Expert Systems With Applications 187 (2022) 115827

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Modelling and optimization of integrated distributed flow shop scheduling and distribution problems with time windows

Yushuang Hou [a], Yaping Fu [a,*,1], Kaizhou Gao [b,*], Hui Zhang [c], Ali Sadollah [d]

[a] School of Business, Qingdao University, Qingdao 266071, China
[b] Macau Institute of Systems Engineering, Macau University of Science and Technology, Macau Taipa 999078, China
[c] School of Economics and Management, Jiangsu University of Science and Technology, Zhenjiang 212003, China
[d] Department of Mechanical Engineering, University of Science and Culture, Tehran, Iran

ARTICLE INFO

Keywords:
Integrated production and distribution
Distributed flow shop scheduling
Distributed production
Vehicle routing problem
Brain storm optimization

ABSTRACT

Production and distribution are two essential activities in supply chain management. Currently, integrated production and distribution problems receive much attention because decision-makers devote to improving the operation efficiency of both stages and try to achieve an optimal solution. This work proposes an integrated distributed production and distribution problem with consideration of time windows, in which a set of jobs (i.e., customer orders) needs to be assigned among factories and the jobs are processed on flow shop environments at their associated factories. Then, the completed jobs are delivered by capacitated vehicles to customers in different regions while satisfying given time windows as much as possible. Accordingly, to optimally solve the proposed problem, a mixed integer programming model with minimizing total weighted earliness and tardiness has been established. For the optimization task, an enhanced brain storm optimization algorithm with some particular strategies is designed to handle the considered problem. To assess the performance of the proposed optimization method, several experiments by adopting a set of benchmark test problems are performed, and state-of-the-art optimizers are chosen for comparisons. The obtained optimization results exhibit that the designed algorithm significantly outperforms its rivals and can be considered as an excellent optimizer for solving the studied problem. Besides, compared with the CPLEX solver, the designed optimizer also performs much better for solving large-size problems.

1. Introduction

Due to interdependency of most activities involving in supply chain, it can be regarded as a complex system. Managing and organizing the supply chain have become a fundamental work in the field of operations management. Production and distribution are two essential activities in the supply chain (Fu, Aloulou, & Triki, 2017). The former focuses on transforming raw materials into finished products, while the latter aims at delivering these products to customers for meeting their demands as much as possible.

Production and distribution have been extensively investigated, however, they are mostly treated separately and sequentially both in industrial applications and the literature (Fu, Li, Huang, & Xiao, 2021; Yağmur & Kesen, 2020). Nevertheless, they need to be considered together in many industry areas such as newspapers, fashion apparels, and perishable products (Liu & Liu, 2020). In such environments, once products have been completed at the production stage, they must be delivered to customers as soon as possible at the distribution stage to guarantee the quality of products. Hence, an integrated framework with consideration of production and distribution stages simultaneously should be employed to reach a global optimal solution. Therefore, modelling and optimization of integrated production and distribution problems (IPDPs) receive significant attention from the fields of industry and academia.

The existing work verifies that integrated approaches contribute to improving the overall performance of both production and distribution (Bo et al., 2021; Han, Yang, Wang, Cheng, & Yin, 2019). Through analyzing the previous researches on the IPDPs, we found that most studies consider only one factory at the production stage which is usually modeled as single-machine, parallel-machine, flow shop and job

* Corresponding authors.
E-mail addresses: fuyaping@qdu.edu.cn (Y. Fu), gaokaizh@aliyun.com (K. Gao), sadollah@usc.ac.ir (A. Sadollah).
[1] https://orcid.org/0000-0001-8645-6647

https://doi.org/10.1016/j.eswa.2021.115827
Received 7 June 2021; Received in revised form 24 August 2021; Accepted 29 August 2021
Available online 5 September 2021

shop, and vehicle routing problems at the distribution stage (Chen, 2010; Moons, Ramaekers, Caris, & Arda, 2017). However, as the economic globalization and demand diversification, manufacturing enterprises have to adopt a distributed production way where multiple factories are set up in various geographical regions to improve production efficiency and respond to customer needs as much as possible (Fu et al., 2021; Li et al., 2020; Pan, Gao, Wang, Liang, & Li, 2019; Wang & Wang, 2020; Wang, Gao, Li, Li, & Tasgetiren, 2020; Zheng, Wang, & Wang, 2020).

Thus, the IPDPs need to consider distributed manufacturing environments at the production stage, which results in a multi-depot vehicle routing problem at the distribution stage. An integration of distributed production and distribution have wide applications in real-life industry. For instance, the glass manufacturing process usually contains a set of operations such as batching, melting, forming, firing, and annealing that need to be sequentially performed as a flow shop. Because of the various kinds of requirements from customers, enterprises usually employ a distributed manufacturing way to improve operation efficiency and decrease production cost (Deng, Wang, Wang, & Zheng, 2016; Pan, Gao, & Wang, 2020; Wang & Peng, 2020; Zhao, Zhao, Wang, & Song, 2020). Thus, the glass production process can be modeled as a distributed flow shop. In addition, the glass products need to be delivered to customers as soon as possible to improve customer satisfaction and reduce inventory cost. To achieve a seamless link of the production and distribution in the glass manufacturing, an integrated distributed production and distribution model needs to be addressed.

To do such work, this study presents an integrated distributed production and distribution problem with time windows to minimize total weighted earliness and tardiness. By comparing this model and its proposed optimizer with the previous studies, two main contributions are drawn as follows:

1) This work proposes an integration of distributed production and distribution optimization problem. At the production stage, a set of jobs (i.e., customer orders) needs to be assigned among multiple factories, modeled as a distributed flow shop scheduling problem. At the distribution stage, the jobs are delivered to customers in diverse regions, abstracted as a multi-depot vehicle routing optimization problem. To better formulate the considered problem, a mixed integer programming model with minimizing total weighted earliness and tardiness is established.
2) To solve the aforementioned model, an enhanced brain storm optimization (EBSO) algorithm is developed by employing some special strategies such as solution encoding and decoding, population initialization, new individual generation, selection, and local search methods. Compared with genetic algorithm (GA) (Zhang, Li, & Yin, 2020), memetic algorithm (MA) (Kurdi, 2020), discrete artificial bee colony algorithm (DABC) (Duan, Meng, Chen, & Pan, 2018), iterative greedy algorithm (IG) (Mao, Pan, Miao, & Gao, 2020), scatter search algorithm (SS) (Pan et al., 2019), and CPLEX, the experiment results verify that the EBSO is an efficient optimizer for solving the considered problem.

The rest of this paper is outlined as follows. Next section concisely reviews the relevant studies on the investigated problem. Section 3 provides a detailed mathematical model along with problem description. Section 4 describes standard brain storm optimization (BSO) algorithm with its searching operators. Section 5 introduces the modified BSO algorithm for solving the proposed problem. Section 6 conducts experiments and performs analysis on the obtained experiment results. Finally, Section 7 summarizes this work and gives some future directions.

2. Literature reviews

Production and distribution are two essential sectors in a supply chain (Fu et al., 2017; Han, Gong, Jin, & Pan, 2017; Zhang, Zhang, Sun, & Zhao, 2021). Managing and organizing both production and distribution from the perspective of integration are regarded as an excellent way to achieve a higher operation efficiency (Yağmur & Kesen, 2020). In the past years, the IPDPs have received much attention and concern (Bo et al., 2021; Chen, 2010; Liu & Liu, 2020; Moons et al., 2017). In the following, some literatures have been reviewed in two classes: scheduling models and distribution models in the IPDPs.

Literature about the scheduling models in the IPDPs dedicated to a single factory. Liu and Liu (2020) proposed an IPDP for perishable products where a single machine, multiple customers, and vehicles with capacity constraints were considered. In their research, they designed an improved large neighborhood search algorithm for handling the IPDP. Ullrich (2013) considered an IPDP with time windows, in which unrelated parallel machines were employed to perform production operations, and a GA was introduced as an optimizer to deal with the suggested IPDP.

Kergosien, Gendreau, and Billaut (2017) addressed an IPDP for a healthcare industry where production process corresponded to a classic parallel machine scheduling problem, and they developed a benders decomposition based heuristic to solve it. Yağmur and Kesen (2020) proposed an IPDP that adopted a flow shop to process jobs at the production stage, and subsequently distributing them by a single capacitated vehicle. They presented an MA to find optimal or near-optimal solutions. Mohammadi, Cheraghalikhani, and Ramezanian (2018) developed an IPDP where products were processed through a flow shop. They designed an improved imperialist competitive algorithm (ICA) to cope with it. Mohammadi, Al-e-Hashem, and Rekik (2020) studied an IPDP for a furniture manufacturing company to minimize a weighted sum of earliness and tardiness, where the production process was modeled as a flexible job shop. To handle it, they proposed a hybrid particle swarm optimization algorithm.

In addition to the aforementioned work, many studies focus on scheduling one factory which can be formulated as single-machine (Karaoğlan & Kesen, 2017; Lacomme, Moukrim, Quilliot, & Vinot, 2018), parallel-machine (Belo-Filho, Amorim, & Almada-Lobo, 2015), flow shop (Ramezanian, Mohammadi, & Cheraghalikhani, 2017) and job shop (Meinecke & Scholz-Reiter, 2014) models.

The distribution models in the IPDPs were discussed as follows. Yan, Banerjee, and Yang (2011) presented an IPDP for perishable products, where one supplier and one customer were considered without performing routing decisions. Schmid, Doerner, Hartl, and Salazar-González (2010) studied an IPDP where a vehicle was allowed to serve only one customer. Thus, routing decisions were not considered at the distribution stage as well. Unlike the aforementioned studies which could not take vehicle routing decisions into account, many studies focused on IPDPs in which the distribution process was performed by multiple vehicles, and thus routing decisions need to be made. Fu et al. (2017) proposed an IPDP for metal packaging enterprises, where jobs were processed on unrelated parallel machines, and then they were delivered to customers within given time windows. They used a two-phase iterative heuristic to solve it.

Kesen and Bektaş (2019) considered an IPDP where delivery operations were modeled as a vehicle routing problem. They employed a mathematical programming solver, i.e., CPLEX, to tackle it. Devapriya, Ferrell, and Geismar (2017) addressed an IPDP for perishable products where the fleet size and vehicles' routes subject to a planning horizon constraint. They designed evolutionary algorithms to settle it. Wang et al. (2019) studied an IPDP where distribution stage aimed at solving a vehicle routing problem. They proposed an improved MA to handle it. In the past years, Chen (2010) and Moons et al. (2017) have made elaborated reviews on IPDPs, respectively. More details regarding the relevant studies on IPDPs can be acquired in their study.

By summarizing the relevant researches on IPDPs, we can find that they possess the following features: 1) Most of studies consider scheduling only one factory with single-machine (Mousavi, Hajiaghaei–Keshteli, & Tavakkoli–Moghaddam, 2020), parallel-machine (Kazemi, Mahdavi Mazdeh, Rostami, & Heydari, 2020), flow shop (Lacomme et al., 2018) and job shop (Mohammadi et al., 2020) models at the production stage; 2) Vehicle routing problems with only one depot are formulated at the distribution stage; 3) Because of the high complexity of IPDPs, swarm intelligence approaches have been extensively employed to handle them (Feng, Chu, Chu, & Huang, 2021; Kabadayi, 2021). In fact, the previous studies have verified the excellent performance of swarm intelligence methods in solving various kinds of complicated optimization problems (Fu, Wang, Tian, Li, & Hu, 2019; Fu, Zhou, Guo, & Qi, 2020; Han et al., 2020).

