Computers & Industrial Engineering 146 (2020) 106604

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Bound-guided hybrid estimation of distribution algorithm for energy-efficient robotic assembly line balancing

Bin-qi Sun[a], Ling Wang[a,*], Zhi-ping Peng[b,*]

[a] Department of Automation, Tsinghua University, Beijing 100084, China
[b] School of Computer, Guangdong University of Petrochemical Technology, Maoming 525000, China

ARTICLE INFO

Keywords:
Estimation of distribution algorithm
Bound-guided sampling
Energy-efficient robotic assembly line balancing
Non-dominated robot allocation

ABSTRACT

Under the pressure of climate change, energy-efficient manufacturing has attracted much attention. Robotic assembly lines are widely-used in automotive and electronic manufacturing. It is necessary to consider the energy saving and economic criteria simultaneously when robots are utilized to operate assembly tasks replacing human labor. This paper addresses an energy-efficient robotic assembly line balancing (EERALB) problem with the criteria to minimize both the cycle time and total energy consumption. We present a multi-objective mathematical model and propose a bound-guided hybrid estimation of distribution algorithm to solve the problem. When designing the optimization algorithm, we adopt estimation of distribution algorithm (EDA) to tackle the task assignment, and design a non-dominated robot allocation (NGRA) heuristic which is embedded into the EDA to allocate suitable robot to each workstation. Moreover, we propose a bound-guided sampling (BGS) method, which is able to reduce the search space of EDA and focus the search on the promising area. The computational complexity of the proposed algorithm is analyzed and the effectiveness of the proposed NGRA and BGS is tested. In addition, we compare the performances of the proposed mathematical model and the proposed algorithm with those of the existing model and algorithms on a set of widely-used benchmark instances. Comparative results demonstrate the effectiveness of the proposed model and algorithm.

1. Introduction

Climate change is one of the most severe challenges that every human being should face seriously. To mitigate climate change, the United Nations member parties have set a target in the Paris Agreement (United Nations, 2020) to limit average warming to 2 °C above pre-industrial temperature. To achieve such a goal, many governments have launched low-carbon projects. According to the United States Environmental Protection Agency (2017), the largest source of greenhouse gas emissions from human being activities is from burning fossil fuels and about one fifth of greenhouse gas emission directly comes from manufacturing industry. Therefore, carbon emission is limited restrictively for manufacturing industries in many countries. Meanwhile, with the increasing energy costs, manufacturing companies are paying more attention to reduce energy consumption. Thus, it is of extraordinary significance to develop energy-efficient manufacturing technologies when considering both energy saving and economic criteria simultaneously.

Assembly line is a typical manufacturing system especially in automotive and electronic industries. With the development of automation technology, robots have been widely-used in the systems to replace human labor. Such a kind of assembly line is called robotic assembly line. Since robots are capable of working continuously without fatigue and can operate different tasks by using different control programs (Nilakantan, Huang, & Ponnambalam, 2015), robotic assembly lines can effectively increase production efficiency and reduce the work stress of workers. However, on the other hand, robotic assembly lines consume more energies than manual assembly lines due to the utilization of robots. Therefore, it is important to design energy-efficient robotic assembly lines, aiming at increasing the production efficiency without much energy consumption.

Assembly line balancing (ALB) is an important issue when considering the design and management of assembly lines (Nilakantan, Li, Tang, & Nielsen, 2017). The most basic version is called the simple assembly line balancing (SALB) (Scholl & Becker, 2006), which aims to balance the workload among workstations by appropriately determining the task assignment subject to some operational constraints. Robotic assembly line balancing (RALB) is an extension of the SALB, where a robot must be additionally allocated to each workstation and is responsible for performing the tasks (Rubinovitz, Bukchin, & Lenz,

* Corresponding authors.
E-mail addresses: sbq17@mails.tsinghua.edu.cn (B.-q. Sun), wangling@mail.tsinghua.edu.cn (L. Wang), pengzp@gdupt.edu.cn (Z.-p. Peng).

https://doi.org/10.1016/j.cie.2020.106604

Available online 11 June 2020

1993). The RALB has two dependent variables: the number of workstations and the cycle time of the line. It evolves two widely-studied types of the RALB problems (Boysen, Fliedner, & Scholl, 2007): RALB-1 to minimize the workstation number with a given cycle time, and RALB-2 to minimize the cycle time with a given workstation number. However, both objectives are economic-oriented for the above problems. To our knowledge based on literature review, very limited research work considers the reduction of energy consumption produced by the robots.

This paper studies an energy-efficient robotic assembly line balancing (EERALB) problem to minimize both the cycle time and energy consumption simultaneously. To formulate the problem, a multi-objective mathematical model is built. Since the objectives of minimizing cycle time and energy consumption are conflicting (Nilakantan, Ponnambalam, Jawahar, & Kanagaraj, 2015), such two objectives should be considered in sense of multi-objective optimization. During the past decade, the algorithms based on swarm intelligence and evolutionary computation have been applied to solve many multi-objective problems (Lei, 2008; Tian, Hao, & Gen, 2019; Zhang et al., 2019), because the population-based nature allows to generate a set of Pareto optimal solutions in a single run (Coello, Lamont, & Van Veldhuizen, 2007). As a population-based evolutionary algorithm, estimation of distribution algorithm (EDA) (Larrañaga & Lozano, 2001) has been successfully applied to solve many optimization problems including some complex combinatorial problems (Wang, Wang, Xu, Zhou, & Liu, 2012; Wu & Wang, 2018; Wang & Chen, 2019). Different from classical evolutionary algorithms, it uses probability estimation and stochastic sampling techniques instead of the search operators like crossover and mutation to learn the distribution of solutions and produce new promising solutions. To solve the EERALB efficiently in this paper, a multi-objective EDA is developed to handle the task assignment of the EERALB, and an efficient non-dominated robot allocation heuristic is designed to enhance the search capability, and a bound-guided sampling method is designed to reduce the search space. With the above special design, a bound-guided hybrid estimation of distribution algorithm (BHEDA) is proposed to solve the EERALB such a large-scale multi-objective integrated optimization problem efficiently.

The remaining of the paper is organized as follows. Section 2 presents a literature review about the related research. Section 3 presents the problem description and mathematical model. Section 4 introduce the problem-specific BHEDA. Section 5 reports the computational results with comparisons to the existing methods. Finally, we end the paper with some conclusions and further work in Section 6.

2. Literature review

In this section, we will review the related literature especially the solution algorithms with two aspects: robotic assembly line balancing and energy-efficient assembly line balancing.

(1) Robotic assembly line balancing

Since the RALB was presented by Rubinovitz et al. (1993), a growing number of mathematical models and optimization algorithms have been developed. The algorithms can be divided into three classes: exact methods, heuristics, and metaheuristics.

For exact methods, Rubinovitz et al. (1993) proposed a frontier-search based branch-and-bound (B&B) algorithm for the RALB to minimize the workstation number with a given cycle time. Although some heuristic rules were incorporated into the algorithm, both the storage space and computation time of the B&B based method are still too large to solve large scale problems. Kim and Park (1995) developed an integer programming model for the RALB and applied the cutting plane algorithm to solve the problem. However, the assignment of robots with different capabilities was not considered in the model. Bukchin and Tzur (2000) proposed a B&B to solve a flexible assembly line designing problem for minimizing total equipment costs with a given cycle time. Different from Rubinovitz et al. (1993), some new dominance rules and lower bounds were developed to help the algorithm solve some large scale instances.

For heuristic methods, Tsai and Yao (1993) developed a line-balance-based capacity planning procedure for serial-type robotic lines. It determines the type and number of robots required in each workstation to minimize the deviation of the workstation output rate. However, the proposed planning procedure was tested on only 10 instances with small scales. Recently, Borba, Ritt, and Miralles (2018) proposed a branch-bound-and-remember (BBR) algorithm with problem-specific dominance rules for the RALB. By limiting the running time of each iteration, the BBR is transformed into an iterative beam search to solve large-scale problems in reasonable time. Besides, they generated a new dataset to explore the different characteristics of the problem and to compare their methods with the state-of-art. However, the proposed methods highly depend on problem-specific characteristics so that they are hard to be generalized to solve other related problems.

For metaheuristics, Levitin, Rubinovitz, and Shnits (2006) proposed a genetic algorithm (GA) for the RALB to minimize the cycle time with given workstation number. Two different decoding procedures were developed for task and robot assignments, including a recursive procedure and a consecutive procedure. Gao, Sun, Wang, and Gen (2009) proposed a hybrid GA for RALB-2. However, the problem assumptions in such two papers are different. In (Levitin et al., 2006) one type of robot can be assigned to different workstations, while in (Gao et al., 2009) each robot can only be assigned to one workstation. Under such an assumption, the problem presented in (Gao et al., 2009) is equivalent to the assembly line worker assignment and balancing (ALWAB) problem that has been widely studied in literature (Blum & Miralles, 2011; Mutlu, Polat, & Supciller, 2013; Vila & Pereira, 2014). Later, a particle swarm optimization (PSO) and a hybrid of cuckoo search and PSO (CS-PSO) were proposed by Nilakantan et al. (2015). These algorithms used the same consecutive procedure as Levitin et al. (2006) to determine the task and robot assignment. In addition to the research of one-sided straight assembly line, a discrete cuckoo search algorithm and a migrating birds optimization were proposed by Li, Dey, Ashour, and Tang (2018) and Li, Janardhanan, Ashour, and Dey (2019) to solve the two-sided and the U-type RALB, respectively.

(2) Energy-efficient assembly line balancing

Facing the serious climate change and environmental issues, the concept of energy-efficient manufacturing (Chen, Wang, & Peng, 2019; Jiang & Wang, 2019; Wang & Wang, 2020) draws more and more attention in recent years. However, the research of energy-efficient assembly line balancing is still limited.

Fysikopoulos, Anagnostakis, Salonitis, and Chryssolouris (2012) presented an empirical study of the energy consumption in an automotive assembly line. They showed that modeling an assembly line with energy considerations can possibly save energy and cost. Nilakantan et al. (2015) investigated the energy consumption in robotic assembly lines. They proposed a time based model and an energy based model to minimize the cycle time and energy consumption. The presented models are quadratic mixed integer programs and the two objectives are not considered simultaneously. So, they adopted particle swam optimization (PSO) to solve the problem. Li, Tang, and Zhang (2016) considered a multi-objective two-sided robotic assembly line balancing and proposed a restarted simulated annealing algorithm to solve the problem. Nilakantan et al. (2017) studied a multi-objective robotic assembly line balancing problem to minimize the total carbon footprint and maximize the line efficiency simultaneously. They proposed a multi-objective co-operative co-evolutionary algorithm and compared their algorithm with three other metaheuristics. Most recently, Zhang, Tang, Li, and Zhang (2019) expanded the research to the U-shape robotic assembly line balancing and proposed a Pareto

artificial bee colony algorithm to minimize the energy consumption and cycle time.

