;

The notation used in this study is listed in Table 1.

2.2 Problem statement

Generally, the mixed-model assembly line balancing problem of Type-II can be depicted as Fig. 5, where S stations ($j = 1, 2, \dots, S$) are installed along a unidirectional transportation system. M ($m = 1, 2, \dots, M$) product models, represented by diamonds, circles and triangles, are serially assembled along the line at a constant pace, which is defined as cycle time (CT). Each product model m has d_m products and I ($i = 1, 2, \dots, N$) tasks. In addition, each workstation j can process any task of every product model.

The following assumptions are introduced for the MMALB-P with sequence-dependent tasks:

- D products from M ($M \geq 2$) models are assembled serially.
- Throughout the planning horizon, demands for each model are given and demand proportions q_m for each model m can be determined in advance.
- Same tasks in different product models may have different processing times.
- Any task should be allocated to exactly one workstation and consequently the task common to several product models will be performed on one workstation [11]. This will improve the proficiency of workers and make full use of tools and fixtures in that station.
- The processing time for each task is predetermined and deterministic.
- The allocation of tasks to workstations must comply with cycle time.
- Every task of each product model must be assigned only once, satisfying precedence relations and sequence-dependent constraints.
- Precedence graphs of all models can constitute a combined precedence graph, which includes N assembly tasks of M product models.

A combined precedence graph, which converts MMALB-P into SALB-P, is the preferred approach to depict similarities and differences among models. Figure 6 illustrates the process of constructing a combined precedence graph, where both sequence-dependent connections

Table 1 The notations used in this study

Indices

i, l : Task ($i, l = 1, 2, \dots, N$)

j : Workstation ($j = 1, 2, \dots, S$)

m : Product model ($m = 1, 2, \dots, M$)

Parameters

N : Number of task

t_{im} : Processing time of task i for model m

S : Number of workstations

t_{\min} : The minimum task time

M : Number of models

t_{\max} : The maximum task time

CT : Cycle time

d_m : Demand for model m

cr : Crossover rate

H : Big value

mr : Mutation rate

D : Total demand of all models, $D = \sum_{m=1}^M d_m$

t_i : Processing time of task i in the combined precedence graph, $t_i = \sum_{m=1}^M t_{im} q_m$

q_m : Proportion of model m , where $q_m = d_m / \sum_{m=1}^M d_m = d_m / D$

W : The number of available positions in logic strings, and $W = \sum_{r=1}^{N-1} Position_r$

PL : The number of precedence links. If there is a precedence relation between tasks, a precedence link exists

y_{lk} : Binary variable for precedence relation. If Task l immediately precedes task k , then y_{lk} equals to 1; otherwise, it is equal to zero

SD_{il} : Binary variable for sequence-dependent connection. If tasks i and l are sequence-dependent with i ahead of l , then SD_{il} equals to 1; otherwise, it is equal to zero

IF_i : Immediate following task set of task i , which is derived directly from the combined precedence graph

Decision variables

$Tabu_i = \begin{cases} 1 & \text{if task } i \text{ has been assigned} \\ 0 & \text{otherwise} \end{cases}$

$Position_r = \begin{cases} 1 & \text{if the task in } r\text{-th position is available to crossover or mutate} \\ 0 & \text{otherwise} \end{cases}$

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

$Y_{ijn} = \begin{cases} 1 & \text{if task } i \text{ is assigned to the } n_{th} \text{ position of workstation } j \\ 0 & \text{otherwise} \end{cases}$

Dynamic sets

PS : Possible Set representing holding tasks which have no predecessor or all their predecessors have been allocated

CS : Candidate Set representing holding tasks which satisfy both precedence relations and sequence-dependent constraints; $y_{kl} = 1, SD_{il} = 1, Tabu_k = 1, i, l, k \in PS$

UD : Updating Set representing holding tasks that are available to be updated, (e.g., $i, l \in PS$, if $Tabu_i = 1, y_{ik} = 1$ and $y_{lk} = 0$, then task k can be updated)

FA : Feasible Area, which is between the latest predecessor and the earliest successor of the mutated task

and precedence relations are included. In Fig. 6a, tasks 4 and 7 are sequence-dependent, and hence, in the combined precedence graph Fig. 6c, this sequence-dependent relation is included. For the processing times, and given that the demand ratio between models A and

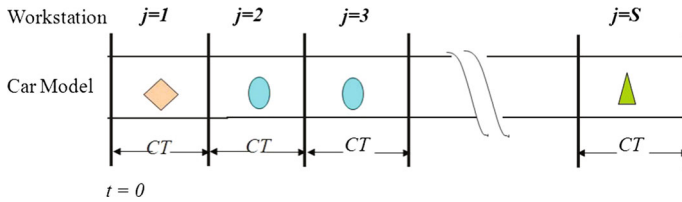


Fig. 5 Mixed-model assembly lines balancing

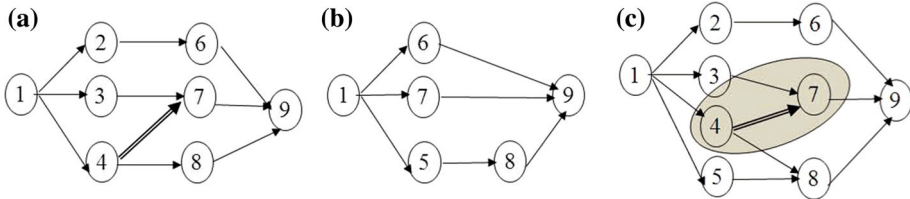


Fig. 6 Precedence graph for: **a** model A, **b** model B, **c** combine precedence graph

Table 2 The processing times for each model and the combined model

Task	1	2	3	4	5	6	7	8	9
t_A	4	24	21	15	0	8	1	5	9
t_B	4	0	0	0	6	2	4	2	9
Weighted t_i	4	8	7	5	4	4	3	3	9

B is 1:2, the respective processing times are listed in lines 2 and 3 of Table 2, while the last line shows the weighted processing time for each task in the combined precedence graph.

2.3 The mathematical model

The MMALB-P with sequence-dependent tasks has two objectives: (a) to minimize the cycle time and (b) to minimize the workload variance for a given number of workstations. The primary objective, to promote throughput and increase productivity, is achieved by minimizing the maximal station time (cycle time) as in Eq. (1). The secondary objective, to smooth workloads along the whole assembly line and ensure fairness among workers, is derived by minimizing the workload variance WV described as Eq. (2).

$$\min CT = \max T_j \tag{1}$$

$$\min WV = \frac{1}{S} \sum_{j=1}^S \left[T_j - \frac{1}{S} \left(\sum_{i=1}^N t_i \right) \right]^2 \tag{2}$$

where S is the number of given workstations, T_j is the j th station time, and t_i is the processing time of task i in the combined precedence graph.