Via the discussed materials in this section, we can conclude that the prior studies give little attention on the integrated distributed production and distribution problems although they have very essential industry applications in reality. Thus, this work proposes an integrated distributed production and distribution problem to achieve an overall optimization of the distributed production and distribution stages. The production stage aims at scheduling a distributed flow shop, while the distribution stage focuses on solving a multi-depot vehicle routing problem. A mixed integer programming model is formulated to minimize total weighted earliness and tardiness. Then, an EBSO is designed by considering this problem's characteristics to find a global optimal solution.

3. Problem formulation

The considered problem consists of two stages: production stage and distribution stage. The production stage has multiple factories, and each one can be regarded as a flow shop in which there are multiple machines in series. A set of jobs needs to be assigned among factories and each job is allowed to be assigned to only one factory. All jobs assigned to a factory are processed on machines following the same route. A machine can process only one job at a time, a job is allowed to be processed on only one machine at a time, and interruption is not allowed. Notice that a job at the production stage can be regarded as a customer at the distribution stage, i.e., each job is associated with a customer.

The distribution stage can be seen as a multi-depot vehicle routing problem aiming at delivering the jobs to customers locating in different regions by employing vehicles with a given maximum capacity. Each customer is visited at most once and each vehicle can be used at most once. Each customer has a requirement for the delivery time which is denoted as a time window including the earliest arrival time and latest departure time. If a vehicle arrives at a customer before the earliest arrival time, an earliness penalty occurs and is applied into the model. It is formulated as a product of a given weight and amount of earliness. Accordingly, a tardiness penalty happens if the time that vehicle departs from the customer is later than the latest departure time. Fig. 1 illustrates an integrated distributed production and distribution in a schematic way.

To establish a mathematical model for the considered problem, a nomenclature is given in Table 1.

By employing the notations and definitions given in Table 1, the mathematical programming model is established as follows:

$$\text{Minimize}\sum\nolimits_{j=1}^{n}\left(W_j^E\times\max\left(a_j-A_j,0\right)+W_j^T\times\max\left(L_j-b_j,0\right)\right) \tag{1}$$

s. t.

$$\sum_{j\in C\setminus\{0\}}X_{0jg}\le 1,\forall g\in F. \tag{2}$$

$$\sum_{j\in C}\sum_{g\in F}X_{kjg}=1,\forall k\in C,k\ne 0. \tag{3}$$

$$\sum_{k\in C}X_{kjg}=\sum_{k\in C}X_{jkg},\forall j\in C,j\ne 0,\forall g\in F. \tag{4}$$

$$S_{ij}\ge S_{ik}+p_{ik}-G\bullet\left(1-\sum_{g\in F}X_{kjg}\right),\forall k,j\in C,k\ne 0,j\ne 0,\forall i\in M. \tag{5}$$

$$S_{ij}\ge S_{(i-1)j}+p_{ij},\forall j\in C,j\ne 0,\forall i\in M\setminus\{1\}. \tag{6}$$

$$C_j\ge S_{mj}+p_{mj},\forall j\in C,j\ne 0. \tag{7}$$

$$\sum_{j\in C}X_{kjg}=\sum_{h\in K}\sum_{j\in N}Y_{kjgh},\forall g\in F,\forall k\in C,k\ne 0. \tag{8}$$

$$\sum_{j\in C\setminus\{0\}}Y_{kjgh}\le 1,\forall k\in F,\forall g\in F,\forall h\in K. \tag{9}$$

$$\sum_{j\in N}Y_{kjgh}=\sum_{j\in N}Y_{jkgh},\forall k\in N,\forall g\in F,\forall h\in K. \tag{10}$$

$$\sum_{g\in F}\sum_{h\in K}\sum_{j\in N}Y_{kjgh}=1,\forall k\in C,k\ne 0. \tag{11}$$

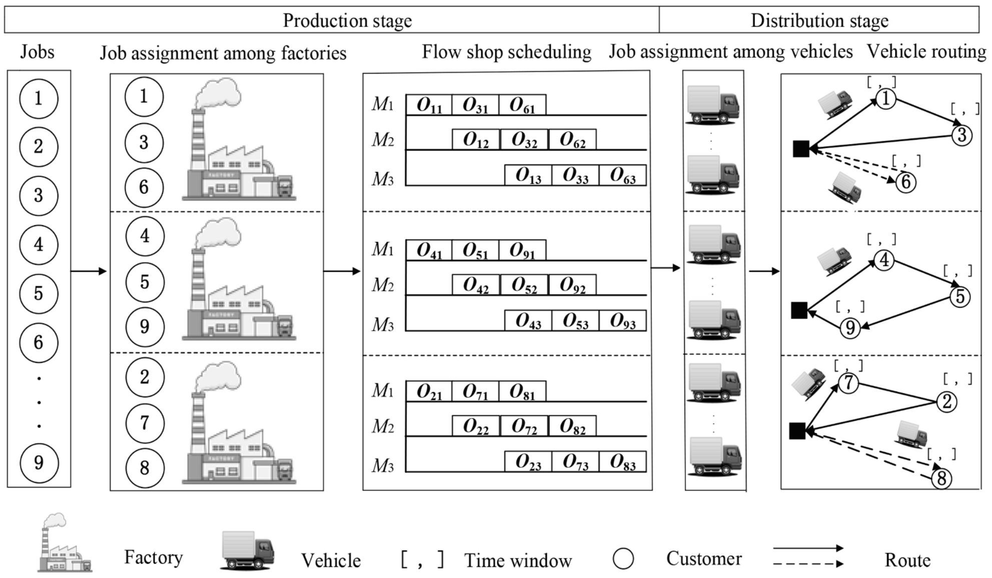

Fig. 1. Illustration of an integrated distributed production and distribution.

Table 1
Notations and symbols used in the proposed mathematical model.

Notations	Descriptions
	Indices
C	Set of jobs,$C = \{0, 1, 2, \cdots, n\}$, where n is the number of jobs and 0 is a dummy job. The jobs are indexed by symbols k, j, l, $k, j, l \in C$.
F	Set of factories,$F = \{n+1, n+2, \ldots, n+f\}$, where f is the number of factories. The factories are indexed by a symbol g, $g \in F$.
M	Set of machines, $M = \{1, 2, \cdots, m\}$, where m is the number of machines at a factory. The machines are indexed by a symbol i, $i \in M$.
K	Set of vehicles, $K = \{1, 2, \cdots, s\}$, where s is the number of vehicles at a factory. The vehicles are indexed by a symbol h, $h \in K$.
N	Set of jobs and factories, $N = C\{0\} \cup F$.
	Parameters
p_{ij}	Processing time of job j on machine i.
τ_{kj}	Travel time between customers k and j.
v_j	Service time at customer j.
d_j	Weight of job j.
$[a_j, b_j]$	Required time windows of customer j, where a_j and b_j are respectively the earliest arrival time and the latest departure time.
W_j^E	Unit earliness penalty of job j.
W_j^T	Unit tardiness penalty of job j.
Q	Maximum capacity of vehicles.
G	An infinite constant.
	Decision variables
X_{kjg}	Binary variable. It equals to 1 if job$j$ is processed immediately after job$k$ at factory g, and 0 otherwise.
Y_{kjgh}	Binary variable. It equals to 1 if vehicle h serves customer k and then customer j at factory g, and 0 otherwise.
S_{ij}	Start time of job j on machine i.
C_j	Completion time of job j at the production stage.
V_{gh}	Departure time of vehicle h at factory g.
U_{ghj}	Delivery time of job j of vehicle h at factory g.
A_j	Arrival time at customer j.
L_j	Leave time from customer j.

$$C_j \le U_{ghj} + G\bullet\left(1 - \sum_{j\in N} \mathrm{Y}_{kjgh}\right), \forall k \in C, j \ne 0, \forall g \in F, \forall h \in K. \tag{12}$$

$$V_{gh} \ge U_{ghj}, \forall j \in C, j \ne 0, \forall g \in F, \forall h \in K. \tag{13}$$

$$A_j \ge V_{gh} + \tau_{gj} - G\bullet\left(1 - \sum_{g\in F} \mathrm{Y}_{kjgh}\right), \forall k \in F, \forall j \in C, j \ne 0. \forall g \in F, \forall h \in K. \tag{14}$$

$$A_j \ge U_{ghj} + \tau_{gj}, \forall k \in F, \forall j \in C, j \ne 0, \forall g \in F, \forall h \in K. \tag{15}$$

$$A_j \ge A_k + \tau_{kj} - G\bullet\left(1 - \sum_{g\in F}\sum_{h\in K} \mathrm{Y}_{kjgh}\right), \forall k, j \in C, k, j \ne 0 \tag{16}$$

$$L_j \ge A_j + v_j, \forall j \in C, j \ne 0 \tag{17}$$

$$\sum_{k\in N}\sum_{j\in N} Y_{kjgh}\bullet \mathrm{d_j} \le Q, \forall g \in F, \forall h \in K. \tag{18}$$

$$X_{kjg} \in \{0, 1\}, \forall k, j \in C, k \ne j, \forall g \in F \tag{19}$$

$$S_{ij} \ge 0, \forall j \in C, \forall i \in M. \tag{20}$$

$$C_j \ge 0, \forall j \in C. \tag{21}$$

$$Y_{kjgh} \in \{0, 1\}, \forall k, j \in N, \forall g \in F, \forall h \in K. \tag{22}$$

$$V_{gh} \ge 0, \forall h \in K, \forall g \in F. \tag{23}$$

$$U_{ghj} \ge 0, \forall g \in F, \forall h \in K, \forall j \in N. \tag{24}$$

$$A_j \ge 0, \forall j \in C. \tag{25}$$

$$L_j \ge 0, \forall j \in C. \tag{26}$$

where Eq. (1) means that the objective is to minimize the total weighted earliness and tardiness. Eq. (2) indicates that the dummy job has only one immediate succession job at factories. Eq. (3) ensures that each job has only one immediate succession job. Eq. (4) defines the order relationship of two adjacent jobs. Eq. (5) guarantees that a machine is allowed to process only one job at a time. Eq. (6) indicates that a job can be processed on only one machine at a time. Eq. (7) defines the completion time of jobs at the production stage. Eq. (8) describes the continuity of two stages. Eq. (9) stipulates that a vehicle departs its associated factory at most once. Eq. (10) ensures that the in-degree and out-degree at customers are equal. Eq. (11) represents that each customer must be visited at most once. Eq. (12) implies that the delivery start time of jobs at the distribution stage must be equal or greater than their completion time at the production stage. Eq. (13) specifies that the departure time of vehicles must be equal or greater than delivery time of jobs in vehicles. Eqs. (14) to (16) define the arrival time of vehicles at customers. Eq. (17) denotes the leave time of vehicles from customers. Eq. (18) explains that the load of vehicles cannot exceed the maximum capacity. Eqs. (19) to (24) give the range of decision variables (i.e., imposed side constraints).