Note that, the assumptions of robot allocation constraints are different in the above studies. Only in Nilakantan et al. (2015), the same type of robot can be assigned to multiple workstations, whereas the problem studied in other papers are equivalent to the ALWAB.

From the literature review, it can be seen that although the RALB and energy-efficient manufacturing have gained some research, the study on the EERALB is still scant. In this paper, we will develop a new multi-objective mathematical model for the EERALB and propose an effective optimization algorithm to minimize the cycle time and energy consumption simultaneously. To the best of our knowledge, Nilakantan et al. (2015) is the only paper that considers energy consumption for the RALB with the same assumption as this research. So, this research enriches the research of the EERALB in terms of modeling and optimization algorithm. Especially, this research will show that the special designed estimation of distribution algorithm with the bound-guided sampling is able to solve the problem more effectively than the existing algorithms based on swarm intelligence and evolutionary computation.

3. Mathematical formulation for EERALB

In an assembly line, components are transferred by a conveyer belt and move from workstation to workstation to be assembled into final products (Scholl, 1999). When a component passes through a workstation, pre-assigned tasks will be performed by a pre-allocated robot. The duration of a component passing through each workstation referred as cycle time is equal (Becker & Scholl, 2006). In each workstation, the total task operating time is called workstation time, which needs to be smaller than the cycle time. The energy consumption of each workstation includes two parts: the operating energy consumption produced by the robot operating tasks, and the stand-by energy consumption produced by the robot standing by during the idle time (Nilakantan et al., 2015). According to the technical requirement, all tasks need to be executed in a specified order given by the precedence constraints. The precedence constraints can be illustrated as a directed acyclic graph (DAG), where the nodes represent the tasks and the arcs represent the precedence relationships between the tasks. An example with 9 tasks and 10 precedence relations is shown in Fig. 1.

The objectives of the EERALB are to minimize the cycle time and energy consumption simultaneously by determining the task assignment and robot allocation to each workstation. Similar to (Nilakantan et al., 2015), some basic assumptions are given as follows.

- A single type of product is assembled on a straight assembly line.
- The assembly line is paced, i.e. the cycle time of all workstations equals to the same value.
- Task time is deterministic and depends on the type of operating robot.
- A task can only be operated in one workstation, and all tasks should be performed without violating the precedence constraints.
- Only one robot can be assigned to a workstation, and any task can be operated by any robot.
- Different workstations can use the same type of robot, and there is no number limit for each type of robot.
- The setup time and tool changing time are included in the task time.
- Only the energy consumed by robots is considered. Total energy consumption includes operating energy consumption and stand-by energy consumption.

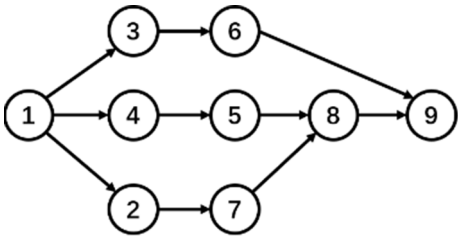


Fig. 1. The illustration of precedence constraints.

Different from Nilakantan et al. (2015), we present a multi-objective mixed 0–1 integer model for the EERALB. The advantage of our model lies in twofold: (1) It can avoid quadratic objective functions by introducing several additional linear constraints to the original model developed by Nilakantan et al. (2015). (2) Our model is a multi-objective model. Using multi-objective optimization algorithms, it can provide a set of Pareto optimal solutions instead of only two single-objective optimal solutions in Nilakantan et al. (2015). The notations used in the model are listed as follows.

Parameters:

N: number of tasks.
i, j: indexes of tasks, $i, j = 1, 2, \cdots, N$.
M: number of workstations.
k, l: index of workstations, $k, l = 1, 2, \cdots, M$.
R: number of robot types.
r: index of robot types, $r = 1, 2, \cdots, R$.
$t_{i,r}$: processing time of task i by robot r.
p_r^o: operating power of robot r.
p_r^s: stand-by power of robot r.
F_i: the set of followers of task i.

Decision variables:

$x_{i,k}$: 1, if task i is assigned to station k; 0, otherwise.
$y_{r,k}$: 1, if robot r is assigned to station k; 0, otherwise.
CT: cycle time.
EC: total energy consumption.
EC_k: energy consumption of workstation k.

$$\min.\ CT \tag{1}$$

$$\min.\ EC \tag{2}$$

s.t.

$$\sum_{i=1}^{N} (t_{i,r} \cdot x_{i,k}) \le CT + M_r^t \cdot (1 - y_{r,k}),\ \forall\ k, r \tag{3}$$

$$\sum_{k=1}^{M} x_{i,k} = 1,\ \forall\ i \tag{4}$$

$$\sum_{r=1}^{R} y_{r,k} = 1,\ \forall\ k \tag{5}$$

$$\sum_{l=1}^{k} x_{i,l} \ge \sum_{l=1}^{k} x_{j,l},\ j \in F_i,\ \forall\ i, k \tag{6}$$

$$EC = \sum_{k=1}^{M} EC_k \tag{7}$$

$$p_r^o \cdot \sum_{i=1}^{N} (t_{i,r} \cdot x_{i,k}) + p_r^s \cdot \left(CT - \sum_{i=1}^{N} (t_{i,r} \cdot x_{i,k}) \right) \le EC_k + M_r^e \cdot (1 - y_{r,k}),\ \forall\ k, r \tag{8}$$

$$x_{i,k} \in \{0, 1\},\ \forall\ i, k \tag{9}$$

$$y_{r,k} \in \{0, 1\},\ \forall\ r, k \tag{10}$$

where (1) is to minimize the cycle time related to constraint (3). The left side of constraint (3) calculates the workstation time operated by robot r in workstation k, and the right side ensures the cycle time is

bigger than each workstation time. Since the right side needs to be free to assume any value when $y_{r,k} = 0$ in constraints (3), we assume that $M_r^t \geq \sum_{i=1}^{N} t_{i,r}$. Objective (2) is to minimize the total energy consumption related to constraints (7) and (8). In (7), the total energy consumption is calculated as the sum of energy consumption among all workstations. In (8), the first and second terms in the left side calculate the operating and stand-by energy consumptions, respectively. Since the right side of constraints (8) needs to be free to assume any value when $y_{r,k} = 0$, we assume that $M_r^e \geq p_r^o \cdot \sum_{i=1}^{N} t_{i,r}$. Constraints (4) and (5) imply that each task should be assigned to one single workstation and each workstation should be assigned a robot, respectively. Constraint (6) ensures that the precedence relationships cannot be violated. Constraints (9) and (10) indicate the type of the decision variables.

4. BHEDA for EERALB

In this section, we first present the overview with a flowchart of the proposed BHEDA for the EERALB, and then introduce the main components of the algorithm in details, and finally analyze the computational complexity of the algorithm.

4.1. Flowchart of BHEDA

To solve the EERALB with the criteria of minimizing both cycle time and energy consumption, we develop a problem-specific multi-objective population-based evolutionary algorithm BHEDA, which fuses the EDA with a bound-guided sampling (BGS) method and a non-dominated robot allocation (NDRA) heuristic. When applying EDA, a probability model (PM) for sampling solutions should be built and an archive set (AS) should be used to record the explored non-dominated solutions. Fig. 2 illustrates the flowchart of the proposed BHEDA with the main procedures described as follows.

(1) Firstly, a PM is initialized uniformly and a AS is initialized using heuristic rules.
(2) In each generation, two populations are generated by sampling the PM, including a CT-oriented population and an EC-oriented population. Each population includes PS individuals, which are encoded to represent the task assignment in each workstation. In particular, different bounds are used in the sampling process to guide the sampling process so as to guarantee the quality of individuals in different population. Then, a non-dominated robot allocation (NDRA) heuristic is performed on each individual to allocate suitable robot to each workstation based on its task assignment. Finally, non-dominated solutions are selected as elite solutions to update the PM and AS.
(3) Repeat the above search until the stopping criterion is met, then output all solutions in the AS.

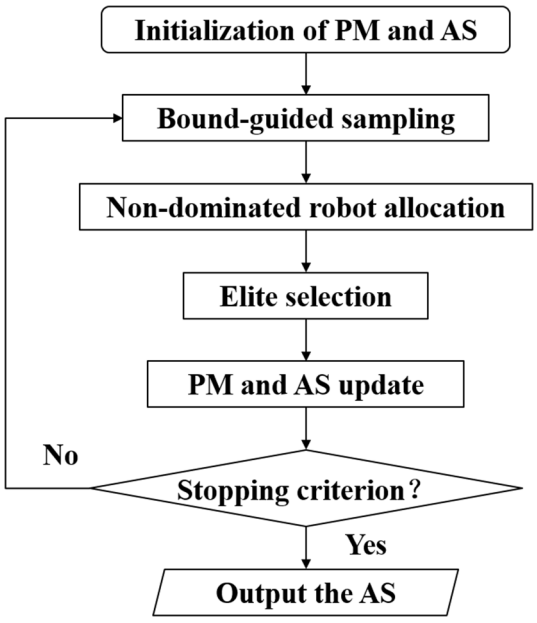


Fig. 2. Flowchart of BHEDA.

4.2. Probability model

Probability model plays an important role in the design of the EDA since it describes the distribution of the solutions in the search space. Meanwhile, the model is the base to be used for generating new solutions. A good probability model should be able to well reflect the characteristics of the problem and easy to implement.

To tackle the task assignment by using EDA for the EERALB, the probability model is designed as a matrix with M lines and N rows. Each element represents the probability of each task being assigned to each workstation. Let $P(g)$ denote the following structure of the probability model.

$$P(g) = \begin{bmatrix} \rho_{11}(g) & \rho_{12}(g) & \cdots & \rho_{1N}(g) \\ \rho_{21}(g) & \rho_{22}(g) & \cdots & \rho_{2N}(g) \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{M1}(g) & \rho_{M2}(g) & \cdots & \rho_{MN}(g) \end{bmatrix}$$

where element $\rho_{ki}(g)$ denotes the probability that task i is assigned to workstation k in the g-th generation. Since a task should be assigned to one workstation, it is clear that $\sum_{k=1}^{M} \rho_{ki}(g) = 1, \forall\, i, g$.

4.3. Initialization of PM and AS

The initialization of the BHEDA includes two parts: the initialization of the probability model, and the initialization of the archive set.