The constraints of the mathematical model are as follows.

$$\sum_{j=1}^S x_{ij} = 1 \quad \forall i \tag{3}$$

$$\sum_{n=1}^N Y_{ijn} = x_{ij} \quad \forall i, j \tag{4}$$

$$\sum_{j=1}^S j \cdot x_{ij} \leq \sum_{j'=1}^S j' \cdot x_{i'j'} \quad \forall i, i', (i, i') \in P \tag{5}$$

$$\sum_{n=1}^N n \cdot Y_{ijn} \leq \sum_{n'=1}^N n' \cdot Y_{i'n'} + H \times (2 - x_{ij} - x_{i'j}) \quad \forall i, i', n, n', j \quad (i, i') \in P \tag{6}$$

$$\sum_{i=1}^N x_{ij} \cdot t_i \leq CT \quad \forall j \tag{7}$$

$$Y_{ijn} - Y_{lj,n+1} = 0 \quad (i, l) \in SD, \forall j, n \tag{8}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \tag{9}$$

$$Y_{ijn} \in \{0, 1\} \quad \forall i, j, n \tag{10}$$

Equations (3) and (4) ensure that each task is assigned to one position in certain workstation. Equations (5) and (6) introduce the precedence relationships between tasks represented by $(i, i') \in P$ which are satisfied by assuring that task i' is performed immediately after task i , no matter whether the immediate following and preceding tasks are allocated to the same workstation or not. Equation (7) imposes that the cycle time is the maximal station time. Equation (8) guarantees that the sequence-dependent tasks i and l are performed sequentially in the same workstation. Equations (9) and (10) are general integrality restrictions.

3 The hybrid genetic algorithm

Genetic algorithm (GA), a stochastic searching algorithm inspired by the natural evolution rule, was introduced by Holland [30]. Being able to search for near optimal solutions efficiently, GA is widely used to address NP-hard problems in many areas, for example, robotics, automatic control, artificial life, image processing, combinatorial optimization and production scheduling.

The hybrid GA starts with an initial population of a set of feasible solutions represented by chromosomes, which will be decoded according to processing times and cycle time, and then be evaluated by a fitness function representing a quality measure of each individual. To search for optimal or near optimal solutions, the selection process is carried out iteratively. A new chromosome is generated through two important operators: crossover and mutation. The crossover operator combines two selected chromosomes by exchanging part of them based on randomly selected cut points, while the mutation operator mutates one or more randomly selected parts. The flow chart of the hybrid GA is shown in the middle column of Fig. 7. It includes the following steps.

- Step 1. Population initialization: formulate initial population of chromosomes.
- Step 2. Individual decoding: assign tasks to ordered workstations.
- Step 3. Fitness evaluation: evaluate each individual.
- Step 4. Selection: choose individuals with higher fitness under an elite preservation strategy and to preserve best individuals into next generation.
- Step 5. Crossover: combine two chromosomes for generating offspring.
- Step 6. Mutation: produce offspring.

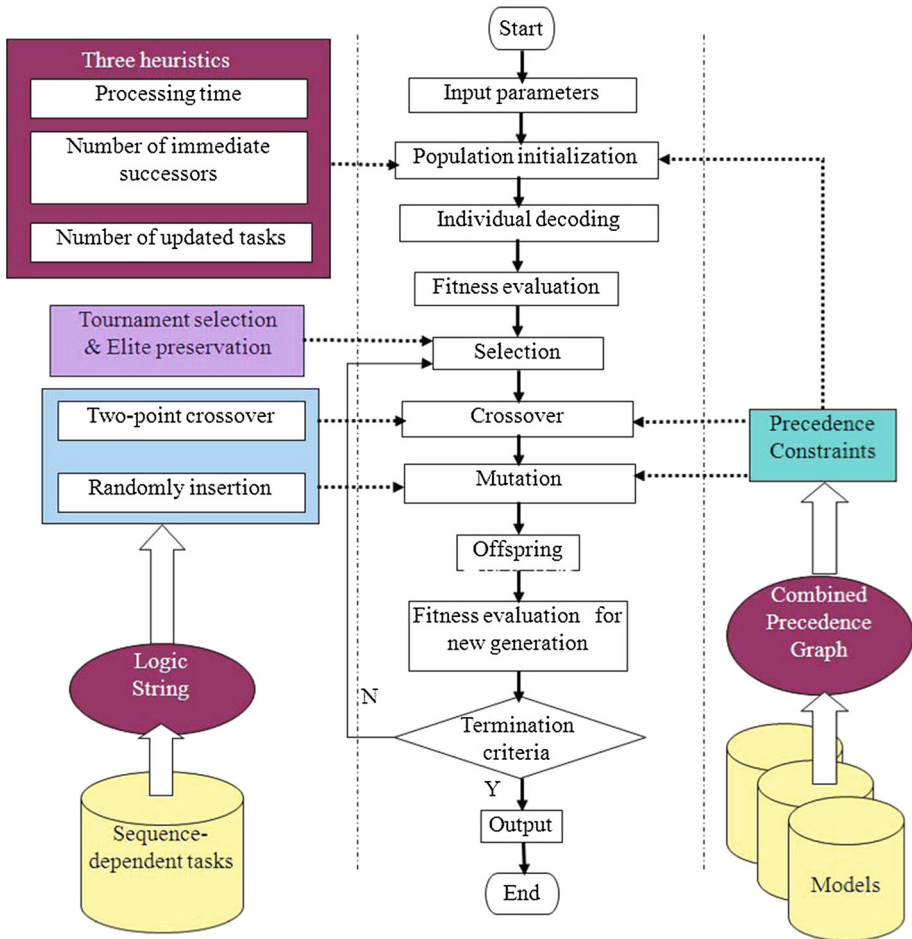


Fig. 7 Flowchat of hybrid GA for MMALB-P with sequence-dependent tasks

Step 7. Fitness evaluation: evaluate each individual of new generation.

Step 8. Termination criteria: end the procedure if certain termination criteria, such as predetermined number of generation, have been satisfied. Otherwise, go to step 4.

Due to multiple product models being involved, the combined precedence graph of all assembled product models should be employed, so that precedence constraints of each product model can be satisfied. This is represented by the right column in Fig. 7. Moreover, as exhibited by the left column in Fig. 7, logic strings are designed to avoid violating sequence-dependent relations between tasks during the crossover and mutation operations, and three novel heuristic factors are introduced to speed up the process of initialization.

3.1 Initialization

A high quality initial population can reduce the computational effort, especially for large-scale industrial applications. Hence, it is important to generate high quality initial solutions. A task needs to be selected from the current set, and one of several alternative task selection

Table 3 Criteria for selecting tasks

No.	Rule for tasks	Decision criteria
1	RPWT	Ranked positional weight technique
2	T	Maximum task processing time
3	N	Minimum task number
4	IS	Maximum number of immediate successors
5	TS	Maximum number of total successors
6	ATS	Maximum average time of successors
7	RT	Uniformly random
8	w_i	Synthesis Weights

rules will be used to choose the task. The decision criteria used to select the next task are presented in Table 3, which displays an adaptation to this problem of 7 well-known priority rules and random selection. These rules are determined either by task processing times, the number of immediate successors, precedence relations, cycle time, or by their combinations. For instance, the ranked positional weight technical (RPWT), associated with the task and its immediate following tasks, was calculated in Helgeson and Birnie [21] so that tasks could be allocated to the lowest numbered feasible stations at descending order of the positional weight. Moodie and Young [29] employed two matrices P (immediate predecessors) and F (immediate successors) to determine the available tasks for assigning based on the maximum task processing time rule.

