A robust MILP and gene expression programming based on heuristic rules for mixed-model multi-manned assembly line balancing

Zikai Zhang [a,b], Qiuhua Tang [a,b,*], Manuel Chica [c,d]

[a] Key Laboratory of Metallurgical Equipment and Control Technology (Wuhan University of Science and Technology), Ministry of Education, Wuhan, Hubei, China
[b] Hubei Key Laboratory of Mechanical Transmission and Manufacturing Engineering (Wuhan University of Science and Technology), Wuhan, Hubei, China
[c] Department of Computer Science and Artificial Intelligence, Andalusian Research Institute in Data Science and Computational Intelligence, DaSCI, University of Granada, Granada, 18071, Spain
[d] School of Electrical Engineering and Computing, The University of Newcastle, Callaghan, NSW 2308, Australia

ARTICLE INFO

ABSTRACT

Current dynamic markets require manufacturing industries to organize a robust plan to cope with uncertain demand planning. This work addresses the mixed-model multi-manned assembly line balancing under uncertain demand and aims to optimize the assembly line configuration by a robust mixed-integer linear programming (MILP) model and a robust solution generation mechanism embedded with dispatching rules. The proposed model relaxes the cycle time constraint and designs robust sequencing constraints and objective functions to ensure the line configuration can meet all the demand plans. Furthermore, two solution generation mechanisms, including a task-operator-sequence and an operator-task-sequence, are designed. To quickly find a suitable line configuration, a gene expression programming (GEP) approach with multi-attribute representation is proposed to obtain efficient dispatching rules which are ultimately embedded into the solution generation mechanisms. Experimental results show that solving the proposed MILP model mathematically is effective when tackling small and medium-scale instances. However, for large instances, the dispatching rules generated by the GEP have significant superiority over traditional heuristic rules and those rules mined by a genetic programming algorithm.

© 2021 Published by Elsevier B.V.

1. Introduction

An assembly line, as one of the flow-oriented production systems, is generally dedicated to process large and complex productions such as cars and electronics due to its high-efficiency and flexible attributes. In an assembly line, products are moved by a transportation system and flows through a set of workstations to perform their tasks [1]. Finding the best configuration of an assembly line is not trivial and the assembly line balancing problem (ALBP) models how to better balance the tasks of the line in workstations. Concretely, the ALBP aims to optimize some specific objectives such as line efficiency and workload while taking task assignment, precedence relation, and cycle time constraints into account. In this problem, each task must be assigned to only one workstation and is executed after all its predecessors complete.

Additionally, the time required by the tasks of each workstation cannot exceed the cycle time.

While ALBP has been studied extensively, it is a simplification of the industrial reality and hence, it presents many practical limitations. For instance, ALBP assumes that an assembly line only produces one homogeneous product and is a one-sided serial line layout within single man stations [2]. Consequently, existing research have focused on variants that add realistic characteristics into ALBP, such as worker assignment [3–5], robot assignment [6–8], mixed-model [9–11], ergonomic risks [12,13], setup times [14, 15] and multi-manned workstations [1,16].

From the previous family of variants, the multi-manned assembly line balancing problem (MALBP) is dedicated to model large-size industries having a high volume of products such as the automotive industry [17]. The MALBP has four advantages over the traditional ALBP: (1) effective reduction of the production line length, (2) effective use of space, (3) increased yield and (4), reduction of the time required for material handling and its setting [18]. Hence this paper focuses on tackling the MALBP, where each workstation is equipped with multiple operators.

Fig. 1 presents an illustrative layout of one multi-manned assembly line. These operators can then accomplish more than one task in the same workstation. The admitted maximum number of operators in each multi-manned workstation is predefined by the system designer according to the real production. If each workstation is assigned to two operators, this problem is also referred as the two-sided assembly line balancing problem.

Existing works on MALBP mainly focus either on a single product or on mixed-model products. However, they neglect that products' demand is uncertain and can suffer from daily variations. In practice, the products' demand is usually dynamic and uncertain in many managerial and operational scenarios [19]. In a multi-manned assembly line, designed to assemble mixed-model products where all the tasks are fixed and assigned to a specific operator and workstation, if the demand changes, task times also change, and the original designed line configuration cannot successfully assemble all the required products. This situation will generate scheduling problems and the company could suffer economic losses.

These challenges encourage us to study the mixed-model multi-manned assembly line balancing problem under uncertain demand (r-MMALBP). There is not any existing research investigating this problem. When providing a robust solution to assembly line balancing, current research typically makes use of exact methods and meta-heuristics. In this work, we propose an efficient approach to solve the problem by dispatching efficient heuristic rules, frequently used due to low computational efforts and convenient implementation [20]. Therefore, this work mainly has the following two contributions:

- The proposal of a novel mixed-integer linear programming (MILP) mathematical model to obtain robust line configurations. The proposed MILP model relaxes the cycle time constraint and designs robust sequencing constraints and objective functions to ensure production is not interrupted when product demand changes.
- The design of a gene expression programming (GEP) with multi-attribute representation to mine efficient dispatching rules. The obtained rules are embedded into two new designed solution generation mechanisms to quickly obtain robust efficient line configuration.

We perform an extensive computational study based on a set of benchmark instances to show the benefits of the problem formulation and the solving approach. This benchmark has 30 training instances and 239 test instances. The training instances are used as a training set to evolve different rules by the GEP algorithm. The 239 test instances are used to test the performance of the proposed MILP model, solution generation mechanisms and dispatching rules. Experimental results show that the proposed MILP model is effective when tackling small-scale instances, and the dispatching rules embedded into the task-operator-sequence solution generation have significant superiority over the traditional heuristic rules and the evolutionary rules by genetic programming.

The remainder of this paper is structured as follows. Section 2 reports the literature review. Section 3 defines the new problem and presents a robust MILP model. Section 4 introduces two designed solution generation mechanisms. Section 5 details the proposed GEP algorithm. Section 6 reports the computational study. Finally, Section 7 concludes with the key findings and suggests future work.

2. Literature review

This section is devoted to present current research on multi-manned assembly line balancing and approaches to cope with uncertain demand. The section first presents a review on the multi-manned assembly line balancing in Section 2.1. Later, we report the latest developments of robustness and uncertain demand for mixed-model multi-manned assembly line balancing in Section 2.2. Finally, we summarize the main optimization methods for multi-manned assembly line balancing in Section 2.3.

2.1. Multi-manned assembly line balancing

The work of Akagi, Osaki and Kikuchi [21] was the initial proposal for an alternative way to assign multiple operators to each workstation. Afterwards, many works were focused on the multi-manned assembly line balancing problems to increase the production rate. Akagi, Osaki and Kikuchi [21] also proposed a parallel assignment method to achieve a higher production rate. This method involves two phases. The first one aims to assign tasks into workstations under the multi-stage upper time limits. The second one assigns a task to specific operators in each workstation. Based on the latter publication, Dimitriadis [17] examined the assembly line balancing with multi-manned workstations and observed that this new line configuration can shorten the line length, improve the space utilization, reduce the amount of throughput time and work in process and lower material handling costs.

According to the objective function, the research can be divided into two categories: type-I with a fixed cycle time and type-II with a given number of stations. For the Type-I with a fixed cycle time, Fattahi, Roshani and Roshani [22] developed an ant colony optimization algorithm; Kellegöz and Toklu [23] proposed an efficient branch and bound algorithm combined a branching scheme and some dominance and feasibility criteria; Roshani, et al. [2] designed a simulated annealing algorithm (SA); And Kellegöz [24] and Chen [18] further put forward two improved versions of SA; Kellegöz and Toklu [25] introduced a priority rule-based constructive heuristic combined with a genetic algorithm-based solution procedure; Michels, et al. [16] developed a benders' decomposition algorithm with combinatorial cuts. With respect to the type-II problem (i.e., with a given number of stations), Roshani and Giglio [26] proposed two variants of simulated annealing, and Li, Wang and Yang [27] designed a simulating annealing algorithm embedded with a new feasible neighborhood generation. This generation contained two stages including encode and adjacent move to guarantee the feasibility of each new solution.

All the previous research focused on the simple multi-manned assembly line balancing problem. However, there are some uncertain factors the original MALBP cannot deal with in practical production systems. Thus, the original MALBP has been extended in several aspects. Roshani and Ghazi Nezami [28] studied mixed-model MALBP and designed a simulated annealing algorithm to minimize the total number of workstations and operators. Chen, Cheng and Li [29] investigated the resource-constrained MALBP where a feasible line configuration does not only contain task assignments but also includes resource allocations (such as machinery, equipment and tool allocations). Lopes, et al. [1] introduced flexible MALBP and presented a novel model-based heuristic procedure to minimize the line length. Sahin and Kellegoz [30] investigated resource investment in MABLP and adapted a particle swarm optimization algorithm hybrid a special constructive heuristic to reduce the cost of workstations and required renewable resources.
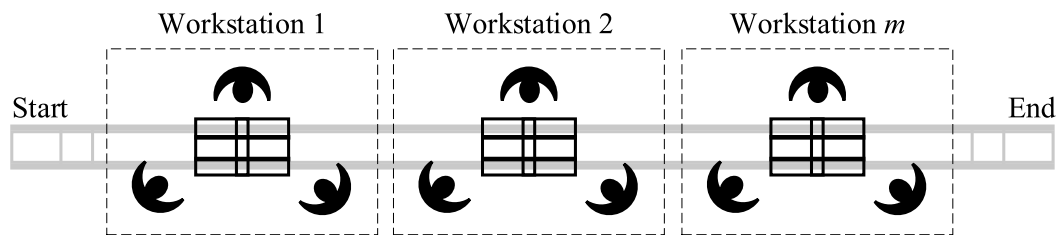
Fig. 1. An illustration of a multi-manned assembly line layout.

2.2. Robustness and uncertain demand for mixed-model assembly line balancing

Single model assembly lines are employed to assemble standardized homogeneous products. However, today's customized production requires the assembly lines to assemble heterogeneous products with similar characteristics [31]. Hence, mixed-model assembly lines are designed to address the above situation. Recently, Ramezanian and Ezzatpanah [32] considered worker assignment into a mixed-model assembly line balancing problem and addressed this problem with the minimization of cycle time and operating costs. Tang, et al. [33] designed a hybrid genetic algorithm to minimize the cycle time and workload variance of mixed-model assembly line balancing. Yuan, et al. [31] proposed an effective hybrid honey bee mating optimization for mixed-model two-sided assembly line balancing. Buyukozkan, et al. [34] investigated the lexicographic bottleneck mixed-model assembly line balancing problem and applied artificial bee colony and tabu search algorithms to obtain a smooth workload distribution. Delice, et al. [35] developed a modified particle swarm optimization to deal with mixed-model two-sided assembly line balancing. Kucukkoc, et al. [9] studied the mixed-model two-sided assembly line balancing with underground workstations and designed an ant colony optimization algorithm to minimize the number of mated workstations and that of workstations. Tiacci and Mimmi [36] integrated the ergonomic risks and stochastic mixed-model assembly line balancing/sequencing, and then applied a genetic algorithm to obtain line configurations. Chen, et al. [10] aimed at the mixed-model assembly line balancing in TFT-LCD module process and proposed a two-phase adaptive genetic algorithm to solve it. Finally, Samouei and Ashayeri [11] considered semi-automated operations into mixed-model assembly line balancing and developed two new mathematical models to minimize the fixed and variable costs.

All the previous research address certain demand in mixed-model balancing. Only limited research focused on the uncertain demand in the literature. Chica, et al. [37] integrated the uncertain demand of mixed-model and space attributes in assembly line balancing and achieved the robustness functions according to the number of overloaded workstations and overloaded size of workstations. Then, Chica, et al. [19] further developed a multi-objective robust model and evolutionary algorithm for the mixed-model time and space assembly line balancing problem under uncertain demand. Moya, Chica and Bautista [38] solved the car sequencing problem under uncertain partial demand and proposed an ant colony optimization with a local search procedure to tackle it.

Based on this review and up to the best of our knowledge, there is no prior research focusing on a mixed-model MALBP under uncertain demand. One of the reasons is mixed-model MALBP is a recent formulation, shown to be important in mass customization production nowadays, and the solving of assembly line balancing under product demand uncertainty only received attention in the last years [19,38]. Therefore, this work focuses on proposing a robust mathematical model for the mixed-model MALBP under uncertain demand, called r-MMALBP.

2.3. Methods applied to multi-manned assembly line balancing

In the assembly line balancing domain, many solution procedures, including mathematical programming approaches, heuristic, and meta-heuristic algorithms, were proposed in the recent literature.

For the mathematical programming approaches, they mainly include a branch and bound method [23], a benders' decomposition algorithm [16] and a constraint programming model [39]. These approaches are effective in solving small- or middle-scale instances. When the problem scales increase, the computation time to obtain an optimal solution increases exponentially. In this situation, most researchers turned their attention to the meta-heuristics. The popular meta-heuristic algorithms contain genetic algorithm (GA) [40], simulated annealing algorithm (SA) [2,18, 24,26], particle swarm optimization (PSO) [30] and ant colony optimization (ACO) [22,41]. These meta-heuristic algorithms are proved to be effective in solving assembly line balancing and can obtain a high-quality solution. However, there are still some limitations in practical production. Due to their time consuming and the complexity of coding and optimization mechanism, these algorithms are difficult to generalize to practical production. The enterprises are more inclined to use heuristic rules to organize production.

Regarding heuristics, Kellegöz and Toklu [25] designed a priority rule-based constructive heuristic to balance the multi-manned assembly lines. Lopes, et al. [1] developed a novel model-based heuristic procedure to short the line lengths. Besides, Baykasoğlu and Özbakır [42] used the genetic programming (GP) to discover task assignment rules to solve the assembly line balancing problems quickly and effectively. Hence, considering the low computational time and convenient implementation of dispatching rules, this paper aims to mine efficient rules by a GEP to deal with the r-MMALBP.