For the probability, since there is no prior knowledge of the elite solution distribution at the beginning, the probability model is initialized uniformly, i.e. $\rho_{ki}(g) = \frac{1}{M}, \forall\, k, i, g$.

For the archive set, we develop two constructive methods to generate two initial solutions. The main motivation is to generate good initial solutions and provide good upper bounds for the two optimization objectives.

The first method is a priority-based heuristic including the following two steps: The first step is to generate a task sequence by sorting the tasks in descending order according to their task time, and the second step is to decode the task sequence into a feasible solution using the consecutive procedure (Levitin et al., 2006). The second method is to construct a solution by assigning all tasks into one single workstation and the robot with the smallest energy consumption.

After generating the two initial solutions, their objective values are calculated and the initial upper bounds are determined, and then they are stored in the AS.

4.4. Bound-guided sampling

The EDA generates a new population via sampling the probability model. Because the EERALB is a bi-objective optimization problem, two populations including a CT-oriented population and an EC-oriented population with respect to different objectives are generated in each generation of the BHEDA. To enhance the sampling efficiency, we propose a bound-guided sampling method to generate the two populations. The bounds related to CT and EC are used to guide the sampling process of the CT-oriented and EC-oriented populations, respectively.

The bound-guided sampling process includes the following M stages with the procedures described as Algorithm 1.

Algorithm 1 (*Procedures of bound-guided sampling*).

for $n = 1$ **to** PS **do**
  Generate a partial individual $I_1(n)$ by assigning tasks to the first workstation using Algorithm 2
end for
for $m = 1$ **to** $M - 2$ **do**
  Expand partial individuals I_m to I_{m+1} using the branching operator in Algorithm 3
end for
for $n = 1$ **to** PS **do**
  Generate a complete individual $I_M(n)$ by assigning all unassigned tasks to the last workstation
end for

In Stage 0, a population is generated consisting of PS partial individuals. Each of them contains only the task assignment of the first workstation. The task assignment procedure is given in Algorithm 2. Then, the lower bound of each partial individual is calculated and the partial individuals are sorted in ascending order according to their lower bounds. The second half of partial individuals with bigger lower bounds are deleted from the population and remaining $PS/2$ partial individuals are selected to enter the next stage (*selecting* operator).

In Stage 1, each partial individual is expanded into two branches by assigning tasks to the second workstation (*branching* operator). In such a way, PS new partial individuals are generated, and each partial individual contains the task assignments of the first two workstations. Then, the *selecting* and *branching* operators are performed iteratively until Stage $M - 2$. Specific procedures of *selecting* and *branching* operator are given in Algorithm 3.

In Stage $M - 1$, the *selecting* and *branching* operator are not performed to avoid duplicate individuals. All the partial individuals remain to this stage and PS complete individuals are generated by assigning all the unassigned tasks to the last workstation.

In Algorithm 2, the task assignment of workstation m is generated by selecting tasks from the *task list* using the Roulette Wheel Method (Lipowski & Lipowska, 2012). The *task list* records the tasks without unassigned predecessors, and the proportions in the Roulette Wheel Method are taken from the PM. Moreover, the workstation objective value is defined as the workstation time and workstation energy consumption for CT-oriented population and EC-oriented population, respectively. The workstation time and energy are calculated under the assumption that the best-fit robot is used. The upper bound of workstation time is calculated as CT_b, where CT_b is the best cycle time obtained so far. The upper bound of workstation energy consumption is calculated as EC_b/M, where EC_b is the best-so-far total energy consumption.

Algorithm 2 (*Procedure of assigning tasks to workstation m* $(1 \le m \le M - 1)$).

Input: the probability model P, the upper bound UB, the *task list* recording the tasks which can be assigned to workstation m without violating precedence constraints
Initialize an empty workstation W_m; initialize the workstation objective value $WO = 0$
while $WO < UB$ **do**
  Select a task s using roulette wheel method: $s \leftarrow RouletteWheelMethod(tasklist, P)$
  Calculate the workstation objective value WO after assigning task s
  if $WO < UB$ **do**
    Add task s to W_m: $s \rightarrow W_m$
    Update the *task list*
  end if
end while
Output: the assignment of workstation W_m $(1 \le m \le M - 1)$

Algorithm 3 (*Procedures of branching partial individuals from stage m to* $m + 1$).

Input: PS partial individuals in stage m: $\boldsymbol{I_m}$, $1 \le m \le M - 2$
Calculate the partial lower bound of each partial individual
Rank the partial individuals $\boldsymbol{I_m}$ in ascending order according to their partial lower bounds
Delete the last $PS/2$ individuals from $\boldsymbol{I_m}$
for $n = 1$ **to** $PS/2$ **do**
  for $l = 1$ **to** 2 **do**
    Expand the nth partial individual $\boldsymbol{I_m}(n)$ to $\boldsymbol{I_{m+1}}(2{\cdot}n - 2 + l)$ by assigning tasks to
    workstation $m + 1$ using Algorithm 1
  end for
  if the task assignment in $\boldsymbol{I_{m+1}}(2{\cdot}n - 1)$ is the same with $\boldsymbol{I_{m+1}}(2{\cdot}n)$ **do**
    Find the last task p assigned to workstation $m + 1$ in $\boldsymbol{I_{m+1}}(2{\cdot}n)$
    if task p can be replaced by other unassigned task q **do**
      Replace task p with task q; update the task list
    end if
    else do
      Delete task p from $\boldsymbol{I_{m+1}}(2{\cdot}n)$; update the task list
    end else
  end if
end for
Output: PS partial individuals in stage $m + 1$: $\boldsymbol{I_{m+1}}$, $1 \le m \le M - 2$

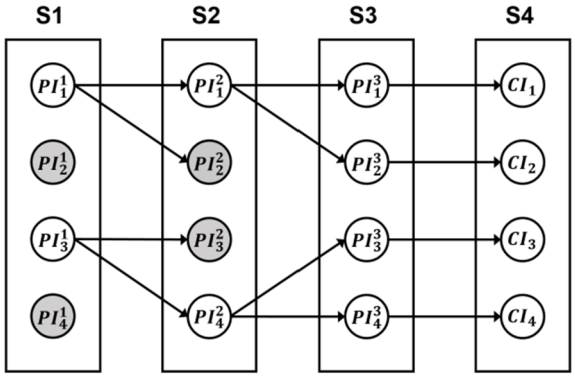
In Algorithm 3, the partial lower bounds of cycle time and energy consumption are used in the sampling of the CT-oriented population and the EC-oriented population, which are defined by (11) and (12), respectively.

$$LB_{CT} = \sum_{k=1}^{m} \min_{1 \le r \le R} \sum_{i \in A_k} t_{i,r} + \sum_{i \in U} \min_{1 \le r \le R} t_{i,r} \tag{11}$$

$$LB_{EC} = \sum_{k=1}^{m} \min_{1 \le r \le R} \sum_{i \in A_k} t_{i,r} \cdot p_r^o + \sum_{i \in U} \min_{1 \le r \le R} t_{i,r} \cdot p_r^o \tag{12}$$

where A_k denotes the set of tasks assigned in workstation k and U denotes the set of unassigned tasks. The partial lower bound LB_{CT} is introduced by Borba et al. (2018). The partial lower bound LB_{EC} is developed by replacing the task operating time used in (11) with the task operating energy consumption.

For easy understanding, an example of the whole bound-guided sampling process is illustrated in Fig. 3. Suppose there are 4 stages and the population size of each stage is 4. Let PI_n^m denote the nth partial solution in stage m. Let CI_n denote the nth complete solution in the last stage. The grey nodes represent the partial individuals which are removed in the sampling process and the white ones represent the partial



In S1: $LB(PI_3^1) < LB(PI_1^1) < LB(PI_2^1) < LB(PI_4^1)$

In S2: $LB(PI_1^1) < LB(PI_4^1) < LB(PI_3^1) < LB(PI_2^1)$

Fig. 3. The illustration of bound-guided sampling process.

Table 1
Example of workstation time.

Robot	Workstation		
	1	2	3
1	27	34	29
2	24	32	28
3	33	31	30

individuals to be expanded in the next stage.

From the figure, it can be seen that the promising partial individuals with better lower bounds are expanded to generate more branches while the inferior ones are removed to avoid invalid search in the following stages. In this way, the search space of the sampling can be reduced and the search can be focused on the promising area so as to generate the individuals with better quality.

4.5. Non-dominated robot allocation

Since the individuals in the EDA only contain the assignments of tasks, it needs to develop a method to allocate suitable robot to each workstation. For each individual, different robot allocation schemes may result in different cycle time and energy consumption. Some robot allocation schemes dominate others if their cycle time and energy consumption are both smaller. The greedy principle, that is, always allocating the robot which performs the tasks in the least time, does not yield the non-dominated solution because it is possible to further reduce the energy consumption under the same cycle time. Moreover, it is very time-consuming to explore all the robot allocation schemes exhaustively. Therefore, we propose a heuristic method to obtain all the non-dominated robot allocation schemes for each individual without examining all the combinations.

The main idea of the proposed non-dominated robot allocation (NDRA) method is to first figure out all the possible cycle times among different robot allocation schemes and then consider only the most energy-efficient robot allocation under each possible cycle time. For an individual containing task assignment $\boldsymbol{\alpha}$, the procedure of generating non-dominated robot allocation scheme set $\boldsymbol{\beta}$ using the NDRA is as follows.

Step 1. For each workstation m, calculate the workstation time WT_m^r under each robot r. Calculate the minimum cycle time $CT_{min} = \max_m \min_r WT_m^r$. Initialize the non-dominated robot allocation scheme set $\boldsymbol{\beta} = \varnothing$. Initialize a robot allocation scheme $\boldsymbol{r}' = \{\arg\min_r WT_1^r, \arg\min_r WT_2^r, \cdots, \arg\min_r WT_M^r\}$.

Step 2. Sort the workstation time $WT_m^r, \forall\, r, m$ in the ascending order: $WT_{m_{(1)}}^{r_{(1)}}, WT_{m_{(2)}}^{r_{(2)}}, \ldots, WT_{m_{(R\cdot M)}}^{r_{(R\cdot M)}}$, where $r_{(q)}$ and $m_{(q)}$ denote the robot and workstation index of the qth smallest workstation time, respectively.

Step 3. Initialize $\hat{q} = \min\{q | WT_{m_{(q)}}^{r_{(q)}} = CT_{min}\}$.

Step 4. Set the cycle time $CT = WT_{m_{(\hat{q})}}^{r_{(\hat{q})}}$, and allocate robot $r_{m_{(\hat{q})}} = r_{(\hat{q})}$ to workstation $m_{(\hat{q})}$.