In this study, the synthesis weights of tasks are calculated based on three heuristic factors: (a) processing times t_i introduced by Tonge [31], (b) the number of immediate successors IF_i suggested by Helgeson and Birnie [21], and the number of updated tasks for task i , defined as UD_i , is proposed here so that tasks with larger number of updated tasks can be selected with higher priority.

Based on these three heuristic factors and their weighted modulus, λ_1, λ_2 and λ_3 , the synthesis weight of each task can be calculated by Eq. (11), where $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

$$w_i = \lambda_1 \frac{t_i}{\sum_{i' \in CS} t_{i'}} + \lambda_2 \frac{UD_i}{\sum_{i' \in CS} UD_{i'}} + \lambda_3 \frac{IF_i}{\sum_{i' \in CS} IF_{i'}}, \forall i \in CS \tag{11}$$

The task with largest synthesis weight is selected. The proposed approach for generating initial feasible solutions is as follows.

Step 1: Select tasks which have no predecessors or all their predecessors have been allocated from the possible set PS according to Eq. (12);

$$Predecessor_k = \sum_{i=1}^N P_{ik} \tag{12}$$

where, $Predecessor_k$ is equal to the sum of unallocated predecessors of task k at given time. Note that, only if $Predecessor_k$ equals to zero, then task k can be selected into the possible set PS .

Step 2: Derive the candidate set CS from the possible set PS , excluding tasks that do not meet the sequence-dependent constraints: $y_{kl} = 1, SD_{il} = 1, Tabu_k = 1, (i, l, k) \in PS$.

Step 3: If CS is empty, (i.e., all tasks are selected) then stop; otherwise, go to step 4;

Step 4: Calculate the synthesis weights of all tasks in CS by Eq. (12), select the task with the largest w_i , and then append it into the task sequence. If there are several tasks with

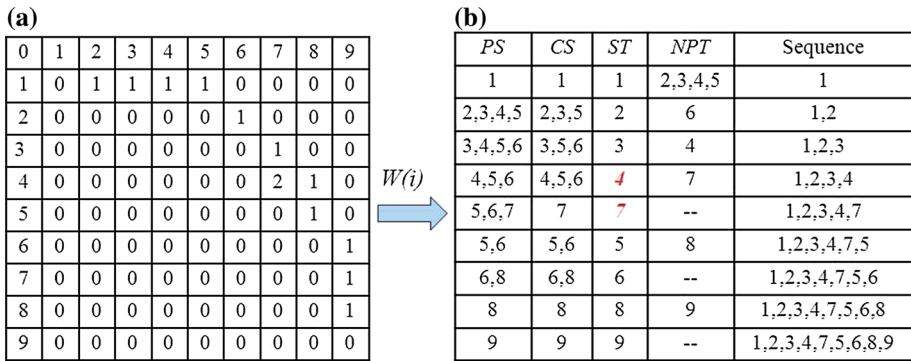


Fig. 8 Formulating feasible task sequence. **a** Precedence matrix. **b** Generation of chromosome sequence

the largest synthesis weight, then select one task randomly. According to the assignment principles about sequence-dependent tasks, two scenarios exist:

- (1) If $SD_{ik} = 1$ and $y_{lk} = 1$, then task l has higher priority than task i ;
- (2) If $SD_{ik} = 1$ and $Tabu_i = 1$, then task k must be selected next; for example, in the 3rd column in Fig. 8b, task 7 must be allocated immediately after task 4.

Step 5: Update PS and CS by removing the selected task and adding new tasks which meet Eq. (12); go to step 3 until termination conditions are satisfied (i.e., sets PS and CS are vacant).

As an example, consider the precedence matrix of Fig. 6c which is shown in Fig. 8a. The number “1” represents that the line is the immediate predecessor of the column, and number “2” shows that the tasks in the line are sequence-dependent to those in the column. By updating the possible set PS and candidate set CS , and appending new possible tasks iteratively, a feasible task sequence is obtained gradually from the selected task ST , as depicted in Fig. 8b.

3.2 Chromosome code and decode

Through the process of initialization outlined in Sect. 3.1, all tasks are ordered into a feasible sequence as represented by (1,2,3,4,7,5,6,8,9) in Fig. 8b. This chromosome simulates the task sequence performed on all ordered stations, without information about the task assignment. To further ascertain the task assignment on each workstation, this chromosome needs to be decoded. Based on the real number coding and decoding processes introduced by Simaria and Vilarinho [23], we apply the process of decoding chromosomes for the minimization of cycle time CT and workload variance WV concurrently:

Step 1: Compute the theoretical minimum cycle time, and set it to CT ;

$$CT = \frac{1}{S} \sum_{i=1}^N t_i$$

Step 2: (a) If no sequence-dependent task is involved, allocate tasks in the feasible sequence to stations from 1st to $(S - 1)th$ so that each station time does not exceed CT ; (b) If tasks i and l are sequence-dependent with i ahead of l , (i.e., $SD_{il} = 1$), assign tasks i and l simultaneously to the same workstation from 1st to $(S - 1)th$; (c) Allocate all the remaining tasks to the last workstation S ;

Fig. 9 Coding and decoding process

Tasks	1	2	3	4	5	6	7	8	9
Station	1	1	1	2	2	3	2	3	3
Coding	↓			↑			Decoding		
Station				1	2			3	
Tasks				1,2,3	4,7,5			6,8,9	
Station time				19	12			16	

- Step 3: Calculate the workstation time $T_j(j = 1, 2, \dots, S)$ and current cycle time CT , $CT = \max(T_j|j = 1, 2, \dots, S)$;
- Step 4: The first task or sequence-dependent tasks, which is/are located in the 1st and 2nd positions in each station (1st station excluded), are reallocated to the current immediate preceding station, and we calculate the new workstation time T_j^* and new cycle time $CT^* = \max(T_j^*|j = 1, 2, \dots, S - 1)$;
- Step 5: If $CT \leq CT^*$, the minimum cycle time CT for the given chromosome is obtained; otherwise, go to Step 2.

The schematic coding and decoding process are shown as Fig. 9.

3.3 Tournament selection with elitism

Selection strategies have different effects on the performance of GA. Generalized selection rules include: proportion selection, level selection, tournament selection, and elite preservation strategy. In this work, two selection mechanisms, elite preservation strategy [32,33] and tournament selection [34–36] have been applied together, and it is executed with two steps:

- Step 1: Best individuals from 15% of the whole population size are inherited directly into the next generation.
- Step 2: The rest of individuals are generated by tournament selection, which is executed as follows.
 - (1) Determine the tournament size ts . Here $ts = 2$. And the selection times $st = P/2$;
 - (2) Randomly choose individuals from the current populations;
 - (3) Compare the objective function value of the selected individuals and choose better chromosome into the next generation.