According to the aforementioned statements, the considered problem contains two well-known optimization problems: distributed flow shop scheduling and multi-depot vehicle routing problems. In fact, both of them have been proven to be NP-hard (Baradaran, Shafaei, & Hosseinian, 2019; Fernandez-Viagas, Perez-Gonzalez, & Framinan, 2018). Therefore, the considered problem (by combining both models) is an NP-hard model as well. Thus, the need of using meta-heuristics such as swarm intelligence approaches is vital to cope with the proposed optimization model.

4. Standard brain storm optimization

Standard BSO algorithm, which has been successfully applied to solve various kinds of engineering optimization problems, is regarded as one of the effective swarm intelligence methods (Hao et al., 2019; Ke, 2018). Inspired by the swarm behaviors of human beings in problem-solving process, i.e., the brainstorming process, Shi (2011) initially introduced the framework of standard BSO. The BSO starts with a population including a set of individuals, and each individual represents a solution of an optimization problem. Its search process contains three phases, i.e., the solution clustering, new individual generation, and selection.

At the solution clustering phase, the population is divided into multiple clusters by using a clustering method, the best solution in a cluster is treated as a center individual, and the rest are considered as normal individuals. At the new individual generation phase, the BSO uses one or two individual(s) from one or two cluster(s) to produce new individuals. At the selection phase, the newly-generated individuals are compared with their corresponding individuals in population, and the better solutions are reserved. Then, the aforementioned phases are repeated until a termination condition is met (e.g., the maximum number of function evaluations), and then the algorithm reports the achieved best solution and its corresponding objective value. The main steps of BSO are given as follows:

Step 1: Initialize parameters of BSO, i.e., population size ρ, number of clusters η, parameters r_s, r_o, and r_t that are employed to select the methods of new individual generation, and a termination condition.

Step 2: Generate ρ individuals at random as an initial population, and then evaluate their objective function values.

Step 3: Divide ρ individuals into η clusters.

Step 4: Perform the next loop for each individual π in population ($\pi^{'}$ represents a newly-generated individual in a loop, π and $\pi^{'}$ are defined as associated individuals):

Step 4.1: Generate a random number r_1 from an interval [0, 1]. If $r_1 < r_s$, proceed to Step 4.2; Otherwise, go to Step 4.3.

Step 4.2: Choose a random cluster, and generate a random number r_2 from [0, 1]. If $r_2 < r_o$, $\pi^{'}$ is generated with this cluster's center; Otherwise, $\pi^{'}$ is produced with a normal individual selected from the chosen cluster.

Step 4.3: Choose two random clusters, and generate a random number r_3 from [0, 1]. If $r_3 < r_t$, $\pi^{'}$ is produced with the two cluster's centers; Otherwise, $\pi^{'}$ is generated with two normal individuals selected from the chosen clusters.

Step 5: Compare newly-generated individuals with their corresponding ones, and reserve the better solutions for the next population.

Step 6: Export the best solution if a given termination criterion is met; Otherwise, go to Step 3.

The BSO has many advantages such as easy implementation, strong search ability, and high solution stability (Shi, 2011). In recent years, it has attained great success in handling various complicated engineering optimization problems such as distributed flow shop scheduling problems (Hao et al., 2019), vehicle routing problems (Ke, 2018), and optimization problems in the electromagnetic field (Duan, Li, & Shi, 2013). However, to the best of our knowledge, it has not been yet employed to solve integrated production and distribution problems in industry systems. Thus, this paper is dedicated to investigate its applications for solving the IPDPs.

5. Proposed enhanced BSO

The standard BSO is initially introduced to solve continuous optimization problems (Shi, 2011). Therefore, in the light of discrete and highly-complex natures of the proposed model, we need to specially design some strategies based on standard BSO. Hence, we develop an EBSO approach to handle the studied problem. The details and discretized approach are described in the following sections.

5.1. Solution encoding and decoding method

To solve the integrated production and distribution problem, we need to make four decisions, i.e., job assignment among factories, job sequence at factories at the production stage, as well as the served customers of vehicles and their delivery routes at the distribution stage. In the proposed approach, an integer string to represent a solution is used. Each integer in the string indicates a job (i.e., customer) index. For instance, a solution $(3, 6, 4, 1, 5, 7, 9, 2, 8)$ indicates that all jobs are scheduled as a sequence 3→6→4→1→5→7→9→2→8. Notice that this solution representation method just gives the scheduled sequence of jobs, and we cannot obtain executable decisions as the above requirements. Therefore, some heuristic rules to decode a solution into a feasible scheme is defined as follows:

At the production stage, two heuristic rules are defined as: 1) Each job is assigned to a factory with the smallest makespan; 2) All jobs assigned to a factory are processed on machines as their assigned sequence. At the distribution stage, two heuristic rules are designed as: 1) All jobs at a factory are assigned to a vehicle sequentially as their processing sequence. If the load of the vehicle exceeds the maximum capacity, a new vehicle comes into use; 2) The jobs loaded in a vehicle are delivered to customers as their processing sequence.

To better understand the solution encoding and decoding method regarding the distribution stage, an example with 2 factories and 6 jobs is given. Let P_g represent a set of jobs that is processed at factory g, $g \in \{1, 2\}$. D_{gk} denotes the delivery jobs of vehicle k at factory g, $g \in \{1, 2\}$. v_{gk} is the index of vehicle k at factory$g$. d_j denotes the weight of jobs, $j \in \{1, 2, 3, 4, 5, 6\}$. Suppose $P_1 = \{1, 3, 5\}$, $P_2 = \{2, 4, 6\}$, $d_1 = 5$, $d_2 = 4$, $d_3 = 2$, $d_4 = 2$, $d_5 = 3$, $d_6 = 4$, and the maximum load capacity of vehicles is 6. When job 1 at factory 1 is assigned, because the vehicle v_{11} is empty, job 1 can be directly assigned to v_{11}, then the current total load of v_{11} is equal to 5. The next job 3 is sequentially assigned to v_{11}. However, the total load of v_{11} is 7, and it exceeds the maximum load capacity. As a consequence, the vehicle v_{12} comes into use and job 3 is finally assigned to v_{12}. Next, job 5 at factory 1 is assigned to v_{12}, the current total load of v_{12} is equal to 5. The method of assigning jobs 2, 4, and 6 at factory 2 is the same with the aforementioned process. Consequently, we obtain decisions of the served customers of vehicles, i.e., $D_{11} = \{1\}$, $D_{12} = \{3, 5\}$, $D_{21} = \{2, 4\}$, and $D_{22} = \{6\}$. We also achieve the delivery routes of vehicles that are consistent with the assigned sequence of jobs to the vehicles.

5.2. Population initialization

A high-quality population with better approximation and diversity contributes to enhancing the search ability of swarm intelligence approaches. Hence, to construct an initial population, this work generates three individuals by adopting heuristic rules, i.e., Earliest Due Date (EDD), unit Earliness and Tardiness Weight (WET), and their combination (EDDWET) (Jing, Pan, Gao, & Wang, 2020), and the rest are randomly generated. The three rules are described as follows:

- EDD rule: All jobs are sorted as non-decreasing order according to b_j.
- WET rule: All jobs are divided into two groups G_1 and G_2 according to their unit earliness and tardiness weights. The jobs in G_1 meet $W_j^E \leq W_j^T$, and are sorted as non-increasing order according to unit tardiness weights, while the jobs in G_2 satisfy $W_j^E > W_j^T$, and are sorted as non-decreasing order in accordance with unit earliness weights. At last, G_1 and G_2 are linked together to generate an individual.
- EDDWET rule: G_1 and G_2 are constructed with the WET rule. The first job with a smaller b_j is moved to a new sequence G_3, and it is deleted from G_1 or G_2. This process is repeated until one of G_1 and G_2 becomes empty. Then, the jobs in G_3 connect with those in G_1 or G_2 to construct an individual.

5.3. Solution clustering

The standard BSO constructs multiple clusters by employing Euclidean distance among individuals, and the similar individuals with the smaller distance to a center are employed to construct the cluster. However, the investigated problem is discrete, and thus we should give a special scheme to divide the population into η clusters. The designed clustering approach is given as follows:

Step 1: Choose the top η best individuals from the population as centers.

Step 2: Calculate the difference of objective function values between the rest individuals and a center.

Step 3: Select the top $\rho/\eta - 1$ individuals with the smaller difference values to construct a cluster.

Step 4: Repeat Steps 2 and 3 to construct clusters for all centers.

5.4. New individual generation

The standard BSO utilizes one or two individual(s) that are selected from one or two cluster(s) to generate new individuals. In the EBSO, the new individual generation method is designed as follows:

1) If a normal individual is chosen, a swap-based method is used where two jobs are selected at random and swapped. If a center individual is chosen, a local search extended by a simulated annealing (SA) method (Bertsimas & Tsitsiklis, 1993) is employed. Let π be a center individual, the local search approach is described as:

Step 1: Initialize the temperature T_o:=O_w −O_b, where O_w and O_b are the objective function values of the worst and best individuals found in the previous search, respectively, and T:=T_o.

Step 2: Construct an individual π' based on π using a swap-based approach.

Step 3: If π' is better than π, then π:=π'.

Step 4: T:=T- $T \times \tau$, where τ is a given constant parameter to reduce the temperature continuously and it is equal to 0.1.

Step 5: Repeat Steps 2 to 4 until $T < T_o \times \tau$.

2) If two center or normal individuals are chosen, an order-based crossover approach is utilized to produce a new individual (Umbarkar & Sheth, 2015). Fig. 2 depicts an example with 9 jobs. As can be seen in Fig. 2, it is performed as follows: Two cut points are randomly chosen, and the job block between them in π_1 is copied to π' at the same positions. The other jobs are chosen from π_2 and added to π' according to their related sequence. Thus, a new individual π' is constructed.

5.5. Referenced local search method

After producing new individuals, a referenced local search (RLS) method (Pan et al., 2019) is applied to refine the best newly-generated individual. The basic idea behind of RLS is to investigate all neighborhood solutions and find the best one. In the RLS, a neighborhood solution is generated by inserting a job into the other position. It has been successfully extended to tackle many scheduling problems such as flow shop (Pan, Tasgetiren, & Liang, 2008) and distributed flow shop problems (Pan et al., 2019). In this work, the same approach has been taken.

Let π_b be the refined solution and the heuristic decoding rules are employed to obtain the decision of job assignment among factories. The RLS extracts a job from the assignment and reinserts it into all possible positions. The job is finally located at a position such that the best neighborhood is found. This process is repeated until all jobs have been considered. Afterwards, the best solution will be stored.

5.6. Selection process

At the selection phase, a set of better individuals needs to be chosen for the next iteration. In the standard BSO, each newly-generated individual is compared with its associated individual in population, and the better one is transferred to the next population. Unlike the above process that is likely to discard some better individuals, the proposed EBSO optimizer combines the population with all newly-generated individuals, and then the top ρ best individuals are employed to construct the next population.

After the discussion on all components of EBSO, its pseudo-code in Algorithm 1 is provided where $rand(0,1)$ returns a random number from an interval [0, 1].