Step 5. For workstation $m \neq m_{(\hat{q})}$, calculate workstation energy consumption $WEC_m^r = WT_m^r \cdot p_r^o + (CT - WT_m^r) \cdot p_r^s, \forall\, r$. Allocate robot $r_m = \arg\min_r \{WEC_{m'}^r | WT_{m'}^r \leq CT\}$ to workstation m.

Step 6. Calculate the total energy consumption $EC = \sum_{m=1}^{M} WEC_m^{r_m}$.

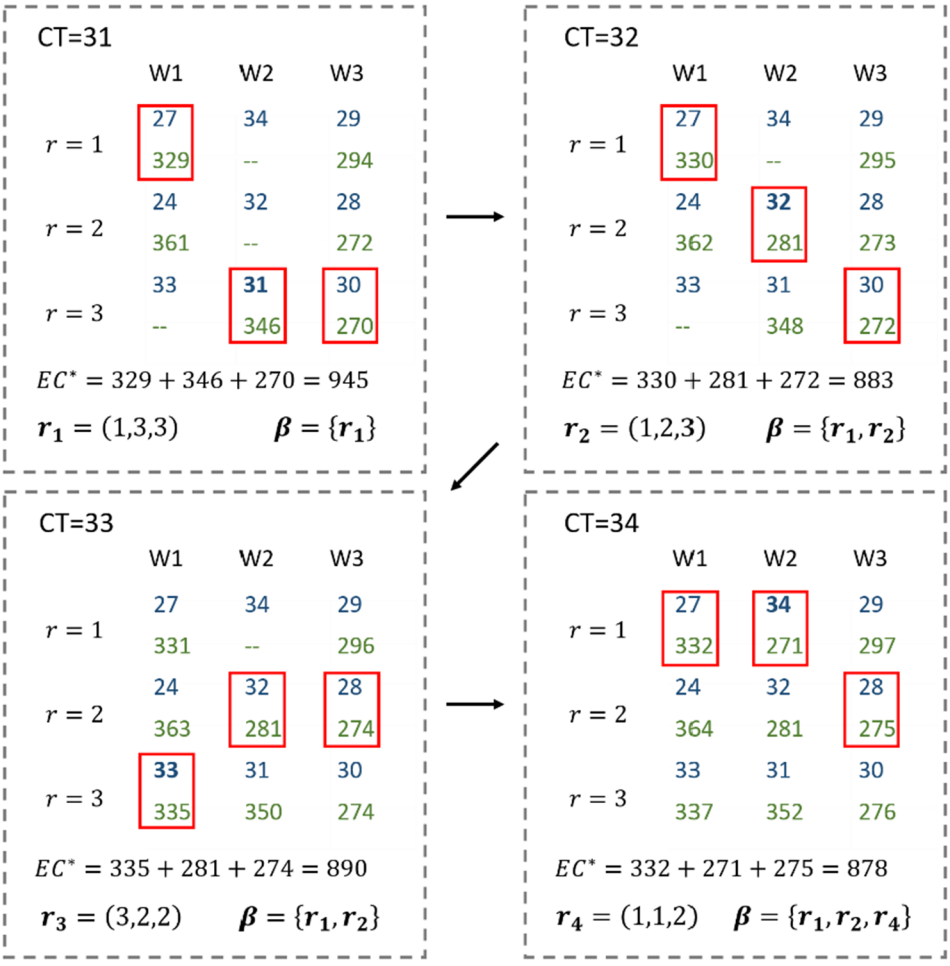


Fig. 4. Example of non-dominated robot allocation procedure.

Table 2
Comparisons between M1 and M2.

N	*M*	CT				EC			
		RPD (%)		Time (s)		RPD (%)		Time (s)	
		M1	M2	M1	M2	M1	M2	M1	M2
25	3	**0.00**	**0.00**	20.94	**1.36**	8.55	**0.00**	1.97	**1.00**
	4	**0.00**	**0.00**	22.34	**2.65**	9.65	**0.00**	7.67	**3.29**
	6	**0.00**	**0.00**	9.43	**1.85**	41.30	**0.00**	18.38	**17.86**
	9	**0.00**	**0.00**	527.30	**141.06**	13.01	**0.00**	**3600.00**	**3600.00**
	Arv	**0.00**	**0.00**	145.00	**36.73**	**18.13**	**0.00**	**907.00**	**905.54**
35	4	**0.00**	**0.00**	24.27	**2.92**	2.16	**0.00**	14.55	**7.75**
	5	**0.00**	**0.00**	98.78	**6.30**	2.59	**0.00**	23.27	**17.08**
	7	0.50	**0.00**	452.99	**130.57**	17.52	**0.00**	607.63	**529.33**
	12	2.11	**0.00**	**3600.00**	**3600.00**	11.23	**0.00**	**3600.00**	**3600.00**
	Arv	0.65	**0.00**	1044.01	**934.95**	8.38	**0.00**	1061.36	**1038.54**
53	5	**0.00**	**0.00**	167.52	**5.05**	39.06	**0.00**	28.45	**17.82**
	7	0.35	**0.00**	1610.93	**46.72**	28.92	**0.62**	**3600.00**	**3600.00**
	10	**0.00**	**0.00**	**3600.00**	**3600.00**	14.43	**0.00**	**3600.00**	**3600.00**
	14	4.48	**0.75**	**3600.00**	**3600.00**	19.50	**2.28**	**3600.00**	**3600.00**
	Arv	1.21	**0.19**	2244.61	**1812.94**	25.48	**0.72**	2707.11	**2704.45**
70	7	0.77	**0.00**	**3600.00**	**3600.00**	4.20	**0.00**	**3600.00**	**3600.00**
	10	2.12	**3.39**	**3600.00**	**3600.00**	15.69	**0.00**	**3600.00**	**3600.00**
	14	40.46	**4.05**	**3600.00**	**3600.00**	15.28	**0.00**	**3600.00**	**3600.00**
	19	54.47	**13.01**	**3600.00**	**3600.00**	104.89	**0.17**	**3600.00**	**3600.00**
	Arv	24.46	**5.11**	**3600.00**	**3600.00**	35.02	**0.04**	3600.00	3600.00
89	8	0.46	**0.00**	3600.00	**842.03**	38.33	**0.00**	**3600.00**	**3600.00**
	12	31.00	**0.67**	**3600.00**	**3600.00**	30.94	**0.00**	**3600.00**	**3600.00**
	16	53.08	**12.32**	**3600.00**	**3600.00**	53.42	**2.13**	**3600.00**	**3600.00**
	21	482.39	**8.18**	**3600.00**	**3600.00**	122.50	**5.56**	**3600.00**	**3600.00**
	Arv	141.73	**5.29**	3600.00	**2910.51**	61.30	**1.92**	**3600.00**	**3600.00**
111	9	0.84	**0.00**	**3600.00**	**3600.00**	22.15	**0.00**	**3600.00**	**3600.00**
	13	509.96	**9.25**	**3600.00**	**3600.00**	20.21	**0.00**	**3600.00**	**3600.00**
	17	538.71	**9.22**	**3600.00**	**3600.00**	6.47	**0.65**	**3600.00**	**3600.00**
	22	72.67	**14.91**	**3600.00**	**3600.00**	61.09	**2.31**	**3600.00**	**3600.00**
	Arv	280.55	**8.34**	**3600.00**	**3600.00**	27.48	**0.74**	**3600.00**	**3600.00**
148	10	2.75	**0.00**	**3600.00**	**3600.00**	34.27	**0.27**	**3600.00**	**3600.00**
	14	212.85	**9.22**	**3600.00**	**3600.00**	18.15	**0.00**	**3600.00**	**3600.00**
	21	413.91	**16.96**	**3600.00**	**3600.00**	32.65	**7.15**	**3600.00**	**3600.00**
	29	725.79	**16.35**	**3600.00**	**3600.00**	74.59	**2.19**	**3600.00**	**3600.00**
	Arv	338.83	**10.63**	**3600.00**	**3600.00**	39.91	**2.40**	**3600.00**	**3600.00**
297	19	185.06	**7.56**	**3600.00**	**3600.00**	22.27	**1.58**	**3600.00**	**3600.00**
	29	174.20	**8.99**	**3600.00**	**3600.00**	363.72	**17.24**	**3600.00**	**3600.00**
	38	268.60	**38.76**	**3600.00**	**3600.00**	417.42	**31.68**	**3600.00**	**3600.00**
	50	2651.56	**78.13**	**3600.00**	**3600.00**	724.89	**37.54**	**3600.00**	**3600.00**
	Arv	819.86	**33.36**	**3600.00**	**3600.00**	382.08	**22.01**	**3600.00**	**3600.00**
Arv		200.91	**7.87**	2679.20	**2511.89**	74.72	**3.48**	2834.43	**2831.07**

The bold values mean the best results.

Table 3
Parameter values.

Parameter	Factor levels		
	1	2	3
PS	100	500	900
LR	0.0001	0.001	0.01

If solution $(\boldsymbol{\alpha}, \boldsymbol{r}) \prec (\boldsymbol{\alpha}, \boldsymbol{r}')$, go to Step 7; otherwise, $\boldsymbol{\beta} = \boldsymbol{\beta} \cup \boldsymbol{r}$, $\boldsymbol{r}' = \boldsymbol{r}$.

Step 7. $\hat{q} = \hat{q} + 1$. If $\hat{q} \leq R{\cdot}M$, go to Step 4; otherwise, output the non-dominated solutions $(\boldsymbol{\alpha}, \boldsymbol{\beta})$, where $\boldsymbol{\beta}$ is a set of non-dominated robot allocation schemes for task assignment $\boldsymbol{\alpha}$.

For the above NDRA, its computational time complexity is analyzed as follows. In Step 1, the complexity of calculating each workstation time under each robot is $O(R{\cdot}N)$. In Step 2, the complexity of sorting WT_m^r, $\forall\, r, m$ is $O((R{\cdot}M){\cdot}\log(R{\cdot}M))$. In Step 4–7, the complexity of generating all non-dominated robot allocation schemes is $O(R^2{\cdot}M^2)$. So, the computational complexity of the NDRA process is $O(R^2{\cdot}M^2 + R{\cdot}N)$, which is much smaller than $O(R^{M+1}{\cdot}M + R{\cdot}N)$ by exhaustive method.

Next, we use an example to explain the above NDRA. Suppose there are three workstations and three robots. The workstation times under different robot allocation are listed in Table 1.

Table 4
Orthogonal array and RVs.