With the combination of elite preservation strategy and tournament selection, on the one hand, elite preservation strategy aims at attaining that the final population contains the best solutions ever found, because the best individuals at each generation are preserved avoiding the loss of best individuals. On the other hand, the tournament selection has no requirement on the fitness value, and it has randomness to some extent.

3.4 Crossover operator

The crossover operator generates new offspring by exchanging contiguous sections of the chromosomes of parent solutions, so that offspring chromosomes inherit features from parents. In this paper, a two-point crossover is adopted.

In the process of a two-point crossover, first, any two chromosomes are chosen from the population as parents, and then two points are selected as the cut points. Note that, the

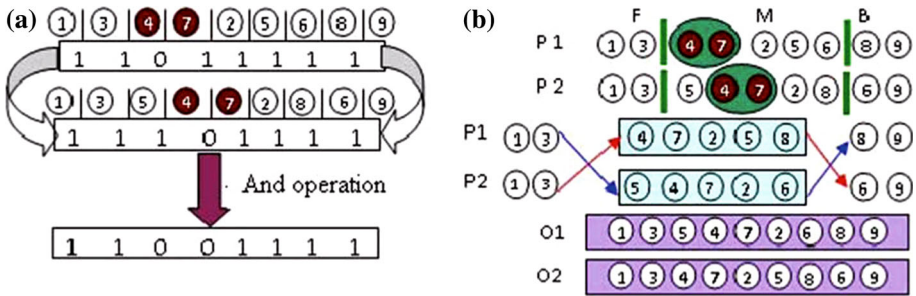


Fig. 10 Crossover operator. **a** And operation between logic strings. **b** Two-points cross over

feasibility of the cut points needs to be verified because sequence-dependent tasks must be performed successively.

3.4.1 Logic string

For sequence-dependent connections between tasks, the following logic string (see Fig. 10a) is proposed.

- Step 1: Choose any two chromosomes from the population as parents $P1$ and $P2$;
- Step 2: For each parent $P1$ and $P2$, set $position_r = 1 (r = 1, 2, \dots, N - 1)$ if sequence-dependent tasks are not involved. Otherwise, set $position_r = 0$.
- Step 3: Carry out logic “AND” operations between these two logic strings (i.e., $position_r = position_{1r} \cap position_{2r}$), resulting in $position_r = 1$ only when both of them are equal to one. Thus, the position with the value of 1 is available for crossover; and the number of available positions is calculated by equation $W = \sum_{r=1}^{N-1} position_r$.

3.4.2 Two-point crossover operator

After available positions have been indicated by logic strings, two positions are randomly selected as the cut points. Each of them divides the parent into three parts: the front (F), middle (M) and back (B). From one parent ($P1$), the F and B parts are inherited directly into the same positions of offspring $O1$, and these elements from the other parent ($P2$) are removed subsequently, and the remaining elements of $P2$ are copied and placed in the M part of $O1$ in the same order in $P2$, so the offspring $O1$ is generated [15, 27]. Similarly, the offspring $O2$ can be generated.

Note that $O1$ and $O2$ are feasible because the task sequence in the middle part has not been changed. The crossover process is shown in Fig. 10b.

3.5 Mutation operator

To avoid convergence to a local optimum and generate new chromosomes, the mutation operator is used. In general, mutation is applied with smaller probability. The mutation operator randomly alters the composition of a chromosome to produce a new offspring, instead of recombining two strings. Herdy [37] proposed four operators: insertion, reciprocal exchange, inversion and displacement. In this article, the insertion mutation is employed.

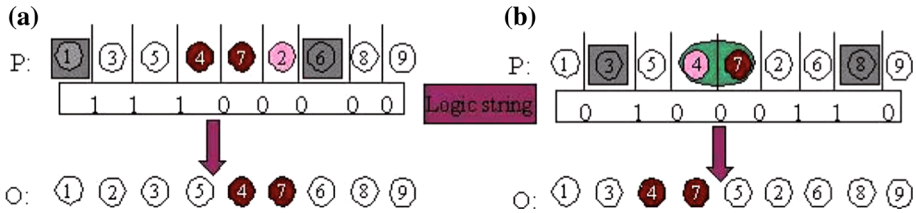


Fig. 11 Mutation operator. **a** Without sequence-dependent tasks. **b** With sequence-dependent tasks

During the insertion mutation operation, first, one chromosome is randomly selected from the population as parent, and then one task is randomly chosen as the mutated task. In order to ensure the feasibility of the population under the constraints of sequence-dependent connections, logic strings are defined first.

3.5.1 Logic string

The logic string shown in Fig. 11 is introduced and generated by the following steps:

Step 1: Choose a chromosome from the population as the parent;

Step 2: In the chromosome, set $position_r = 1 (r = 1, 2, \dots, N - 1)$ and randomly select task i in position r as the mutated task, and set $position_{r-1} + position_r = 0$. There are two cases:

Case 1: without sequence-dependent tasks

If no sequence-dependent task exists (i.e., $SD_{ik} = 0, \forall k$), determine the feasible area (FA) of the mutated task according to its latest predecessor and earliest successor; set $position_n = 0$ when ($n \notin FA$) and select exactly one position from those $position_n = 1 (k \in FA)$.

Case 2: with sequence-dependent tasks

If ($SD_{ik} = 1$ or $SD_{kl} = 1, \forall k$), determine the feasible area (FA) ranging from the latest predecessor and the earliest successor of tasks i and k , and set $position_n = 0 (n \notin FA)$.

Note that, the logic string is obtained and the number of available positions W can be calculated.

3.5.2 Insertion mutation

Upon generation of the available positions, one position from W is randomly selected as the insertion point. There are two cases:

Case 1: without sequence-dependent tasks

If no sequence-dependent relation exists between tasks i and others, insert i into the insertion position. For instance, in Fig. 11a task 2 is selected as the mutation point. According to the combined precedence graph of Fig. 6c, tasks 1 and 6 are the immediate preceding and following task of task 2, respectively.

Case 2: with sequence-dependent tasks

If a sequence-dependent relation exists between tasks i and k , insert these two tasks into the insertion position. This approach ensures that both precedence constraints and sequence-dependent relations are satisfied. For instance, in Fig. 11b, task 4 is selected as the mutation point, and tasks 3 and 8 are their latest preceding and earliest following tasks of the sequence-dependent tasks 4 and 7.