Algorithm 1. Pseudo-code of EBSO

```
Input: Algorithm parameters $\rho$, $\eta$, $r_s$, $r_o$, $r_t$, and a termination condition.
Output: The obtained best solution.
```

(continued on next column)

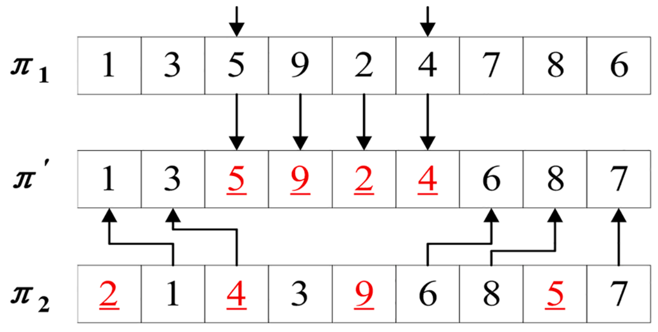

Fig. 2. Illustration of the order-based crossover approach.

(continued)

Algorithm 1. Pseudo-code of EBSO

```
Begin
  Initialize algorithm parameters.
  Generate $\rho$ individuals as an initial population.
  Repeat
  //Clustering method//
  Divided $\rho$ individuals into $\eta$ clusters.
  //New individual generation//
  For (each individual in population)
  If ($rand(0,1) < r_s$) Then
    //Use a swap-based local search method to generate a new individual//
  If ($rand(0,1) < r_o$) Then
    Select a center individual to produce a new individual.
  Else
    Select a normal individual to obtain a new individual.
  End If
Else
  //Adopt an order-based crossover approach to generate a new individual//
  If ($rand(0,1) < r_t$) Then
    Select two center individuals to produce a new individual.
Else
  Select two normal individuals from chosen clusters to produce a new individual.
End If
End If
End For
  Perform the RLS method on the best newly-generated individual.
  //Selection//
  Choose the top $\rho$ best individuals as the next population.
Until a given termination condition is met.
End
```

6. Experiment results and discussions

To assess the performance of EBSO for solving the proposed model, a set of benchmark problems is considered for experiments. In addition, a mathematical programming solver (i.e., CPLEX) and five well-known optimization algorithms, i.e., genetic algorithm (GA) (Zhang et al., 2020), memetic algorithm (MA) (Kurdi, 2020), discrete artificial bee colony algorithm (DABC) (Duan et al., 2018), iterative greedy algorithm (IG) (Mao et al., 2020), and scatter search algorithm (SS) (Pan et al., 2019) are chosen for comparisons. All the studied algorithms are coded in C++ programming language and carried out on the platform VC++ 2017 on a PC with an Intel Core i5-8265U CPU @ 1.60 GHz with 8 GB RAM.

6.1. Test instance generation

In this section, three types of benchmark problems, i.e., Taillard benchmark (Taillard, 1993) on flow shop scheduling problems, Gehring & Homberger benchmark (Homberger & Gehring, 2005) and Solomon benchmark (Solomon, 1987) on vehicle routing problems, are examined in this work. At the production stage, 10 instances are selected as follows: Ta001, Ta011, Ta031, Ta041, Ta051, Ta061, Ta071, Ta081, Ta091, and Ta101 from Taillard benchmark (Taillard, 1993), where $n \in \{20, 50, 100, 200\}$, $m \in \{5, 10, 20\}$, and we define $f \in \{3, 5, 7\}$. At the distribution stage, the problems C1_2_1, C1_6_1, and C2_2_1 from Gehring & Homberger benchmark (Homberger & Gehring, 2005) and C101 from Solomon benchmark (Solomon, 1987) are chosen. It is worth mentioning that the first instance is selected for each problem. Consequently, 30 benchmark test instances are constructed by developing a set of combinations based on the selected benchmark on flow shop and vehicle routing problems.

6.2. Parameter setting and sensitivity analysis

In order to analyze the influence of user parameters on the performance of EBSO, several experiments are carried out with diverse parameter settings and the obtained results are analyzed using the Taguchi approach (Montgomery, 2005). An instance via combining

Ta041 and C1_6_1 with 5 factories, 10 machines, and 50 jobs (recorded as Ta041_5 & C1_6_1 for convenience) is employed. The EBSO contains four key parameters, i.e., ρ, r_s, r_o, and r_t. Each parameter has four levels, i.e., $\rho \in \{30, 60, 90, 120\}$, $r_s \in \{0.2, 0.4, 0.6, 0.8\}$, $r_o \in \{0.2, 0.4, 0.6, 0.8\}$, and $r_t \in \{0.2, 0.4, 0.6, 0.8\}$. Therefore, we select an orthogonal array $L_{16}(4^4)$ including 16 parameter combinations, as tabulated in Table 2.

The EBSO performs each parameter combination for 20 runs independently, and the average objective function value over 20 independent runs is calculated as the response variable (RV) as can be seen in Table 2. In addition, we set a termination criterion that the maximum running time is 300nm milliseconds, where n and m are the number of jobs and machines, respectively. The significance rank of parameter combinations is given in Table 3, and the factor level trend of each parameter is accordingly plotted in Fig. 3.

By analyzing the experiment results, it can be seen that ρ and r_o play the most important and second roles, respectively, as they greatly influence EBSO's exploration and exploitation abilities. Besides, r_s and r_t play the third and fourth roles, respectively. According to the aforementioned results, a promising parameter combination is suggested as follows: $\rho = 60$, $r_s = 0.4$, $r_o = 0.6$ and $r_t = 0.6$, and thus, these preference parameters are used in the considered experiments.

6.3. Comparisons between EBSO and CPLEX

To analyze the gap between the acquired results by the EBSO and the optimal solutions, this section compares the EBSO with a mathematical programming solver, i.e., CPLEX, to handle the considered problem. An example with 20 jobs, 3 factories where each factory contains 2 machines is tested. The processing time of jobs is from a benchmark test instance named Ta001 in Taillard benchmark (Taillard, 1993), and the coordinate (c_x, c_y) of customers is from a benchmark instance named C1_2_1 in Gehring & Homberger benchmark (Homberger & Gehring, 2005). The maximum load capacity is 6, and the coordinates of factories are $(33, 78)$, $(59, 52)$, and $(10, 137)$, respectively. Table 4 tabulates the job information.

Next, four instances with the first 6 jobs, 10 jobs, 15 jobs and all jobs are examined. Through preliminary experiments, it can be found that the CPLEX solves an instance with 6 jobs in around 24 s, while for the instance with 10 jobs it takes about 48 min. In fact, the CPLEX needs more computational time for solving the instances with 15 and 20 jobs. In general, working with the CPLEX method, computational time dramatically increases as the number of jobs increases.

In this work, the maximum computational time as a termination condition of CPLEX is considered. Therefore, it also achieves an approximately optimal value (AOV) for large-size problems. Table 5 reports the experiment results attained by the EBSO and CPLEX. Via the preliminary experiments, it can be observed that the CPLEX can reach the optimal solution when $n = 6$, and achieve approximately optimal solution when $n \in \{10, 15, 20\}$. According to the results in Table 5, it can be seen that the CPLEX can obtain better optimization results compared with the EBSO when $n = 6$, while the EBSO outperforms the CPLEX when $n \in \{10, 15, 20\}$.

It is noted that the EBSO spends less computational time compared with the CPLEX for solving all the instances. By comparing the EBSO with CPLEX, we can draw a conclusion that the EBSO exhibits better performance and efficiency than CPLEX in solving the considered problem.

Table 2
Orthogonal experiments and response values of EBSO with different parameter combinations.

Trail number	Factor level				RV
	ρ	r_s	r_o	r_t	
1	30	0.2	0.2	0.2	1460.2050
2	30	0.4	0.4	0.4	1432.2200
3	30	0.6	0.6	0.6	1425.6500
4	30	0.8	0.8	0.8	1484.9600
5	60	0.2	0.4	0.6	1449.3600
6	60	0.4	0.2	0.8	1415.9150
7	60	0.6	0.8	0.2	1444.6000
8	60	0.8	0.6	0.4	1427.7450
9	90	0.2	0.6	0.8	1450.9300
10	90	0.4	0.8	0.6	1455.2300
11	90	0.6	0.2	0.4	1498.2250
12	90	0.8	0.4	0.2	1450.2450
13	120	0.2	0.8	0.4	1445.5100
14	120	0.4	0.6	0.2	1442.9500
15	120	0.6	0.4	0.8	1446.4450
16	120	0.8	0.2	0.6	1448.3500

Table 3
Response and rank of parameters for the EBSO.

Level	ρ	r_s	r_o	r_t
1	1450.7588	1451.5013	1455.6738	1449.5000
2	1434.4050	1436.5788	1444.5675	1450.9250
3	1463.6575	1453.7300	1436.8188	1444.6475
4	1445.8138	1452.8250	1457.5750	1449.5625
Delta	29.2525	17.1512	20.7562	6.2775
Rank	1	3	2	4

6.4. Comparisons between EBSO and metaheuristics

In this section, five swarm intelligence methods, i.e., GA (Zhang et al., 2020), MA (Kurdi, 2020), DABC (Duan et al., 2018), IG (Mao et al., 2020), and SS (Pan et al., 2019), as competitive approaches are entered into the comparison pool for the sake of fair assessment. The parameter setting of the GA, MA, DABC, IG, and SS refers to their corresponding literature and finally we summarize the parameters in Table 6. The maximum running time is 300nm milliseconds. Twenty independent runs are performed on each instance for all optimization methods. In this work, we test the performance of algorithms by relative percentage deviation (RPD) as follows:

$$RPD = \frac{\theta_v - \theta^*}{\theta^*} \times 10\%. \tag{27}$$

where θ^* is the best objective function value obtained by the EBSO and its peers, and θ_v is the objective function value of a certain algorithm in the v-th replication. In the following experiments, the average RPD ($aRPD$), the best RPD ($bRPD$), and the standard deviation of RPD ($sRPD$) are calculated over 20 replications to analyze the experiment results. Notice that the smaller $aRPD$, $bRPD$, and $sRPD$ values indicate that the EBSO is much better than its competitors.

The experiment results of EBSO and its peers regarding $aRPD$, $bRPD$, and $sRPD$ are shown in Tables 7 and 8. Through observing the obtained optimization results in Table 7, it can be found that the EBSO exhibits better performance than its peers on most instances with respect to the $aRPD$. The results regarding the $aRPD$ reveal that the EBSO has obvious advantage over the DABC and IG on all the instances, and it outperforms GA, MA, and SS on 26, 29, 27 out of 30 instances, respectively. Besides, the average $aRPD$ values of EBSO and GA, MA, DABC, IG, SS on all the instances are 0.0087, 8.2094, 5.0748, 9.1517, 7.4047, and 0.5646, respectively.

Accordingly, it can be observed that the EBSO significantly performs better than the DABC on all the instances regarding the $bRPD$. The proposed optimizer in this work shows better performance compared with the GA, MA, and IG on 26, 27, 27 out of 30 instances, respectively. Moreover, the $bRPD$ values suggest that the SS can find better solutions on some instances. Nevertheless, the EBSO is superior to SS since the former can find more satisfactory results. The average $bRPD$ values on all instances obtained by the GA, MA, DABC, IG, SS, and EBSO are equal to 1.7086, 1.4067, 2.4619, 1.7256, 0.0206, and 0.0010, respectively.