3. r-MMALBP definition

MALBP states that a set of tasks with processing time and precedence graph must be assigned into a set of multi-manned workstations. In each multi-manned workstation, u_{max} operators can perform different tasks simultaneously. Tasks assignments should satisfy the following constraints: (i) precedence relation constraint that each task starts after all its immediate predecessors have completed, (ii) cycle time constraint that no workstation time is greater than cycle time, and (iii) task assignment constraint that each task is only assigned to one operator in one workstation.

In mixed-model MALBP, a set of mixed products needs to be assembled and the task time depends on the products. However, due to the dynamic market, the product demand always changes, which will impact the capacity and hence lead to an unsatisfied cycle time constraint [11]. Hence, it is essential to obtain a robust line configuration that can ensure production under different demands. This work makes use of the methodology presented in

Table 1
Description of all the r-MMALBP parameters.

Parameters	Description
i, h	Index of task.
j	Index of workstation.
k	Index of operator.
e	Index of demand plan.
q	Index of product type.
I	Set of all the tasks.
J	Set of all the workstations.
K	Set of all the operators in each workstation.
E	Set of all demand plans.
Q	Set of all product types.
P^0	Set of tasks that have no immediate predecessors.
$P(i)$	Set of immediate predecessors of task i.
n	The total number of tasks.
m	The upper bound on the number of workstations.
u_{max}	The admitted maximum number of operators in each workstation.
d_{qe}	The demand of product model q in plan e.
CT	Cycle time.
Δ_{CT}	The maximum exceeding cycle time allowed for all the stations.
t_{iq}	The processing time of task i in product model q.
$\bar{t}_{ie}$	The average processing time of task i in plan e, $\bar{t}_{ie} = \frac{1}{\sum_{q \in Q} d_{qe}} \sum_{q \in Q} d_{qe} t_{iq}$.

Table 2
Description of all variables in r-MMALBP model.

Variables	Description
FT_{ie}	(continuous) The finishing time of task i in plan e.
W_{ih}	(binary) If task i and h are assigned to the same operator and task i is performed before task h, the value is 1. 0, otherwise.
X_{ijk}	(binary) If task i is operated by operator k at workstation j, the value is 1. 0, otherwise.
Y_{jk}	(binary) If operator k is employed in workstation j, the value is 1. 0, otherwise.
Z_j	(binary) If workstation j is opened, the value is 1. 0, otherwise.
G_{ie}	(binary) If the finishing time of task i in plan e exceeds the cycle time, the value is 1. 0, otherwise.
R_{je}	(binary) If the finishing time of workstation j in plan e exceeds the cycle time, the value is 1. 0, otherwise.
O_{jke}	(binary) If the finishing time of operator k of workstation j in plan e exceeds the cycle time, the value is 1. 0, otherwise.
XG_{ijke}	(binary) If the finishing time of task i in plan e exceeds the cycle time and task i is operated by operator k at workstation j, the value is 1. 0, otherwise.

Chica, et al. [19], where authors define a set of demand plans to model changes in the mixed products' demand.

In the r-MMALBP model, the cycle time is first relaxed to permit a higher workload than the allowed maximum cycle time. Then, two robust objective functions are optimized to ensure flexibility in all the demand plans. Finally, the sequencing constraints are extended from one demand plan to multiple demand plans. The goal is to minimize the number of operators and workstations and that of the operators and workstations whose finishing times exceed cycle time where some constraints exist such as precedence relation and task assignment constraints. Note that, since the proposed model aims to obtain a solution that can be applied in uncertain demand plans, we name it a robust model. Meanwhile, the corresponding objective functions, the number of operators and workstations whose finishing times exceed cycle time, are named robust objective functions.

3.1. Assumptions and notations

The corresponding assumptions for our r-MMALBP model are as follows:

1. Mixed products are assembled on the multi-manned assembly line.
2. The task time and the precedence relationships between tasks are known deterministically, the task time depends on products.
3. The travel times of operators are ignored.
4. Parallel tasks and parallel workstations are not allowed.
5. Each task must be assigned to one operator in one workstation.
6. Product demand plans are determined.
7. The admitted maximum number of operators in each workstation is determined.

Additionally, Table 1 shows the definition of all the parameters of the model and Table 2 presents its variables.

3.2. Cycle time relaxation

In most real-world cases, we cannot ensure that the mixed products of any demand plan are completed within the cycle

time. If there is a workstation whose workload exceeds the cycle time, then the production cannot be completed smoothly. In this situation, we relax the original cycle time to an allowed maximum cycle time: $CT + \Delta_{CT}$. Hence, each workstation time in all demand plans is in $(0, CT + \Delta_{CT}]$, satisfying the cycle time constraint. We introduce Eqs. (1) and (2) as the new robust cycle time constraint:

$$FT_{ie} \leq CT + \Delta_{CT} G_{ie}, \forall i \in I, e \in E \tag{1}$$

$$FT_{ie} > CT \cdot G_{ie}, \forall i \in I, e \in E \tag{2}$$

In the above equations, the finishing time of each task in all demand plans are limited within the $CT + \Delta_{CT}$. If the finishing time is in $(CT, CT + \Delta_{CT}]$, the variable G_{ie} is equal to 1. And these constraints generate a linear reformulation.

Property 1. *If $\Delta_{CT}=0$ and $|E| =1$, the r-MMALBP is converted to a mixed-model MALBP.*

Equations (1) and (2) define the robust cycle time constraint, and also limit the variable G_{ie}. Subsequently, the variable G_{ie} is used to limit the variables O_{jke} and R_{je} which determine two of the objective functions in our model. For the variable O_{jke}, constraints (3) to (5) introduce a new variable XG_{ijke} which defines whether the finishing time of task i in plan e exceeds the cycle time and task i is operated by operator k at workstation j. Then, if $G_{ie} = 1$ and $X_{ijk} = 1$, constraint (3) limits that XG_{ijke} must equal to 1; otherwise, constraints (4) and (5) limit $XG_{ijke} = 0$.

$$XG_{ijke} \geq G_{ie} + X_{ijk} - 1, \forall i \in I, j \in J, k \in K, e \in E \tag{3}$$

$$XG_{ijke} \leq G_{ie}, \forall i \in I, j \in J, k \in K, e \in E \tag{4}$$

$$XG_{ijke} \leq X_{ijk}, \forall i \in I, j \in J, k \in K, e \in E \tag{5}$$

Based on the variable XG_{ijke}, we control the variable O_{jke} by the linearization equations (6) and (7). For each operator k in workstation j of demand plan e, if there is one task assigned to this operator and its finishing time exceeds the original cycle time, the finishing of this operator exceeds the original cycle time and hence $O_{jke} = 1$. Otherwise, it shows that this operator finishes all its assigned tasks within the cycle time.

$$\sum_{i \in I} XG_{ijke} \leq nO_{jke}, \forall j \in J, k \in K, e \in E \tag{6}$$

$$\sum_{i \in I} XG_{ijke} \geq O_{jke}, \forall j \in J, k \in K, e \in E \tag{7}$$

Following the similar limitation of the variable O_{jke}, the variable R_{je} is controlled by the linearization of constraints defined in Eqs. (8) and (9).

$$\sum_{k \in K} O_{jke} \leq u_{max} R_{je}, \forall j \in J, e \in E \tag{8}$$

$$\sum_{k \in K} O_{jke} \geq R_{je}, \forall j \in J, e \in E \tag{9}$$

For other constraints such as sequencing and precedence relation constraints, an average task time of each demand plan $\bar{t}_{ie}$ is defined by $\frac{1}{\sum_{q \in Q} d_{qe}} \sum_{q \in Q} d_{qe} t_{iq}$, and then the robustness of these constraints is achieved by replacing the original task time with $\bar{t}_{ie}$ in each demand plan. The original constraints of the multi-manned assembly line balancing problem can be found in Chen [18]. Constraints for the robust problem are presented in Section 3.4.

3.3. Robustness objective functions

In our model, four objective functions are involved: minimizing the total number of operators f^1, minimizing the number of workstations f^2, minimizing the total number of operators whose finishing times exceed cycle time f^3, and minimizing the total number of workstations whose finishing times exceed cycle time f^4. The latter two objectives, f^3 and f^4, are the ones considering the robustness of the solutions with different demand plans and therefore, they help to reflect the ability of the proposed model to cope with uncertain demand planning. These functions are defined in Eqs. (10) to (13) where the former two functions are regular objectives in MALBP and the latter two are new robustness objective functions.

$$f^1 = \sum_{j \in J} \sum_{k \in K} Y_{jk} \tag{10}$$

$$f^2 = \sum_{j \in J} Z_j \tag{11}$$

$$f^3 = \sum_{j \in J} \sum_{k \in K} \sum_{e \in E} O_{jke} \tag{12}$$

$$f^4 = \sum_{j \in J} \sum_{e \in E} R_{je} \tag{13}$$

Lemma 1. *The lower bound of the total number of operators is* $\max_{e \in E} \left\lceil \frac{\sum_{i \in I} \bar{t}_{ie}}{CT + \Delta_{CT}} \right\rceil$.

Proof. For each demand plan, all the tasks should be assembled by only one operator and the finishing times are relaxed to $CT + \Delta_{CT}$. The number of operators required in demand plan e is defined as f^1. Then, the required time $f^1(CT + \Delta_{CT})$ cannot be less than the total task time $\sum_{i \in I} \bar{t}_{ie}$. In this case, the low bound of the total number of operators in demand plan e is $\left\lceil \frac{\sum_{i \in I} \bar{t}_{ie}}{CT + \Delta_{CT}} \right\rceil$. And hence the final lower bound of the total number of operators is the maximum value under all demand plans and equal to $\max_{e \in E} \left\lceil \frac{\sum_{i \in I} \bar{t}_{ie}}{CT + \Delta_{CT}} \right\rceil$.

Lemma 2. *The lower bound of the total number of workstations is* $\max_{e \in E} \left\lceil \frac{\sum_{i \in I} \bar{t}_{ie}}{u_{max}(CT + \Delta_{CT})} \right\rceil$.

Proof. The maximum available workload of each workstation is $u_{max}(CT + \Delta_{CT})$. In other words, the workstation runs with full load. The number of workstations required in demand plan e is defined as f^2. Then, since the available time $f^2 u_{max}(CT + \Delta_{CT})$ must be greater than or equal to the total task time $\sum_{i \in I} \bar{t}_{ie}$, the lower bound of f^2 can be defined as $\left\lceil \frac{\sum_{i \in I} \bar{t}_{ie}}{u_{max}(CT + \Delta_{CT})} \right\rceil$. And the final lower bound of the total number of workstations is the maximum value under all demand plans and equal to $\max_{e \in E} \left\lceil \frac{\sum_{i \in I} \bar{t}_{ie}}{u_{max}(CT + \Delta_{CT})} \right\rceil$.

Lemma 3. *The lower bound of the robustness objective functions is 0.*

Proof. It is assumed that in all demand plans Δ_{CT} is equal to 0, then the cycle time constraints are converted to $FT_{ie} \leq CT, \forall i \in I, e \in E$, which means the workloads of all employed operators and workstations are not allowed to exceed the original cycle time. Hence, the low bound of the robustness objective functions is 0.

In this paper we consider the above-mentioned objectives have the same importance and hence, these functions can be normalized, as done in Eq. (14). For the weighted function, we refer to Chen [18] and Fattahi, Roshani and Roshani [22] where they normalized the different objectives by their upper bound if the functions have the same importance. Therefore, the function to be minimized in this paper is the one of Eq. (14).

$$\text{minimize } \frac{f^1}{n \cdot u_{max} + 1} + \frac{f^2}{n + 1} + \frac{f^3}{n \cdot u_{max} |E| + 1} + \frac{f^4}{n \cdot |E| + 1} \tag{14}$$

3.4. Constraints of the model

Apart from those constraints related to the cycle time relaxation (i.e., those from Eqs. (1) to (9) in Section 3.2), the additional constraints of the r-MMALBP model are stated as follows:

$$\sum_{j \in J} \sum_{k \in K} X_{ijk} = 1, \forall i \in I \tag{15}$$

$$\sum_{j \in J} \sum_{k \in K} (j X_{hjk}) - \sum_{j \in J} \sum_{k \in K} (j X_{ijk}) \leq 0, \forall i \in I - P^0, h \in P(i) \tag{16}$$

$$FT_{ie} - FT_{he} + Max\left(1 - \sum_{k \in K} X_{ijk}\right) + Max\left(1 - \sum_{k \in K} X_{hjk}\right) \tag{17}$$
$$\geq \bar{t}_{ie}, \forall i \in I - P^0, h \in P(i), j \in J, e \in E$$

$$FT_{he} - FT_{ie} + Max(1 - X_{ijk}) + Max(1 - X_{hjk}) \tag{18}$$
$$+ Max(1 - W_{ih}) \geq \bar{t}_{he}, \forall (i, h)|i \neq h \wedge i, h \in I, j \in J, k \in K, e \in E$$

$$FT_{ie} - FT_{he} + Max(1 - X_{ijk}) + Max(1 - X_{hjk}) + Max W_{ih} \tag{19}$$
$$\geq \bar{t}_{ie}, \forall (i, h)|i \neq h \wedge i, h \in I, j \in J, k \in K, e \in E$$

$$\sum_{i \in I} X_{ijk} \leq n \cdot Y_{jk}, \forall j \in J, k \in K \tag{20}$$

$$\sum_{i \in I} X_{ijk} \geq Y_{jk}, \forall j \in J, k \in K \tag{21}$$

$$Y_{j,k+1} \leq Y_{jk}, \forall j \in J, k = 1, 2, \ldots, u_{max} - 1 \tag{22}$$

$$\sum_{i \in I} \sum_{k \in K} X_{ijk} \leq (n u_{max}) Z_j, \forall j \in J \tag{23}$$

$$\sum_{i \in I} \sum_{k \in K} X_{ijk} \geq Z_j, \forall j \in J \tag{24}$$

$$Z_{j+1} \leq Z_j, \forall j = 1, 2, \ldots, m - 1 \tag{25}$$

$$FT_{ie} \geq \bar{t}_{ie}, \forall i \in I, e \in E \tag{26}$$

$$X_{ijk} \in \{0, 1\}, \forall i \in I, j \in J, k \in K \tag{27}$$

$$W_{ih} \in \{0, 1\}, \forall (i, h)|i \neq h \wedge i, h \in I, j \in J, k \in K \tag{28}$$

$$Y_{jk} \in \{0, 1\}, \forall j \in J, k \in K \tag{29}$$

$$Z_j \in \{0, 1\}, \forall j \in J \tag{30}$$

$$G_{ie} \in \{0, 1\}, \forall i \in I, e \in E \tag{31}$$

$$R_{je} \in \{0, 1\}, \forall j \in J, e \in E \tag{32}$$

$$O_{jke} \in \{0, 1\}, \forall j \in J, k \in K, e \in E \tag{33}$$

$$XG_{ijke} \in \{0, 1\}, \forall i \in I, j \in J, k \in K, e \in E \tag{34}$$

Constraint (15) ensures that each task is assigned exactly to one operator at one workstation. Constraint (16) limits the precedence relation constraint that each task starts after all its predecessors have been completed. Constraints (17)–(19) control the sequence-dependent starting time of each task in all demand plans. For each pair of tasks i and h, if task h is the immediate predecessor of task i and they are assigned to the same workstation, constraint (17) ensures that task i starts after finishing the task h in all demand plans. Constraints (18)–(19) limit each pair of tasks that have no precedence relation and are assigned to the adjacent position of the same operator. Among them, *Max* is a very large positive number. Constraints (20)–(22) define that Y_{jk} is equal to 1 if tasks are assigning to the kth operator of jth workstations, and the operators are loaded in an increasing manner. Constraints (23)–(25) confine the variable Z_j, similar to the Y_{jk}. Constraint (26) ensures that the finishing time of each task is greater than or equal to the processing time of each task. Constraints (27)–(33) state the binary variables.