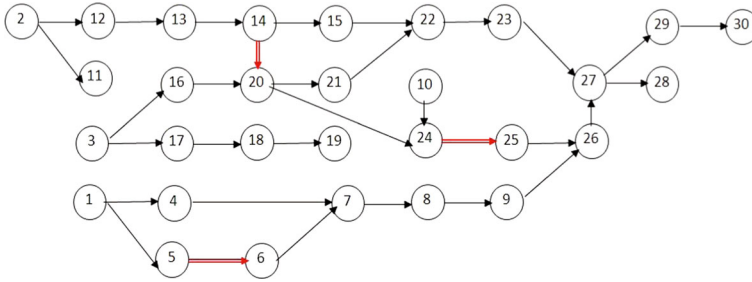


Fig. 12 Modified Sawyer 30 with sequence-dependent tasks

Table 4 Processing times and weighted average task times for Sawyer 30

Task	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
t_A	9.5	1.3	4.8	3.3	1.5	4.5	3.6	0	12.3	0.0	2.5	4.3	6.5	1.7	7.0
t_B	9.5	1.3	4.8	3.3	1.7	4.1	3.6	2.0	12.3	8.0	2.5	4.3	0.0	1.7	7.0
Weighted t_i	9.5	1.3	4.8	3.3	1.58	4.34	3.6	0.8	12.3	3.2	2.5	4.3	3.9	1.7	7.0
Task	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
t_A	1.4	7.8	2.9	1.6	7.0	8.7	3.9	6.4	2.8	8.5	6.7	1.9	9.9	4.6	4.0
t_B	1.4	7.8	2.9	1.6	7.0	8.7	4.1	6.4	2.7	8.5	6.7	1.9	9.9	0.0	4.2
Weighted t_i	1.4	7.8	2.9	1.6	7.0	8.7	3.98	6.4	2.76	8.5	6.7	1.9	9.9	2.76	4.08

4 Computational studies and results

4.1 Benchmark case studies

To evaluate the performance of the proposed hybrid genetic algorithm for MMALB-P with sequence-dependent tasks, 5 benchmark case studies including Thomopoulos 19, Sawyer 30, Lutz 32, Tonge 70 and Arcus 111 from Scholl [38] are investigated. These data sets can be downloaded from <http://www.assembly-line-balancing.de>. Data about demand ratios among models are taken from Kim and Kim [39,40].

To integrate the sequence-dependent tasks into the benchmark case studies, each of them has been modified. For example, in Sawyer 30 as shown in Fig. 12, 30 tasks exist in the combined precedence graph, the number of workstations is three, two models *A* and *B* are assembled, and the demands for each model are 300 and 200 units, respectively. Subsequently, three sequence-dependent task pairs (5, 6), (14, 20) and (24, 25) are appended. In addition, the processing time for every task of each model and the weighted average processing times are listed in Table 4.

4.2 Implementation and parameter settings

The proposed hybrid genetic algorithm is implemented with visual C++, and carried out on a PC with AMD Sempron LE-1250 processor and 512 MB of Ram.

Related parameters are shown in Table 5, where *N*, *M* and *P* represent the number of tasks of the combined precedence graph, the number of models and the population size, respectively. The last two columns denote crossover rate (*cr*) and mutation rate (*mr*).

Note that as the number of tasks increases, then the population size increases.

Table 5 The parameters of tested problems

Problem	N	M	P	cr	mr
Thomopoulos	19	3	100	0.6	0.1
Sawyer	30	3	100	0.6	0.1
Lutz	32	3	100	0.6	0.1
Tonge	70	4	200	0.6	0.1
Arcus	111	5	300	0.6	0.1

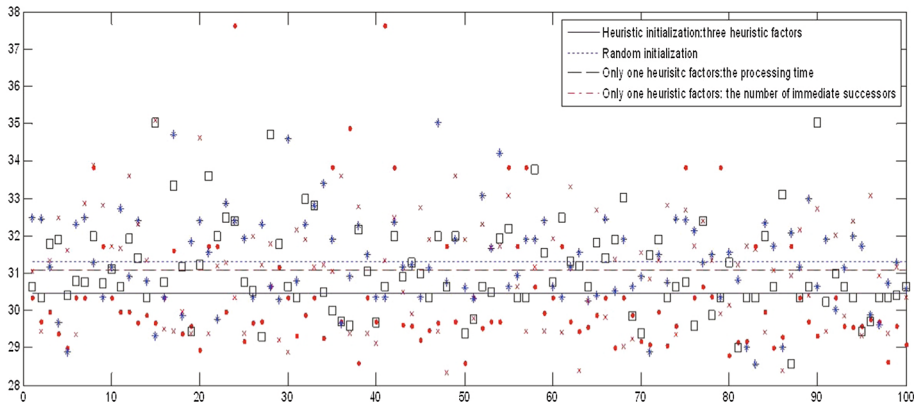


Fig. 13 Initialization comparison of four approaches

4.3 Comparison of different initializations

In Sect. 3.1, the proposed algorithm is initialized by hybridizing 3 predesigned heuristic factors. To test its performance, the case study of Sawyer 30 which features 30 tasks that must be allocated to 5 stations (see Fig. 13), is initialized by four approaches, respectively: (1) the proposed heuristic initialization, (2) random initialization, (3) the task with largest processing time first, and (4) the task with the largest number of successors first.

With each approach, this case study was run 100 times, and thus derived 100 results about cycle time, which are denoted by bullet symbol, asterisk symbol, square box symbol and times symbol, respectively in Fig. 13. The obtained results are different even with the same approach. The mean values of these 400 results are represented by four horizontal lines.

For this problem, the average and standard deviation of each approach are reported in Table 6. From this table, we can observe that the average of the heuristic initialization is 30.4820, which is less than 31.3324, 31.0964 and 31.0902 from the other three approaches, although the standard deviation is larger than that of other three approaches.

In addition, the significance of the differences between the average results are analyzed with Students t test. And the significance and confidence intervals of the t test results are reported in Table 7, where significance is the probability that the observed value of CT could be as small or smaller by chance under the null hypothesis that the mean of CT obtained by H is equal to that by $R/P/S$. ci is a 95 % confidence interval for the true difference in means. The result $h = 1$ means that the null hypothesis is rejected. For example, in the comparison between the heuristic initialization and maximum processing time, the significance is 0.0061, which means that by chance the observed values are more extreme than the one in this example

Table 6 The average and standard deviation of each approach

Initialization method	Heuristic Initialization (H)	Random initialization (R)	Maximum processing time (P)	Maximum number of immediate successors (S)
Average	30.4820	31.3324	31.0964	31.0902
Standard deviation	1.8357	1.2463	1.2410	1.4803

Table 7 The significance and confidence interval of the *t* test results

Comparison	<i>h</i>	sig	ci	
H & R	1	1.7017e−004	−1.2880	−0.4128
H & P	1	0.0061	−1.0514	−0.1774
H & S	1	0.0106	−1.0732	−0.1432

Table 8 Task assignment of modified Sawyer 30

Station	Tasks	<i>T_i</i>	Model A		Model B	
			Tasks	<i>T_i</i>	Tasks	<i>T_i</i>
1	10,2,3,12,16,13, <i>14,20</i>	27.60	2,3,12,16,13, <i>14,20</i>	27.00	10,2,3,12,16, <i>14,20</i>	22.3
2	15,24,25,17,18	28.96	15,24,25,17,18	29.00	15,24,25,17,18	28.90
3	21,1,5,6,22	28.10	21,1,5,6,22	28.10	21,1,5,6,22	28.10
4	4,19,7,8,23,9	28.00	4,19,7,23,9	27.20	4,19,7,8,23,9	29.20
5	26,27,28,11,29,30	27.84	26,27,28,11,29,30	29.60	26,27,28,11,30	25.20

in only 61 of 10,000 similar experiments. And a 95% confidence interval on the mean is [−1.0514, −0.1774]. All these computational studies show that the heuristic initialization outperforms other three approaches.