Experiment number	Factor level		RV
	PS	LR	
1	1	1	0.244
2	1	2	0.253
3	1	3	0.219
4	2	1	0.214
5	2	2	0.295
6	2	3	0.254
7	3	1	0.193
8	3	2	0.260
9	3	3	0.226

First, the minimum cycle time is calculated as 31. Then, the workstation time is sorted and the possible cycle times are derived as 31, 32, 33, and 34. After that, we examine the most energy-efficient robot allocation scheme under each possible cycle time. The procedure is illustrated in Fig. 4, where the workstation time and energy consumption are denoted in blue and green, respectively. Finally, four robot

Table 5
One-way ANOVA.

Source	DF	SS	MS	*F*	*p*
PS	2	0.0035	0.0017	0.51	0.604
Error	24	0.0812	0.0034		
Total	26	0.0847			
LR	2	0.0131	0.0065	2.19	0.134
Error	24	0.0716	0.0030		
Total	26	0.0847			

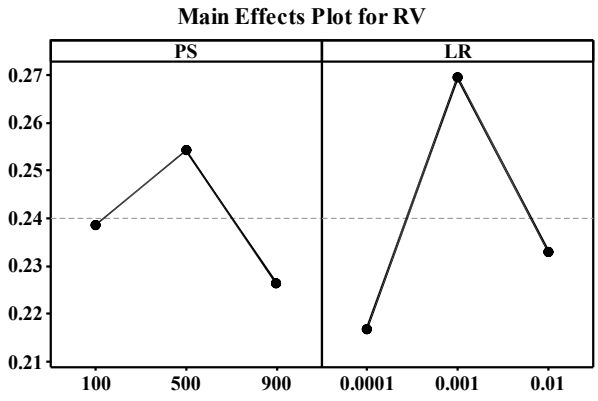


Fig. 5. Trends of parameters.

Table 6
Comparisons of BEDA, HEDA, and BHEDA.

N	*M*	BEDA		HEDA		BHEDA
		CŌN	p-value	*CŌN*	p-value	*CŌN*
25	3	0.58	0.00	0.65	0.00	**1.00**
	4	0.35	0.00	0.73	0.01	**0.94**
	6	0.50	0.00	0.85	0.04	**0.93**
	9	0.32	0.00	0.72	0.44	**0.74**
35	4	0.44	0.00	0.81	0.01	**0.95**
	5	0.03	0.00	0.39	0.00	**0.74**
	7	0.11	0.00	0.13	0.00	**0.79**
	12	0.30	0.00	0.57	0.12	**0.61**
53	5	0.53	0.07	0.57	0.01	**0.67**
	7	0.28	0.00	0.39	0.01	**0.67**
	10	0.50	0.00	0.08	0.00	**0.82**
	14	0.52	0.03	0.02	0.00	**0.81**
70	7	0.53	0.08	0.33	0.01	**0.80**
	10	0.16	0.00	0.52	0.04	**0.73**
	14	0.51	0.04	0.00	0.00	**0.89**
	19	0.44	0.02	0.14	0.00	**0.74**
89	8	0.28	0.00	0.05	0.00	**0.83**
	12	0.22	0.00	0.25	0.00	**0.58**
	16	0.29	0.00	0.13	0.00	**0.66**
	21	0.42	0.02	0.16	0.00	**0.73**
111	9	0.53	0.02	0.05	0.00	**0.63**
	13	0.57	0.02	0.12	0.00	**0.83**
	17	0.54	0.01	0.09	0.00	**0.73**
	22	0.50	0.03	0.07	0.00	**0.75**
148	10	0.52	0.04	0.04	0.00	**0.94**
	14	0.68	0.02	0.00	0.00	**0.92**
	21	0.30	0.00	0.00	0.00	**0.90**
	29	0.70	0.03	0.05	0.00	**0.90**
297	19	0.90	0.02	0.00	0.00	**1.00**
	29	0.64	0.01	0.00	0.00	**0.96**
	38	0.51	0.01	0.00	0.00	**0.79**
	50	0.48	0.03	0.01	0.00	**0.82**

The bold values mean the best results.

allocation schemes $\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3, \boldsymbol{r}_4$ are generated and three of them, i.e., $\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_4$, constitute the non-dominated robot allocation set $\boldsymbol{\beta}$.

4.6. Update of PM and AS

For a single-objective optimization problem, the individuals with better objectives are used to update the probability model. When solving the bi-objective EERALB by the BHEDA, the non-dominated solutions found in each generation are used to update the AS with Pareto ranking criterion.

As for the PM in the BHEDA, it is updated by using the population-based incremental learning (PBIL) method (Baluja, 1994). The basic idea is to increase the probability of using construction steps that are contained in high quality solutions via positive feedback (Grahl, 2008) as follows.

$$\rho_{ki}(g+1) = (1-\alpha)\cdot\rho_{ki}(g) + \alpha\cdot\frac{1}{|NS|}\cdot\sum_{h=1}^{|NS|} I_{k,i}^{h}(g),\ \forall\ i,\ k \tag{13}$$

where $\alpha \in (0, 1)$ represents the learning rate, $|NS|$ is the number of non-dominated solutions, and $I_{k,i}^{h}(g)$ is the following indicator function corresponding to the hth non-dominated solution.

$$I_{k,i}^{h}(g) = \begin{cases} 1, \textit{ if task } i \textit{ is assigned to workstation } k \textit{ in the } h\textit{th} \\ \quad \textit{non-dominated solution} \\ 0, \textit{ otherwise} \end{cases} \tag{14}$$

4.7. Computational time complexity analysis

According to the flowchart in Fig. 2, in each generation the BHEDA needs to execute sampling, robot allocation, and update sequentially. So, its computational time complexity is analyzed as follows.

For sampling, most time are consumed in *assigning* and *branching* procedures as Algorithm 2 and Algorithm 3, respectively. In Algorithm 2, there are at most N tasks assigned to one workstation. Since the Roulette Wheel Method is used in assigning each task, the time complexity of assigning all tasks is $O(N^2)$. Moreover, the time complexity of calculating the workstation objective is $O(N{\cdot}R)$. Thus, the time complexity of generating the assignment for a workstation is $O(N^2 + N{\cdot}R)$. In Algorithm 3, the complexity of ranking PS partial individuals is $O(PS{\cdot}logPS)$, and the complexity of expanding $PS/2$ partial individuals into PS branches for next stage is $O(PS{\cdot}N^2 + PS{\cdot}N{\cdot}R)$. So, the total time complexity of the sampling procedure as Algorithm 1 is $O(M{\cdot}PS{\cdot}\log PS + PS{\cdot}N^2 + PS{\cdot}N{\cdot}R)$.

According to the analysis in Section 4.5, the time complexity of non-dominated robot allocation is $O(R^2{\cdot}M^2 + R{\cdot}N)$. Since it performs on $2 \times PS$ individuals, the time complexity of robot allocation in each generation is $O(PS{\cdot}(R^2{\cdot}M^2 + R{\cdot}N))$. In addition, the complexity of updating the PM is $O(N_p{\cdot}logN_p + N{\cdot}M)$, where N_p is the total solutions generated in each generation; the complexity of updating the AS is $O(N_P^2{\cdot}G)$, where G is the number of total generations. So, the time complexity of updating process is $O(N_p{\cdot}logN_p + N{\cdot}M + N_P^2{\cdot}G)$.

Based on the above analysis, the total computational time complexity of the proposed BHEDA is $O(G{\cdot}(PS{\cdot}(N^2 + N{\cdot}R + R^2{\cdot}M^2 + M{\cdot}\log PS) + M{\cdot}N) + N_p{\cdot}logN_p + N{\cdot}M + N_P^2{\cdot}G)$.

5. Computational results

To evaluate the performance of our proposed mathematical model and the BHEDA, extensive numerical tested are conducted by using benchmarking instances. First, we compare our mathematical model with the existing model proposed by Nilakantan et al. (2015). Then, we investigate the effectiveness of the proposed bound-guided sampling method and non-dominated robot allocation heuristic. Finally, we compare the proposed BHEDA with two algorithms based on swarm

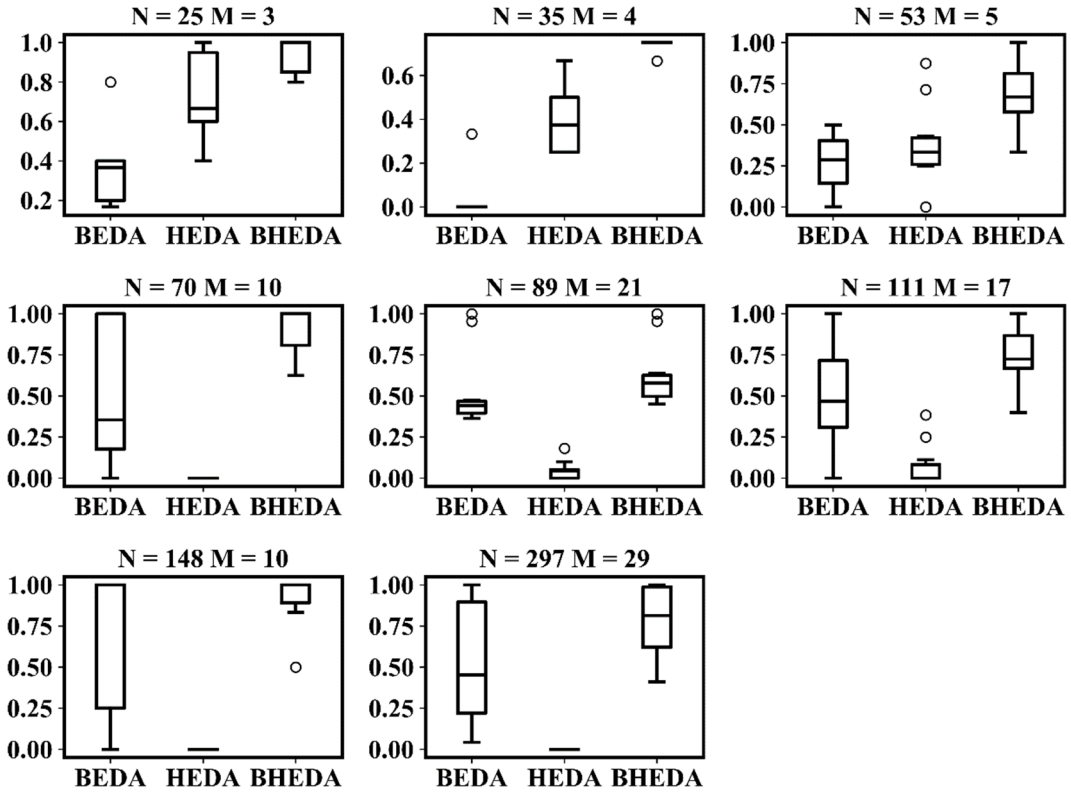


Fig. 6. Boxplots comparisons of BEDA, HEDA, and BHEDA.

