APEM journal
Advances in Production Engineering & Management
Volume 19 | Number 1 | March 2024 | pp 31–45
https://doi.org/10.14743/apem2024.1.491
ISSN 1854-6250
Journal home: apem-journal.org
Original scientific paper

Human-robot collaboration assembly line balancing considering cross-station tasks and the carbon emissions

Li, Y.C.[a,*], **Wang, X.**[b]

[a]School of Economics and Management, Beijing University of Technology, P.R. China
[b]School of Economics and Management, Beijing University of Technology, P.R. China

A B S T R A C T

With the growth of industrialization, the global manufacturing industry is continually evolving and reforming in the direction of intelligence and green production. Industrial robots have replaced human workers because of the benefit of production efficiency. However, the large-scale application of robots requires a large amount of energy consumption and generates a large amount of CO_2, which will lead to energy waste and environmental pollution. In addition, in term of performing some particular tasks, current robot technology cannot achieve the same level of intelligence as human. Therefore, the design trend of assembly lines in industry has shifted from traditional configuration to human-robot collaboration to achieve higher productivity and flexibility. This paper investigates the human-robot collaboration (HRC) assembly line balancing problem, taking cycle time and carbon emission as primary and secondary objectives. A new mixed-integer programming model that features a cross-station design is formulated. A particle swarm algorithm (PSO) with two improvement rules is designed to solve the problems. The comparative experiments on ten benchmark datasets are conducted to assess the performance of the proposed algorithm. The experimental results indicate that the improved particle swarm algorithm is superior to the other two heuristics: simulated annealing (SA) and the late acceptance hill-climbing heuristic (LAHC).

A R T I C L E   I N F O

Keywords:
Assembly line balancing problem;
Human-robot collaboration;
Cross-station tasks;
Carbon emissions;
Collaborative robot (cobot);
Particle swarm algorithm (PSO)

**Corresponding author:*
15831029622@163.com
(Li, Y.C.)

Article history:
Received 26 February 2024
Revised 17 April 2024
Accepted 19 April 2024

1. Introduction

Assembly line (AL) plays an essential role in industrial production. The assembly line balancing problem (ALBP) refers to an actual production scheduling problem of assigning the assembly sequence, distributing the production tasks, and dispatching agents to workstations appropriately to meet production targets in the manufacturing process. With the development of intelligent manufacturing technology, the application of industrial robots can not only improve product quality but also improve production efficiency. Robot assembly lines (RAL) have been widely used in various production fields of the manufacturing industry.

In recently years, we find that even though the robot is highly advanced, it can still not carry out some production jobs with high accuracy or flexibility. Furthermore, robots consume a large amount of electricity during operation, leading to energy consumption and the generation of greenhouse gases such as CO_2, exacerbating the trend of global warming. To address these issues, human-robot collaboration (HRC) is evolving into a new mode of production, directing the continuous development regarding the intelligent and environmental friendly manufacturing.

This study considers a design of "cross-station task" which has already been used in the actual production process of manufacturing enterprises. It is possible for a "cross-station task" to be processed simultaneously at multiple stations. By employing this design, it can reduce the idle time of the workstations on the AL, boost the workstations' production efficiency and reduce the energy consumption and carbon emissions.

In this paper, we build a multi-objective programming model for balancing collaborative assembly lines between humans and robots while considering carbon emissions. The model considers the carbon emissions of various types of robots when optimizing two objectives, production efficiency represented by cycle time and carbon emissions. The contributions are twofold. Firstly, an ALBP-HRC considering a "cross-station task" design, which has never been discussed in studies of human-robot interactions on AL, is well studied. Secondly, this research utilizes and enhances the particle swarm optimization (PSO) algorithm to solve medium-scale and large-scale problems. Two improvement rules, "Task exchange" and "Set expansion," are designed for the proposed PSO algorithm, which can reduce the cycle time or find more Pareto solutions. The proposed PSO are compared with two other heuristics, simulated annealing (SA) and the late acceptance hill-climbing heuristic (LAHC) on solving a set of benchmark problems. The results show that PSO outperforms the other algorithms in terms of three different metrics.

The remainder of this paper is organized as follows. Section 2 describes the current research progress in the literature. We propose a mixed-integer programming model in Section 3. An algorithm, based on the particle swarm optimization algorithm (PSO) is developed in Section 4 to find the optimal solution. Experimental studies are conducted in Section 5. Section 6 concludes the paper. The notations are defined in Table 1 and used throughout the paper.

Table 1 Notations

N	Total number of the agent types
n	Total number of the tasks
m	Total number of the workstations
a	Index of agent
i,j	Index of task ($\forall\ i,j\ = 1,2,\dots,n$)
s,h	Index of workstation ($\forall\ s,h\ = 1,2,\dots,m$)
t_{ia}	The operation time of task i by agent a
Wl_{as}	The workload of agent a at station s
$P_r(i)$	Index of the immediate predecessors of task i
OPC_a	The operation energy consumption of the agent a per unit time
SEC_a	The standby energy consumption of the agent a per unit time
ECF_{elc}	Carbon emissions per unit of electricity consumption
TCF	The total carbon emissions
c	The cycle time
γ	The maximum time that can be shared between two stations
$\emptyset$	A large positive number
x_{is}	0-1 variables, $x_{is} = 1$ if task i is allocated to workstation s and 0 otherwise
y_{as}	0-1 variables, $y_{as} = 1$if agent a is allocated to workstation s and 0 otherwise
z_{ias}	0-1 variables, $z_{ias} = 1$ if task i operated by agent a is allocated to workstation s and 0 otherwise
k_{sh}	0-1 variables, $k_{sh} = 1$ if workstation s utilizes the cycle time of workstation h and 0 otherwise
u_{as}	A non-negative value depicts the idle time of the agent a at workstation s
v_{sh}	A non-negative value indicates how much of workstation h's cycle time that workstation s occupies

2. Literature review

This part reviews relevant research on robotic assembly line balancing problems (RALBPs) in Section 2.1, the ALBP-HRC studies in Section 2.2, and the studies on ALBPs considered the carbon emissions in Section 2.3.

2.1 Review of robotic assembly line balancing problem

The research on RALBP can be traced back to the 1990s. Rubinovitz*et al.* [1] first proposed the concept of the robot assembly line in 1993, and they proposed a linear programming model for the RALBP-I problem. Yoosefelahi *et al.* [2] studied the RALBP-II problem to simultaneously optimize multiple objectives, namely, robot costs, and cycle time. A mixed-integer programming (MILP) model was established. As the problem is NP-hard, the article proposed three metaheuristic methods to solve the above problem. More recently, Nilakantan [3] were the first to solve the RALBP-II problem in a U-shaped assembly line layout, they proposed a 0-1 IP model for small-scale problems. Wang *et al.* [4] improved traditional particle swarm optimization algorithms by introducing several extension operators, enabling them to solve process planning (PP) problems such as robot assembly lines. Borba *et al.* [5] proposed two new algorithms, namely a "branch-bound and remember" algorithm and an "iterative beam search" algorithm with problem-specific dominance rules to minimize the cycle time in RALBP. Li *et al.* [6] proposed a mixed integer linear programming model to minimize the cycle time. Raatz *et al.* [7] proposed a RALBP that minimized the cost. A multi-objective optimization method, namely the genetic algorithm, was used to solve the proposed problem. Jiang *et al.* [8] proposed an improved genetic algorithm to optimize the balance problem in the clothing production line in response to the low balance rate and the uneven work intensity of employees. The effectiveness of the algorithm was verified through simulation experiments. Şahin *et al.* [9] studied a robot stochastic assembly line balancing problem (RSALBP) based on the assumption that the task time is fixed. They proposed a mixed-integer second-order cone programming model and a constraint programming model to solve the problem given the number of workstations and robots.