4.4 Validation studies

With the proposed hybrid genetic algorithm, the case study of Sawyer 30 (see Fig. 12) was solved and the corresponding task assignment result is shown in Table 8.

Note that all sequence-dependent tasks, represented by italic numbers, have been assigned sequentially into the same workstation.

To verify the convergence of the proposed algorithm, the average and best solutions of workload variances (WV) of 100 generations are employed and illustrated in Figs. 14 and 15. Note that, the horizontal axis represents generations and the vertical axis depicts the workload variances (WV). (WV is in logarithmic scale).

It can be seen in Fig. 14 that the average workload variances of four out of the five cases, (except Archus 111), reduce progressively and almost stabilize after the 40th generation. For the largest case, Archus 111, its average workload variance fluctuates among 100 generations, implying that more generations are required for such large-scale case studies. Further experiments show that Archus 111 stabilizes after the 150th generation.

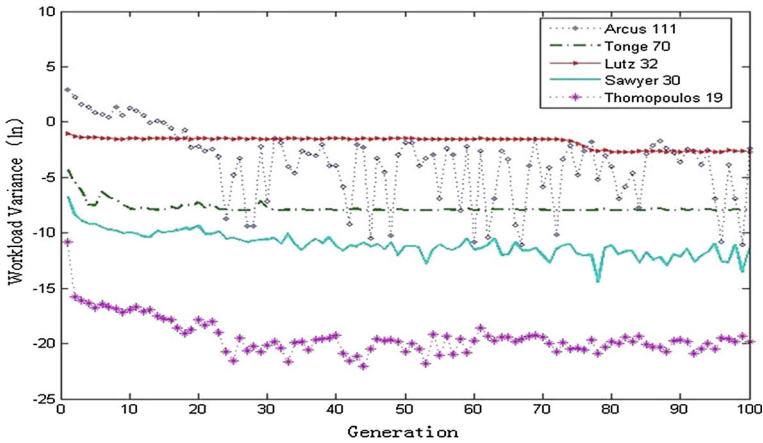


Fig. 14 Convergence of average workload variance

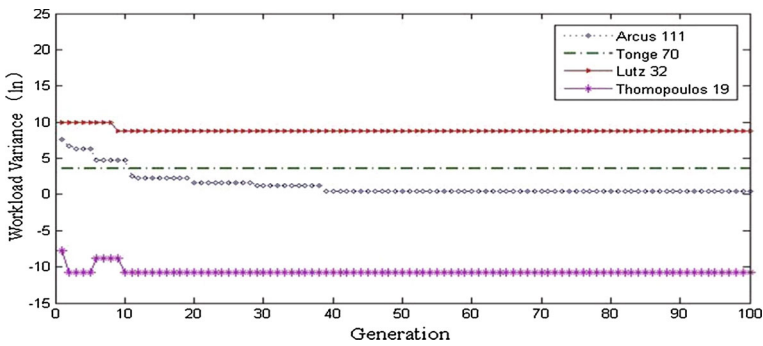


Fig. 15 Convergence of best solutions

Figure 15 depicts the best solutions and it can be seen that four cases, except Sawyer 30, show distinct convergence. For Sawyer 30, its best WV value is 0 and so no natural logarithm exists.

It should be pointed out that the MMALB-P with sequence-dependent tasks has not been addressed in the literature. The mathematical model of Eqs. 1–10 is a Mixed-Integer Linear Programming (MILP) problem and each case of the modified five cases has been solved using the General Algebra Modeling System (GAMS/CPLEX). The computational time is set to 3 days and all case studies were solved to optimality. Hence, the obtained solutions from the MILP model through GAMS/CPLEX represent the global optimum solutions.

From the comparison of the two approaches shown in Table 9, the relative gap is small for most cases, especially for Sawyer 30 with three workstations, Tonge 70 with 23 workstations and Arcus 111 with 23 workstations; the relative gaps are as low as 0. Furthermore, for several cases including Thomopoulos 19, Tonge 70 with three workstations, and Arcus 111 with three workstations, their relative gaps are less than 1%. Finally, the largest relative gap is less than 10%.

Through the comparison between the hybrid genetic algorithm and MILP model using GAMS/CPLEX, we can conclude that although the objective CT obtained by the hybrid genetic algorithm is not as good as that of GAMS/CPLEX, the relative gaps are small,

Table 9 Comparison results of the proposed genetic algorithm and the MILP model using GAMS/CPLEX (with sequence-dependent tasks)

Problem	S	Sequence-dependent tasks	The proposed hybrid genetic algorithm		MILP model using GAMS/CPLEX		Relative gap of CT (%)
			CPU Time(s)		CT		
			CT	CPU Time(s)	CT	CPU Time(s)	
Thomopoulos 19 (3:2:1)	3	(3,9), (11,14)	1.89	0.16	1.883	0.33	0.37
Sawyer 30 (1:1:1)	3	(5,6), (14,20), (24,25)	108	25.28	108	46.35	0
	12		30	62.42	29	105.14	3.45
Lutz 32 (1:1:1)	8	(6,7), (12,13), (17,19), (27,28)	1902	61.48	1836	190.28	3.59
	3	(6,8), (3,68),	1178	217.22	1170	12,327.24	0.68
Tonge 70 (1:1:1)	8	(23,33), (35,36),	455	257.23	439	15,407.89	3.64
	13	(64,65)	289	134.84	272	18,934.52	6.25
	18		227	153.44	216	21,063.85	5.09
	23		216	182.77	216	67,348.12	0
	3	(2,3), (14,24),	48,288	362.25	48,285.35	176,435.67	0.005
Arcus 111 (1:2:4:5:8)	8	(43,49), (57,65),	19,121.5	470.25	18,107.1	194,675.35	5.60
	13	(72,74), (75,79),	11,545	792.83	11,303.9	214,637.94	2.13
	18	(102,106)	8438	1091.88	8234	234,761.21	2.48
	23		6615	1110.53	6615	259,200	0

and the CPU time is much less. For example, as for Arcus 111 with 23 workstations, the CPU time of GAMS/CPLEX is as long as 259,200s, while that of hybrid genetic algorithm is 1110.53 s. Hence, the hybrid genetic algorithm has better computational efficiency than GAMS/CPLEX. These comparisons illustrate that the proposed hybrid genetic algorithm is effective and efficient for solving mixed-model assembly line balancing problem with sequence-dependent tasks.

4.5 Other performance metrics

To further compare the performance, three other metrics are employed: (a) single-point crossover & random initialization (SC&RI), (b) single-point crossover & heuristic initialization (SC&HI), and (c) two-point crossover & random initialization (TC&RI). Comparisons are introduced with other existing best solutions in the literature. Similar to Sect. 4.4, the number of workstations for each case is changed so as to expand the original five cases into 14 case studies.