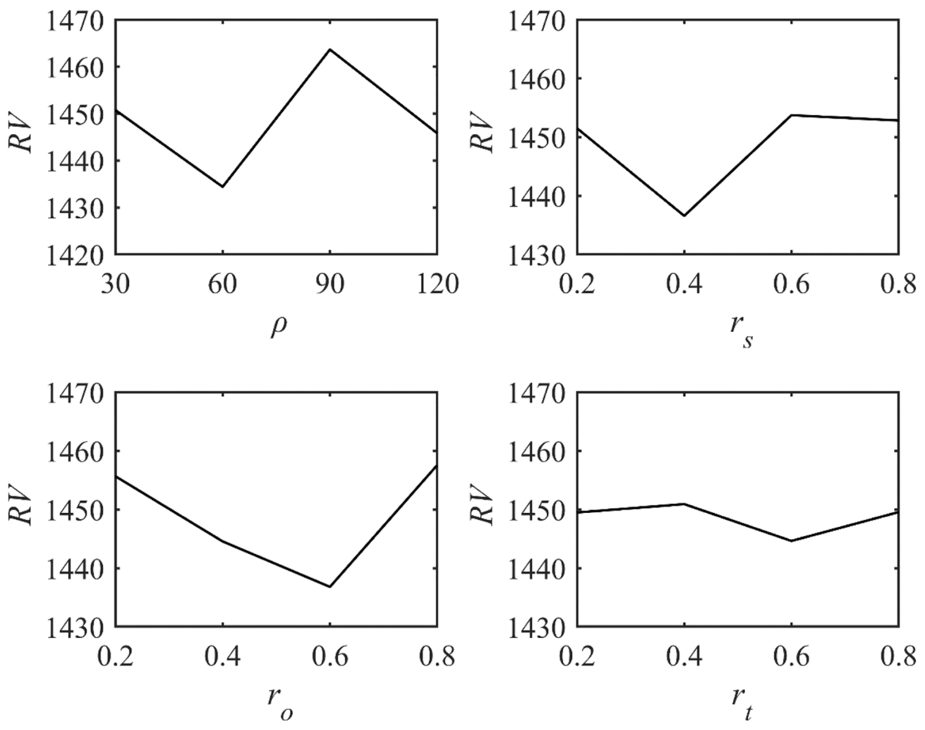

Fig. 3. Factor level trend of the EBSO for each user parameter.

Table 4
Information of an example with 3 factories, 2 machines, and 20 jobs.

Job	Processing time		Coordinate		Load	Time window	
	p_{1j}	p_{2j}	c_x	c_y	d_j	a_j	b_j
1	54	79	24	26	3	279	331
2	83	3	103	69	4	93	153
3	15	11	84	99	2	92	162
4	71	99	104	106	3	450	494
5	77	56	121	112	3	281	337
6	36	70	1	34	4	137	192
7	53	99	103	34	3	637	696
8	38	60	113	71	2	346	401
9	27	5	57	53	2	701	757
10	87	56	94	72	3	707	763
11	76	3	107	63	3	808	860
12	91	61	86	102	4	153	219
13	14	73	94	104	3	814	873
14	29	75	34	138	2	764	809
15	12	47	42	136	4	370	441
16	77	14	99	26	3	347	423
17	32	21	118	114	4	125	186
18	87	86	0	27	1	301	352
19	68	5	28	33	3	115	188
20	94	77	131	31	2	664	717

Table 5
Comparison results between the EBSO and CPLEX.

n	CPLEX	EBSO	
	AOV	AOV	Time (in seconds)
6	148.0000	247.8000	3.6500
10	666.7000	261.1500	7.0093
15	1184.3000	352.3450	10.0483
20	2317.9000	490.2600	13.1479

Besides, by calculating the *sRPD* values on all instances, it is found that the EBSO far surpasses the GA, MA, DABC, IG, and SS on 26, 28, 29, 30, 27 out of 30 instances respectively, and the average *sRPD* values of EBSO, GA, MA, DABC, IG, and SS are 0.0116, 6.3036, 3.4033, 5.4531, 4.6740, and 0.7787, respectively. By analyzing the experiment results of EBSO and its rivals, it can be concluded that the EBSO is a promising approach in tackling the investigated problem.

Table 6
User parameters of studied algorithms.

Algorithms	Parameter setting
GA	population size: 50, crossover rate: 1.0, mutation rate: 0.05
MA	population size: 100, crossover probability: 0.8, mutation probability: 0.05; SA probability: 0.05
DABC	number of food sources: 40, local search times: 15, limit: 25
IG	dropout rate: 0.1, destruction size: 3, temperature: 0.5
SS	population size: 5, number of factory assignment vectors: 2, probability in solution combination method: 0.15

To analyze the performance of EBSO with its peers more obviously, we group all the instances according to the number of factories, machines, and jobs, respectively. The experiment results are given in Table 8. It can be found that the EBSO outperforms its peers on all the considered groups regarding *aRPD*, *bRPD*, and *sRPD*. The average *aRPD* values of EBSO, GA, MA, DABC, IG, and SS are 0.0084, 8.2904, 5.1827, 9.1580, 7.4002, and 0.6105, while the average *bRPD* values are 0.0010, 1.6873, 1.3969, 2.4178, 1.6803, and 0.0219, respectively. Concerning the average *sRPD* metric, values of 0.0114, 6.5484, 3.5717, 5.6075, 4.8049, and 0.8361 are obtained by the aforementioned optimizers, respectively. Through the given analysis in this section, we can verify that the EBSO is an outstanding optimizer in settling the considered problem.

Furthermore, to exhibit the experiment results visually, boxplot graphs of EBSO and its competitors with different number of jobs, machines, and factories are displayed in Fig. 4. We can see that the results obtained by the EBSO are more concentrated and stable than those acquired by its peers.

In addition, statistical methods are utilized to analyze and show the statistical significance of the obtained results using the EBSO and the approaches in the comparison pool. This study adopts two methods, i.e.,

Table 7
Comparison results of the EBSO and its peers in terms of*aRPD*, *bRPD*, and *sRPD*.

Instance	GA			MA			DABC			IG			SS			EBSO		
	aRPD	*bRPD*	*sRPD*	*aRPD*	*bRPD*	*sRPD*	*aRPD*	*bRPD*	*sRPD*	*aRPD*	*bRPD*	*sRPD*	*aRPD*	*bRPD*	*sRPD*	*aRPD*	*bRPD*	*sRPD*
Ta001_3 & C1_2_1	0.0210	0.0130	0.0044	0.0225	0.0136	0.0045	0.0221	0.0130	0.0079	0.0116	0.0000	0.0096	0.0074	0.0001	0.0044	0.0005	0.0000	0.0015
Ta011_3 & C1_2_1	0.0484	0.0437	0.0030	0.0162	0.0091	0.0041	0.0170	0.0097	0.0052	0.0057	0.0000	0.0045	0.0052	0.0000	0.0036	0.0006	0.0000	0.0012
Ta001_5 & C1_2_1	0.0310	0.0153	0.0094	0.0308	0.0147	0.0087	0.0320	0.0189	0.0085	0.0237	0.0080	0.0110	0.0089	0.0002	0.0058	0.0000	0.0000	0.0000
Ta011_5 & C1_2_1	0.0181	0.0106	0.0030	0.0169	0.0072	0.0035	0.0193	0.0142	0.0030	0.0113	0.0008	0.0078	0.0061	0.0000	0.0036	0.0002	0.0000	0.0006
Ta001_7 & C1_2_1	0.0228	0.0109	0.0066	0.0182	0.0061	0.0057	0.0234	0.0107	0.0085	0.0248	0.0059	0.0140	0.0025	0.0000	0.0029	0.0021	0.0000	0.0048
Ta011_7 & C1_2_1	0.0061	0.0016	0.0025	0.0076	0.0000	0.0043	0.0104	0.0029	0.0045	0.0103	0.0000	0.0052	0.0028	0.0000	0.0023	0.0002	0.0000	0.0005
Ta031_3 & C1_6_1	0.0722	0.0490	0.0194	0.1403	0.0997	0.0310	0.1527	0.1176	0.0287	0.3757	0.3045	0.0602	0.0354	0.0020	0.0204	0.0000	0.0000	0.0000
Ta041_3 & C1_6_1	0.0846	0.0682	0.0082	0.1815	0.1420	0.0287	0.1903	0.1543	0.0153	0.1921	0.0762	0.0646	0.0359	0.0084	0.0163	0.0000	0.0000	0.0000
Ta051_3 & C1_6_1	0.1366	0.0972	0.0204	0.0734	0.0499	0.0131	0.0879	0.0517	0.0231	0.0513	0.0083	0.0367	0.0424	0.0023	0.0216	0.0000	0.0000	0.0000
Ta031_5 & C1_6_1	0.0026	0.0000	0.0042	0.1409	0.1243	0.0124	0.1467	0.1307	0.0111	0.1326	0.0761	0.0242	0.0259	0.0000	0.0102	0.0076	0.0000	0.0080
Ta041_5 & C1_6_1	0.0283	0.0088	0.0120	0.1800	0.1390	0.0237	0.2345	0.1773	0.0390	0.3188	0.2273	0.0492	0.0295	0.0000	0.0207	0.0002	0.0000	0.0009
Ta051_5 & C1_6_1	0.4024	0.3477	0.0258	0.2657	0.2196	0.0316	0.4560	0.3709	0.0550	0.3979	0.2194	0.0778	0.0333	0.0039	0.0195	0.0000	0.0000	0.0000
Ta031_7 & C1_6_1	0.0579	0.0410	0.0101	0.1024	0.0787	0.0117	0.1087	0.0917	0.0102	0.3442	0.2877	0.0303	0.0062	0.0000	0.0075	0.0026	0.0000	0.0044
Ta041_7 & C1_6_1	0.0614	0.0459	0.0086	0.1461	0.1132	0.0162	0.1573	0.1302	0.0144	0.3793	0.1673	0.6935	0.0144	0.0000	0.0154	0.0032	0.0000	0.0062
Ta051_7 & C1_6_1	0.0458	0.0288	0.0108	0.1159	0.0652	0.0269	0.1049	0.0745	0.0112	0.3368	0.2124	0.0729	0.0115	0.0000	0.0083	0.0005	0.0000	0.0020
Ta061_3 & C101	0.0795	0.0315	0.0271	0.2504	0.1654	0.0490	0.2838	0.1969	0.0533	0.2698	0.1901	0.0493	0.0052	0.0000	0.0095	0.0168	0.0000	0.0169
Ta071_3 & C101	0.3735	0.2679	0.0587	0.8743	0.6867	0.1121	0.9905	0.7709	0.1188	1.1098	0.8347	0.1721	0.0070	0.0000	0.0092	0.0104	0.0000	0.0191
Ta081_3 & C101	0.1148	0.0758	0.0257	0.2571	0.1740	0.0464	0.3764	0.2723	0.0696	0.0904	0.0013	0.0560	0.0491	0.0000	0.1386	0.0109	0.0000	0.0187
Ta061_5 & C101	0.0435	0.0205	0.0154	0.0000	0.0000	0.0000	0.1005	0.0670	0.0187	0.1696	0.1216	0.0286	0.1031	0.0556	0.0320	0.0765	0.0286	0.0280
Ta071_5 & C101	0.4096	0.2909	0.0651	0.4600	0.3017	0.0835	0.5155	0.3775	0.0794	0.5187	0.3125	0.1252	0.0178	0.0000	0.0239	0.0051	0.0000	0.0103
Ta081_5 & C101	53.9744	14.9600	22.3659	62.6455	24.9975	27.1358	14.6471	4.7000	5.9402	47.8359	15.1225	18.7593	0.5981	0.0000	1.0362	0.0537	0.0000	0.1294
Ta061_7 & C101	0.0086	0.0000	0.0097	0.0153	0.0000	0.0163	0.0882	0.0599	0.0214	0.1064	0.0506	0.1451	0.0149	0.0000	0.0145	0.0115	0.0000	0.0126
Ta071_7 & C101	0.0012	0.0000	0.0026	0.1900	0.1445	0.0221	0.1676	0.1348	0.0241	0.2466	0.1660	0.0456	0.0195	0.0000	0.0190	0.0258	0.0020	0.0111
Ta081_7 & C101	102.4031	24.8423	45.9617	23.3608	4.5417	12.3400	185.8871	55.8231	81.9948	114.5344	29.5115	49.0680	1.4843	0.0000	1.6241	0.0010	0.0000	0.0044
Ta091_3 & C2_2_1	1.4725	0.5558	0.5835	2.7025	0.9367	1.0701	2.7321	0.9985	1.0859	1.0147	0.3297	0.4328	0.1906	0.0200	0.1004	0.0000	0.0000	0.0000
Ta101_3 & C2_2_1	0.2483	0.1989	0.0316	0.4585	0.3514	0.0591	0.4768	0.3898	0.0594	0.5028	0.3940	0.0643	0.0351	0.0065	0.0175	0.0000	0.0000	0.0000
Ta091_5 & C2_2_1	31.5114	2.5721	66.2734	3.6523	0.2077	7.8957	6.2647	0.2728	14.1327	14.9700	0.6013	33.9115	2.6383	0.0000	7.9546	0.0086	0.0000	0.0170
Ta101_5 & C2_2_1	8.2291	2.1565	7.4319	15.8131	3.7396	16.8290	16.7495	3.9591	15.9705	3.1109	0.7409	2.9318	0.2168	0.0000	0.2949	0.0222	0.0000	0.0488
Ta091_7 & C2_2_1	14.2650	3.4774	11.2503	17.5394	4.2611	13.4349	16.3337	3.6051	13.5903	3.4600	0.6374	2.6071	2.5225	0.3747	2.4602	0.0000	0.0000	0.0000
Ta101_7 & C2_2_1	32.1075	1.0275	34.9389	22.5677	0.6098	22.7775	27.1757	0.8615	30.1874	31.5849	1.1606	30.6626	8.7675	0.1452	9.4648	0.0000	0.0000	0.0000
Average	8.2094	1.7086	6.3063	5.0748	1.4067	3.4033	9.1517	2.4619	5.4531	7.4047	1.7256	4.6740	0.5646	0.0206	0.7787	0.0087	0.0010	0.0116