Similar to RALBP, the disassembly line balancing of RAL is also NP-hard. Intelligent optimization algorithms have performed well in solving this type of problem. Based on analyzing the disassembly information of automotive components, Yu *et al.* [10] established a disassembly model for automotive components. The optimal disassembly sequence was obtained by considering the mapping between the Floyd Warhill algorithm and car disassembly patterns. Wang *et al.* (2021) [11] studied the multi-objective disassembly line balance problem and proposed an improved genetic algorithm to solve the model.

2.2 Review of human-robot collaboration assembly line balancing problem

The flexibility of HRC-AL is much higher than that of traditional robotic or manual assembly lines, which can improve the efficiency of ALs, enhance the work enthusiasm of workers, and become the primary production mode of ALs chosen by many enterprises today. Ding *et al.* [12] adopted a human and robot collaborative hybrid assembly cell to develop an automatic subtask allocation strategy. Nikolakis *et al.* [13] adopted a two-level breakdown where tasks were translated to specific operations, being carried out by humans or robots to realize an adaptive and more efficient execution of the production schedule. Mura *et al.* [14] , who developed a genetic algorithm to reduce the cost of the assembly line, the number of qualified workers needed, and the variation in worker energy loads. Zanchettin *et al.* [15] used a fuzzy-timed Petri net with uncertain task time to minimize idle time. Vieira *et al.* [16] developed a novel optimization simulation based on the Recursive Optimization-Simulation Approach (ROSA) methodology to prioritize reducing costs and the makespan. Aljinovic *et al.* [17] utilized a systematic framework with the mathematical model to minimize cycle time. Riedel *et al.* [18] provided the architecture and implementation details of an assembly assistance system based on object detection models using deep learning and machine learning algorithms. Nourmohammadi *et al.* (2022) [19] studied ALBP with HRC, where human workers and robots share the same workplace to process tasks simultaneously. They developed a new mixed-integer linear programming model and proposed a neighborhood-search simulated annealing algorithm to solve the problem.

2.3 Review of assembly line balancing problem with carbon emissions

Global climate change has triggered a series of ecological, social, and economic problems, gradually increasing people's environmental awareness. Enterprises have also begun to explore low-carbon and low-energy production methods, taking the path of sustainable development. Many existing studies on ALs also incorporate the goal of reducing carbon emissions and energy consumption. Lin *et al.* [20] developed an integrated model for parameter optimization and process workshop scheduling to minimize the number of workstations and carbon emissions during the production process. Li *et al.* [21] presented a restarted simulated annealing algorithm to minimize energy consumption and cycle time simultaneously. Nilakantan *et al.* [22] proposed a multi-objective coevolutionary algorithm to minimize the carbon emissions of RAL. It is the first time carbon emissions have been considered in a robot assembly line system. Zhang *et al.* [23] established a multi-objective mathematical model with energy consumption and line balance rate. They proposed a multi-objective algorithm combining cellular strategy and local search to solve this multi-objective problem. Sun *et al.* [24] proposed an energy-efficient robot assembly line balancing (EERALB) problem to minimize cycle time and total energy consumption. They proposed a multi-objective mathematical model and a boundary-oriented mixed distribution estimation algorithm to solve this problem. Zhou *et al.* [25] considered robots with different efficiency and energy consumption rates in their programming model, with the total energy consumption and workload as optimization objectives. They proposed an improved MOEA/D algorithm and evaluated its superiority through computational experiments.

With the continuous development of industrialization, the relevant literature on human-robot collaborative assembly lines is constantly increasing, but there are still some research gaps worth exploring and investigating. For example, current research emphasizes traditional production objectives such as the number of stations, cycle time and cost but neglects carbon emissions. The related research which seeks the balance between carbon emissions and production efficiency in collaborative assembly lines is still absent. To this end, we propose a mixed-integer programming model for optimizing the cycle time and total carbon emissions for HRC-AL with "cross-station task" design.

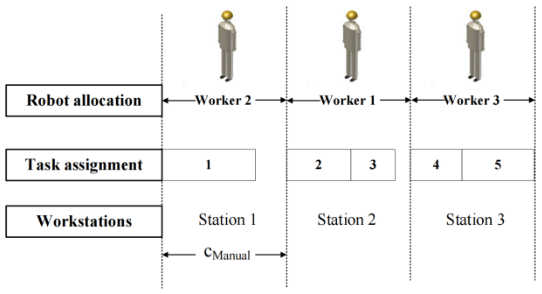
3. Mathematical model

In this section, we explain the structure of the research problem and establish a mixed-integer programming model to formulate the above problem.

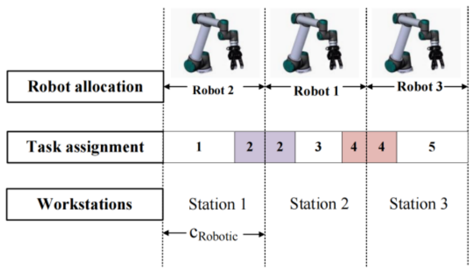
3.1 Problem description

In this paper, we consider a homogeneous product. Assume that there are N types of agents where the former $(N-1)_{th}$ agents refer to robots and the N_{th} agent refers to human worker. The task operation time varies depending on the agent type. In the "cross-station task" design, one task might be operated concurrently at the assigned station and its rear station or the front station (if it exists). Because the industrial robotic arms contain multiple joints that act as axes controlling movement, the application of industrial robotic arms allows a task can be processed at a pair of stations simultaneously. On the contrary, the human workers cannot process more than one task. We provide some examples to illustrate our statements. In Fig. 1(a), tasks cannot be shared between stations resulting in some idle time which may lead to low production efficiency. In Fig. 1(b), task 2 and task 4 can be processed by the robots in advance at the previous station. In Fig. 1(c), cycle time cannot be shared between station 1 and 2 due to the assignment of human worker, but task 4 can be processed in advance by the robot at station 3. Given the number of stations and the task processing sequence, we can observe that the "cross-station task" design can reduce cycle time by comparing c_{Manual}, $c_{Robotic}$ and c_{HRC}.

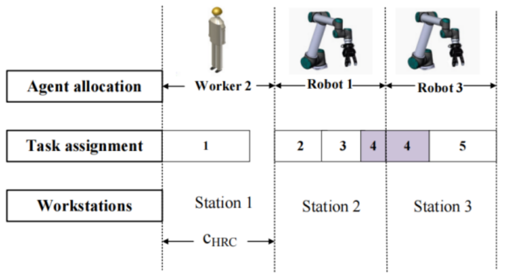
The objective of the optimization problem is to minimize the cycle time and the total carbon emissions. To address the aforementioned problem, we propose a mixed-integer programming model called "ALBP-HRC-CS," aiming at minimizing the two objectives in the Pareto optimization framework.