To describe the problem complexity, other evaluation indicators are introduced to measure the computational complexity, which are order strength (*OS*), flexibility ratio (*FR*), west ratio (*WR*), time interval (*TI*) and weighted efficiency (*WE*) respectively. *OS* measures the relative number of precedence relations in the combined precedence graph, and problems with higher values of order strength, holding intensified constraints, are typically more difficult to assign tasks. *FR* is equal to one minus the order strength. *WR* represents the average number of tasks per station. *TI*, defined as the time interval measure, is a two-part expression indicating the range of task times relative to the cycle time. The last evaluation indicator, *WE*, is introduced to further measure the performance of the assembly line. All five evaluation indicators are calculated as follows:

$$OS = \frac{2PL}{N(N-1)}, \quad FR = 1 - OS, \quad WR = \frac{N}{S}, \quad TI = \left[\frac{t_{min}}{CT}, \frac{t_{max}}{CT} \right],$$

$$WE = \sum_{m=1}^M \left(\frac{q_m \cdot \sum_{i=1}^N t_{im}}{S \cdot CT} \right)$$

As demonstrated in Table 10, *OS* and *FR* are the same as long as the combined precedence graph is predefined, and the west ratios decrease while the time intervals increase with the number of workstations.

As for the quality of the obtained solutions, eight out of 14 instances outperform the current best solutions, which are cited from literature [38].

Moreover, compared with three initialization and crossover approaches of SC&RI, SC&HI and TC&RI, it can be observed that the TC&HI in the proposed algorithm, leads into better solutions for 10 out of 14 instances. It can be seen that the two objectives of cycle time and workload variance are much less, and the weighted efficiency is higher.

5 Conclusions and future research

In real practical production, sequence-dependent tasks are common in mixed-model assembly lines such as automotive and electronic industries, although few studies have been published. In this study, the mathematical model of the mixed-model assembly line balancing problem with sequence-dependent tasks was formulated. A hybrid genetic algorithm has been proposed to solve the mixed-model assembly line balancing problem with sequence-dependent

Table 10 Computational complexity and measurement performance (without sequence-dependent tasks)

Problem	Parameter	Proposed hybrid genetic algorithm										Current best solution	Relative Gap (%)								
		Computational complexity					SC&RI							TC&HI							
		S	Gene	OS	FR	WR	TI	CT	WV	WE	CT			WV	WE	CT	WV	WE	CT		
Thomopoulos 19 (3:2:1)	3	100	0.135	0.865	6.333	0.49	1.89	2.22×10^{-5}	0.996	1.89	2.22×10^{-5}	0.996	1.89	2.22×10^{-5}	0.996	1.89	2.22×10^{-5}	0.996	1.89	0	
Sawyer30 (1:1:1)	3	100	0.145	0.855	10	0.51	108	0	1.00	108	0	1.00	108	0	1.00	108	0	1.00	108	0	
Lutz 32 (1:1:1)	12			2.5	0.83	30	4.5	0.900	30	4.5	0.900	30	4.5	0.900	30	3.667	0.900	28	0.950	1860	7.14
Tongue 70 (1:1:1:1)	8	100	0.141	0.859	4	0.75	1874	6424.50	0.950	1860	4712.75	0.950	1860	4712.75	0.950	1860	4712.75	0.950	1860	0	
Arcus 111 (1:2:4:5:8)	3	200	0.065	0.935	23.33	0.13	1178	32.6667	0.993	1170	0.222	0.999	1178	32.6667	0.993	1170	0.222	0.999	1170	0	
	8			8.75	0.35	456	258.938	0.962	449	93.26	0.977	449	93.26	0.977	441	1.69	0.995	439	0.975	271	0.46
	13			5.385	0.55	285	196.615	0.947	277	15.539	0.975	277	15.539	0.975	277	15.539	0.975	271	0.951	196	2.21
	18			3.889	0.70	221	170.556	0.882	216	121.333	0.903	221	170.556	0.882	205	25.889	0.903	196	0.903	156	4.59
	23			3.043	0.91	170	65.433	0.898	170	65.433	0.898	171	87.462	0.892	169	30.673	0.903	156	0.903	156	8.33
	3	300	0.015	0.985	37	0.11	48294.5	42.021	0.999	48287.1	2.296	0.999	48287.1	2.296	0.999	48286.9	3.893	0.999	50133	-0.04	
	8			13.88	0.30	18495.9	57030.6	0.979	18495.9	57030.6	0.979	18195.1	3398.4	0.995	18195.1	3398.4	0.995	18800	0.965	11570	-0.03
	13			8.538	0.49	11693	234147	0.953	11693	234147	0.953	11542	142335	0.965	11542	142335	0.965	11570	0.954	8376	-0.24
	18			6.167	0.67	8454	58454.26	0.952	8454	58454.26	0.952	8454	58454.26	0.952	8438	40935.7	0.952	8376	0.952	8376	7.35
	23			4.826	0.84	6754	38136.7	0.932	6754	38136.7	0.968	6754	38136.7	0.932	6615	27092.6	0.932	6615	0.952	6615	0

tasks. And the objective is to minimize the cycle time and workload variance simultaneously, considering that sequence-dependent tasks should be allocated to the same workstation. The contributions of this paper are as follows.

- (1) The initial population is improved by hybridizing three heuristics: processing time, number of immediate successors and number of updated tasks. Additionally, with respect to these three heuristic factors, the synthesis weight w_i of task i was used to calculate the weight of each task in the possible set PS , in which λ_1 , λ_2 and λ_3 are introduced as weights for the three terms, respectively.
- (2) During the initialization, encoding & decoding and genetic operators, the precedence constraints and sequence-dependent relations are considered.
- (3) Novel logic strings are introduced for dealing with sequence-dependent relations during crossover and mutation operations to guarantee the feasibility of solutions.
- (4) Numerous computational experiments demonstrate that the quality of initial feasible solution has been improved by hybridizing those three heuristics. The best and average solutions converge fast to the near-optimal or even optimal solutions. Finally, the objective CT and CPU time are compared between the hybrid genetic algorithm and the MILP model solved to optimality using GAMS/CPLEX. All the comparison results illustrate that the proposed hybrid genetic algorithm functions effectively and efficiently for the given problem.

Although the computational studies indicate that the hybrid genetic algorithm is effective and efficient, there are still more improvements for the mixed-model assembly line balancing problem with sequence-dependent tasks. There are several research directions that can be studied in future. Firstly, the deterministic processing times for each task, can be extended to stochastic cases. Secondly, the assembly line may be extended to U-type /Two-sided assembly line systems which are common in practice. Finally, the algorithm can be combined with other algorithms, such as simulated annealing, ant colony algorithm and so on.

Acknowledgments The authors would like to say thanks to the Computer-Aided Systems Laboratory of Princeton University and Dongfeng Peugeot Citroen Automobile Company.