3.5. A numerical example

We present a numerical example to give a better insight of the r-MMALBP. The example shows a mixed-model MALBP with 10 products, 9 tasks, and 3 demand plans. Table 3 presents the processing time and precedence graph between the tasks while Table 4 shows the demand values for the available products (columns) in three different demand plans (rows). If some tasks for some products are not needed, the corresponding values are set to zero in Table 3. For instance, task 1 is not needed for products 7, 9 and 10; task 3 is not needed for products 3, 6 and 8; and task 7 is just needed for product 1. We also assume in this example that the admitted maximum number of operators in each workstation is set 3, and the original cycle time and Δ_{CT} are 3.7 and 0.6, respectively. Finally, Table 5 shows the average task time for each demand plan, pre-calculated according to the definition of $\bar{t}_{ie}$.

Figs. 2 to 4 provide the details of the obtained line configuration by presenting the Gantt charts for the three demand plans. The green vertical line of the figures means the original cycle time while the red one is the allowed maximum cycle time. Specifically, for workstation 1, tasks 1 and 2 are assembled by operator 1; task 3 is assembled by operator 2; task 4 is assembled by operator 3. For workstation 2, tasks 5 and 8 are assigned to operator 1, and tasks 6 and 7 are assigned to operator 3. For workstation 3, only task 9 belongs to operator 1. To sum up, this line configuration includes 6 operators and 3 workstations. We can see operators 2, 2 and 1 cannot complete the assembly work within the original cycle time for demand plan 1, 2, and 3, respectively. In other words, there are 5 operators and 5 workstations with finishing time exceeding the available cycle time.

The values for the four weighted functions composing the model objective are: $f^1 = 6$, $f^2 = 3$, $f^3 = 5$ and $f^4 = 5$. Besides, it is observed that although there are some operators or workstations whose finishing times exceed the original cycle time,

Table 3
The processing time and precedence graph of a numerical example.

Task i	Immediate predecessors $P(i)$	Processing time of task i in product model q (t_{iq})									
		1	2	3	4	5	6	7	8	9	10
1	0	5	1	3	1	1	3	0	1	0	0
2	1	3	2	1	0	1	1	0	2	2	2
3	1	4	3	0	2	3	0	3	0	1	3
4	1	5	1	1	3	0	4	1	3	0	0
5	2, 3	4	2	3	2	1	2	2	0	0	0
6	4	5	2	1	1	4	3	0	0	1	2
7	6	1	0	0	0	0	0	0	0	0	0
8	5	4	2	3	3	3	2	2	2	3	1
9	8	6	3	4	4	1	0	4	0	4	1

Table 4
The demand plans of the numerical example.

Demand plan e	Products									
	1	2	3	4	5	6	7	8	9	10
1	27	8	13	17	19	21	12	22	14	24
2	18	19	12	12	25	14	10	8	11	21
3	18	29	23	6	7	6	8	25	8	8

Table 5
The average processing time of task i in demand plan e.

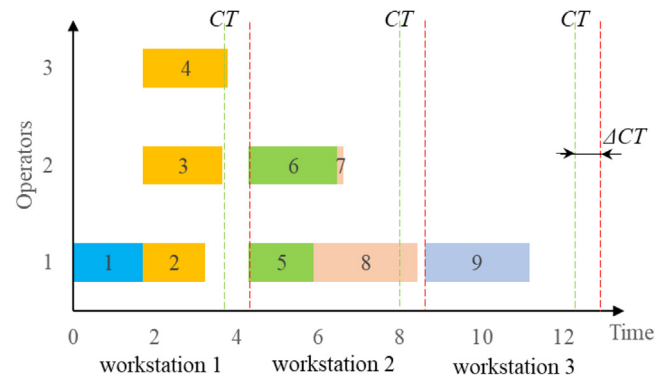| Demand plan e	The average processing time of task i								
	1	2	3	4	5	6	7	8	9
1	1.71	1.53	1.95	2.08	1.59	2.16	0.15	2.53	2.56
2	1.55	1.49	2.21	1.65	1.62	2.31	0.12	2.5	2.61
3	1.77	1.67	1.8	1.93	1.78	1.79	0.13	2.52	2.83

Fig. 2. Gantt chart for demand plan 1.

all the operators and workstations complete their assigned tasks within the maximum cycle time $CT + \Delta_{CT}$, which satisfies the robust cycle time constraints. Due to the limitation of precedence relation constraint, all tasks start after finishing the assemble of their immediate predecessors such as tasks 2, 3 and 4, followed by task 1.

4. Solution generation mechanisms

When the size of the r-MMALBP increases, the solving difficulty increases exponentially and then it is hard to obtain the best line configuration through the MILP model. Besides, the consideration of the uncertain demand plans makes the original multi-manned assembly line balancing more difficult. Hence, this work proposes to mine efficient dispatching rules to select the tasks for each operator due to their convenience and effectiveness in practice. The dispatching rules are the combination of
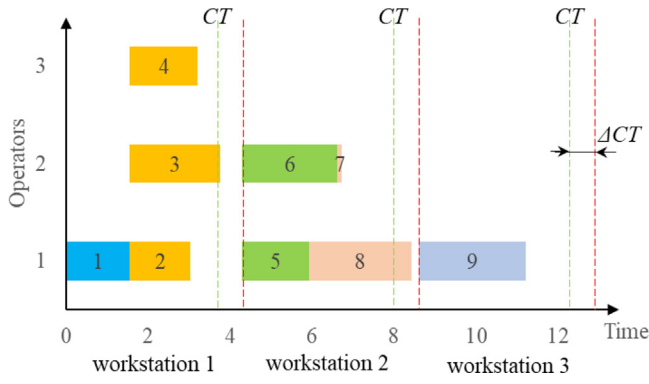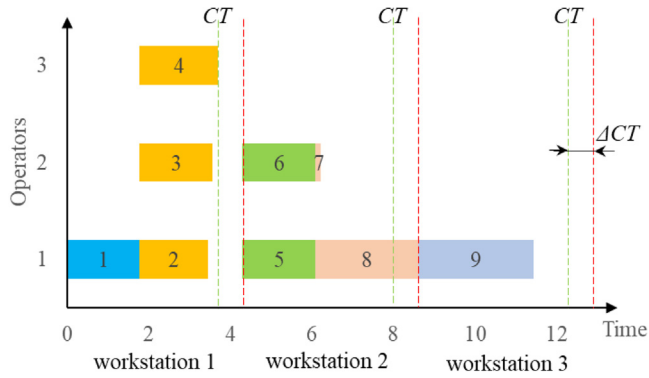
Fig. 3. Gantt chart of demand plan 2.

Fig. 4. Gantt chart of demand plan 3.

heuristics or empirical rules and are defined as algebraic expressions. They are embedded into the solution generation or decoding mechanisms to select a task from the set of candidates for the next assignment. In our problem, these dispatching rules (i.e., heuristic, or empirical rules) first assign a priority value for each task, and during the task selection step, they select the task with the maximum priority value to perform the next operation. A good solution generation mechanism, together with the rules, generates a great line configuration.

We have designed two solution generation mechanisms named operator-task-sequence solution generation and task-operator-sequence solution generation. These two mechanisms generate line configuration in a way that task assignment constraints, robust cycle time constraints, robust sequencing constraints should be satisfied. The difference between the two generations is how and when to select either the task or the operator. Both two solution generation mechanisms have the same computational complexity $O(n)$. The robustness of the two generations is mainly reflected in the consideration of uncertain demand plans when finding a better solution. The details of two solution generation mechanisms are defined in next Sections 4.1 and 4.2.

4.1. Operator-task-sequence solution generation

This generation aims at balancing the workload of each operator. It first determines an operator with the maximum average remaining capacity in all demand plans. And then, a task is selected from the candidate set according to the dispatching rule and assigned to the determined operator. The specific steps are detailed below. Before generating a line configuration, the current workstation (wc) is set to 0.

Step 1: Open a new multi-manned workstation.

Open a new multi-manned workstation ($wc = wc + 1$) with u_{max} operator.

Step 2: Determine the operator.

First, we calculate the available operators. An operator is available when (s)he has remaining capacity in all demand plans. The remaining capacity in each demand plan is equal to the allowed maximum cycle time ($CT + \Delta_{CT}$) minus the finishing time of the current operator. If no operator has remaining processing capacity, return to step 1. Otherwise, the operator with the maximum average remaining capacity in all demand plans is determined.

Step 3: Construct the candidate set of available tasks.

The candidate set of available tasks for the determined operator should be constructed. A task is available for this set based on the following criteria:

(1) It has not been assigned (task assignment constraint).
(2) Its predecessors have been assigned (precedence relation constraint).
(3) The determined operator can assemble this task in all the demand plans (robust cycle time constraint).

If the candidate set is empty (i.e., the current workstation is full) the procedure returns to Step 1. Otherwise, it goes to step 4.

Step 4: Select a task.

The task having the maximum priority provided by the dispatching rules (or heuristic rules) is selected from the candidate set and is assigned to the current determined operator. If there are unassigned tasks, the procedure returns to step 2. Otherwise, we calculate the robust objective functions and hence a complete line configuration is generated. The pseudocode of operator-task-sequence solution generation is presented in Algorithm 1.

4.2. Task-operator-sequence solution generation

This solution generation process first determines the task to be selected based on the dispatching rules and the available candidate set. Then, it determines the operator to assemble the selected task. This solution generation refers to the decoding mechanism proposed by Zhang et al. [43] and aims to reduce the sequence-dependent idle times and remaining idle times. The details of this solution generation are presented below. As done with the previous generation method, the current workstation (wc) is initially set to 0.

Step 1: Open a new multi-manned workstation.

Open a new multi-manned workstation ($wc = wc + 1$) with u_{max} operator.

Step 2: Construct the candidate set of available tasks.

A task can be put into the candidate set based on the following criteria:

(1) It has not been assigned (task assignment constraint).
(2) Its predecessors have been assigned (precedence constraint).

Procedure: Operator-task-sequence solution generation

1:    $wc=1$;

2:    **While** there are unassigned tasks **do**

3:        Construct the candidate set of available operators;

4:        **If** the candidate set of operators is not empty, **then**

5:           Select an operator ws with maximum average remaining capacity;

6:           Construct the candidate set of available tasks according to criteria in step 3;

7:           **If** the candidate set of tasks is not empty, **then**

8:               Select a task i according to criteria in step 4;

9:               Assign the task i to operator ws in wc;

10:           **Else**

11:               $wc = wc + 1$;

12:           **End If**

13:        **Else**

14:           $wc = wc + 1$;

15:        **End If**

16:    **End While**

17:    **Return** the complete solution.

Algorithm 1 The pseudocode of operator-task-sequence solution generation.

If the candidate set is empty, which means all the tasks have been assigned, the procedure returns to step 6. Otherwise, it goes to step 3.

Step 3: Select a task.

Same as the operator-task-sequence solution generation, a task with the maximum priority provided by dispatching rules (or heuristic rules) is selected from the candidate set for the next step.

Step 4: Determine the set of available operators.

Since more than one operator is allowed in the multi-manned workstation, the selected task should be assigned to a specific operator. Hence, the set of available operators needs to be determined. An operator k is available if this operator can assemble this task in all demand plans. If no operator is available, it means that the current workstation is full, and the procedure returns to step 1. Otherwise, it goes to step 5.

Step 5: Select an operator.

In MALBP two types of idle times are involved: sequence-dependent idle times and remaining idle times. Hence, to reduce these idle times, an operator from the set of available operators is selected according to the following rules:

(1) Select an operator who can assemble the selected task first.
(2) From the set of filtered operators from (1), select the one with the minimum sequencing idle time between the former task assigned to this operator and the selected task.
(3) If there is more than one operator from (2), select an operator with the minimum number.

It should be noted that all the times mentioned in step 5 refers to the average time in all demand plan. The pseudocode of task-operator-sequence solution generation is presented in Algorithm 2.