(a) The assembly line with only human workers



(b) The assembly line with only robots and cross-station task design



(c) The assembly line with the human-robot collaboration and cross-station task design

Fig. 1 The layout of the assembly line

3.2 Mathematical formulation

The formulation of the mathematical model is as follows.

$$min\ c \tag{1}$$

$$min\ TCF \tag{2}$$

$$EC = OEC + SEC \tag{3}$$

$$OEC = \sum_{s=1}^{m} OEC_s \tag{4}$$

$$SEC = \sum_{s=1}^{m} SEC_s \tag{5}$$

$$OEC_s = \sum_{a=1}^{N}\sum_{i=1}^{n} OPC_a \times z_{ias} \times t_{ia}, \forall s = 1, \dots m \tag{6}$$

$$OEC_s = \sum_{a=1}^{N} u_{as} \times SPC_a, \forall s = 1, \dots m \tag{7}$$

The objective function (Eq. 1) minimizes the cycle time of all product models. The objective function (Eq. 2) minimizes the total carbon emissions. Robots in operation or on standby consume electricity and produce carbon emissions. Therefore, the total energy consumption is equal to the sum of all robotic workstations' standby and operation energy consumption. The total carbon emissions are equal to the product of total energy consumption and carbon emission coefficient. Eqs. 3 to 7 compute the overall energy consumption for all workstations and the individual energy consumption of each workstation. It should be noted that the agent's OPC_a and SPC_a values are 0 if it is a human worker.

Now we first introduce the basic constraints of the proposed model.

$$\sum_{s=1}^{m} x_{is} = 1, \forall s = 1, \ldots m, \forall i = 1, \ldots n \tag{8}$$

$$\sum_{a=1}^{N} y_{as} = 1, \forall s = 1, \ldots m, \forall a = 1, \ldots N \tag{9}$$

$$x_{is} + y_{as} \leq z_{ias} + 1, \forall s = 1, \ldots m, \forall i = 1, \ldots n, \forall a = 1, \ldots N \tag{10}$$

$$x_{is} + (1 - y_{as}) \leq (1 - z_{ias}) + 1, \forall s = 1, \ldots m, \forall i = 1, \ldots n, \forall a = 1, \ldots N \tag{11}$$

$$(1 - x_{is}) + (1 - y_{as}) \leq (1 - z_{ias}) + 1, \forall s = 1, \ldots m, \forall i = 1, \ldots n, \forall a = 1, \ldots N \tag{12}$$

$$(1 - x_{is}) + y_{as} \leq (1 - z_{ias}) + 1, \forall s = 1, \ldots m, \forall i = 1, \ldots n, \forall a = 1, \ldots N \tag{13}$$

$$\sum_{s=1}^{m} s \times x_{js} \leq \sum_{s=1}^{m} s \times x_{is}, \forall j \in P_r(i), \forall s = 1, \ldots m, \forall i = 1, \ldots, n \tag{14}$$

Constraints (Eqs. 8 and 9) are indivisibility of tasks/agents, meaning that a task/agent can only be allocated to one workstation. Constraints (Eqs. 10 to 13) ensure that the task can be executed, that is, when a task and an agent are allocated to the same workstation, the agent must process the task. Constraint (Eq. 14) is the task precedence constraints.

Next, the constraints related to the cross-station design are illustrated.

$$\begin{gathered} u_{as} \leq c + v_{s,s+1} + v_{s,s-1} - v_{s+1,s} - v_{s-1,s} - \sum_{a=1}^{N}\sum_{i=1}^{n} z_{ias} \times t_{ia} + \emptyset(1 - y_{as}), \\ \forall s = 1, \ldots m, \forall i = 1, \ldots n, \forall a = 1, \ldots N \end{gathered} \tag{15}$$

$$\begin{gathered} u_{as} \geq c + v_{s,s+1} + v_{s,s-1} - v_{s+1,s} - v_{s-1,s} - \sum_{a=1}^{N}\sum_{i=1}^{n} z_{ias} \times t_{ia} - \emptyset(1 - y_{as}), \\ \forall s = 1, \ldots m, \forall i = 1, \ldots n, \forall a = 1, \ldots N \end{gathered} \tag{16}$$

$$v_{s,s+1} \leq \gamma, \forall s = 1, \ldots m \tag{17}$$

$$v_{s+1,s} \leq \gamma, \forall s = 1, \ldots m \tag{18}$$

$$k_{s,s+1} \leq \sum_{a=1}^{N-1} y_{as}, \forall s = 1, \ldots m - 1 \tag{19}$$

$$k_{s,s-1} \leq \sum_{a=1}^{N-1} y_{as}, \forall s = 1, \ldots m \tag{20}$$

$$k_{s,s+1} \leq 1 - y_{Ns}, \forall s = 1, \ldots m - 1 \tag{21}$$

$$k_{s,s-1} \leq 1 - y_{Ns}, \forall s = 1, \ldots m \tag{22}$$

$$k_{s,s+1} + k_{s+1,s} \leq 1, \forall s = 1, \ldots m \tag{23}$$

$$v_{s,s+1} \leq \emptyset \cdot k_{s,s+1}, \forall s = 1, \ldots m \tag{24}$$

$$v_{s+1,s} \leq \emptyset \cdot k_{s+1,s}, \forall s = 1, \ldots m \tag{25}$$

$$v_{s,s+1} \geq 0.1 \cdot k_{s,s+1}, \forall s = 1, \ldots m \tag{26}$$

$$v_{s+1,s} \geq 0.1 \cdot k_{s+1,s}, \forall s = 1, \dots m \tag{27}$$

Constraints (Eqs. 15 and 16) are in effect when agent a is allocated to workstation s ($y_{as} = 1$), and they calculate the agent's idle time if that agent is allocated. Constraints (Eqs. 17 to 18) restrict the maximum amount of time that the robot at one station consumes the cycle time of another station. Constraints (Eqs. 19 and 20) ensure that the ability to utilize the cycle time of the front or rear station is only possible if the station is furnished with robots. Constraints (Eqs. 21 to 22) ensure that human workers cannot utilize the cycle time of the front or rear station. Constraints (Eqs. 23 to 27) denote that tasks shared between adjacent stations can only occur once.

$$v_{s,s+1}, v_{s+1,s} \geq 0, \forall s = 1, \dots, m \tag{28}$$

$$u_{as} \geq 0, \forall s = 1, \dots m, \forall a = 1, \dots N \tag{29}$$

$$v_{0,1},\ v_{1,0}, v_{m,m+1}, v_{m+1,m} = 0 \tag{30}$$

Constraints (Eqs. 28 to 30) are the domain constraints.