Table 8
Comparison results of the EBSO and its peers for all the instances grouped by$f$, m, and n.

Instance		GA			MA			DABC			IG			SS			EBSO		
		aRPD	*bRPD*	*sRPD*	*aRPD*	*bRPD*	*sRPD*	*aRPD*	*bRPD*	*sRPD*	*aRPD*	*bRPD*	*sRPD*	*aRPD*	*bRPD*	*sRPD*	*aRPD*	*bRPD*	*sRPD*
f	3	0.2651	0.1401	0.0782	0.4977	0.2628	0.1418	0.5330	0.2975	0.1467	0.3624	0.2139	0.0950	0.0413	0.0039	0.0342	0.0039	0.0000	0.0057
	5	9.4650	2.0382	9.6206	8.3205	2.9751	5.2024	3.9166	1.0089	3.6258	6.7489	1.7430	5.5926	0.3678	0.0060	0.9401	0.0174	0.0029	0.0243
	7	14.8979	2.9475	9.2202	6.4063	0.9820	4.8656	23.0057	6.0794	12.5867	15.1028	3.2199	8.3344	1.2846	0.0520	1.3619	0.0047	0.0002	0.0046
m	5	0.0377	0.0201	0.0118	0.0801	0.0558	0.0155	0.1065	0.0785	0.0187	0.1620	0.1161	0.0414	0.0233	0.0064	0.0119	0.0131	0.0032	0.0085
	10	4.0234	0.6119	6.5226	2.1639	0.5791	1.8916	2.3027	0.5540	2.4261	1.8531	0.2794	3.1766	0.4575	0.0336	0.8858	0.0045	0.0002	0.0056
	20	21.9625	4.8594	12.3125	13.9509	3.8610	8.8066	27.3291	7.3892	14.9235	22.0495	5.2634	11.3033	1.2487	0.0175	1.4028	0.0098	0.0000	0.0226
n	20	0.0246	0.0158	0.0048	0.0187	0.0085	0.0051	0.0207	0.0116	0.0063	0.0146	0.0024	0.0087	0.0055	0.0001	0.0038	0.0006	0.0000	0.0014
	50	0.0991	0.0763	0.0133	0.1496	0.1146	0.0217	0.1821	0.1443	0.0231	0.2810	0.1755	0.1233	0.0260	0.0018	0.0156	0.0016	0.0000	0.0024
	100	17.4898	4.4988	7.6146	9.7837	3.4457	4.4228	22.5619	6.9336	9.8134	18.3202	5.1456	7.6055	0.2555	0.0062	0.3230	0.0235	0.0034	0.0278
	200	14.6390	1.6647	20.0849	10.4556	1.6844	10.3444	11.6221	1.6811	12.5044	9.1072	0.6440	11.7683	2.3951	0.0911	3.3821	0.0051	0.0000	0.0110
Average		8.2904	1.6873	6.5484	5.1827	1.3969	3.5717	9.1580	2.4178	5.6075	7.4002	1.6803	4.8049	0.6105	0.0219	0.8361	0.0084	0.0010	0.0114

Friedman test (Friedman, 1937) and Nemenyi post-hoc test (Pereira, Afonso, & Medeiros, 2015), to rank the overall performance of six reported algorithms. The methodology and analysis of the statistical methods are provided as follows:

1) To perform the Friedman test at a significance level of 0.05, the six studied algorithms according to the obtained *aRPD* values as decreasing order are sorted and assigned order values of 1, 2, 3, 4, 5, and 6. Based on their assigned indices, the average order values over all instances are calculated. The results are given in Table 9. By using the Friedman test, we can conclude that the performance of six algorithms is statistically significant and thus there are significant differences among the obtained optimization results using the considered optimizers.
2) To further exhibit the difference among the EBSO and its rivals, the Nemenyi post-hoc test at a significance level of 0.05 is employed to calculate a critical range value of the difference of average order values. The difference of average order values among the EBSO and GA, MA, DABC, and IG are 2.3666, 2.8666, 3.8333, and 3.4666, respectively, which are greater than the critical range value of 1.3766. In addition, the difference of average order values between the EBSO and SS is equal to 0.8666, indicating that their performance is statistically equivalent.

Besides, *t*-test (Li & Yin, 2013) at a significance level of 0.05 is considered to analyze the experiment results. The statistical results are listed in Table 9. The *t*-test results are nominated as '+', '-' and '~' when the EBSO is significantly better than, significantly worse than, or statistically equivalent to its peers, respectively. By observing the results via *t*-test, it can be observed that the EBSO has significantly better performance compared with the GA, MA, DABC, IG, and SS on 26, 28, 30, 30 and 23 instances. Furthermore, the EBSO shows significantly worse performance than the GA on 3 instances, and performs obviously worse than the MA and SS on only one instance, and for the rest instances, the EBSO exhibits statistical equivalence.

6.5. *Effectiveness of combining the EBSO with some special strategies*

To validate the performance of two components in the EBSO, i.e., heuristic rules and local search methods, this work develops two variants of the EBSO to verify their performance, i.e., EBSO-w/o-EW and EBSO-w/o-LS. They are defined as: 1) EBSO-w/o-EW is designed to verify the performance of three heuristic rules for generating an initial population in the EBSO. It does not use the three heuristic rules and randomly generates an initial population; 2) EBSO-w/o-LS is used to test the performance of local search methods, i.e., the SA and RLS, in the EBSO. It employs a swap-based approach to take the place of SA and RLS methods.

We construct the comparison experiments on 12 instances by comparing the EBSO with the EBSO-w/o-EW and EBSO-w/o-LS, respectively. Table 10 shows the *aRPD*, *bRPD*, and *sRPD* results of small-size instances (Ta011_3 & C1_2_1, Ta011_5 & C1_2_1 and Ta011_7 & C1_2_1 with 20 jobs), medium-size instances (Ta041_3 & C1_6_1, Ta041_5 & C1_6_1 and Ta041_7 & C1_6_1 with 50 jobs) and large-size instances (Ta071_3 & C101, Ta071_5 & C101 and Ta071_7 & C101 with 100 jobs; Ta091_3 & C2_2_1, Ta091_5 & C2_2_1 and Ta091_7 & C2_2_1 with 200 jobs).

The experiment results obtained by the EBSO and its variants are given in Table 10. Looking at Table 10, on the one hand, it can be observed that the EBSO reveals significantly better performance than the EBSO-w/o-EW on 10 out of 12 instances regarding *aRPD* metric. The EBSO can achieve better optimization results compared with the EBSO-w/o-EW in terms of *bRPD*, and it shows better performance than the EBSO-w/o-EW on 10 out of 12 instances regarding *sRPD*. Additionally, the average values of *aRPD*, *bRPD*, and *sRPD* of EBSO on all instances acquired by the EBSO are 0.0500, 0.0000, and 0.1027, respectively,

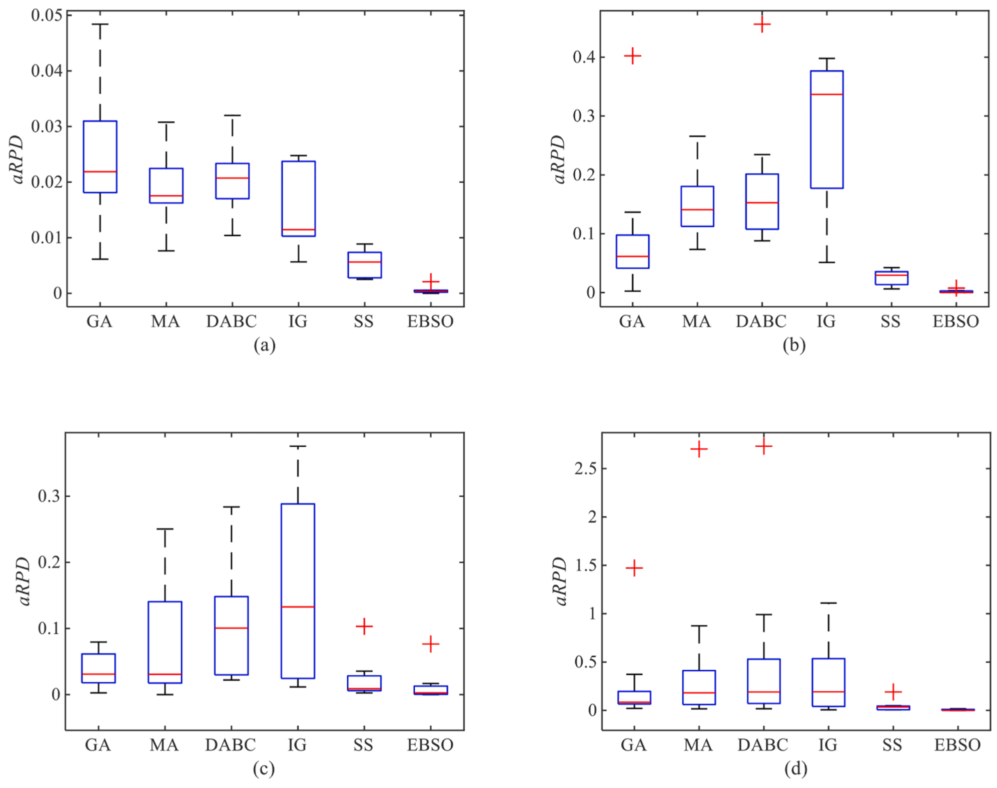

Fig. 4. Boxplot graphs of different groups on the instances regarding *aRPD*, (a)$n = 20$, (b)$n = 50$, (c)$m = 5$, and (d)$f = 3$.