References

1. Salveson, M.E.: The assembly line balancing problem. *J. Ind. Eng.* **6**, 18–25 (1955)
2. Scholl, A.: *Balancing and Sequencing of Assembly Lines*, 2nd edn. Physica, Heidelberg (1999)
3. Wee, T.S., Magazine, M.J.: Assembly line balancing as generalized bin packing. *Oper. Res. Lett.* **1**, 56–58 (1982)
4. Bukchin, Y., Rabinowitch, I.: A branch-and-bound based solution approach for the mixed-model assembly line-balancing problem for minimizing stations and task duplication costs. *Eur. J. Oper. Res.* **174**, 492–508 (2006)
5. Merengo, C., Nava, F., Pozzetti, A.: Balancing and sequencing manual mixed-model assembly lines. *Int. J. Prod. Res.* **37**, 2835–2860 (1999)
6. Vilarinho, P.M., Simaria, A.S.: A two-stage heuristic method for balancing mixed-model assembly lines with parallel workstations. *Int. J. Prod. Res.* **40**, 1405–1420 (2002)
7. Boysen, N., Fliedner, M., Scholl, A.: Assembly line balancing: Which model to use when? *Int. J. Prod. Econ.* **111**, 509–528 (2008)
8. Thomopoulos, N.T.: Mixed-model line balancing with smoothed station assignments. *Manag. Sci.* **16**, 593–603 (1970)
9. Macaskill, J.L.C.: Production-line balancing for mixed-model lines. *Manag. Sci.* **19**, 423–434 (1972)
10. Van Zante-de Fokkert, J., de Kok, T.G.: The mixed and multi model line balancing problem: a comparison. *Eur. J. Oper. Res.* **100**, 399–412 (1997)

11. Noorul, A., Jayaprakash, J., Rengarajan, K.: A hybrid genetic algorithm approach to mixed-model assembly line balancing. *Int. J. Adv. Manuf. Technol.* **28**, 337–341 (2006)
12. Boysen, N., Fliedner, M., Scholl, A.: Assembly line balancing joint precedence graphs under high product variety. *IIE Trans.* **41**, 183–193 (2009)
13. Giard, V., Jeunet, J.: Optimal sequencing of mixed-models with sequence-dependent setups and utility workers on an assembly line. *Int. J. Prod. Econ.* **123**, 290–300 (2010)
14. Ozturk, C., Tunali, S., Hnich, B., Ornek, M.A.: Simultaneous balancing and scheduling of flexible mixed-model assembly lines with sequence-dependent setup times. *Electron. Notes Discrete Math.* **36**, 65–72 (2010)
15. Yolmeh, A., Kianfar, F.: An efficient hybrid genetic algorithm to solve assembly linebalancing problem with sequence-dependent setup times. *Comput. Ind. Eng.* **62**, 936–945 (2012)
16. Tang, Q., Li, J., et al.: Optimization framework for process scheduling of operation-dependent automobile assembly lines. *Optim. Lett.* **6**, 797–824 (2012)
17. Mosadegha, H., Zandieh, M., Ghomia, S.M.T.F.: Simultaneous solving of balancing and sequencing problems with station-dependent assembly times for mixed-model assembly lines. *Appl. Soft Comput.* **12**, 1359–1370 (2012)
18. Hamta, N., Ghomia, S.M.T.F., Jolaib, F., Shirazi, M.A.: A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect. *Int. J. Prod. Econ.* **141**(1), 99–111 (2013)
19. Scholl, A., Boysen, N., Fliedner, M.: The sequence-dependent assembly line balancing problem. *OR Spectr.* **30**, 579–609 (2006)
20. Hadi, G., Erel, E.: A goal programming approach to mixed-model assembly line balancing problem. *Prod. Econ.* **48**, 177–185 (1997)
21. Helgeson, W.B., Birnie, D.P.: Assembly line balancing using the ranked positional weight technique. *J. Ind. Eng.* **12**, 394–397 (1961)
22. Bock, S.: Using distributed search methods for balancing mixed-model assembly lines in the automotive industry. *OR Spectr.* **30**, 551–578 (2006)
23. Simaria, A., Vilarinho, P.: A genetic algorithm based approach to the mixed-model assembly line balancing problem of type II. *Comput. Ind. Eng.* **47**, 391–407 (2004)
24. Zhang, W., Gen, M.: An efficient multiobjective genetic algorithm for mixed-model assembly line balancing problem considering demand ratio-based cycle time. *J. Intell. Manuf.* **22**, 367–378 (2009)
25. Manavizadeh, N., Rabbani, M., Moshtaghi, D., Jolai, F.: Mixed-model assembly line balancing in the make-to-order and stochastic environment using multi-objective evolutionary algorithms. *Expert Syst. Appl.* **39**, 12026–12031 (2012)
26. ElMihoub, T.A., Hopgood, A.A., Nolle, L., Battersby, A.: Hybrid genetic algorithms: a review. *Eng. Lett.* **3**, 2–16 (2006)
27. Akpinar, S., Bayhan, G.M.: A hybrid genetic algorithm for mixed-model assembly line balancing problem with parallel workstations and zoning constraints. *Eng. Appl. Artif. Intell.* **24**, 449–457 (2011)
28. Kilbridge, M.D., Wester, L.: A heuristic method of assembly line balancing. *J. Ind. Eng.* **7**, 292–298 (1961)
29. Moodie, C.L., Young, H.H.: A heuristic method of assembly line balancing for assumptions of constant or variable work element times. *J. Ind. Eng.* **1**, 23–29 (1965)
30. Holland, J.H.: *Adaptation in Natural and Artificial System*. University of Michigan Press, Ann Arbor (1975)
31. Tonge, F.M.: Summary of a heuristic line balancing procedure. *Manag. Sci.* **7**, 21–42 (1960)
32. Yamamoto, A.: A quantitative comparison of loading pattern optimization methods for in-core fuel management of PWR. *J. Nucl. Sci. Technol.* **34**, 339–347 (1997)
33. Yang, S.M., Shao, D.G., Luo, Y.J.: A novel evolution strategy for multiobjective optimization problem. *Appl. Math. Comput.* **170**, 850–873 (2005)
34. Deb, K.: An efficient constraint handling method for genetic algorithms. *Comput. Methods Appl. Mech. Eng.* **186**, 311–338 (2000)
35. Miettinen, K., Makela, M.M., Toivanen, J.: Numerical comparison of some penalty-based constraint handling techniques in genetic algorithms. *J. Glob. Optim.* **27**, 427–446 (2003)
36. Alvarenga, G.B., Mateus, G.R.: Hierarchical tournament selection genetic algorithm for the vehicle routing problem with time windows. In: *Proceedings of the Fourth International Conference on Hybrid Intelligent Systems*, pp. 410–415 (2004)
37. Herdy, M.: Application of the evolution strategy to discrete optimization problems. In: *Proceedings of 1st International Conference Parallel Problem Solving from Nature (PPSN)*, Lecture Notes in Computer Science, vol. 496, pp. 188–192 (1991)
38. Scholl, A.: *Data of assembly line balancing problems*. Working Paper, TH Darmstadt (1993)

39. Kim, Y.K., Kim, S.J., Kim, J.Y.: Balancing and sequencing mixed-model U-lines with a co-evolutionary algorithm. *Prod. Plan. Control* **11**, 754–764 (2000)
40. Kim, Y.K., Kim, J.Y., Kim, Y.: An endosymbiotic evolutionary algorithm for the integration of balancing and sequencing in mixed-model U-lines. *Eur. J. Oper. Res.* **68**, 838–852 (2006)