4. Methods

In this section, a particle swarm optimization algorithm (PSO) with two improvement rules is adapted to address the "ALBP-HRC-CS". The general framework of PSO is introduced in section 4.1. Variable neighborhood search mechanism is developed in section 4.2. Encoding and decoding schemes are designed in section 4.3. Finally, two improvement rules are delineated in section 4.4.

4.1 The general framework of PSO

Kennedy *et al.* [26] developed the simplified version of PSO in 1995 by the insight of birds' feeding behavior. Birds can locate most food locations through collective information sharing. In the mathematical model of "ALBP-HRC-CS", the PSO algorithm minimizes the whole assembly line's cycle time and carbon emissions by changing agent sequences and task sequences. The relevant parameters are displayed in Table 2.

Table 2 Related parameters

$iter, iter'$	The iteration index
$iter_{max}$	The maximal number of iterations
k_1, k_2	The index for agent particles and task particles
X_{k_1}, X_{k_2}	The position for agent particles and task particles
V_{k_1}, V_{k_2}	The velocity for agent particles and task particles
P_{best}, G_{best}	The local and global best of the particles
l_1, l_2	Learning coefficients
U_1, U_2	Uniform random numbers between [0,1]

In PSO, each particle in the search domain is constantly moving. During the movement, the particle's speed is changing, resulting in the change of the final position of the particle. The position update for the agent and task follow Eq. 31 and Eq. 32, respectively.

$$X_{k_1}^{iter+1} = X_{k_1}^{iter} + V_{k_1}^{iter+1} \tag{31}$$

$$X_{k_2}^{iter+1} = X_{k_2}^{iter} + V_{k_2}^{iter+1} \tag{32}$$

The velocitiy updates for the agent and task follow Eqs. 33 and 34, respectively.

$$V_{k_1}^{iter+1} = V_{k_1}^{iter} + l_1 \times U_1 \times (P_{best_{iter}} - X_{k_1}^{iter}) + l_2 \times U_2 \times (G_{best_{iter}} - X_{k_1}^{iter}) \tag{33}$$

$$V_{k_2}^{iter+1} = V_{k_2}^{iter} + l_1 \times U_1 \times (P_{best_{iter}} - X_{k_2}^{iter}) + l_2 \times U_2 \times (G_{best_{iter}} - X_{k_2}^{iter} \tag{34}$$

PSO is an intelligent optimization algorithm with high local convergence and a "premature" phenomenon. If the PSO falls into a local extreme value early in the iteration, it will be difficult to jump out of it later, dramatically reducing the efficiency of the algorithm and significantly reducing the probability that the algorithm will find a globally optimal solution. To address this problem, “Task exchange” and “Set expansion” rules are developed to enhance the quality of the Pareto set’s solutions. The PSO also features a variable neighborhood search (VNS) mechanism.

The main body of the improved PSO is shown in Fig. 2.

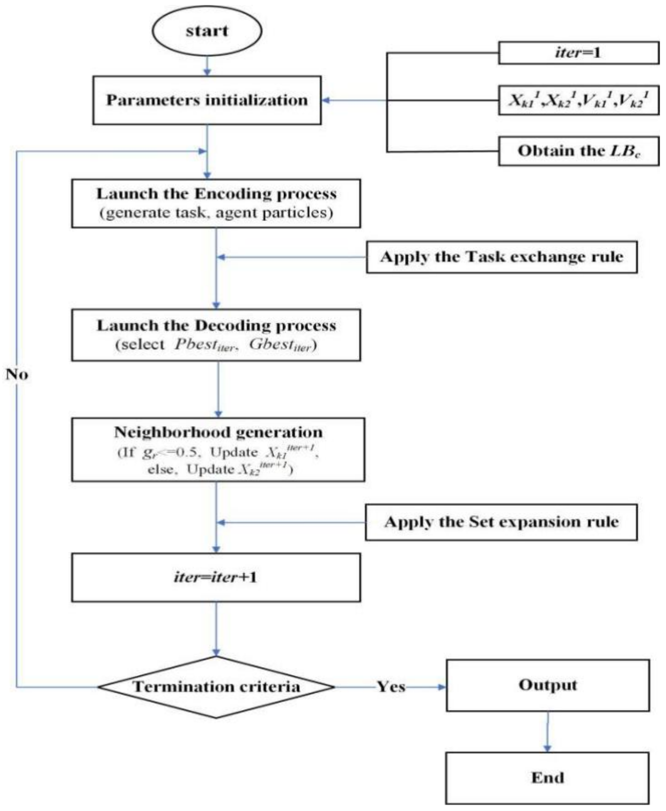

start
Parameters initialization
iter=1
$X_{k1}^1, X_{k2}^1, V_{k1}^1, V_{k2}^1$
Obtain the LB_c
Launch the Encoding process (generate task, agent particles)
Apply the Task exchange rule
Launch the Decoding process (select $Pbest_{iter}$, $Gbest_{iter}$)
No
Neighborhood generation (If g_r<=0.5, Update X_{k1}^{iter+1}, else, Update X_{k2}^{iter+1})
Apply the Set expansion rule
iter=iter+1
Termination criteria
Yes
Output
End

Fig. 2 Main body of the improved PSO

4.2 Encoding and decoding

Encoding

In this section, the task sequence Sq_{task} and agent sequence Sq_{agent} according to the constraints are arrayed. First, we obtain the initial Sq_{task} and Sq_{agent} as follows.

Initial Sq_{task}

Step 1: Set $Sq_{task} = [\,]$

Step 2: Set $X_{k_1}^1 = 1 + random[0,1] * n$, where [0,1] is a randomly generated number that conforms to the uniform distribution [0,1]

Step 3: The task is arrayed in a non-increasing order according to the $X_{k_1}^1$ under the constraint of task priority, and the Sq_{task} is attained

Step 4: If a task violates the task priority relationship when it is assigned, skip the task, assign the next task, and return to Step 3; otherwise, go to Step 5

Step 5: Continue Steps 3 and 4 until all tasks have been assigned

Initial Sq_{agent}

Step 1: Set $Sq_{agent} = [\,]$

Step 2: Set $X_{k_2}^1 = 1 + random[0,1] * N$, where [0,1] is a randomly generated number that conforms to the uniform distribution [0,1]

Step 3: Assign the agent with the largest values to the workstation

Step 4: Repeat Step 3 until all the stations are equipped with an agent

Decoding

As we know, lower bounds for the objective variables is crucial for the decoding process. We develop the lower bounds for c and TCF (LB_c and LB_{TCF}). It is straightforward that the lower bound for TCF: $LB_{TCF} = 0$ (i.e., the manual assembly line). Next, according to the necessary conditions for the feasibility of the model, LB_c can be determined as: $LB_c = [max(\sum_{i=1}^{n} min_{a=1}^{N} t_{ia}/m, max_{i=1}^{n} min_{a=1}^{N} t_{ia})]$. The decoding scheme is listed below.