Table 7
Comparison between BHEDA and PSO.

N	*M*	PSO		BHEDA			
		RPD_{CT}	RPD_{EC}	RPD_{CT}	RPD_{EC}	|AS|	DN
25	3	**0.00**	2.96	**0.00**	**0.00**	2	1
	4	0.69	2.19	**0.00**	**0.57**	6	**2**
	6	13.92	3.20	**0.00**	**0.15**	3	**2**
	9	0.92	6.90	**0.00**	**0.86**	7	**2**
35	4	**0.00**	8.16	**0.00**	**0.40**	4	**2**
	5	8.51	9.33	**0.00**	**1.77**	4	**2**
	7	12.44	9.98	**0.00**	**1.28**	15	**2**
	12	10.53	13.08	**2.11**	**0.20**	7	**2**
53	5	1.11	6.78	**0.00**	**2.49**	3	1
	7	3.53	3.87	**0.00**	**0.00**	13	**2**
	10	10.34	5.39	**0.00**	**0.18**	16	**2**
	14	8.96	12.42	**0.00**	**0.00**	15	**2**
70	7	14.95	7.18	**4.90**	**3.15**	5	**2**
	10	9.75	7.89	**0.00**	**0.58**	6	**2**
	14	12.14	10.56	**0.00**	**0.20**	12	**2**
	19	13.01	9.42	**0.00**	**0.00**	14	**2**
89	8	7.41	7.68	**2.08**	**1.39**	16	**2**
	12	5.67	7.84	**0.00**	**0.05**	19	**2**
	16	3.79	9.53	**0.00**	**0.00**	10	**2**
	21	10.69	5.90	**0.00**	**0.00**	11	**2**
111	9	10.97	9.09	**2.95**	**3.94**	27	**2**
	13	12.81	11.97	**0.00**	**2.76**	22	**2**
	17	15.21	10.18	**0.00**	**0.00**	7	**2**
	22	14.91	13.03	**0.00**	**0.00**	15	**2**
148	10	2.02	4.62	**0.55**	**0.00**	5	1
	14	17.32	13.85	**0.00**	**1.46**	11	**2**
	21	18.26	10.15	**0.00**	**0.00**	11	**2**
	29	19.50	14.87	**0.00**	**0.00**	9	**2**
297	19	9.59	12.94	**0.00**	**0.00**	30	**2**
	29	24.06	13.78	**0.00**	**0.00**	13	**2**
	38	14.34	6.25	**0.00**	**0.00**	21	**2**
	50	33.33	10.84	**0.00**	**0.00**	22	**2**
Average		10.65	8.81	**0.42**	**0.67**	11.91	1.91

The bold values mean the best results.

intelligence and evolutionary computation. One is the elitist non-dominated sorting genetic algorithm (NSGA-II) (Deb, Pratap, Agarwal, & Meyarivan, 2002), which is the most famous multi-objective optimization algorithm. The other is the PSO proposed by Nilakantan et al. (2015), which was used to solve the problem with the same objectives and problem assumptions among the related literature.

The used dataset is directly taken from Nilakantan et al. (2015), which consists of 32 benchmarking instances with 8 precedence graphs, i.e., Roszieg, Gunther, Hahn, Tonge, Lutz3, Arc111, Barthol2 and Scholl. For more details, please refer to http://www.assembly-linebalancing.de/. The task number in these precedence graphs varies from 25 to 297. For each precedence graph, there are four instances with different numbers of workstations and robots. The time and power data are generated by Gao et al. (2009) and Nilakantan et al. (2015), respectively.

For fair comparison, all the algorithms are coded in C++ and tested on the same computer with an Intel Core i5-6500 CPU @ 3.2-GHz.

5.1. Comparison of mathematical models

First we compare our mathematical model (M2) to the model (M1) given by Nilakantan et al. (2015). Since both M1 and M2 have two objectives, we solve two single-objective problems with CT and EC respectively to test the performance of M1 and M2. The models are solved by Gurobi 8.0 with a time limit of 3600 s. The results are given in Table 2. The relative percentage deviation (RPD) is calculated as $(Obj - Obj^*)/Obj^* \times 100$, where Obj is the objective value found by a certain model and Obj^* is the best objective value among all tests.

For objective CT, it can be seen that 10 instances are solved optimally by M2, compared to 9 by M1. Moreover, M2 costs less time to achieve these optimal solutions than M1. For instances that are not solved optimally, the objective values obtained by M2 are consistently smaller than those by M1 with the same CPU time.

For objective EC, it can be seen that both M1 and M2 solve 7

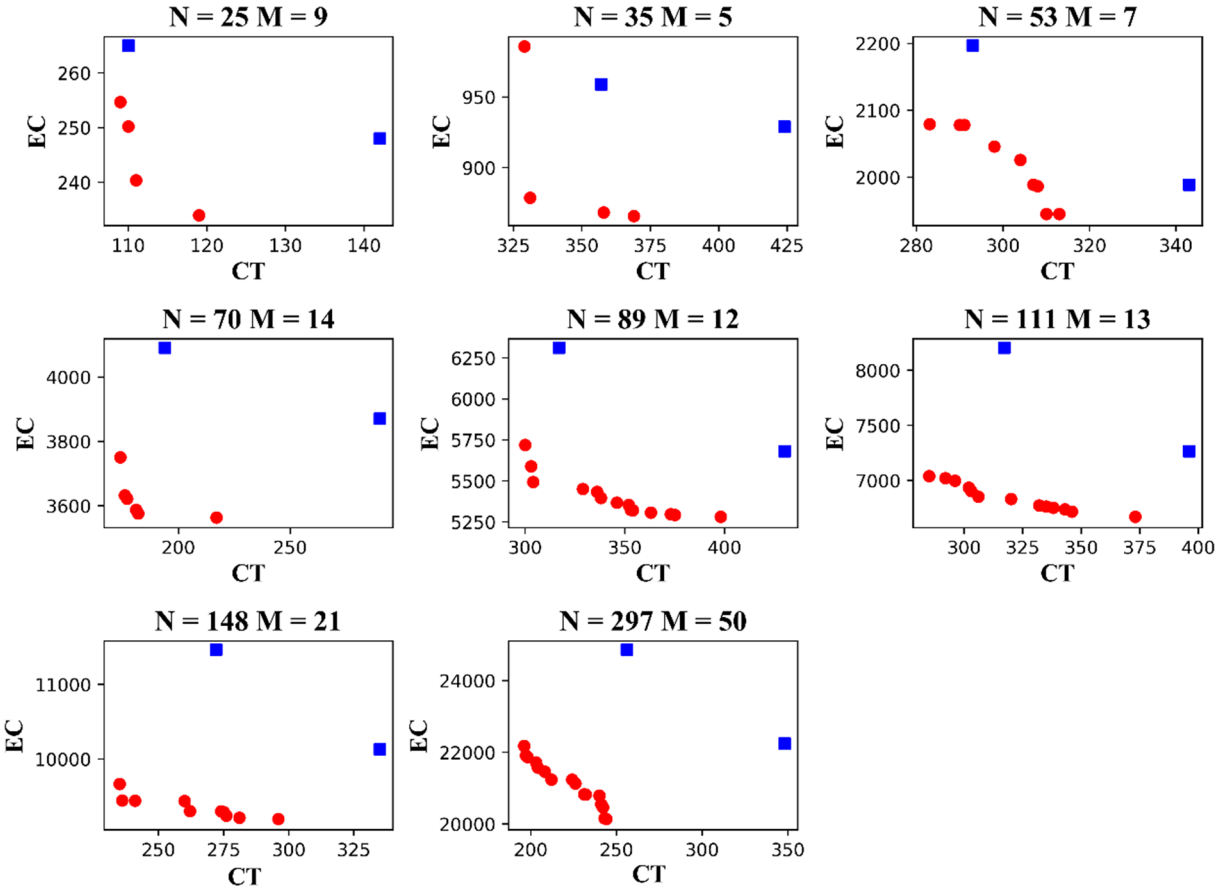


Fig. 7. Pareto fronts obtained by BHEDA and best solutions obtained by PSO.

instances optimally. However, note that the stand-by energy consumption is not included in the objective function of EC in M1 but added to the total energy consumption of the best-found solution instead. Thus, the EC values of the optimal solutions obtained by M1 are worse than those by M2.

In addition, it can be also seen that the performances of both M1 and M2 are strongly influenced by the problem scale. For the instances with the same task number, they both perform better when the number of task or workstation is smaller. For the Instances with a larger number of tasks, it takes much longer time and the resulted average RPDs becomes larger. So, it is a challenge for the model to solve the problems with larger scales. This is just our motivation to develop an effective optimization algorithm in this paper.

5.2. Parameter settings of BHEDA

From the procedure of the BHEDA, it only contains two parameters: population size PS and learning rate LR. To investigate their influence on the performance of the BHEDA, the analysis of variance (ANOVA) (Tabachnick & Fidell, 2007) is adopted. Three levels are considered for each parameter as listed in Table 3. Accordingly, an orthogonal array $L_9(3^2)$ is selected, which includes 16 different (PS, LS) combinations.

We use 8 instances with different problem scales for investigation. For each instance, the BHEDA is run 10 times independently with each combination of (PS, LR) to obtain Archive set AS_i ($i = 1, 2, \cdots, 9$). For each run, the stopping criterion is set as $0.1 \times N \times M$ seconds CPU time. The non-dominated solutions among AS_{1-9} constitute the final set FS. Then the contribution of each AS_i is calculated as $CON_i = |AS_i'|/|FS|$, where $AS_i' = AS_i \cap FE$. After testing all the instances, the average CON_i of each parameter combination i is calculated as the response value (RV). The orthogonal array and RVs are listed in Table 4. Besides, the results of one-way ANOVA with 95% confidence interval are listed in Tables 5, and the main effect is shown in Fig. 5.

From Tables 5 and Fig. 5, it can be seen that the influence of LR is more significant than PS. Since LR is used in the update of the probability model, a good convergence will benefit from a moderate value of LR. For PS, a small value will result in insufficient search in each generation, while a large value will result in insufficient evolution when the total running time is given. According to the full factorial experiments, the recommended values for parameters are PS = 500 and LR = 0.001, which will be used in the following tests.