Table 9
Comparison results of *t*-test and Friedman test for six algorithms.

Instance	GA	MA	DABC	IG	SS	EBSO
Ta001_3 & C1_2_1	4(+)	6(+)	5(+)	3(+)	2(+)	1
Ta011_3 & C1_2_1	6(+)	4(+)	5(+)	3(+)	2(+)	1
Ta001_5 & C1_2_1	5(+)	4(+)	6(+)	3(+)	2(+)	1
Ta011_5 & C1_2_1	5(+)	4(+)	6(+)	3(+)	2(+)	1
Ta001_7 & C1_2_1	4(+)	3(+)	5(+)	6(+)	2(~)	1
Ta011_7 & C1_2_1	3(+)	4(+)	6(+)	5(+)	2(+)	1
Ta031_3 & C1_6_1	3(+)	4(+)	5(+)	6(+)	2(+)	1
Ta041_3 & C1_6_1	3(+)	4(+)	5(+)	6(+)	2(+)	1
Ta051_3 & C1_6_1	6(+)	4(+)	5(+)	3(+)	2(+)	1
Ta031_5 & C1_6_1	1(-)	5(+)	6(+)	4(+)	3(+)	2
Ta041_5 & C1_6_1	2(+)	4(+)	5(+)	6(+)	3(+)	1
Ta051_5 & C1_6_1	5(+)	3(+)	6(+)	4(+)	2(+)	1
Ta031_7 & C1_6_1	3(+)	4(+)	5(+)	6(+)	2(~)	1
Ta041_7 & C1_6_1	3(+)	4(+)	5(+)	6(+)	2(+)	1
Ta051_7 & C1_6_1	3(+)	5(+)	4(+)	6(+)	2(+)	1
Ta061_3 & C101	3(+)	4(+)	6(+)	5(+)	1(-)	2
Ta071_3 & C101	3(+)	4(+)	5(+)	6(+)	1(~)	2
Ta081_3 & C101	4(+)	5(+)	6(+)	3(+)	2(~)	1
Ta061_5 & C101	2(+)	1(+)	4(+)	6(+)	5(+)	3
Ta071_5 & C101	3(-)	4(-)	5(+)	6(+)	2(+)	1
Ta081_5 & C101	5(+)	6(+)	3(+)	4(+)	2(+)	1
Ta061_7 & C101	1(~)	4(~)	5(+)	6(+)	3(~)	2
Ta071_7 & C101	1(-)	5(+)	4(+)	6(+)	2(~)	3
Ta081_7 & C101	4(+)	3(+)	6(+)	5(+)	2(+)	1
Ta091_3 & C2_2_1	4(+)	5(+)	6(+)	3(+)	2(+)	1
Ta101_3 & C2_2_1	3(+)	4(+)	5(+)	6(+)	2(+)	1
Ta091_5 & C2_2_1	6(+)	3(+)	4(+)	5(+)	2(+)	1
Ta101_5 & C2_2_1	4(+)	5(+)	6(+)	3(+)	2(+)	1
Ta091_7 & C2_2_1	4(+)	6(+)	5(+)	3(+)	2(+)	1
Ta101_7 & C2_2_1	6(+)	3(+)	4(+)	5(+)	2(+)	1
Average	3.6333	4.1333	5.1000	4.7333	2.1333	1.2667

while those attained by the EBSO-w/o-EW are 0.0905, 0.0014, and 0.1258, respectively. On the other hand, it can be seen that the EBSO outperforms the EBSO-w/o-LS on all instances regarding the *aRPD*, *bRPD*, and *sRPD*. Via calculating the average of *aRPD*, *bRPD*, and *sRPD* values on all instances, we can see that the EBSO obviously exceeds the EBSO-w/o-LS since the average results acquired by the EBSO are better than those obtained by the EBSO-w/o-LS. By analyzing the discussed results, it can be concluded that the heuristic rules and local search methods all play essential roles in enhancing the performance of EBSO, thus we choose them in this work.

According to the comparisons and analysis of the obtained results in aforementioned sections, it can be observed that the EBSO has strong search ability in both exploration and exploitation phases for solving the studied problem. In the EBSO, three heuristic rules are designed to generate initial individuals, which can obtain a high-quality population. Additionally, an order-based crossover approach is adopted to produce a new individual with two chosen individuals, which contributes to enhancing the EBSO's exploration ability. Besides, the local search approach is employed to refine a center individual and the RLS method is performed on the best new individual, which can strengthen the EBSO's exploitation ability. Therefore, the EBSO has better capacity to balance the global and local searches, and consequently achieves a better performance compared with its rivals for tackling the proposed optimization model.

7. Conclusions and future directions

This paper addressed an integrated distributed production and distribution problem where the production stage aimed at scheduling a distributed flow shop and the distribution stage focused on solving a multi-depot vehicle routing problem. A mixed integer programming model was formulated to minimize the total weighted earliness and tardiness. To handle the proposed problem, an enhanced brain storm optimization algorithm was designed by considering its features and searching operators. To validate the performance of the considered modified optimizer, experiments on a set of benchmark test instances were carried out. For the sake of comparison, a mathematical programming solver (i.e., CPLEX) and five state-of-the-art approaches existing in the literature were chosen. The attained optimization results

Table 10
Comparison results of the EBSO and its variants regarding *aRPD*, *bRPD*, and *sRPD*.

Instance	EBSO-w/o-EW			EBSO			EBSO -w/o-LS			EBSO		
	aRPD	*bRPD*	*sRPD*	*aRPD*	*bRPD*	*sRPD*	*aRPD*	*bRPD*	*sRPD*	*aRPD*	*bRPD*	*sRPD*
Ta011_3 & C1_2_1	0.0057	0.0000	0.0029	0.0001	0.0000	0.0005	0.0165	0.0088	0.0042	0.0000	0.0000	0.0000
Ta011_5 & C1_2_1	0.0076	0.0019	0.0035	0.0000	0.0000	0.0000	0.0205	0.0129	0.0040	0.0000	0.0000	0.0000
Ta011_7 & C1_2_1	0.0022	0.0000	0.0018	0.0002	0.0000	0.0006	0.0132	0.0060	0.0030	0.0000	0.0000	0.0000
Ta041_3 & C1_6_1	0.0386	0.0151	0.0158	0.0000	0.0000	0.0000	0.1843	0.1251	0.0288	0.0000	0.0000	0.0000
Ta041_5 & C1_6_1	0.0332	0.0000	0.0210	0.0003	0.0000	0.0013	0.2031	0.1447	0.0311	0.0000	0.0000	0.0000
Ta041_7 & C1_6_1	0.0199	0.0000	0.0197	0.0015	0.0000	0.0035	0.1615	0.1170	0.0184	0.0000	0.0000	0.0000
Ta071_3 & C101	0.0283	0.0000	0.0287	0.0073	0.0000	0.0208	0.3961	0.2025	0.0734	0.0000	0.0000	0.0000
Ta071_5 & C101	0.0449	0.0000	0.0349	0.0012	0.0000	0.0037	0.6489	0.4004	0.1239	0.0000	0.0000	0.0000
Ta071_7 & C101	0.0091	0.0000	0.0121	0.0095	0.0000	0.0126	0.4003	0.2865	0.0509	0.0000	0.0000	0.0000
Ta091_3 & C2_2_1	0.0837	0.0000	0.1141	0.0105	0.0000	0.0295	2.1186	0.8229	0.7797	0.0000	0.0000	0.0000
Ta091_5 & C2_2_1	0.2776	0.0000	0.5080	0.5578	0.0000	1.1098	48.2991	2.2376	109.1585	0.0000	0.0000	0.0000
Ta091_7 & C2_2_1	0.5345	0.0000	0.7476	0.0121	0.0000	0.0502	11.5473	2.4865	8.7681	0.0000	0.0000	0.0000
Average	0.0905	0.0014	0.1258	0.0500	0.0000	0.1027	5.3341	0.5709	9.9203	0.0000	0.0000	0.0000

verified that the proposed approach exhibited better performance in solving the considered problem. This work provides an effective approach to solve integrated distributed production and distribution problems where the production stage is modeled as a distributed flow shop scheduling problem, while the distribution stage is formulated as a multi-depot vehicle routing problem.

As further research, two directions can be considered as follows: 1) Formulating an integrated production and distribution model with consideration of uncertainties, e.g., machine breakdown, maintenance actives, and random processing time (Fu, Zhou, Guo, & Qi, 2021); and 2) Designing more effective swarm intelligence approaches to handle the studied problem (Gao, He, Huang, Duan, & Suganthan, 2020; Gao, Huang, Sadollah, & Wang, 2019).

CRediT authorship contribution statement

Yushuang Hou: Writing – original draft, Software, Formal analysis. **Yaping Fu:** Methodology, Conceptualization, Writing - review & editing. **Kaizhou Gao:** Supervision, Investigation, Validation. **Hui Zhang:** Software. **Ali Sadollah:** Writing - review & editing, Conceptualization.

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.

Acknowledgements

This work is supported in part by National Natural Science Foundation of China under Grant Nos. 62173356, 61703220 and 61603169; Shandong Province Outstanding Youth Innovation Team Project of Colleges and Universities of China under Grant No. 2020RWG011; China Postdoctoral Science Foundation Funded Project under Grant No. 2019T120569; Shandong Province Colleges and Universities Youth Innovation Talent Introduction and Education Program, the Faculty Research Grants (FRG) from Macau University of Science and Technology, and Shandong Provincial Key Laboratory for Novel Distributed Computer Software Technology.

References

Bo, V., Bortolini, M., Malaguti, E., Monaci, M., Mora, C., & Paronuzzi, P. (2021). Models and algorithms for integrated production and distribution problems. *Computers & Industrial Engineering, 154*, 107003. https://doi.org/10.1016/j.cie.2020.107003

Belo-Filho, M. A. F., Amorim, P., & Almada-Lobo, B. (2015). An adaptive large neighbourhood search for the operational integrated production and distribution problem of perishable products. *International Journal of Production Research, 53*(20), 6040–6058.

Baradaran, V., Shafaei, A., & Hosseinian, A. H. (2019). Stochastic vehicle routing problem with heterogeneous vehicles and multiple prioritized time windows: Mathematical modeling and solution approach. *Computers & Industrial Engineering, 131*(3), 187–199.

Bertsimas, D., & Tsitsiklis, J. (1993). Simulated annealing. *Statistical Science, 8*(1), 10–15.