Step 1: Load $t_{ia}, OPC_a, SPC_a, ECF_{elc}, LB_c, P(i), m, \gamma, Sq_{task}, Sq_{agent}$, set $s = 1, ll = 1$
Step 2: Arrange agents for each station according to Sq_{agent}
Step 3: Arrange the ll_{th} task in the Sq_{task} to station s and compute the workload of station s, Wl_{as}, go to the Step 4; otherwise, if there is no task to be assigned, go to Step 13
Step 4: If $Wl_{as} \geq LB_c + v_{s,s-1} - v_{s-1,s}$ AND $s < m$, go to Step 5; if $Wl_{as} \geq LB_c + v_{s,s-1} - v_{s-1,s}$ AND $s == m$, go to Step 12; otherwise, go back to Step 3, and $ll = ll + 1$
Step 5: If $OPC_a(a_s(s)) == 0$, the agent assigned to the station s is a human worker who cannot utilize the cycle time of the front or rear station, go to Step 9; otherwise, go to step 6
Step 6: If $Wl_{as} + t_{ia} \leq LB_c + v_{s,s-1} - v_{s-1,s} + \gamma$ AND $t_{ia} \leq LB_c + v_{s,s-1} - v_{s-1,s} - min(\gamma, LB_c + v_{s,s-1} - v_{s-1,s} - Wl_{as}) - Wl_{as} + t_{ia}$, go to Step 7; otherwise, go to Step 9
Step 7: Task $i = Sq_{task}(ll)$ is assigned to the station s, update Wl_{as} and $ll = ll + 1$, and then, repeat this step; otherwise, go to Step 8
Step 8: Set $v_{s,s+1} = Wl_{as} + t_{ia} - (LB_c + v_{s,s-1} - v_{s-1,s}), v_{s+1,s} = 0$, and go to Step 11
Step 9: Task $i = Sq_{task}(ll)$ is assigned to the station $s + 1$, update Wl_{as} and $ll = ll + 1$, go to Step 10
Step 10: Set $v_{s+1,s} = min(\gamma, LB_c + v_{s,s-1} - v_{s-1,s} - \sum_{j=1}^{n} z_{jas} t_{ja}), v_{s,s+1} = 0$, go to Step 11
Step 11: OEC_s and SEC_s are computed according to Eqs. 5 to 10, $s = s + 1$, and then, return to Step 3
Step 12: The total task time is computed for all unassigned tasks TR, and set $LB_c = LB_c + [TR/m]$, $s = 1, ll = 1$, return to the Step 3
Step 13: Compute TCF and $c = LB_c$

In the decoding process, the initial parameters are loaded in Step 1. The agent is assigned to each station according to Sq_{agent} in Step 2. Step 3 arranges tasks and computes the workload of the station. In Step 4, three situations may occur: if the current station is overloaded and is not the last station, go to Step 5; if the current station is overloaded but is the last station, go to Step 12; otherwise, return to Step 3. Step 5 shows that the station equipped with a human worker cannot utilize the time from the adjacent stations. Step 6 schedules this task to the current station through the "cross-station task" design and goes to Step 7; if it does not, go to Step 9. Step 7 updates the workload of the current station. Step 8 calculates the time it takes from its rear station. Step 9 assigns this task to the next station and updates the workload of the current station; Step 10 calculates the idle time of the current station (which can be used by other stations). The total energy consumption for the workstation is computed in Step 11. In Step 12, LB_c is increased to maintain feasibility. The above steps are designed to obtain a feasible task sequence. Based on it, we can calculate the values of the objective functions (TCF and c) in step 13.

4.3 Variable Neighborhood Search

Variable Neighborhood Search (VNS) is an enhanced local search mechanism. For alternating searches, it utilizes the neighborhood structure of various actions, striking a suitable balance between concentration and dispersion. First, the initial solution is generated and its neighborhoods is defined. Given the initial solution, VNS systematically selects new solution between two neighborhoods related to task and agent, respectively. A random number g_r is generated to select the part of the particle subject to change during each iteration. For instance, if $g_r \leq 0.5$, the positions and velocities are updated for the agent sequences while the task sequences remain unchanged. The new solution becomes the best if it can lead to a better objective. The algorithm terminates when the termination criterion (e.g., runtime limit) is satisfied.

4.4 Improvement rules

This section considers “Task exchange” and “Set expansion” as two enhanced mechanisms to find more solutions. “Task exchange” can enhance the efficiency of the PSO algorithm by swapping the processing orders of two different tasks operated by two stations within a feasible task sequence. “Set expansion” is proposed to address problem where the PSO algorithm is vulnerable to “premature convergence” and “local optimal solutions”. “Set expansion” can be used to discover more Pareto solutions.

Task exchange

The “task exchange” mechanism can be executed as follows: after obtaining a feasible solution, exchange the processing order and stations of two different tasks; assume that task a and task b are processed by agent r and r^* and assigned to station s and s^*, respectively (agents here can be human workers or robots); the tasks can be exchanged if the following criteria are met.

1. The exchange between task a and task b does not violate the precedence relationship;
2. “$t_{ar} + t_{br^*} > t_{ar^*} + t_{br}$” or “$t_{ar} + t_{br^*} == t_{ar^*} + t_{br}$ and $TCF_{ar} + TCF_{br^*} > TCF_{ar^*} + TCF_{br}$”;

The above criteria ensure that following task exchange, the cycle time can be decreased or the overall carbon emissions can be reduced without increasing the cycle time. When task exchange is applied, the decoding process is executed again to find a new cycle time and a new TCF.

Set expansion

Among the non-dominated solutions in the Pareto set, some solutions are far away from each other, resulting in a blank area in the middle of two solutions. The appearance of the blank area indicates some solutions in Pareto set are not available. To address that issue, the search area shall be expanded. The rule is explained as follows.

Step 1: Input the initial Pareto set
Step 2: Calculate the cycle time difference between two adjacent feasible solutions respectively, and calculate the mean value of these differences to obtain c_{diff}
Step 3: Select the two solutions with the largest differentials of cycle times, called c_1 and c_2 (assume $c_1 > c_2$)
Step 4: If $c_1 - c_2 > \lambda * c_{diff}$, go to Step 5, otherwise, stop the PSO
Step 5: Set $c_3 = (c_1 + c_2)/2$, and calculate the TCF_3
Step 6: Update the Pareto solution set and return to Step 2

In the “Set extension”, the following points should be noted: (1) The “Set extension” is added to expand the search neighborhood and explore better feasible solutions; for example, if $TCF_2 > TCF_3 > TCF_1$, it means that we have found a new feasible solution, and the Pareto set will be updated; (2) λ is a constant value which determines the termination criterion. The smaller the λ is chosen, the newer feasible solutions can be obtained, but it also takes more computational time.

5. Experimental design

5.1 Experimental setting

The benchmark datasets are extracted from Otto *et al.* [27]. According to the number of tasks, the experiments are divided into medium-scale ($n = 100$) and large-scale ($n = 1000$) problems. The values of OPC_a and $SPC_a$of different types of robots can be referred to Nilakantan *et al.* [22]. The number of agents (i.e., robot or human worker) ranges from 6 to 50. This experiment includes 10 instances, each is provided with 5 different values of m. Therefore, there is a total of 50 independent tests, and the corresponding results are presented.