5.3. Effect of BGS and NDRA

To demonstrate the effectiveness of the proposed bound-guided sampling (BGS) and non-dominated robot allocation (NDRA), we compare the BHEDA to the algorithm without NDRA (denoted as BEDA) and the algorithm without BGS (denoted as HEDA). In the BEDA, the robots are allocated using two greedy rules, i.e., allocating the robots that perform the tasks in the least cycle time and energy consumption. Accordingly, a fixed number of two robot allocation schemes are generated for each individual. In the HEDA, the lower bounds are not used in the sampling process while all partial individuals remain to the next stage without being expanded or removed.

Due to the random nature of the search process, each algorithm is run 20 times in solving each instance. The running time for each run is set as $0.1 \times N \times M$ seconds CPU time. In addition, the final set is defined as the Pareto front obtained by three algorithms for each run. Thus, the contribution of each algorithm is calculated as $CON = |AS'|/|FS|$, $AS' = AS \cap FS$, where AS is the archive set consisting of the non-dominated solutions obtained by a certain algorithm and FS is the final set. The average contribution $\bar{CON}$ of 20 runs is

Table 8
Comparisons between BHEDA and NSGA-II.

N	*M*	IGD			HV		
		NSGA-II	BHEDA	p-value	NSGA-II	BHEDA	p-value
25	3	3.306	**0.685**	**0.001**	0.052	**0.178**	**0.006**
	4	1.276	**0.337**	**0.001**	0.149	**0.303**	**0.000**
	6	4.660	**0.404**	**0.000**	0.039	**0.570**	**0.000**
	9	2.970	**0.645**	**0.000**	0.512	**0.736**	**0.002**
35	4	3.069	**1.668**	**0.000**	0.337	**0.545**	**0.000**
	5	4.298	**0.320**	**0.000**	0.718	**0.851**	**0.007**
	7	2.752	**0.129**	**0.000**	0.463	**0.759**	**0.000**
	12	4.238	**0.000**	**0.000**	0.079	**0.785**	**0.000**
53	5	4.046	**2.071**	**0.000**	0.053	**0.353**	**0.000**
	7	3.024	**0.596**	**0.000**	0.442	**0.690**	**0.006**
	10	3.066	**0.133**	**0.000**	0.234	**0.714**	**0.000**
	14	3.114	**0.000**	**0.000**	0.114	**0.737**	**0.000**
70	7	7.776	**0.000**	**0.000**	0.052	**0.979**	**0.000**
	10	5.152	**0.000**	**0.000**	0.100	**0.871**	**0.000**
	14	3.618	**0.000**	**0.000**	0.115	**0.894**	**0.000**
	19	3.106	**0.000**	**0.000**	0.039	**0.823**	**0.000**
89	8	3.794	**0.000**	**0.000**	0.247	**0.816**	**0.000**
	12	2.723	**0.000**	**0.000**	0.100	**0.756**	**0.000**
	16	3.336	**0.000**	**0.000**	0.107	**0.896**	**0.000**
	21	3.178	**0.000**	**0.000**	0.068	**0.839**	**0.000**
111	9	2.382	**0.000**	**0.000**	0.329	**0.736**	**0.000**
	13	2.681	**0.000**	**0.000**	0.076	**0.792**	**0.000**
	17	4.453	**0.000**	**0.000**	0.003	**0.884**	**0.000**
	22	3.102	**0.000**	**0.000**	0.007	**0.833**	**0.000**
148	10	3.892	**0.000**	**0.000**	0.086	**0.846**	**0.000**
	14	3.360	**0.000**	**0.000**	0.024	**0.867**	**0.000**
	21	2.769	**0.000**	**0.000**	0.014	**0.882**	**0.000**
	29	3.740	**0.000**	**0.000**	0.002	**0.984**	**0.000**
297	19	2.289	**0.000**	**0.000**	0.009	**0.963**	**0.000**
	29	3.380	**0.000**	**0.000**	0.009	**0.993**	**0.000**
	38	2.330	**0.000**	**0.000**	0.006	**0.969**	**0.000**
	50	2.268	**0.000**	**0.000**	0.007	**0.965**	**0.000**

The bold values mean the best results.

reported in Table 6. To analyze whether the advantage of the BHEDA is significant, we carry out the nonparametric Mann-Whitney test with 95% confidence level. The related p-values are also listed in Table 6.

From Table 6, it can be seen that the BHEDA achieves the biggest contribution over three algorithms on all instances. As the task number increases, the advantage of the BHEDA over the HEDA becomes more obvious. This implies that the BGS is able to reduce the search space effectively, especially in solving the large-scale problems. According to the nonparametric Mann-Whitney test, it shows that the BHEDA is significantly better on almost all the instances.

In addition, we select 8 instances with different task numbers and illustrate the boxplots of the obtained *CON* values in Fig. 6. Each box represents the variation of the *CON* value over 20 runs. From the boxplots, it can be seen that the results of the BHEDA is the best. Moreover, the box sizes of the BHEDA and the HEDA are smaller than the BEDA. This implies that the non-dominated robot allocation heuristic is more effective than greedy methods to produce more non-dominated solutions and to achieve better stability. Besides, the boxplots clearly show that the performance of the HEDA gets worse as the problem-scale grows. This is consistent with the results in Table 6.

So, it can be concluded that fusing NDRA and BGS into EDA is effective in solving the EERALB.

5.4. Comparison between BHEDA and PSO

Next, we compare our BHEDA with the PSO proposed by Nilakantan et al. (2015). Since both the time based and energy based models are used for the evaluation in the PSO, a time based solution and an energy based solution are produced for each instance accordingly. For fair comparison, both algorithms are run 20 times on each isntance and the running time is set as $0.1 \times N \times M$ seconds CPU time. For each run, the two solutions with the best cycle time and energy consumption are selected from the archive set of the BHEDA to compare with the two solutions obtained by the PSO. The RPD values of CT and EC from the obtained best objective values by both algorithms are used as the comparison indicator. $\text{RPD} = (Obj - Obj^*)/Obj^* \times 100$, where Obj is the objective value obtained by a certain algorithm and Obj^* is the best objective value among all tests. The comparative results are listed in Table 7, where RPD_{CT} and RPD_{EC} denote the average RPD values of CT and EC, respectively, and $|\text{AS}|$ denotes the average number of non-dominated solutions obtained by the BHEDA, and DN shows how many solutions obtained by the PSO are dominated by the solutions obtained by the BHEDA.

From the Table, it can be seen that the BHEDA can yield better CT and EC than PSO on all instances. The average RPD values of CT and EC by the BHEDA are 0.42% and 0.67%, against 10.65% and 8.81% by the PSO, respectively. Moreover, the BHEDA is able to obtain an average number of 11.91 non-dominated solutions, while the PSO can only produce two solutions. For 29 out of all 32 instances, the solutions obtained by the BHEDA dominate the solutions obtained by the PSO. For other 3 instances, one of the two solutions obtained by the PSO is dominated by those obtained by the BHEDA, while the other solution obtained by the PSO does not dominate the solutions obtained by the BHEDA each other.

We also select 8 instances with different task numbers and illustrate the solutions obtained by two algorithms in Fig. 7. The red round points are the non-dominated solutions found by the BHEDA, and the blue square points are the solutions obtained by the PSO. It is clear that the solutions obtained by the BHEDA are better. So, it can be concluded that the BHEDA is superior to the PSO in obtaining more non-dominated solutions with better quality.

5.5. Comparisons between BHEDA and NSGA-II

Next we compare the BHEDA with the most famous multi-objective evolutionary algorithm NSGA-II which also can produce a set of non-dominated solutions. To adapt the NSGA-II to solve the EERALB, the same encoding and decoding mechanisms as (Nilakantan et al., 2015) are adopted. The parameters of the NSGA-II are set as those suggested by Deb et al. (2002).

Two widely-used metrics are used to evaluate the performance of the BHEDA and the NSGA-II: inverted generational distance (IGD) (Sierra & Coello, 2005) and hyper-volume metric (HV) (Zitzler & Thiele, 1999). For fair comparison, both algorithms run 20 times on each instance with a total of $0.1 \times N \times M$ seconds CPU time as the stopping criterion. The resulted average IGD and HV are reported in Table 8. To furthur analyze whether the advantage of the BHEDA is significant, we carry out the nonparametric Mann-Whitney test with 95% confidence level. The p-values are also listed in Table 8.

From Table 8, it can be seen that both IGD and HV of the BHEDA are better than those of the NSGA-II on all instances. Especially, the IGD of the BHEDA equals to zero when $N \geq 70$, which means the non-dominated solutions found by NSGA-II are all dominated by those found by the BHEDA. In addition, the HV of BHEDA increases as the task number grows, which implies that the advantage of the BHEDA is more apparent when solving the problems with larger scales. Moreover, according to the nonparametric Mann-Whitney test, it shows that the performance of the BHEDA is significantly better than that of the NSGA-II.

We also illustrate the Pareto fronts obtained by the BHEDA and the NSGA-II for 8 instances, as Fig. 8, where the red circle and blue square points represent the non-dominated solutions found by the BHEDA and the NSGA-II, respectively. Clearly, the Pareto fronts obtained by the BHEDA are better than those obtained by the NSGA-II. In particular, the solutions obtained by the NSGA-II are dominated by those of the BHEDA on all instances, while only on the instance with the smallest scale ($N = 25$, $M = 6$) the NSGA-II obtains a same solution.