5. Gene expression programming

Gene expression programming (GEP) is an evolutionary computation method, proposed by Ferreira [44], and used in mining dispatching rules of complete problems. GEP been successfully applied in regression, prediction, classification, and optimization [45]. Inspired by the traditional design and components of a genetic algorithm, GEP uses evolutionary operators such as selection, mutation, transposition, and recombination to mines the dispatching rules iteratively. Our proposal applies a GEP method using a multi-attribute representation and two solution generation mechanisms presented in Section 4 to tackle the r-MMALBP.

Procedure: Task-operator-sequence solution generation

1:   *wc*=1;

2:   **While** there are unassigned tasks **do**

3:        Construct the candidate set of available tasks;

4:        **If** the candidate set of tasks is not empty, **then**

5:             Select a task *i* according to criteria in step 3;

6:             Construct the candidate set of available operators according to criteria in step 4;

7:             **If** the candidate set of operators is not empty, **then**

8:                  Select a task *ws* according to criteria in step 5;

9:                  Assign the task *i* to operator *ws* in *wc*;

10:            **Else**

11:                 *wc*=*wc*+1;

12:            **End If**

15:        **End If**

16:   **End While**

17:   **Return** the complete solution.

Algorithm 2 The pseudocode of task-operator-sequence solution generation.

Fig. 5 shows the general scheme of the GEP algorithm for solving the r-MMALBP.

In the next Section 5.1. we define the representation of the solutions for the GEP method. In Section 5.2 we present the designed evolutionary operators. We specify how we evaluate the individuals in Section 5.3.

5.1. Multi-attribute representation and initialization

In GEP, each chromosome represents a mined dispatching rule and is composed of two parts: head and tail, where the former contains the functional symbols and terminal symbols while the latter just includes the terminal symbols. For tackling our problem, we have defined 20 heuristic rules, shown in Table 6 and named from H_1 to H_{20}. They are defined by terminal symbols and six basic arithmetic operators as the functional symbols: (a) +, (b) −, (c) * for multiplication, (d) / for division, (e) Q for squared root and (f), S for square. The head and tail's lengths are fixed. The tail's length is given as a function of the head's length, determined by Eq. (35), as done in Zhong, Ong and Cai [46].

$$l = h(s - 1) + 1, \tag{35}$$

where *l* and *h* are the lengths of tail and head respectively, and *s* is the maximum argument in the functional symbols.

For example, a chromosome defined as $^* + H_1 + - H_3 \, Q \, H_{12} - \underline{H_3} \, \underline{H_5} \, \underline{H_7} \, \underline{H_{12}} \, \underline{H_9} \, \underline{H_1} \, \underline{H_{20}} \, \underline{H_{13}} \, \underline{H_8} \, \underline{H_2}$, where the non-underlined elements belong to the head and the underlined elements are part of the tail. To calculate the priority of each candidate task, this chromosome should be transformed into an algebraic expression. We first transform the chromosome into an expression tree.

Specifically, the elements of the head are first put into the root nodes of the expression tree in turn, and then leaf nodes are filled with the tail. Once all the leaf nodes are filled with terminal symbols, the expression tree is finished. The expression tree of the above chromosome is shown in Fig. 6. When constructing the expression tree, only the front part of the chromosome $^* + H_1 + - H_3 \, Q \, H_{12} - \underline{H_3} \, \underline{H_5} \, \underline{H_7}$ is used. That is, after the elements $^* + H_1 + - H_3 \, Q \, H_{12} - \underline{H_3} \, \underline{H_5} \, \underline{H_7}$ are set into the nodes of the expression tree in turn, all the leaf nodes are filled with the terminal symbols. According to the expression tree, the final algebraic expression (dispatching rule) is $(H_3 + H_{12} + H_7 - H_5 + \sqrt{H_3}) \times H_1$. Hence, a line configuration can be obtained based on the solution generation mechanisms (see previous Section 4) which are embedded in this dispatching rule.

Since heads having large lengths reduce the search efficiency while a short one cannot express the complexity of algebraic expression, this paper refers to the method proposed by Zhang, et al. [20] and applies a multi-attribute representation that each chromosome contains several heads and one tail. These heads share the same tail and relate to the algebraic operation "+". For example, a chromosome has three heads with a length of 3 and a tail with a length of 4, which is represented as $^* + H_1 - H_1 / S - H_5 \, \underline{H_{12}} \, \underline{H_{14}} \, \underline{H_{20}} \, \underline{H_9}$. The expression trees of the individuals and combination are depicted in Fig. 7. And the final algebraic expression is $(H_{12} + H_{14}) \times H_1 + H_1 - \frac{H_{12}}{H_{14}} + \sqrt{H_5 - H_{12}}$. In the initiation *PS* (population size) chromosomes are randomly generated using the functional symbols and terminal symbols.
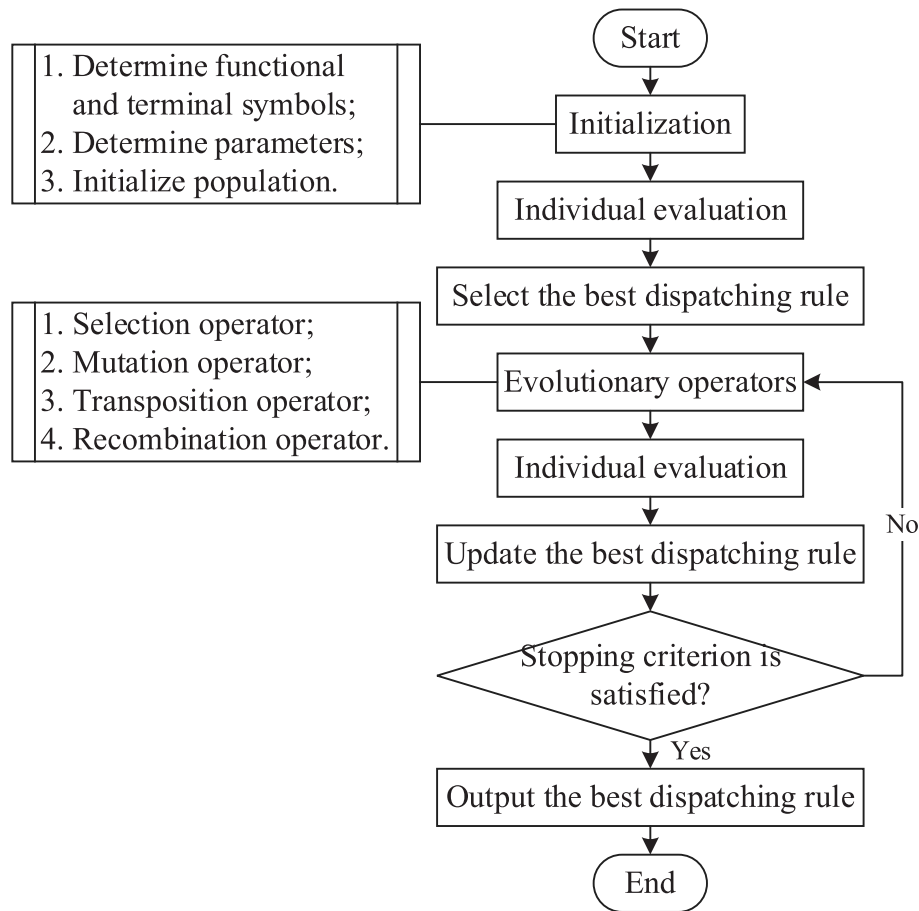
Fig. 5. The procedure of our proposed GEP.

Table 6
The heuristic rules used in the GEP method.

Symbol	Heuristic rule
H_1	The largest task number
H_2	The smallest task number
H_3	The shortest processing time
H_4	The largest processing time
H_5	The maximum number of all predecessors
H_6	The least number of all predecessors
H_7	The maximum number of all successors
H_8	The least number of all successors
H_9	The largest processing time of all predecessors
H_{10}	The shortest processing time of all predecessors
H_{11}	The largest processing time of all successors
H_{12}	The shortest processing time of all successors
H_{13}	The maximum number of all immediate predecessors
H_{14}	The least number of all immediate predecessors
H_{15}	The maximum number of all immediate successors
H_{16}	The least number of all immediate successors
H_{17}	The largest processing time of all immediate predecessors
H_{18}	The shortest processing time of all immediate predecessors
H_{19}	The largest processing time of all immediate successors
H_{20}	The shortest processing time of all immediate successors

Fig. 6. The expression tree of the example.

5.2. Evolutionary operators

The proposed GEP mines the dispatching rules iteratively by the evolution operators including selection, mutation, transposition, and recombination. Since each chromosome contains several heads and one tail, it is important to ensure its flexibility when executing evolutionary operators. Hence, we have designed original evolutionary operators, described in the next sub-sections.
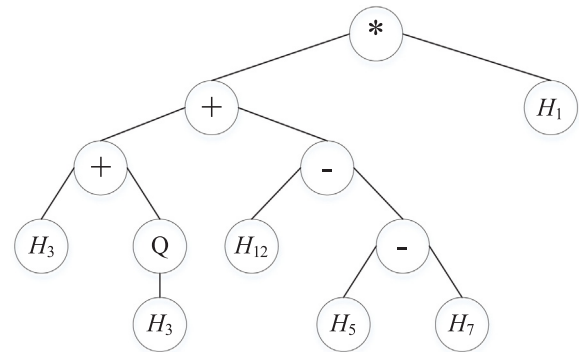
5.2.1. Selection operator

Tournament selection is adopted in this paper to guarantee that *PS* individuals are selected from the combination of parent and offspring populations into the next iteration. Through this selection, the individuals with higher fitness are endowed with more opportunity to enter the next generation while the individuals with the worst fitness are hardly selected. The selection starts by selecting some individuals from the population at random. The individual with the best fitness is selected for the next generation. The above process is repeated until *PS* individuals are selected.
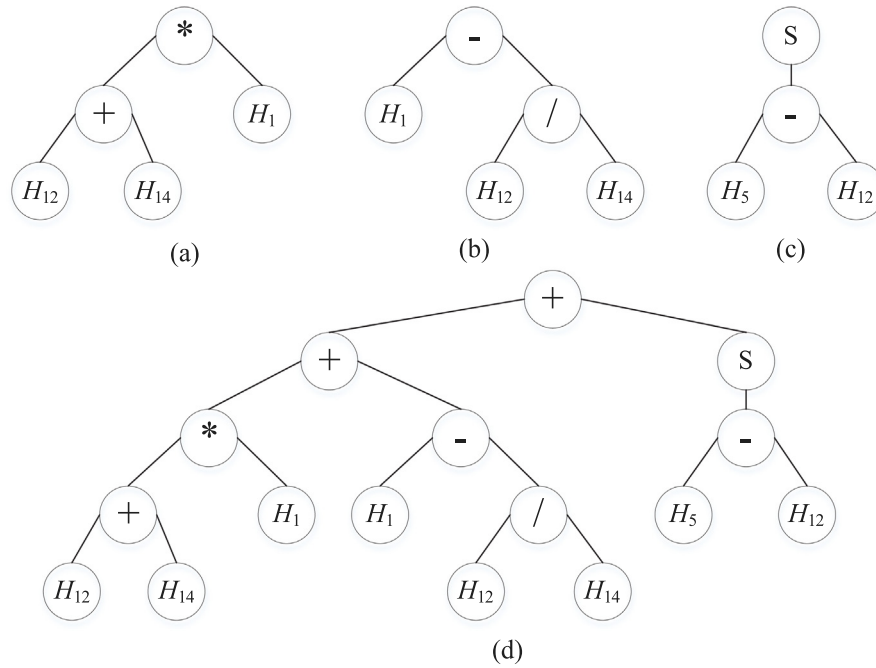
Fig. 7. The expression trees of the individuals (a, b, and c) and their combination (d).

5.2.2. Mutation operator

The mutation operator is a perturbation to the whole evolution process to prevent the algorithm from falling into local optimization. Since several heads and one tail are involved, this operator is performed at two different positions. If the position to be mutated occurs in heads, the element is mutated into any functional or terminal symbols. On the contrary, if the position is at the tail, the element is only mutated into any terminal symbols. This design then ensures the flexibility of the new generated individuals.

For example, consider a parent chromosome represented as $^* + H_1 - H_1 / S - H_5 \underline{H}_{12} \underline{H}_{14} \underline{H}_{20} \underline{H}_9$ and the element in the fifth position is determined to be mutated. Since this selected position is in the head, the element in this position is mutated into H_{16}, one of the functional and terminal symbols. The final mutated chromosome is $^* + H1 - H_{16} / S - H_5 \underline{H}_{12} \underline{H}_{14} \underline{H}_{20} \underline{H}_9$.

5.2.3. Transposition operator

The transposition operator aims to activate the invalid codes of the individuals. The root insertion sequence transposition is employed in this paper to implement rule mining. Specifically, a conversion segment is randomly selected, and this segment is inserted at the beginning position. It should be noted that the elements in the tail stay the same. For example, consider a chromosome represented as $^* + H_1 - H_1 / S - H_5 \underline{H}_{12} \underline{H}_{14} \underline{H}_{20} \underline{H}_9$, the conversion segment $S - H_5 \underline{H}_{12} \underline{H}_{14}$ is selected and inserted into the original position. Therefore, the new generated chromosome is $S - H_5 H_{12} H_{14} \, ^* + H_1 - \underline{H}_{12} \underline{H}_{14} \underline{H}_{20} \underline{H}_9$.

5.2.4. Recombination operator

The recombination operator can keep the favorable segment into the next generation and further mine dispatching rules [47]. We use a two-point recombination operator, as used done in other evolutionary algorithms such as a genetic algorithm. For each individual, two points are randomly determined, and the segment between two points are exchanged with one from the other parent individual. For example, in the case of two chromosomes represented as $^* + H_1 - H_1 / S - H_5 \underline{H}_{12} \underline{H}_{14} \underline{H}_{20} \underline{H}_9$ and $^* + H_1 + - H_3 Q H_{12} - \underline{H}_3 \underline{H}_5 \ \underline{H}_7 \ \underline{H}_{12}$, respectively, the segments between the fourth and tenth positions are selected for swapping. Hence, the first chromosome is transformed into $^* + H_1 + - H_3 Q H_{12} - \underline{H}_3 \underline{H}_{14} \underline{H}_{20} \underline{H}_9$.