5.2 Parameter setting

To show the superiority of the proposed algorithm, the PSO is compared with the other two algorithms, namely the simulated annealing algorithm (SA), which is extended on the SA proposed by Li *et al.* [21] and the late acceptance hill climbing heuristic algorithm (LAHC) which is extended on the LAHC developed by Yuan *et al.* [28]. All algorithms are coded in MATLAB R2022a and executed on a computer with inter Core i5, 4.7 GHz. Since the meta-heuristic algorithms are stochastic, each algorithm is run ten times for each instance to find the average result, and the algorithm runtime is limited to rt $(rt = n \times n)$ milliseconds. The algorithm's parameters are obtained from the related literature. Table 3 shows the parameters and their values used in the three algorithms.

To evaluate the effectiveness of the three algorithms, three appropriate metrics, namely the ratio of non-dominated solutions (R_p), the non-dominated solution's convergence (C_p), and the spread measure (S_p) have been applied. R_p measures the solution in the Pareto set that is not dominated by other solutions; the higher the value of R_p is, the more effective the algorithm will be. C_p describes the difference between one Pareto-optimal set and the true Pareto-optimal set; the smaller the value of C_p is, the better the algorithm's performance. S_p captures the distributions of the solutions of the Pareto-optimal set; the algorithm with lower S_p performs better. Li *et al.* [29] introduced the formulae for these three metrics.

Table 3 Parameter settings for PSO, LAHC, and SA algorithms

Parameters	PSO	LAHC	SA
The number of task particles in medium (large) instances	100(30)	---	---
The number of agent particles in medium (large) instances	50(30)	---	---
The learning coefficient $l_1(l_2)$	2(2)	---	---
The expansion parameter λ	1.5	---	---
The initial temperature	---	---	100
The cooling rate	---	---	0.9
The length of the cost list	---	100	---

5.3 Results and analysis

The results for the three assessment indicators in medium and large-scale situations are shown in Tables 4 and 5, and the best results are denoted in bold.

The PSO outperforms the other two algorithms in Table 4 for medium-scale instances. For 24 out of 25 instances, PSO finds the best Pareto solutions based on the R_p metric, showing that the PSO algorithm generates more non-dominated solutions in the Pareto solution set than the other two algorithms. The PSO algorithm finds all of the best Pareto solutions based on the C_p metric, demonstrating that the optimal set of the PSO algorithm converges more quickly than those of other algorithms; for 22 out of 25 instances, PSO also generates the best Pareto solutions based on the S_p metric, demonstrating that the spread of Pareto set is superior.

For large-scale problem, as shown in Table 5, PSO provides the best Pareto solutions for 19 out of 25 problems using the R_p metric. The PSO algorithm finds all of the best Pareto solutions based on the C_p metric. For the S_p metric, PSO finds the best Pareto solutions for 23 out of 25 instances.

We also display the performance of the three algorithms on representative examples by using scatter plots. The results are shown in Fig. 3. The first number denotes the problem scale (M for medium problems; L for large problems); the second number denotes the index of instance in Table 4 and Table 5; and the third number denotes the number of stations.

Table 4 Computational results on the Medium-scale problems

Problem	m	SA			LAHC			PSO		
		R_p	C_p	S_p	R_p	C_p	S_p	R_p	C_p	S_p
	6	0.70	0.01	1.09	0.00	0.02	0.78	**0.79**	**0.00**	**0.77**
	8	0.48	0.02	0.94	0.00	0.02	**0.60**	**0.84**	**0.00**	0.68
1	10	0.58	0.01	0.98	0.00	0.01	0.75	**0.79**	**0.00**	**0.59**
	12	0.50	0.01	0.85	0.09	0.02	0.70	**0.75**	**0.00**	**0.57**
	14	0.53	0.01	1.02	0.00	0.01	0.81	**0.86**	**0.00**	**0.73**
	6	0.30	0.01	0.96	0.17	0.01	**0.64**	**0.83**	**0.00**	0.78
	8	0.31	0.00	1.03	0.21	0.01	0.92	**0.88**	**0.00**	**0.64**
2	10	0.64	0.01	0.98	0.00	0.02	0.81	**0.86**	**0.00**	**0.60**
	12	0.57	0.01	0.98	0.06	0.01	0.84	**0.85**	**0.00**	**0.55**
	14	0.37	0.02	0.76	0.14	0.01	0.69	**0.77**	**0.00**	**0.52**
	6	**0.86**	0.01	1.01	0.00	0.02	0.97	0.85	**0.00**	**0.76**
	8	0.23	0.02	0.89	0.03	0.01	1.02	**0.92**	**0.00**	**0.68**
3	10	0.29	0.01	0.78	0.00	0.01	0.92	**0.87**	**0.00**	**0.59**
	12	0.40	0.01	0.85	0.00	0.01	0.70	**0.84**	**0.00**	**0.58**
	14	0.25	0.02	0.65	0.10	0.01	0.58	**0.79**	**0.00**	**0.48**
	6	0.22	0.02	0.76	0.37	0.01	0.73	**0.81**	**0.00**	**0.72**
	8	0.39	0.01	0.90	0.04	0.01	0.69	**0.82**	**0.00**	**0.67**
4	10	0.35	0.02	0.84	0.00	0.01	0.93	**0.84**	**0.00**	**0.56**
	12	0.45	0.01	0.78	0.04	0.01	0.94	**0.78**	**0.00**	**0.74**
	14	0.08	0.01	0.77	0.42	0.00	0.83	**0.79**	**0.00**	**0.54**
	6	0.50	0.01	0.94	0.00	0.01	0.83	**0.84**	**0.00**	**0.77**
	8	0.52	0.01	1.07	0.00	0.01	0.94	**0.83**	**0.00**	**0.67**
5	10	0.45	0.01	0.85	0.00	0.01	0.77	**0.86**	**0.00**	**0.62**
	12	0.45	0.01	0.81	0.03	0.02	**0.69**	**0.73**	**0.00**	0.76
	14	0.26	0.02	0.87	0.00	0.02	0.73	**0.77**	**0.00**	**0.52**