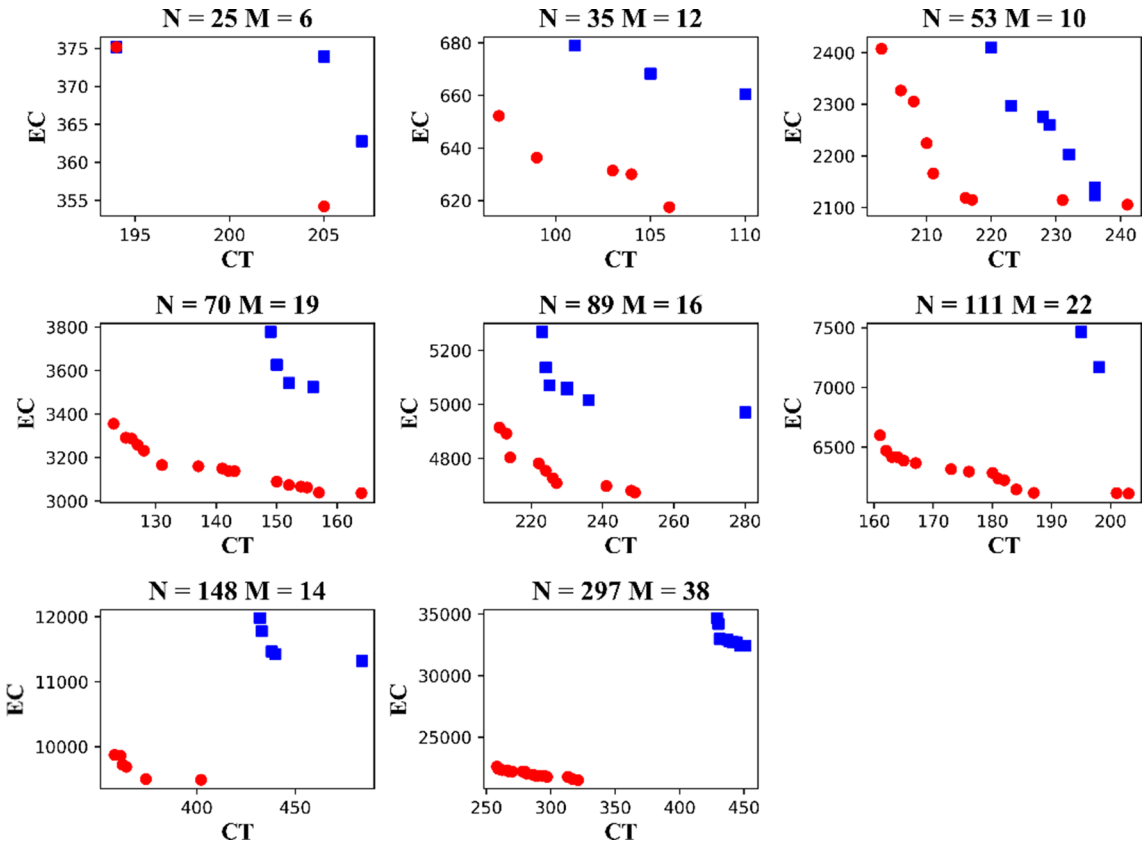


Fig. 8. Pareto fronts obtained by BHEDA and NSGA-II.

So, it can be concluded that the proposed BHEDA is more effective in solving the EERALB with the criteria of minimizing both the cycle time and energy consumption simultaneously.

6. Conclusions

In this paper, we propose a multi-objective mathematical model and a bound-guided hybrid estimation of distribution algorithm to solve the energy-efficient robotic assembly line balancing problem. The computational complexity of the proposed algorithm is analyzed and the performance is tested. Numerical comparison shows that the proposed model and the BHEDA is more effective. The effectiveness of the BHEDA mainly owes to the following aspects: (1) The probability model and its updating mechanism are effective in learning the distribution of elite task assignments and sampling the promising solutions. (2) The bound-guided sampling method is able to reduce the search space while focus on the promising area. (3) The non-dominated robot allocation heuristic is effective in allocating suitable robot to yield the non-dominated solutions efficiently.

Since problem-specific lower bounds and heuristics are adopted in the BHEDA for solving the EERALB, it may be difficult to apply it for other optimization problems directly. However, the idea of fusing learning-based intelligent algorithms with model-based mathematical approaches is of the value for designing effective algorithms to solve other complex optimization problems.

In the future, we will study the energy consumption in other production lines like the mixed-model robotic assembly lines. It is also interesting to develop other kinds of intelligent algorithms hybrid with model-based approaches and the reinforcement learning mechanisms to achieve more powerful optimization capability.

CRediT authorship contribution statement

Bin-qi Sun: Data curation, Methodology. **Ling Wang:** Conceptualization, Methodology, Funding acquisition. **Zhi-ping Peng:** Data curation, Funding acquisition.

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.

Acknowledgement

This research is supported by the National Natural Science Foundation of China (No. 61873328 and No. 61772145).

References

Baluja, S. (1994). *Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning.* Pittsburgh: Carnegie-Mellon Universtiy.

Becker, C., & Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research, 168*(3), 694–715.

Blum, C., & Miralles, C. (2011). On solving the assembly line worker assignment and balancing problem via beam search. *Computers & Operations Research, 38*(1), 328–339.

Borba, L., Ritt, M., & Miralles, C. (2018). Exact and heuristic methods for solving the robotic assembly line balancing problem. *European Journal of Operational Research, 270*(1), 146–156.

Boysen, N., Fliedner, M., & Scholl, A. (2007). A classification of assembly line balancing problems. *European Journal of Operational Research, 183*(2), 674–693.

Bukchin, J., & Tzur, M. (2000). Design of flexible assembly line to minimize equipment cost. *IIE Transactions, 32*(7), 585–598.

Chen, J., Wang, L., & Peng, Z.-P. (2019). A collaborative optimization algorithm for energy-efficient multi-objective distributed no-idle flow-shop scheduling. *Swarm and Evolutionary Computation, 50*, 100557.

Coello, C. A. C., Lamont, G. B., & Van Veldhuizen, D. A. (2007). *Evolutionary algorithms for solving multi-objective problems.* New York: Springer.
Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.
Fysikopoulos, A., Anagnostakis, D., Salonitis, K., & Chryssolouris, G. (2012). An empirical study of the energy consumption in automotive assembly. *Procedia CIRP, 3*, 477–482.
Gao, J., Sun, L., Wang, L., & Gen, M. (2009). An efficient approach for type II robotic assembly line balancing problems. *Computers & Industrial Engineering, 56*(3), 1065–1080.
Grahl, J. (2008). Estimation of distribution algorithms in logistics: Analysis, design, and application. (Dissertation), University of Mannheim, Mannheim.
Jiang, E.-D., & Wang, L. (2019). An improved multi-objective evolutionary algorithm based on decomposition for energy-efficient permutation flow shop scheduling problem with sequence-dependent setup time. *International Journal of Production Research, 57*(6), 1756–1771.
Kim, H., & Park, S. (1995). A strong cutting plane algorithm for the robotic assembly line balancing problem. *International Journal of Production Research, 33*(8), 2311–2323.
Larrañaga, P., & Lozano, J. A. (2001). *Estimation of distribution algorithms: A new tool for evolutionary computation.* Berlin: Springer Science & Business Media.
Lei, D. (2008). A Pareto archive particle swarm optimization for multi-objective job shop scheduling. *Computers & Industrial Engineering, 54*(4), 960–971.
Levitin, G., Rubinovitz, J., & Shnits, B. (2006). A genetic algorithm for robotic assembly line balancing. *European Journal of Operational Research, 168*(3), 811–825.
Li, Z., Dey, N., Ashour, A. S., & Tang, Q. (2018). Discrete cuckoo search algorithms for two-sided robotic assembly line balancing problem. *Neural Computing and Applications, 30*(9), 2685–2696.
Li, Z., Janardhanan, M. N., Ashour, A. S., & Dey, N. (2019). Mathematical models and migrating birds optimization for robotic U-shaped assembly line balancing problem. *Neural Computing and Applications,* 1–17.
Li, Z., Tang, Q., & Zhang, L. (2016). Minimizing energy consumption and cycle time in two-sided robotic assembly line systems using restarted simulated annealing algorithm. *Journal of Cleaner Production, 135*, 508–522.
Lipowski, A., & Lipowska, D. (2012). Roulette-wheel selection via stochastic acceptance. *Physica A. 391*(6), 2193–2196.
Mutlu, Ö., Polat, O., & Supciller, A. A. (2013). An iterative genetic algorithm for the assembly line worker assignment and balancing problem of type-II. *Computers & Operations Research, 40*(1), 418–426.
Nilakantan, J. M., Huang, G. Q., & Ponnambalam, S. G. (2015). An investigation on minimizing cycle time and total energy consumption in robotic assembly line systems. *Journal of Cleaner Production, 90*, 311–325.
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, 156*, 124–136.
Nilakantan, J. M., Ponnambalam, S. G., Jawahar, N., & Kanagaraj, G. (2015). Bio-inspired search algorithms to solve robotic assembly line balancing problems. *Neural Computing & Applications, 26*(6), 1379–1393.
Rubinovitz, J., Bukchin, J., & Lenz, E. (1993). RALB – a heuristic algorithm for design and balancing of robotic assembly lines. *CIRP Annals - Manufacturing Technology, 42*(1), 497–500.
Scholl, A. (1999). *Balancing and sequencing of assembly lines.* Heidelberg: Physica-Verlag.
Scholl, A., & Becker, C. (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research, 168*(3), 666–693.
Sierra, M. R., & Coello, C. A. C. (2005). Improving PSO-based multi-objective optimization using crowding, mutation and∈-dominance. *International Conference on Evolutionary Multi-Criterion Optimization,* 505–519.
Tabachnick, B. G., & Fidell, L. S. (2007). *Experimental designs using ANOVA.* Belmont: Duxbury.
Tian, J., Hao, X., & Gen, M. (2019). A hybrid multi-objective EDA for robust resource constraint project scheduling with uncertainty. *Computers & Industrial Engineering, 130*, 317–326.
Tsai, D.-M., & Yao, M.-J. (1993). A line-balance-based capacity planning procedure for series-type robotic assembly line. *International Journal of Production Research, 31*(8), 1901–1920.
United Nations (2020). The Paris Agreement. https://unfccc.int/process-and-meetings/the-paris-agreement/the-paris-agreement.
United States Environmental Protection Agency (2017). Sources of greenhouse gas emissions. https://www.epa.gov/ghgemissions/sources-greenhouse-gas-emissions.
Vila, M., & Pereira, J. (2014). A branch-and-bound algorithm for assembly line worker assignment and balancing problems. *Computers & Operations Research, 44*, 105–114.
Wang, J.-J., & Wang, L. (2020). A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 50*(5), 1805–1819.
Wang, L., Wang, S., Xu, Y., Zhou, G., & Liu, M. (2012). A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem. *Computers & Industrial Engineering, 62*(4), 917–926.
Wang, Y., & Chen, W. (2019). A decomposition-based hybrid estimation of distribution algorithm for practical mean-cvar portfolio optimization. *International Conference on Intelligent Computing,* 38–50.
Wu, C.-G., & Wang, L. (2018). A multi-model estimation of distribution algorithm for energy efficient scheduling under cloud computing system. *Journal of Parallel and Distributed Computing, 117*, 63–72.
Zhang, B., Pan, Q.-K., Gao, L., Li, X.-Y., Meng, L.-L., & Peng, K.-K. (2019). A multi-objective evolutionary algorithm based on decomposition for hybrid flowshop green scheduling problem. *Computers & Industrial Engineering, 136*, 325–344.
Zhang, Z., Tang, Q., Li, Z., & Zhang, L. (2019). Modelling and optimisation of energy-efficient U-shaped robotic assembly line balancing problems. *International Journal of Production Research, 57*(17), 5520–5537.
Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation, 3*(4), 257–271.