5.3. Individual evaluation

To evaluate the quality of the mined rules, we first calculate the objective values of the training instances according to the solution generation mechanisms in Section 4, and then calculate the fitness of each dispatching rule. The fitness of an individual i can be calculated by Eq. (36):

$$f_i = \frac{1}{N} \sum_{j=1}^{N} (Obj_{ij} - minObj_j)/\min Obj_j, \tag{36}$$

where N is the number of the training instances, Obj_{ij} is the objective value when dispatching rule i is applied to instance j. $minObj_j$ is the minimum objective value of instance j in the current population.

It can be observed from Eq. (36) that, when the fitness is close to 0, the corresponding dispatching rule obtains the best objective values in nearly all training instances, and hence performs best in the current population. Note that since the fitness value depends on the minimum values in the current population, the same dispatching rule may have different fitness in different populations. Hence, when updating the best dispatching rule, the previous best dispatching rule needs to be re-evaluated under the current population.

6. Computational study and analysis of the results

6.1. Benchmark instances and experimental setup

We have generated a set of benchmark instances to analyze the performance of proposed solution generation mechanisms and evolutionary dispatching rules. These instances are 269, having 30 training instances and 239 test instances. These instances organized into 25 groups, mainly including Mertens7, Bowman8, Jaeschke9, Jackson11, Mansoor11, Mitchell21, Roszieg25, Heskiaoff28, Buxey29, Sawyer30, Lutz32, Gunther35, Kilbridge45, Hahn53, Warnecke58, Tonge70, Wee-Mag75, Arcus83, Lutz89-1, Lutz89-2, Mukherje94, Arcus111, Barthold148-1, Barthold148-2 and Scholl297. The original task times and precedence relation of

Table 7
Parameters used in GEP.

Parameter	Value
Population size	20
Number of iterations	30
Head length	6
Head number	3
Tail length	7
Mutation rate	0.2
Recombination rate	0.4
Transposition rate	0.3
Number of individuals in selection	5

these instances can be downloaded from https://assembly-line-balancing.de. For each instance, the number of product model is uniformly generated between [3,10], and the number of demand plan is drawn out from a uniform distribution in [5,10]. Under each plan, the demand of each model is randomly generated [5, 30]. Since our study focuses on mixed-model assembly line balancing, the processing times of all models are also different. Hence, in this paper, the original task times are given to the first model. task times of other model obey uniform distribution in [0, t_{i0}]. Due to the limited space, the detailed data is not provided, but available upon request.

We have also determined the parameters used by the GEP method. Regarding head length, a larger value will bring a better result but with higher complexity. Hence, we need to select an appropriate value to balance the efficiency and complexity. Besides, the selection of population size, mutation, recombination and transposition rates as well as the number of individuals in selection have a great influence on the performance of GEP. We refer to previous research from [20,48] as well as a preliminary experimentation to determine the values of these parameters. For each parameter, we set several values. Later, we use the GEP to obtain an evolutionary rule for each parameter combination. Finally, we selected the best parameter combination by considering the fitness function values. The list of parameters is shown in Table 7.

All the heuristic and evolutionary rules and GEP method are programmed with C++ language and are run on a computer with Intel Core i5 4440, 2.80 GHz and 2 GB RAM. In addition, the relative percentage deviation (RPD) is used to measure the effectiveness of the generations and rules. This indicator shows how far the solution of a heuristic is from the best one found for a given problem [49], and can be calculated by Eq. (37).

$$RPD_{rule} = \frac{Obj_{rule} - Obj_{best}}{Obj_{best}} \cdot 100 \quad (37)$$

where Obj_{rule} is the objective value obtained by rule and Obj_{best} is the minimum objective value obtained by all the compared rules. When each RPD is obtained, the average relative percentage deviation (ARPD) is calculated according to next Equation (38), where S is the number of instances for a given class:

$$ARPD_{rule} = \frac{\sum_{\forall s} RPD_{s,rule}}{S} \quad (38)$$

6.2. Comparison between the MILP model and other algorithms

To evaluate the performance of the proposed MILP model, this section compares it with two well-known algorithms based on 66 comparison instances including small- and middle-scale instances. The MILP model is coded in IBM CPLEX and its running time is controlled within 1800 s. For the CPLEX solver, this solver first uses the emphasis parameter 0 (balancing the optimality and feasibility) to solve each instance. If no solution is found or the found solution is feasible, this solver continues to use

the emphasis parameter 1 (emphasize feasibility over optimality) to find a feasible solution. If the solution is optimal, this solver also continues to use the emphasis parameter 2 (emphasize optimality over feasibility) to verify the optimal solution. The compared algorithms mainly include simulated annealing algorithm (SA) [18] and genetic algorithm (GA) [40]. These algorithms have been proved to be effective in tackling multi-manned assembly line balancing problem. In the comparison experiments, these algorithms are run with the same CPU time limits of $n \times n \times 5$ milliseconds. The final compared results are shown in Table 8.

From this table, it can be observed that the MILP model finds the optimal solutions of 49 instances and the feasible solutions of 6 instances in r-MMALBP. Regarding the compared algorithms, they achieve the same results as the MILP model in 49 instances. As the instance size increases, the MILP model cannot achieve the optimal results, and hence the results obtained by algorithms are better than those by the MILP model. The reason for this situation is that with the increase of the instance size, the number of variables and constraints are increasing exponentially and the CPLEX solver cannot effectively cut branches. Therefore, it can be concluded that the MILP model solved by the CPLEX solver can effectively deal with the small- and middle-scale instances of r-MMALBP and can ensure the optimality of the solutions. Hence, it is suggested to use the MILP model to tackle the instances whose task numbers are less than or equal to 35.

6.3. Comparison between the two solution generation mechanisms

This section aims to investigate the performance of two solution generation mechanisms. In the following comparisons, we name operator-task-sequence solution generation as generation 1, and task-operator-sequence solution generation as generation 2. The heuristic rules listed in Table 6 are respectively named as *rule1-rule20*. In the comparison, we solve the 269 instances with the two solution generation mechanisms embedded with 20 heuristic rules. Thus, each instance is solved 40 times. A total of 10,760 experiments are involved. We calculate the RPD values of two generations for each instance under each heuristic rule. The ARPD values (269 data per average) of the performance of the heuristic rules are shown in Table 9. It can be observed from this table that generation 2 is significantly more advanced than generation 1 under all rules, which can suggest that task-operator-sequence solution generation outperforms operator-task-sequence solution generation in tackling mixed-model multi-manned assembly line balancing under uncertain demand.

Besides, statistical analysis is performed to further confirm the statistically significant difference between the two generations. After conducting the normality test and equal variance test, a multiple ANOVA test is carried out where RPD is regarded as the response variable, and generation and rule types are regarded as the controlled factor. The ANOVA results for the heuristic rule and generation types are shown in Table 10. We see that *p*-values of generation and rule types are both less than 0.001 with a total of 269 instances, and that of the interaction of generation and rule is also less than 0.001. These results suggest that the generation and rule types have a statistically significant effect on the performance of mixed-model multi-manned assembly line balancing under uncertain demand. Meanwhile, the *F*-ratio of generation type is much larger than that of rule type. This observation indicates that the influence of generation types greater than that of rule type. Based on the above analysis of ANOVA, we further depict the means plots of *RPD* with Tukey's Honest Significant Difference (HSD) 95% confidence intervals for all heuristic rules under two generations in Fig. 8. This figure states that Statistically, generation 2 (task-operator-sequence solution generation) is superior to generation 1 (operator-task-sequence solution generation).

Table 8
Compared results for CPLEX solver, SA, and GA.

No.	Instances	n	CT	CPLEX solver			SA		GA	
				Emphasis values	Results	Status	Results	CPU/s	Results	CPU/s
1	Mertens	7	6	0, 2	0.170455	Optimal	0.170455	0.245	0.170455	0.245
2	Mertens	7	7	0, 2	0.170455	Optimal	0.170455	0.245	0.170455	0.245
3	Mertens	7	8	0, 2	0.170455	Optimal	0.170455	0.245	0.170455	0.245
4	Mertens	7	10	0, 2	0.170455	Optimal	0.170455	0.245	0.170455	0.245
5	Mertens	7	15	0, 2	0.170455	Optimal	0.170455	0.245	0.170455	0.245
6	Mertens	7	18	0, 2	0.170455	Optimal	0.170455	0.245	0.170455	0.245
7	Bowman	8	20	0, 2	0.151111	Optimal	0.151111	0.32	0.151111	0.32
8	Jaeschke	9	6	0, 2	0.135714	Optimal	0.135714	0.405	0.135714	0.405
9	Jaeschke	9	7	0, 2	0.135714	Optimal	0.135714	0.405	0.135714	0.405
10	Jaeschke	9	8	0, 2	0.135714	Optimal	0.135714	0.405	0.135714	0.405
11	Jaeschke	9	10	0, 2	0.135714	Optimal	0.135714	0.405	0.135714	0.405
12	Jaeschke	9	18	0, 2	0.135714	Optimal	0.135714	0.405	0.135714	0.405
13	Jackson	11	7	0, 2	0.112745	Optimal	0.112745	0.605	0.112745	0.605
14	Jackson	11	9	0, 2	0.112745	Optimal	0.112745	0.605	0.112745	0.605
15	Jackson	11	10	0, 2	0.112745	Optimal	0.112745	0.605	0.112745	0.605
16	Jackson	11	13	0, 2	0.112745	Optimal	0.112745	0.605	0.112745	0.605
17	Jackson	11	14	0, 2	0.112745	Optimal	0.112745	0.605	0.112745	0.605
18	Jackson	11	21	0, 2	0.112745	Optimal	0.112745	0.605	0.112745	0.605
19	Mansoor	11	48	0, 2	0.112745	Optimal	0.112745	0.605	0.112745	0.605
20	Mansoor	11	62	0, 2	0.112745	Optimal	0.112745	0.605	0.112745	0.605
21	Mansoor	11	94	0, 2	0.112745	Optimal	0.112745	0.605	0.112745	0.605
22	Mitchell	21	14	0, 2	0.06108	Optimal	0.06108	2.205	0.06108	2.205
23	Mitchell	21	15	0, 2	0.06108	Optimal	0.06108	2.205	0.06108	2.205
24	Mitchell	21	21	0, 2	0.06108	Optimal	0.06108	2.205	0.06108	2.205
25	Mitchell	21	26	0, 2	0.06108	Optimal	0.06108	2.205	0.06108	2.205
26	Mitchell	21	35	0, 1	No solution found		0.06108	2.205	0.06108	2.205
27	Mitchell	21	39	0, 2	0.06108	Optimal	0.06108	2.205	0.06108	2.205
28	Roszieg	25	14	0, 2	0.051619	Optimal	0.051619	3.125	0.051619	3.125
29	Roszieg	25	16	0, 2	0.051619	Optimal	0.051619	3.125	0.051619	3.125
30	Roszieg	25	18	0, 2	0.051619	Optimal	0.051619	3.125	0.051619	3.125
31	Roszieg	25	21	0, 2	0.051619	Optimal	0.051619	3.125	0.051619	3.125
32	Roszieg	25	25	0, 2	0.051619	Optimal	0.051619	3.125	0.051619	3.125
33	Roszieg	25	32	0, 2	0.051619	Optimal	0.051619	3.125	0.051619	3.125
34	Heskiaoff	28	138	0, 1	0.336308	Non-optimal	0.046247	3.92	0.046247	3.92
35	Heskiaoff	28	205	0, 1	No solution found		0.046247	3.92	0.046247	3.92
36	Heskiaoff	28	216	0, 1	No solution found		0.046247	3.92	0.046247	3.92
37	Heskiaoff	28	256	0, 1	No solution found		0.046247	3.92	0.046247	3.92
38	Heskiaoff	28	324	0, 1	0.31359	Non-optimal	0.046247	3.92	0.046247	3.92
39	Heskiaoff	28	342	0, 2	0.046247	Optimal	0.046247	3.92	0.046247	3.92
40	Buxey	29	27	0, 2	0.044697	Optimal	0.044697	4.205	0.044697	4.205
41	Buxey	29	30	0, 2	0.044697	Optimal	0.044697	4.205	0.044697	4.205
42	Buxey	29	33	0, 2	0.044697	Optimal	0.044697	4.205	0.044697	4.205
43	Buxey	29	36	0, 2	0.044697	Optimal	0.044697	4.205	0.044697	4.205
44	Buxey	29	41	0, 2	0.044697	Optimal	0.044697	4.205	0.044697	4.205
45	Buxey	29	47	0, 2	0.044697	Optimal	0.044697	4.205	0.044697	4.205
46	Buxey	29	54	0, 2	0.044697	Optimal	0.044697	4.205	0.044697	4.205
47	Sawyer	30	25	0, 2	0.043247	Optimal	0.043247	4.5	0.043247	4.5
48	Sawyer	30	27	0, 2	0.043247	Optimal	0.043247	4.5	0.043247	4.5
49	Sawyer	30	30	0, 2	0.043247	Optimal	0.043247	4.5	0.043247	4.5
50	Sawyer	30	33	0, 2	0.043247	Optimal	0.043247	4.5	0.043247	4.5
51	Sawyer	30	36	0, 2	0.043247	Optimal	0.043247	4.5	0.043247	4.5
52	Sawyer	30	41	0, 1	No solution found		0.043247	4.5	0.043247	4.5
53	Sawyer	30	47	0, 2	0.043247	Optimal	0.043247	4.5	0.043247	4.5
54	Sawyer	30	54	0, 1	No solution found		0.043247	4.5	0.043247	4.5
55	Sawyer	30	75	0, 1	0.119461	Non-optimal	0.043247	4.5	0.043247	4.5
56	Lutz1	32	1414	0, 1	No solution found		0.040612	5.12	0.040612	5.12
57	Lutz1	32	1572	0, 1	0.061231	Non-optimal	0.040612	5.12	0.040612	5.12
58	Lutz1	32	1768	0, 1	0.206169	Non-optimal	0.040612	5.12	0.040612	5.12
59	Lutz1	32	2020	0, 1	0.112152	Non-optimal	0.040612	5.12	0.040612	5.12

Table 8 (*continued*).