Table 5 Computational results on the Large-scale problems

Problem	m	SA			LAHC			PSO		
		R_p	C_p	S_p	R_p	C_p	S_p	R_p	C_p	S_p
	42	0.66	0.01	0.92	0.33	0.01	0.79	**0.57**	**0.00**	**0.72**
	44	0.69	0.00	0.80	**0.85**	0.00	0.90	0.48	**0.00**	**0.75**
1	46	0.33	0.01	0.75	0.00	0.02	0.83	**0.85**	**0.00**	**0.69**
	48	0.32	0.01	0.69	0.47	0.01	0.76	**0.64**	**0.00**	**0.67**
	50	0.31	0.01	0.65	**0.65**	0.00	0.75	0.29	**0.00**	**0.64**
	42	0.57	0.01	0.81	0.24	0.01	0.83	**0.63**	**0.00**	**0.68**
	44	0.00	0.02	0.80	0.18	0.02	0.83	**0.99**	**0.00**	**0.61**
2	46	0.38	0.01	0.77	0.41	0.01	0.79	**0.59**	**0.00**	**0.52**
	48	0.45	0.01	0.81	0.31	0.02	0.76	**0.77**	**0.00**	**0.72**
	50	0.36	0.01	0.80	0.58	0.01	0.90	**0.64**	**0.01**	**0.73**
	42	0.46	0.01	0.71	0.11	0.01	0.80	**0.71**	**0.00**	**0.66**
	44	0.37	0.01	0.85	0.16	0.01	**0.70**	**0.67**	**0.00**	0.78
3	46	0.53	0.01	0.77	0.27	0.01	0.82	**0.59**	**0.00**	**0.67**
	48	0.48	0.01	0.68	0.31	0.01	0.91	**0.51**	**0.00**	**0.68**
	50	0.41	0.01	0.72	**0.91**	0.00	0.74	0.23	**0.00**	**0.68**
	42	0.35	0.01	0.81	0.00	0.02	0.76	**0.71**	**0.00**	**0.64**
	44	0.24	0.01	0.74	0.39	0.01	0.83	**0.64**	**0.00**	**0.66**
4	46	0.31	0.01	0.31	0.00	0.02	0.85	**0.85**	**0.00**	**0.67**
	48	**0.69**	0.00	0.78	0.41	0.01	0.81	0.34	**0.00**	**0.56**
	50	0.46	0.01	0.76	**0.71**	0.00	0.85	0.53	**0.00**	**0.68**
	42	0.38	0.01	0.80	0.09	0.02	0.84	**0.70**	**0.00**	**0.63**
	44	0.44	0.01	0.82	0.00	0.02	0.81	**0.83**	**0.00**	**0.50**
5	46	0.08	0.01	0.83	0.00	0.02	0.92	**0.99**	**0.00**	**0.56**
	48	0.33	0.01	0.71	0.50	0.02	0.88	**0.59**	**0.00**	**0.67**
	50	0.26	0.01	0.78	0.00	0.02	0.95	**0.92**	**0.00**	**0.64**

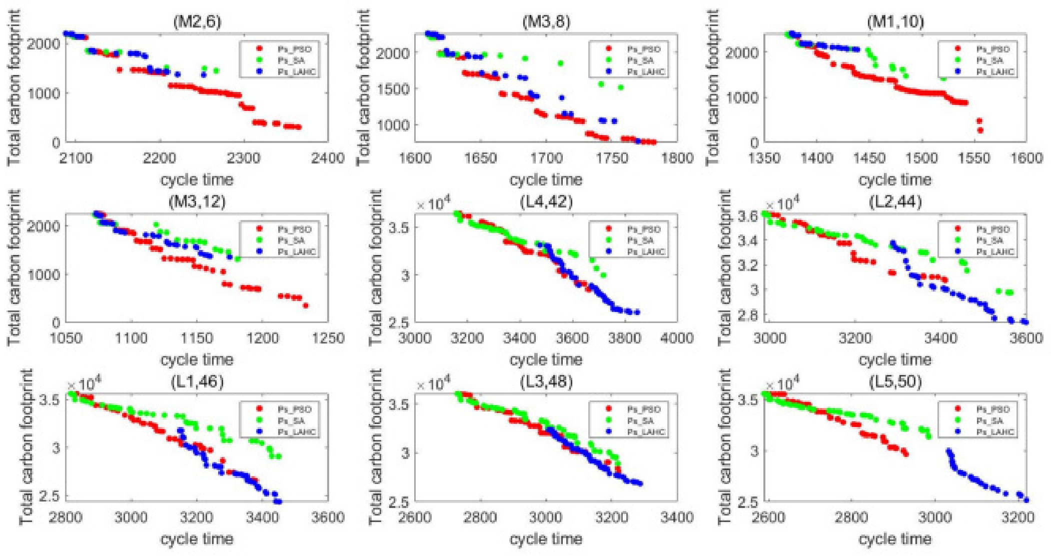


Fig. 3 Some Pareto fronts of different instances

6. Conclusion

In this paper, a multi-objective human-robot collaborative assembly line balancing problem that takes into account production efficiency and carbon emissions is investigated. A mixed-integer programming model is proposed which considers a cross-station task design. This design is intended to improve the assembly line's overall production efficiency and flexibility. A particle swarm optimization algorithm is devised to solve the problem. The performance of the proposed particle swarm optimization algorithm is validated by comparing against simulated annealing and late acceptance hill-climbing algorithms.

The model and algorithm proposed provide some management insights for production in the real world: (1) for assembly line designers, this research can provide a reasonable workstation configurations for HRC production in terms of the goals of production efficiency and carbon emissions; (2) for production managers, the model proposed in this paper can enable them to have a clear understanding of the constitution of carbon emissions in the production process.

In future work, the design for the collaboration of human workers and robots within the same station can be studied. Further, the ALBP with various levels of automation can be examined and contrasted, and management recommendations for the automation transition of small and medium-sized businesses will be offered. Additionally, the research problem can be transformed into a hybrid ALBP-HRC problem by eliminating the assumptions of the single product and the straight AL structure. Finally, the solution methods, such as deep-learning and machine learning algorithms are worth researched.

Acknowledgement

This work was partially supported by the National Natural Science Foundation of China (71901006).

Disclosure and conflicts of interest

No potential conflict of interest was reported by the author(s).

Data availability statements

The data that support the findings of this study are openly available in Wang, Wang xin (2023), "data and result for the “CS-ALBP-HRC”, Mendeley Data, V1, doi: 10.17632/xb5mmtvz6j.1.

Ethics statements

There are no human subjects in this article and informed consent and ethics statement is not applicable.

References

[1] Rubinovitz, J., Bukchin, J., Lenz, E. (1993). RALBP - A heuristic algorithm for design and balancing of robotic assembly lines, *CIRP Annals*, Vol. 42, No. 1, 497-500, doi: 10.1016/S0007-8506(07)62494-9.

[2] Yoosefelahi, A., Aminnayeri, M., Mosadegh, H., Davari Ardakani, H. (2012). Type II robotic assembly line balancing problem: An evolution strategies algorithm for a multi-objective model, *Journal of Manufacturing Systems*, Vol. 31, No. 2, 139-151, doi: 10.1016/j.jmsy.2011.10.002.

[3] Mukund Nilakantan, J., Ponnambalam, S.G. (2016). Robotic u-shaped assembly line balancing using particle swarm optimization, *Engineering Optimization*, Vol. 48, No. 2, 231-252, doi: 10.1080/0305215X.2014.998664.

[4] Wang, J.F., Kang, W.L., Zhao, J.L., Chu, K.Y. (2016). A simulation approach to the process planning problem using a modified particle swarm optimization, *Advances in Production Engineering & Management*, Vol. 11, No. 2, 77-92, doi: 10.14743/apem2016.2.211.

[5] Borba, L., Ritt, M., Miralles, C. (2018). Exact and heuristic methods for solving the robotic assembly line balancing problem, *European Journal of Operational Research*, Vol. 270, No. 1, 146-156, doi: 10.1016/j.ejor.2018.03.011.

[6] Li, Z., Janardhanan, M.N., Tang, Q.H., Ponnambalam, S.G. (2019). Model and metaheuristics for robotic two-sided assembly line balancing problems with setup times, *Swarm and Evolutionary Computation*, Vol. 50, Article No. 100567, doi: 10.1016/j.swevo.2019.100567.

[7] Raatz, A., Blankemeyer, S., Recker, T., Pischke, D., Nyhuis, P. (2020). Task scheduling method for HRC workplaces based on capabilities and execution time assumptions for robots, *CIRP Annals*, Vol. 69, No. 1, 13-16, doi: 10.1016/j.cirp.2020.04.030.

[8] Jiang, L., Duan, J.J., Zheng, R.P., Shen, H.N., Li, H., Xu, J. (2023). Optimization and simulation of garment production line balance based on improved GA, *International Journal of Simulation Modelling*, Vol. 22, No. 2, 303-314, doi: 10.2507/IJSIMM22-2-CO6.

[9] Şahin, M.C., Tural, M.K. (2023). Robotic stochastic assembly line balancing, *Flexible Services and Manufacturing Journal*, Vol. 35, No. 4, 1076-1115, doi: 10.1007/s10696-023-09494-x.

[10] Yu, B., Wu, E., Chen, C., Yang, Y., Yao, B.Z., Lin, Q. (2017). A general approach to optimize disassembly sequence planning based on disassembly network: A case study from automotive industry, *Advances in Production Engineering & Management*, Vol. 12, No. 4, 305-320, doi: 10.14743/apem2017.4.260.

[11] Wang, Y.J., Wang, N.D., Cheng, S.M., Zhang, X.C., Liu, H.Y., Shi, J.L., Ma, Q.Y., Zhou, M.J. (2021). Optimization of disassembly line balancing using an improved multi-objective Genetic Algorithm, *Advances in Production Engineering & Management*, Vol. 16, No. 2, 240-252, doi: 10.14743/apem2021.2.397.

[12] Ding, H., Schipper, M., Matthias, B. (2014). Optimized task distribution for industrial assembly in mixed human-robot environments - Case study on IO module assembly, In: *Proceedings of 2014 IEEE International Conference on Automation Science and Engineering (CASE)*, New Taipei, Taiwan, 19-24, doi: 10.1109/CoASE.2014.6899298.

[13] Nikolakis, N., Kousi, N., Michalos, G., Makris, S. (2018). Dynamic scheduling of shared human-robot manufacturing operations, *Procedia CIRP*, Vol. 72, 9-14, doi: 10.1016/j.procir.2018.04.007.

[14] Dalle Mura, M., Dini, G. (2019). Designing assembly lines with humans and collaborative robots: A genetic approach, *CIRP Annals*, Vol. 68, No. 1, 1-4, doi: 10.1016/j.cirp.2019.04.006.

[15] Zanchettin, A.M., Casalino, A., Piroddi, L., Rocco, P. (2019). Prediction of human activity patterns for human-robot collaborative assembly tasks, *IEEE Transactions on Industrial Informatics*, Vol. 15, No. 7, 3934-3942, doi: 10.1109/TII.2018.2882741.

[16] Vieira, M., Moniz, S., Gonçalves, B.S., Pinto-Varela, T., Barbosa-Póvoa, A.P., Neto, P. (2022). A two-level optimisation-simulation method for production planning and scheduling: The industrial case of a human-robot collaborative assembly line, *International Journal of Production Research*, Vol. 60, No. 9, 2942-2962, doi: 10.1080/00207543.2021.1906461.

[17] Aljinovic, A., Gjeldum, N, Bilic, B, Mladineo, M. (2022). Optimization of industry 4.0 implementation selection process towards enhancement of a manual assembly line, *Energies*, Vol. 15, No. 1, Article No. 30, doi: 10.3390/en15010030.

[18] Riedel, A., Gerlach, J., Dietsch, M., Herbst, S., Engelmann, F., Brehm, N., Pfeifroth, T. (2021). A deep learning-based worker assistance system for error prevention: Case study in a real-world manual assembly, *Advances in Production Engineering & Management*, Vol. 16, No. 4, 393-404, doi: 10.14743/apem2021.4.408.

[19] Nourmohammadi, A., Fathi, M., Ng, A.H.C. (2022). Balancing and scheduling assembly lines with human-robot collaboration tasks, *Computers & Operations Research,* Vol. 140, Article No.105674, doi: 10.1016/j.cor.2021.105674.

[20] Lin, W., Yu, D.Y., Zhang, C., Liu, X., Zhang, S., Tian, Y., Liu, S., Xie, Z. (2015). A multi-objective teaching–learning-based optimization algorithm to scheduling in turning processes for minimizing makespan and carbon footprint, *Journal of Cleaner Production*, Vol. 101, 337-347, doi: 10.1016/j.jclepro.2015.03.099.

[21] Li, Z., Tang, Q., Zhang, L.P. (2016). Minimizing energy consumption and cycle time in two-sided robotic assembly line systems using restarted simulated annealing algorithm, *Journal of Cleaner Production*, Vol. 135, 508-522, doi: 10.1016/j.jclepro.2016.06.131.

[22] Nilakantan, J.M., Li, Z., Tang, Q., Nielsen, P. (2017). Multi-objective co-operative co-evolutionary algorithm for minimizing carbon footprint and maximizing line efficiency in robotic assembly line systems, *Journal of Cleaner Production*, Vol. 156, 124-136, doi: 10.1016/j.jclepro.2017.04.032.

[23] Zhang, B., Xu, L., Zhang, J. (2020). A multi-objective cellular genetic algorithm for energy-oriented balancing and sequencing problem of mixed-model assembly line, *Journal of Cleaner Production*, Vol. 244, Article No. 118845, doi: 10.1016/j.jclepro.2019.118845.

[24] Sun, B.-Q., Wang, L., Peng, Z.-P. (2020). Bound-guided hybrid estimation of distribution algorithm for energy-efficient robotic assembly line balancing, *Computers & Industrial Engineering,* Vol. 146, Article No. 106604, doi: 10.1016/j.cie.2020.106604.

[25] Zhou, B., Wu, Q. (2020). Decomposition-based bi-objective optimization for sustainable robotic assembly line balancing problems, *Journal of Manufacturing Systems*, Vol. 55, 30-43, doi: 10.1016/j.jmsy.2020.02.005.

[26] Kennedy, J., Eberhart, R. (1995). Particle swarm optimization, In: *Proceedings of ICNN'95 - International Conference on Neural Networks,* Perth, Australia,1942-1948, doi: 10.1109/ICNN.1995.488968.

[27] Otto, A., Otto, C., Scholl, A. (2013). Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing, *European Journal of Operational Research*, Vol. 228, No. 1, 33-45, doi: 10.1016/j.ejor.2012.12.029.

[28] Yuan, B., Zhang, C., Shao, X. (2015). A late acceptance hill-climbing algorithm for balancing two-sided assembly lines with multiple constraints, *Journal of Intelligent Manufacturing*, Vol. 26, 159-168, doi: 10.1007/s10845-013-0770-x.

[29] Li, Y., Peng, R., Kucukkoc, I., Tang, X., Wei, F. (2020). System reliability optimization for an assembly line under uncertain random environment, *Computers & Industrial Engineering,* Vol. 146, Article No. 106540, doi: 10.1016/j.cie.2020.106540.