No.	Instances	n	CT	CPLEX solver			SA		GA	
				Emphasis values	Results	Status	Results	CPU/s	Results	CPU/s
60	Lutz1	32	2357	0, 1	No solution found		0.040612	5.12	0.040612	5.12
61	Lutz1	32	2828	0, 1	No solution found		0.040612	5.12	0.040612	5.12
62	Gunther	35	41	0, 2	0.037212	Optimal	0.037212	6.125	0.037212	6.125
63	Gunther	35	44	0, 1	No solution found		0.037212	6.125	0.037212	6.125
64	Gunther	35	49	0, 1	No solution found		0.037212	6.125	0.037212	6.125
65	Gunther	35	54	0, 2	0.037212	Optimal	0.037212	6.125	0.037212	6.125
66	Gunther	35	61	0, 2	0.037212	Optimal	0.037212	6.125	0.037212	6.125
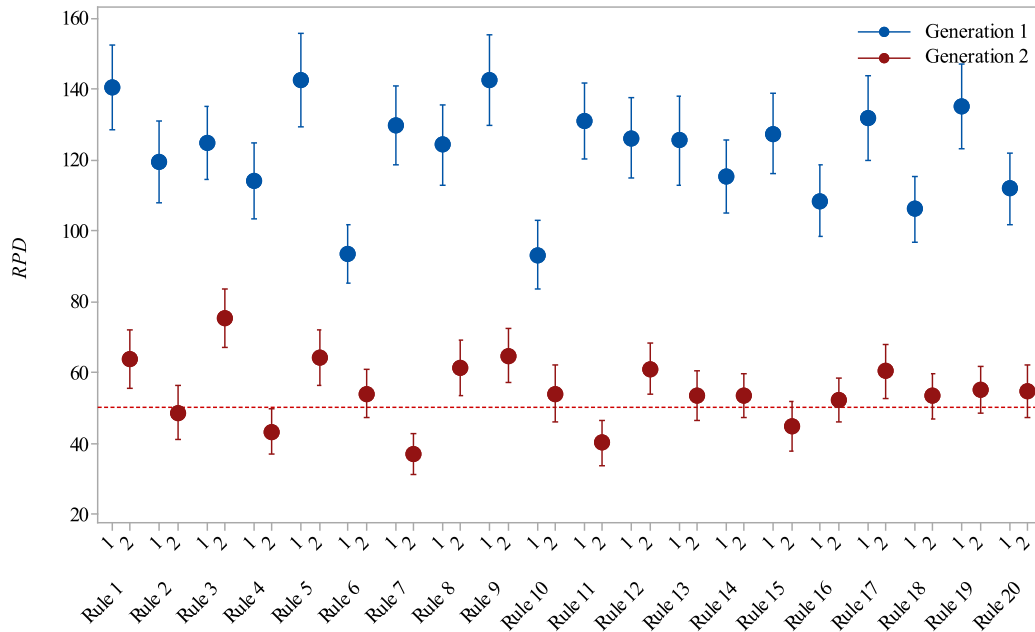
Fig. 8. Means plots of *RPD* with Tukey's Honest Significant Difference (HSD) 95% confidence intervals for all heuristic rules under two generations (values 1 and 2 in the X-axis). A horizontal red line is added as a reference to highlight the best rules (2, 4, 7, 11, and 15).

Table 9
ARPD values (269 data per average) of the performance of the heuristic rules.

Rule	Generation 1	Generation 2	Rule	Generation 1	Generation 2
1	140.47	63.81	11	131.09	40.01
2	119.36	48.65	12	126.22	61.00
3	124.86	75.27	13	125.48	53.30
4	114.13	43.14	14	115.42	53.52
5	142.57	64.21	15	127.47	44.62
6	93.34	53.92	16	108.45	52.13
7	129.68	36.85	17	131.83	60.35
8	124.28	61.19	18	106.10	53.25
9	142.66	64.73	19	135.00	55.04
10	93.22	54.03	20	111.88	54.61

6.4. Analysis of the GEP sets of heuristic rules

30 training instances are employed to train the GEP to mine efficient dispatching rules. After a preliminary experiment, we find that if we define all 20 heuristic rules as the terminal symbols, the performance of mined dispatching rules is not adequate. Hence, we decide to filter out the good heuristic rules before training the GEP. From Fig. 8, it can be observed that heuristic rules 2, 4, 7, 11 and 15 have a significant superiority over other rules, and accordingly we define these five rules as the final terminal symbols. Finally, GEP is run for 10 times to obtain 10 evolutionary rules. These rules and their corresponding expressions are shown in Table 11.

10 evolutionary rules are embedded into task-operator-sequence solution generation to solve 269 benchmark instances. Furtherly, the RPD values of different rules for each instance are calculated. To confirm the statistically significant difference

Table 10
ANOVA results for the heuristic rule and generation types.

Sources	df	Type III sum of squares	Mean square	F-ratio	P-value
Corrected Model	39	1376.17	35.29	58.41	<0.001
Generation	1	1225.48	1225.48	2028.40	<0.001
Rule	19	91.92	4.84	8.01	<0.001
*Generation * Rule*	19	58.77	3.09	5.12	<0.001
Error	10 720	6476.59	0.60		
Total	10 759	7852.76			

df: degree of freedom.

Table 11
10 optimal evolutionary rules obtained by GEP for 10 different runs.

No	Evolutionary rule	Expression
1	$H_7H_7 + -H_7 - H_{11} - H_4H_2/+H_{11}/H_7H_7H_{11}/H_{11}H_{15}H_2H_{15}H_{11}H_7H_{15}$	$2 \times H_{11} + H_7$
2	$H_7H_4H_7H_{11}H_{15}/+H_7H_{11}*S/H_4+/H_7H_4+H_7H_2H_{11}H_7H_{11}H_{11}H_{15}$	$2 \times H_7 + H_{11} + H_4$
3	$/H_2H_2 - *SH_7Q + H_{11}H_2H_7H_7/-H_{11} * +H_7H_7H_4H_{11}H_2H_7H_{11}$	$2 \times H_7 + 1$
4	$H_{11}H_4QH_2H_{15} - *H_7 + H_{11}Q/+/++ Q + H_2H_4H_2H_7H_7H_{11}H_2$	$H_7+H_2+2\times H_{11}+H_7\cdot\left(\sqrt{\dfrac{H_2}{H_4}}+H_{11}\right)+\dfrac{H_4+H_2}{\sqrt{H_7}}$
5	$H_7H_{11} + *+*H_4/H_{11}H_{11}/H_7QS + H_7H_7QH_4H_2H_7H_2H_2H_{11}H_4$	$H_4 + 3 \times H_7$
6	$/H_{15}SH_{15}Q -++H_7H_2H_4 + S+S+H_7*H_7H_1H_{11}H_7H_4H_2H_{15}$	$H_7 + H_2 + H_4 + \dfrac{1}{H_{15}} + \left(H_7{}^2 + H_{11}{}^2 + H_7\right)^2$
7	$Q/+*/H_7/H_7H_{15}H_7H_{11}QH_7H_{11} + H_2H_2 - H_2H_4H_{11}H_4H_7H_2H_{11}$	$\dfrac{1}{H_4}\times\sqrt{\dfrac{H_{11}+H_7\times H_4}{H_2}}+H_7+\dfrac{H_7}{H_{15}}$
8	$/H_7H_2//H_7H_7S - /H_{15}*Q/*H_{11}H_2H_{11}H_{11}H_2H_{11}H_{11}H_{15}H_7H_4$	$\dfrac{H_7}{H_2}+H_7+\sqrt{H_2}$
9	$H_7H_4H_{11}H_{15} *+H_7 - Q - *H_4//H_{15}H_7/SH_{11}H_2H_4H_7H_4H_7H_4$	$2 \times H_7 + \dfrac{H_7\times H_{11}}{H_2{}^2\times H_{15}}$
10	$H_7H_4H_7 + SH_7/H_4H_2H_{11}H_7H_7 *+*+SH_7H_4H_7H_{15}H_7H_7H_{15}H_7$	$H_7 + \dfrac{H_4}{H_2} + H_7\times H_4\times\left(H_7+H_{15}+H_7{}^2\right)$

between the different evolutionary rules, a multiple ANOVA test is carried out where RPD and evolutionary rule type are regarded as the response variable and the controlled factor respectively. The ANOVA results for the evolutionary rule type is shown in Table 12. It can be observed that *P*-value of the evolutionary rule type is equal to 0.001 with a total of 269 instances, which suggests that the 10 mined rules have a different effect on the performance of our new problem. We further depict the means plots of *RPD* with Tukey's Honest Significant Difference (HSD) 95% confidence intervals for all evolutionary rules in Fig. 9. From this figure, it can be observed that the evolutionary rules 10 and 7 perform worst. Then we delete these two rules and use the ANOVA test to analyze the remaining 8 evolutionary rules. The ANOVA results for the remaining 8 evolutionary rules are shown in Table 13. In this table, the *P*-value of 8 evolutionary rules is equal to 0.971 with a total of 269 instances, which suggests that the remaining 8 evolutionary rules have a similar effect on the performance of our new problem. Hence, we can conclude that the proposed GEP has strong stability in solving this problem, and this method is suitable for solving the robust optimization problem.

Finally, we further count the number of the best solutions found by evolutionary rules and rank them from 1 to 10 in each instance to analyze their robustness. Fig. 10 depicts the average rank and the number of the best solutions found by each evolutionary rule considering all the benchmark instances. The closer to 1 the average rank is, the better the rule performs. Regarding the number of the best solutions found, the evolutionary rule 5 finds 85 best solutions out of 269 instances and performs best, followed by evolutionary rules 1, 9, 8, 3, 4, 2, 6, 7 and 10 in turn. If using the average rank values to rank these rules, evolutionary rule 1 ranks first, followed by evolutionary rules 3, 6, 4, 2, 5, 8, 9, 7 and 10 in turn.

In summary, through the above experiments, we select 5 evolutionary rules with better performance. *E_rule1*, *E_rule2*, *E_rule3*, *E_rule4*, and *E_rule5* are selected and will be compared with the existing heuristic rules and evolutionary rules mined by GP to further explore this performance.

6.5. Comparison between the evolutionary and existing heuristic rules

This section discusses the performance of the evolutionary rules by comparing them with the existing heuristic rules. In the above sections, five evolutionary rules and five heuristic rules with greater performance are determined. Hence in this section,

Table 12
ANOVA results for the 10 evolutionary rules.

Sources	df	Type III sum of squares	Mean square	F-ratio	P-value
E_rule	9	3002	333.5	3.18	0.001
Error	2680	280874	104.8		
Total	2689	283876			

df: degree of freedom.

Table 13
ANOVA results for the 8 evolutionary rules.

Sources	df	Type III sum of squares	Mean square	F-ratio	P-value
E_rule	7	177	25.25	0.26	0.971
Error	2144	211773	98.77		
Total	2151	211950			

df: degree of freedom.

we focus on the comparisons between these 10 rules. The selected evolutionary rules are *E_rule1*, *E_rule3*, *E_rule4*, *E_rule6* and *E_rule10*. The selected heuristic rules are *Rule 2*, *Rule4*, *Rule 7*, *Rule 11* and *Rule 15*. In this comparison, a total of 269 instances is solved by the task-operator-sequence solution generation embedded with 10 evolutionary or heuristic rules so that each instance is solved 10 times. Since these rules are embedded into the solution generation to calculate the priority values, the computational complexity is the same as that of the solution generation and equals to $O(n)$ when using these rules to solve the r-MMALBP. Therefore, a total of 2690 experiments are involved. We calculate the RPD values of 10 rules for each instance. Table 14 presents the ARPD values and average running times (269 data per average) of 10 rules.

As illustrated in Table 14, there is a slight difference in the running time of different rules. However, the ARPD values of the 5 evolutionary rules are lower than those of the 5 heuristic rules. This observation again suggests that the evolutionary rules outperform the heuristic rules. Moreover, to explore the quality and differentiation of the evolutionary and heuristic rules, this section also carries out a multiple ANOVA test where the response variable is RPD and the single controlled factor is the rule type with 10 levels. The ANOVA results are shown in Table 15. Fig. 11 shows the mean plots of *RPD* with Tukey's Honest Significant Difference (HSD) 95% confidence intervals for 10 heuristic and evolutionary rules.

As shown in Table 15, *p*-values are lower than 0.001 from a total number of 269 instances. These values demonstrate that the quality of these rules is different graphically, and the type of the rule has a statistically significant effect on the performance when
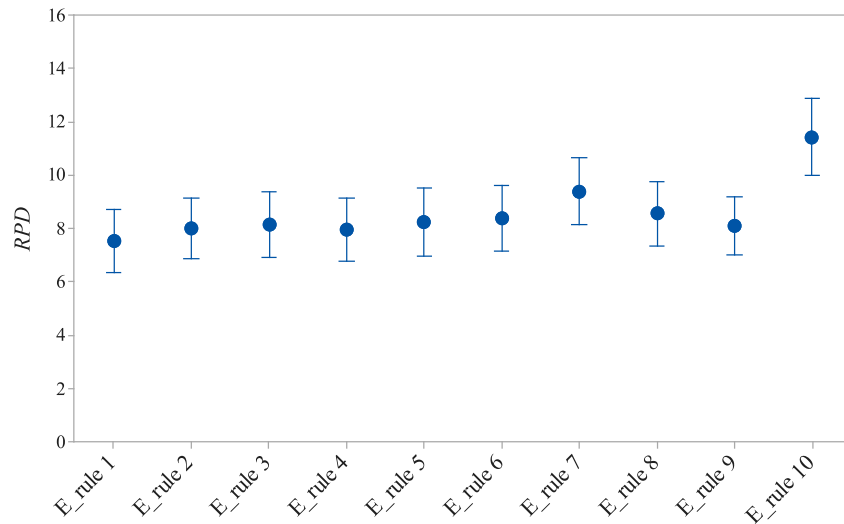
Fig. 9. Means plots of *RPD* with Tukey's Honest Significant Difference (HSD) with a 95% confidence interval for all evolutionary rules.
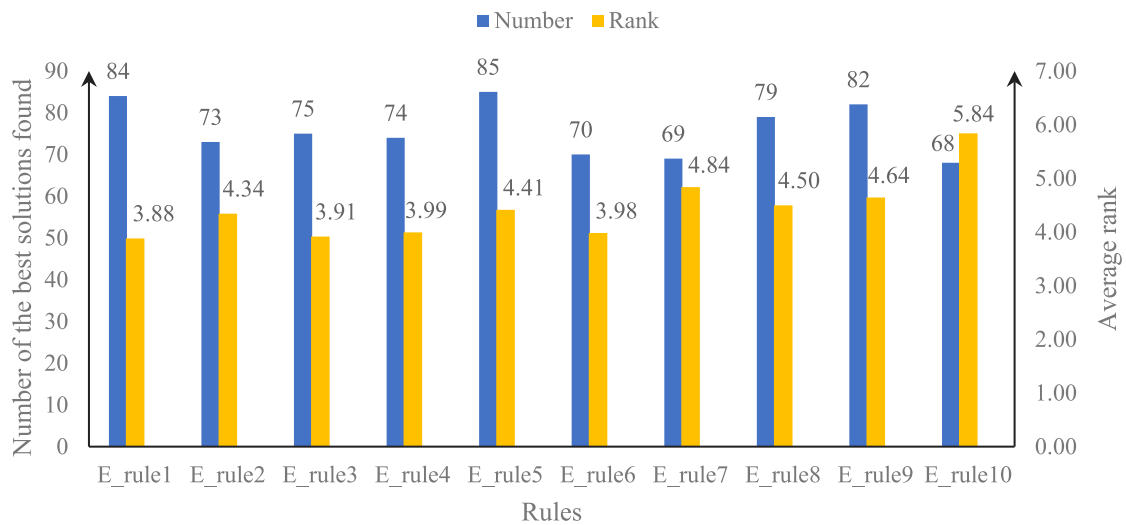

Fig. 10. The average rank and the number of the best solutions found by each evolutionary rule considering all the benchmark instances.

solving the problem. Furthermore, as observed in Fig. 11, the means plots of 5 evolutionary rules dominate those of 5 heuristic rules. This means that the evolutionary rules have great performance in tackling the multi-manned assembly line balancing problem under uncertain demand and the heuristic rules perform worse.

Finally, we count the number of the best solutions found by evolutionary or traditional rules and rank them from 1 to 10 in each instance to analyze their robustness. Fig. 12 depicts the average rank and the number of the best solutions found by each evolutionary or traditional rule considering all the benchmark instances. Regarding the number of the best solutions found, the evolutionary rules 1–5 respectively find 106, 99, 95, 90 and 100 best solutions out of 269 instances; while the heuristic rules 2, 4, 7, 11 and 15 only find 27, 31, 34, 29 and 27 best solutions respectively. If using the average rank values to rank these rules, the evolution rules 1–5 respectively rank first, fourth, third, second and fifth; the heuristic rules 2, 4, 7, 11 and 15 rank 10th, 9th, 6th, 7th, and 8th respectively. Therefore, we can conclude that the evolutionary rules are superior to traditional heuristic rules.

In summary, since the evolutionary rules are trained on 30 instances with different scales, they have strong adaptability in different instances. In other words, the evolutionary rules have great robustness and can obtain a better solution in all instances. Meanwhile, the evolutionary rules are the combinations of the heuristic rules 2, 4, 7, 11 and 15 with better performance. The combinations of these heuristic rules are adjusted and perturbed through different genetic operators in each training phase and are filtered by the fitness functions. Therefore, the final combinations (evolutionary rules) outperform the original heuristic rules.

6.6. Comparison between the evolutionary rules mined by GEP and GP

This section further compares the evolutionary rules obtained by GEP and GP. We use the GP proposed by Baykasoğlu and Özbakır [42] to generate 5 rules. The expressions of the generated rules are $\left(H_{15}^2 + H_4\right)^2 - H_{15}$, $H_4^{1/4} + H_7^{1/4} + H_{15} + H_2 \times \left(H_{11} + \sqrt{H_7}\right)$, $\left(\frac{H_2}{\sqrt{H_2 \times H_7}} + \frac{1}{H_7^2}\right) \times \frac{H_7 \times H_4 + H_{11} - H_4}{(H_2 + H_4)^2} + \left(\frac{H_2 \times \sqrt{H_{11}}}{H_7} + H_{11}^2 - \sqrt{H_{11}}\right) \times$
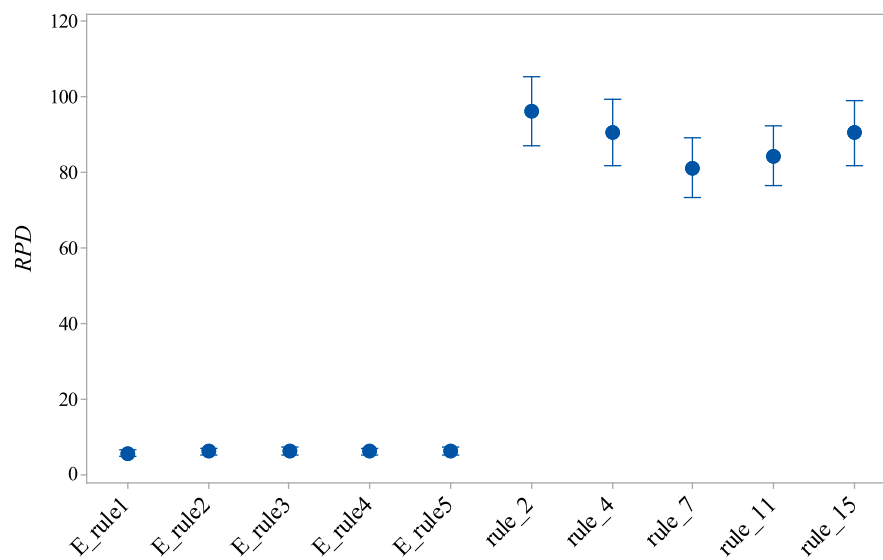
Fig. 11. Means plots of *RPD* with Tukey's Honest Significant Difference (HSD) 95% confidence intervals for 10 heuristic and evolutionary rules.
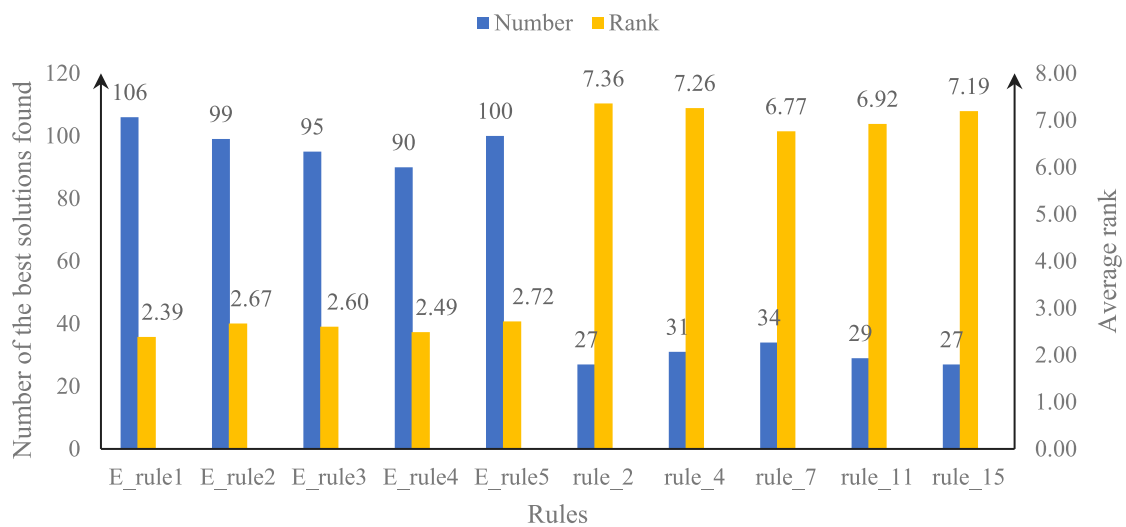

Fig. 12. The average rank and the number of the best solutions found by each evolutionary or heuristic rule considering all the benchmark instances.
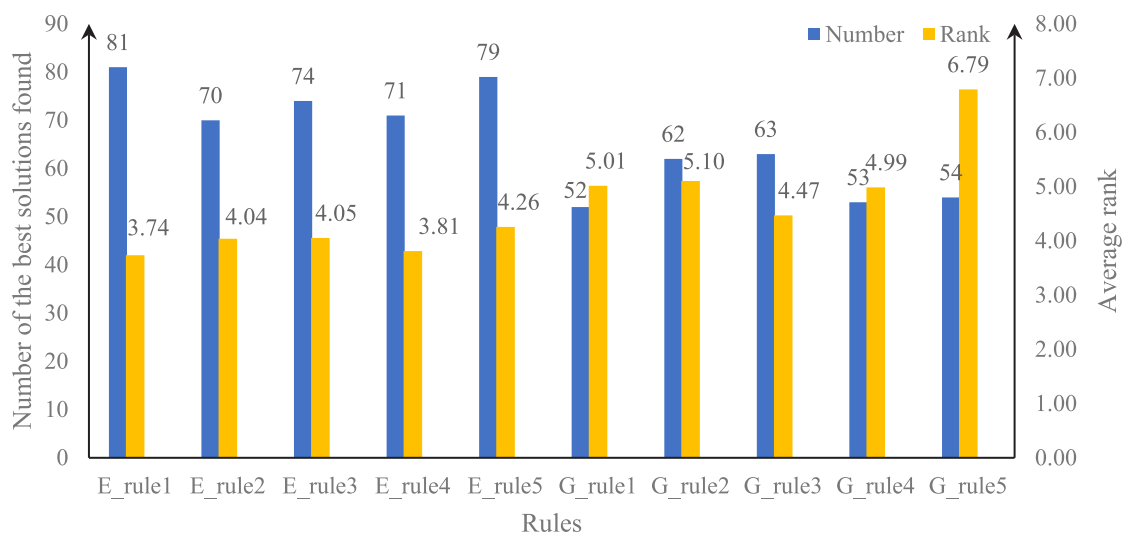

Fig. 13. The average rank and the number of the best solutions found by each evolutionary rule mined by GEP or GP considering all the benchmark instances.

Table 14
ARPD values (269 data per average) and average running times of the heuristic and evolutionary rules.

Traditional rules	ARPD	Time (ms)	Evolutionary rules	ARPD	Time (ms)
Rule2	96.27	282	E_rule1	5.49	3270
Rule4	90.65	176	E_rule2	5.97	2942
Rule7	81.31	562	E_rule3	6.09	3375
Rule11	84.38	560	E_rule4	5.91	3534
Rule15	90.57	313	E_rule5	6.19	3010

Table 15
ANOVA results for the traditional and evolutionary rules.

Sources	df	Type III sum of squares	Mean square	F-ratio	P-value
Rule	9	4 637 182	5 15 242	202.91	<0.001
Error	2680	6 805 321	2539		
Total	2689	11 442 503			

$(H_{11} \times H_{15} - H_4 + H_{11})^2$, $(H_4 - H_7 + H_{15} - H_{15}^2)^2$ and $\frac{H_{15}}{H_2^2} + H_4^2$. We name them as G_rule1, G_rule2, G_rule3, G_rule4 and G_rule5, respectively. The selected evolutionary rules by GEP are E_rule1, E_rule2, E_rule3, E_rule4 and E_rule5.

In this comparison, a total of 269 instances are solved by the task-operator-sequence solution generation embedded with 10 evolutionary rules mined by GEP and GP so that each instance is solved 10 times. Since these rules are embedded into the solution generation to calculate the priority values, the computational complexity is the same as that of the solution generation and equals to $O(n)$ when using these rules to solve the r-MMALBP. Therefore, a total of 2690 experiments are involved. We calculate the RPD values of 10 rules for each instance. Table 16 presents the ARPD values and average running times (269 data per average) of 10 rules mined by GEP and GP. As illustrated in Table 16, the ARPD values of the 5 evolutionary rules by GEP are 7.99, 8.45, 8.60, 8.42 and 8.68 respectively, which are lower than those of 5 evolutionary rules by GP. Besides, these rules have the same average running times. This observation suggests that the evolutionary rules by GEP outperform those by GP.

Apart from analyzing the robustness of the evolutionary rules, this section also employs statistical methods to count the average rank and the number of the best solutions found by each evolutionary rule. The statistical result results are shown in Fig. 13. Regarding the number of the best solutions found, the E_rules 1–5 respectively find 81, 70, 74, 71 and 79 best solutions out of 269 instances; while the G_rules 1–5 only find 52, 62, 63, 53 and 54 best solutions respectively. Besides, if using the average rank values to rank these rules, E_rules 1–5 make the top five, while G_rules 1–5 are ranked in the bottom five. Therefore, we can conclude that the evolutionary rules mined by GEP outperform those by GP.

Although GEP and GP train 30 instances to find the better combinations of heuristic rules 2, 4, 7, 11 and 15, the proposed GEP designs a multi-attribute representation to ensure the search efficiency. Besides, GP works on the tree structure, while GEP works on the linear string. When using the genetic operators to improve the population, we can easily find the differences among the individual of GEP. Meanwhile, the mutation operator aims to perturb the individual with a certain small probability. However, in a GP algorithm, if the mutation point is on the root node, and the while tree structure will be changed, and no information remains. To sum up, the mined evolutionary rules by GEP are superior to those by GP.

Table 16
ARPD values (269 data per average) and average running times of the evolutionary rules mined by GEP and GP.

GP rules	ARPD	CPU/ms	GEP rules	ARPD	CPU/ms
G_rule1	13.01	3572	E_rule1	7.99	3270
G_rule2	12.23	2799	E_rule2	8.45	2942
G_rule3	9.63	3194	E_rule3	8.60	3375
G_rule4	12.93	2728	E_rule4	8.42	3534
G_rule5	16.60	1418	E_rule5	8.68	3010

7. Conclusions, limitations, and future work

This paper studied the mixed-model assembly line balancing problem under uncertain demand. To obtain an efficient line configuration, a robust MILP model (called r-MMALBP), two robust solution generation mechanisms and a gene expression programming are designed. In the robust model, we relax the cycle time constraint and do a robust treatment of sequence constraint and objective functions. We consider a linear combination of four indicators in a single objective function by including the total number of operators, the number of workstations, the total number of operators whose finishing times exceed cycle time and the total number of workstations whose finishing times exceed cycle time.

A GEP is designed and applied to the problem to mine 10 efficient dispatching rules. In the GEP, multiple attributes related to the processing time and precedence relations are abstracted into 20 heuristic rules. Subsequently, 10 mined dispatching rules are embedded into two robust solution generation mechanisms to quickly capture efficient line configuration. 269 benchmark instances are generated to test the performance of the MILP model, two proposed solution generation mechanisms and 10 mined dispatching rules. The main three conclusions of our study are the following:

(1) The proposed MILP model solved by the CPLEX solver is effective in tackling those instances having 35 or less tasks.
(2) Task-operator-sequence solution generation outperforms operator-task-sequence solution generation when tackling mixed-model assembly line balancing problem under uncertain demand.
(3) The mined dispatching rules have significant superiority over the 20 existing heuristic rules and 5 dispatching rules mined by GP. Each of them can be embedded into task-operator-sequence solution generation to solve the real problem quickly and efficiently.

The dispatching rules can be directly applied in the practical assembly line balancing problems because of their low computational time and convenient implementation. They can also obtain a feasible solution quickly and efficiently; they can also be used to deal with the online balancing or rebalancing for mixed-model multi-manned assembly lines. However, the proposed method still has some limitations. The dispatching rules are mined according to a specific objective function. When the objective changes, it is necessary to re-train the GEP with the new instances to extract the appropriate dispatching rules. Besides, the accuracy of rules-based solutions is sometimes lower than those obtained by algorithms which do not generate a set of rules, but the rules-based solutions are more interpretability [50]. Hence, future work could improve the current results by developing a metaheuristic method to use the dispatching evolutionary rules to initialize the population to have a quicker convergence speed. Additionally, we may focus on enriching the model by considering the human–machine coordination in multi-manned or multi-machine or human–machine workstations. A multi-objective variant of the model can be also a promising future research path.

CRediT authorship contribution statement

Zikai Zhang: Methodology, Writing - original draft. **Qiuhua Tang:** Writing - review & editing. **Manuel Chica:** Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.asoc.2021.107513.

References

[1] T.C. Lopes, G.V. Pastre, A.S. Michels, L. Magatao, Flexible multi-manned assembly line balancing problem: Model, heuristic procedure, and lower bounds for line length minimization, Omega-Int. J. Manage. Syst. (2020) 95.

[2] A. Roshani, A. Roshani, A. Roshani, M. Salehi, A. Esfandyari, A simulated annealing algorithm for multi-manned assembly line balancing problem, J. Manuf. Syst. 32 (1) (2013) 238–247.

[3] M.N. Janardhanan, Z. Li, P. Nielsen, Model and migrating birds optimization algorithm for two-sided assembly line worker assignment and balancing problem, Soft Comput. (2018).

[4] J. Pereira, The robust (minmax regret) assembly line worker assignment and balancing problem, Comput. Oper. Res. 93 (2018) 27–40.

[5] Z. Zhang, Q. Tang, D. Han, Z. Li, Enhanced migrating birds optimization algorithm for U-shaped assembly line balancing problems with workers assignment, Neural Comput. Appl. 31 (11) (2018) 7501–7515.

[6] Z. Li, Q. Tang, L. Zhang, Minimizing energy consumption and cycle time in two-sided robotic assembly line systems using restarted simulated annealing algorithm, J. Cleaner Prod. 135 (2016) 508–522.

[7] Z. Zhang, Q. Tang, Z. Li, L. Zhang, Modelling and optimisation of energy-efficient U-shaped robotic assembly line balancing problems, Int. J. Prod. Res. 57 (17) (2018) 5520–5537.

[8] Z.K. Zhang, Q.H. Tang, L.P. Zhang, Mathematical model and grey wolf optimization for low-carbon and low-noise U-shaped robotic assembly line balancing problem, J. Cleaner Prod. 215 (2019) 744–756.

[9] I. Kucukkoc, Z. Li, A.D. Karaoglan, D.Z. Zhang, Balancing of mixed-model two-sided assembly lines with underground workstations: A mathematical model and ant colony optimization algorithm, Int. J. Prod. Econ. 205 (2018) 228–243.

[10] J.C. Chen, Y.-Y. Chen, T.-L. Chen, Y.-H. Kuo, Applying two-phase adaptive genetic algorithm to solve multi-model assembly line balancing problems in TFT–LCD module process, J. Manuf. Syst. 52 (2019) 86–99.

[11] P. Samouei, J. Ashayeri, Developing optimization & robust models for a mixed-model assembly line balancing problem with semi-automated operations, Appl. Math. Model. 72 (2019) 259–275.

[12] A. Otto, A. Scholl, Incorporating ergonomic risks into assembly line balancing, European J. Oper. Res. 212 (2) (2011) 277–286.

[13] J. Bautista, C. Batalla-Garcia, R. Alfaro-Pozo, Models for assembly line balancing by temporal, spatial and ergonomic risk attributes, European J. Oper. Res. 251 (3) (2016) 814–829.

[14] M.N. Janardhanan, Z. Li, G. Bocewicz, Z. Banaszak, P. Nielsen, Metaheuristic algorithms for balancing robotic assembly lines with sequence-dependent robot setup times, Appl. Math. Model. 65 (2019) 256–270.

[15] U. Özcan, Balancing and scheduling tasks in parallel assembly lines with sequence-dependent setup times, Int. J. Prod. Econ. 213 (2019) 81–96.

[16] A.S. Michels, T.C. Lopes, C.G.S. Sikora, L. Magatão, A benders' decomposition algorithm with combinatorial cuts for the multi-manned assembly line balancing problem, European J. Oper. Res. 278 (3) (2019) 796–808.

[17] S.G. Dimitriadis, Assembly line balancing and group working: A heuristic procedure for workers' groups operating on the same product and workstation, Comput. Oper. Res. 33 (9) (2006) 2757–2774.

[18] Y.-Y. Chen, A hybrid algorithm for allocating tasks, operators, and workstations in multi-manned assembly lines, J. Manuf. Syst. 42 (2017) 196–209.

[19] M. Chica, J. Bautista, Ó. Cordón, S. Damas, A multiobjective model and evolutionary algorithms for robust time and space assembly line balancing under uncertain demand, Omega 58 (2016) 55–68.

[20] L. Zhang, Q. Tang, Z. Wu, F. Wang, Mathematical modeling and evolutionary generation of rule sets for energy-efficient flexible job shops, Energy 138 (2017) 210–227.

[21] F. Akagi, H. Osaki, S. Kikuchi, A method for assembly line balancing with more than one worker in each station, Int. J. Prod. Res. 21 (5) (1983) 755–770.

[22] P. Fattahi, A. Roshani, A. Roshani, A mathematical model and ant colony algorithm for multi-manned assembly line balancing problem, Int. J. Adv. Manuf. Technol. 53 (1–4) (2010) 363–378.

[23] T. Kellegöz, B. Toklu, An efficient branch and bound algorithm for assembly line balancing problems with parallel multi-manned workstations, Comput. Oper. Res. 39 (12) (2012) 3344–3360.

[24] T. Kellegöz, Assembly line balancing problems with multi-manned stations: a new mathematical formulation and gantt based heuristic method, Ann. Oper. Res. 253 (1) (2016) 377–404.

[25] T. Kellegöz, B. Toklu, A priority rule-based constructive heuristic and an improvement method for balancing assembly lines with parallel multi-manned workstations, Int. J. Prod. Res. 53 (3) (2015) 736–756.

[26] A. Roshani, D. Giglio, Simulated annealing algorithms for the multi-manned assembly line balancing problem: minimising cycle time, Int. J. Prod. Res. (2016).

[27] Y. Li, H. Wang, Z. Yang, Type II assembly line balancing problem with multi-operators, Neural Comput. Appl. (2018).

[28] A. Roshani, F. Ghazi Nezami, Mixed-model multi-manned assembly line balancing problem: a mathematical model and a simulated annealing approach, Assem. Autom. 37 (1) (2017).

[29] Y.-Y. Chen, C.-Y. Cheng, J.-Y. Li, Resource-constrained assembly line balancing problems with multi-manned workstations, J. Manuf. Syst. 48 (2018) 107–119.

[30] M. Sahin, T. Kellegoz, A new mixed-integer linear programming formulation and particle swarm optimization based hybrid heuristic for the problem of resource investment and balancing of the assembly line with multi-manned workstations, Comput. Ind. Eng. 133 (2019) 107–120.

[31] B. Yuan, C. Zhang, X. Shao, Z. Jiang, An effective hybrid honey bee mating optimization algorithm for balancing mixed-model two-sided assembly lines, Comput. Oper. Res. 53 (2015) 32–41.

[32] R. Ramezanian, A. Ezzatpanah, Modeling and solving multi-objective mixed-model assembly line balancing and worker assignment problem, Comput. Ind. Eng. 87 (2015) 74–80.

[33] Q. Tang, Y. Liang, L. Zhang, C.A. Floudas, X. Cao, Balancing mixed-model assembly lines with sequence-dependent tasks via hybrid genetic algorithm, J. Global Optim. 65 (1) (2015) 83–107.

[34] K. Buyukozkan, I. Kucukkoc, S.I. Satoglu, D.Z. Zhang, Lexicographic bottleneck mixed-model assembly line balancing problem: Artificial bee colony and tabu search approaches with optimised parameters, Expert Syst. Appl. 50 (2016) 151–166.

[35] Y. Delice, E. Kızılkaya Aydoğan, U. Özcan, M.S. Ilkay, A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing, J. Intell. Manuf. 28 (1) (2017) 23–36.

[36] L. Tiacci, M. Mimmi, Integrating ergonomic risks evaluation through OCRA index and balancing/sequencing decisions for mixed model stochastic asynchronous assembly lines, Omega 78 (2018) 112–138.

[37] M. Chica, Ó. Cordón, S. Damas, J. Bautista, A robustness information and visualization model for time and space assembly line balancing under uncertain demand, Int. J. Prod. Econ. 145 (2) (2013) 761–772.

[38] I. Moya, M. Chica, J. Bautista, Constructive metaheuristics for solving the car sequencing problem under uncertain partial demand, Comput. Ind. Eng. (2019) 137.

[39] Z. Abidin Çil, D. Kizilay, Constraint programming model for multi-manned assembly line balancing problem, Comput. Oper. Res. (2020) 124.

[40] X. Qian, Q. Fan, International conference on information management, in: 2011 International Conference on Information Management, Innovation Management and Industrial Engineering, 2011, pp. 320–323.

[41] Z. Zhang, Q. Tang, M. Chica, Multi-manned assembly line balancing with time and space constraints: A MILP model and memetic ant colony system, Comput. Ind. Eng. (2020) 150.

[42] A. Baykasoğlu, L. Özbakır, Discovering task assignment rules for assembly line balancing via genetic programming, Int. J. Adv. Manuf. Technol. 76 (1–4) (2014) 417–434.

[43] Z. Zhang, Q. Tang, Z. Li, D. Han, An efficient migrating birds optimization algorithm with idle time reduction for type-i multi-manned assembly line balancing problem, J. Syst. Eng. Electron. 32 (2) (2021) 286–296.

[44] C. Ferreira, Gene Expression Programming in Problem Solving, in: Soft computing and industry, Springer, 2002, pp. 635–653.

[45] Y. Yang, X. Li, L. Gao, X. Shao, Modeling and impact factors analyzing of energy consumption in CNC face milling using GRASP gene expression programming, Int. J. Adv. Manuf. Technol. 87 (5–8) (2013) 1247–1263.

[46] J. Zhong, Y.-S. Ong, W. Cai, Sef-learning gene expression programming, IEEE Trans. Evol. Comput. 20 (1) (2016) 65–80.

[47] L. Zhang, Z. Li, G. Królczyk, D. Wu, Q. Tang, Mathematical modeling and multi-attribute rule mining for energy efficient job-shop scheduling, J. Cleaner Prod. (2019).

[48] K. Xu, Y. Liu, R. Tang, J. Zuo, J. Zhu, C. Tang, A novel method for real parameter optimization based on gene expression programming, Appl. Soft Comput. 9 (2) (2009) 725–737.

[49] H.H. Miyata, M.S. Nagano, J.N.D. Gupta, Integrating preventive maintenance activities to the no-wait flow shop scheduling problem with dependent-sequence setup times and makespan minimization, Comput. Ind. Eng. 135 (2019) 79–104.

[50] S. García, A. Fernández, J. Luengo, F. Herrera, A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability, Soft Comput. 13 (10) (2008) 959.