Chen, Z.-L. (2010). Integrated production and outbound distribution scheduling: Review and extensions. *Operations Research, 58*(1), 130–148.

Deng, J., Wang, L., Wang, S.-Y., & Zheng, X.-L. (2016). A competitive memetic algorithm for the distributed two-stage assembly flow-shop scheduling problem. *International Journal of Production Research, 54*(12), 3561–3577.

Duan, H. B., Li, S. T., & Shi, Y. H. (2013). Predator–prey brain storm optimization for DC brushless motor. *IEEE Transactions on Magnetics, 49*(10), 5336–5340.

Duan, J. H., Meng, T., Chen, Q. D., & Pan, Q. K. (2018). In *An effective artificial bee colony for distributed lot-streaming flowshop scheduling problem* (pp. 795–806). Cham: Springer.

Devapriya, P., Ferrell, W., & Geismar, N. (2017). Integrated production and distribution scheduling with a perishable product. *European Journal of Operational Research, 259*(3), 906–916.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association, 32*(200), 675–701.

Feng, X., Chu, F., Chu, C., & Huang, Y. (2021). Crowdsource-enabled integrated production and transportation scheduling for smart city logistics. *International Journal of Production Research, 59*(7), 2157–2176.

Fernandez-Viagas, V., Perez-Gonzalez, P., & Framinan, J. M. (2018). The distributed permutation flow shop to minimise the total flowtime. *Computers & Industrial Engineering, 118*(2), 464–477.

Fu, Y., Hou, Y., Wang, Z., Wu, X., Gao, K., & Wang, L. (2021). Distributed scheduling problems in intelligent manufacturing systems. *Tsinghua Science and Technology, 26*(5), 625–645.

Fu, Y., Wang, H., Tian, G., Li, Z., & Hu, H. (2019). Two-agent stochastic flow shop deteriorating scheduling via a hybrid multi-objective evolutionary algorithm. *Journal of Intelligent Manufacturing, 30*(5), 2257–2272.

Fu, Y., Zhou, MengChu, Guo, X., & Qi, L. (2020). Scheduling dual-objective stochastic hybrid flow shop with deteriorating jobs via bi-population evolutionary algorithm. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 50*(12), 5037–5048.

Fu, Y. P., Zhou, M. C., Guo, X. W., & Qi, L. (2021). Multiverse optimization algorithm for stochastic biobjective disassembly sequence planning subject to operation failures. IEEE Transactions on Systems, Man, and Cybernetics: Systems. https://doi.org/10.1109/TSMC.2021.3049323.

Fu, Y. P., Li, H. B., Huang, M., & Xiao, H. (2021). Bi-objective modelling and optimization for stochastic two-stage open shop scheduling problems in the sharing economy. *IEEE Transactions on Engineering Management*. https://doi.org/10.1109/TEM.2021.3095954

Fu, L.-L., Aloulou, M. A., & Triki, C. (2017). Integrated production scheduling and vehicle routing problem with job splitting and delivery time windows. *International Journal of Production Research, 55*(20), 5942–5957.

Gao, K. Z., Huang, Y., Sadollah, A., & Wang, L. (2019). A review of energy-efficient scheduling in intelligent production systems. *Complex & Intelligent Systems, 6*(3), 237–249.

Gao, K. Z., He, Z. M., Huang, Y., Duan, P. Y., & Suganthan, P. N. (2020). A survey on meta-heuristics for solving disassembly line balancing, planning and scheduling problems in remanufacturing. *Swarm and Evolutionary Computation, 57*, 100719. https://doi.org/10.1016/j.swevo.2020.100719

Homberger, J., & Gehring, H. (2005). A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research, 162*(1), 220–238.

Han, Y., Li, J., Sang, H., Liu, Y., Gao, K., & Pan, Q. (2020). Discrete evolutionary multi-objective optimization for energy-efficient blocking flow shop scheduling with setup time. *Applied Soft Computing, 93*, 106343. https://doi.org/10.1016/j.asoc.2020.106343

Han, Y. Y., Gong, D. W., Jin, Y. C., & Pan, Q. K. (2017). Evolutionary multiobjective blocking lot-streaming flow shop scheduling with machine breakdowns. *IEEE Transactions on Cybernetics, 49*(1), 184–197.

Han, D. Y., Yang, Y. J., Wang, D. J., Cheng, T. C. E., & Yin, Y. Q. (2019). Integrated production, inventory, and outbound distribution operations with fixed departure times in a three-stage supply chain. *Transportation Research Part E: Logistics and Transportation Review, 125*(3), 334–347.

Hao, J.-H., Li, J.-Q., Du, Y.u., Song, M.-X., Duan, P., & Zhang, Y.-Y. (2019). Solving distributed hybrid flowshop scheduling problems by a hybrid brain storm optimization algorithm. *IEEE Access, 7*, 66879–66894.

Jing, X.-L., Pan, Q.-K., Gao, L., & Wang, Y.-L. (2020). An effective Iterated Greedy algorithm for the distributed permutation flowshop scheduling with due windows. *Applied Soft Computing, 96*, 106629. https://doi.org/10.1016/j.asoc.2020.106629

Kurdi, M. (2020). A memetic algorithm with novel semi-constructive evolution operators for permutation flowshop scheduling problem. *Applied Soft Computing, 94*, 106458. https://doi.org/10.1016/j.asoc.2020.106458

Ke, L. (2018). A brain storm optimization approach for the cumulative capacitated vehicle routing problem. *Memetic Computing, 10*(4), 411–421.

Kabadayi, N. (2021). *A Memetic Algorithm for Integrated Production Distribution Problem in a Supply Chain. In Demand Forecasting and Order Planning in Supply Chains and Humanitarian Logistics* (pp. 198–224). IGI Global.

Kazemi, H., Mahdavi Mazdeh, M., Rostami, M., & Heydari, M. (2020). The integrated production-distribution scheduling in parallel machine environment by using improved genetic algorithms. *Journal of Industrial and Production Engineering, 38*(3), 1–14.

Kesen, S. E., & Bektaş, T. (2019). Integrated production scheduling and distribution planning with time windows. In *Lean and Green Supply Chain Management* (pp. 231–252). Cham: Springer.

Karaoğlan, İ., & Kesen, S. E. (2017). The coordinated production and transportation scheduling problem with a time-sensitive product: A branch-and-cut algorithm. *International Journal of Production Research, 55*(2), 536–557.

Kergosien, Y., Gendreau, M., & Billaut, J.-C. (2017). A benders decomposition-based heuristic for a production and outbound distribution scheduling problem with strict delivery constraints. *European Journal of Operational Research, 262*(1), 287–298.

Liu, L., & Liu, S. (2020). Integrated production and distribution problem of perishable products with a minimum total order weighted delivery time. *Mathematics, 8*(2), 146.

Lacomme, P., Moukrim, A., Quilliot, A., & Vinot, M. (2018). Supply chain optimisation with both production and transportation integration: Multiple vehicles for a single perishable product. *International Journal of Production Research, 56*(12), 4313–4336.

Li, J.-Q., Song, M.-X., Wang, L., Duan, P.-Y., Han, Y.-Y., Sang, H.-Y., & Pan, Q.-K. (2020). Hybrid artificial bee colony algorithm for a parallel batching distributed flow-shop problem with deteriorating jobs. *IEEE Transactions on Cybernetics, 50*(6), 2425–2439.

Li, X. T., & Yin, M. H. (2013). An opposition-based differential evolution algorithm for permutation flow shop scheduling based on diversity measure. *Advances in Engineering Software, 55*(1), 10–31.

Moons, S., Ramaekers, K., Caris, A., & Arda, Y. (2017). Integrating production scheduling and vehicle routing decisions at the operational decision level: A review and discussion. *Computers & Industrial Engineering, 104*(1), 224–245.

Mohammadi, S., Cheraghalikhani, A., & Ramezanian, R. (2018). A joint scheduling of production and distribution operations in a flow shop manufacturing system. *Scientia Iranica, 25*(2), 911–930.

Mohammadi, S., Al-e-Hashem, S. M., & Rekik, Y. (2020). An integrated production scheduling and delivery route planning with multi-purpose machines: A case study from a furniture manufacturing company. *International Journal of Production Economics, 219*(1), 347–359.

Mousavi, M., Hajiaghaei–Keshteli, M., & Tavakkoli–Moghaddam, R. (2020). Two calibrated meta-heuristics to solve an integrated scheduling problem of production and air transportation with the interval due date. *Soft Computing, 24*(21), 16383–16411.

Mao, J. Y., Pan, Q. K., Miao, Z. H., & Gao, L. (2020). An effective multi-start iterated greedy algorithm to minimize makespan for the distributed permutation flowshop scheduling problem with preventive maintenance. *Expert Systems with Applications, 169*(1), 114495. https://doi.org/10.1016/j.eswa.2020.114495

Montgomery, D. C. (2005). *Design and analysis of experiments*. Arizona: John Wiley & Sons.

Meinecke, C., & Scholz-Reiter, B. (2014). A heuristic for the integrated production and distribution scheduling problem. *International Science Index, 8*(2), 290–297.

Pereira, D. G., Afonso, A., & Medeiros, F. M. (2015). Overview of Friedman's test and post-hoc analysis. *Communications in Statistics-Simulation and Computation, 44*(10), 2636–2653.

Pan, Q.-K., Tasgetiren, M. F., & Liang, Y.-C. (2008). A discrete differential evolution algorithm for the permutation flowshop scheduling problem. *Computers & Industrial Engineering, 55*(4), 795–816.

Pan, Q. K., Gao, L., Wang, L., Liang, J., & Li, X. Y. (2019). Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem. *Expert Systems with Applications, 124*(3), 309–324.

Pan, Q. K., Gao, L., & Wang, L. (2020). An effective cooperative Co-Evolutionary algorithm for distributed flowshop group scheduling problems. *IEEE Transactions on Cybernetics, 12*, 1–14. https://doi.org/10.1109/TCYB.2020.3041494

Ramezanian, R., Mohammadi, S., & Cheraghalikhani, A. (2017). Toward an integrated modeling approach for production and delivery operations in flow shop system: Trade-off between direct and routing delivery methods. *Journal of Manufacturing Systems, 44*(4), 79–92.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research, 35*(2), 254–265.

Schmid, V., Doerner, K. F., Hartl, R. F., & Salazar-González, J.-J. (2010). Hybridization of very large neighborhood search for ready-mixed concrete delivery problems. *Computers & Operations Research, 37*(3), 559–574.

Shi, H. Y. (2011). Brain storm optimization algorithm. In *International conference in swarm intelligence* (pp. 303–309). Berlin Heidelberg: Springer.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research, 64*(2), 278–285.

Ullrich, C. A. (2013). Integrated machine scheduling and vehicle routing with time windows. *European Journal of Operational Research, 227*(1), 152–165.

Umbarkar, A. J., & Sheth, P. D. (2015). Crossover operators in genetic algorithms: a review. ICTACT Journal on Soft Computing, 6(1), 1083–1092.

Wang, G., Gao, L., Li, X., Li, P., & Tasgetiren, M. F. (2020). Energy-efficient distributed permutation flow shop scheduling problem using a multi-objective whale swarm algorithm. *Swarm and Evolutionary Computation, 57*(5), 100716. https://doi.org/10.1016/j.swevo.2020.100716

Wang, J. J., & Wang, L. (2020). A bi-population cooperative memetic algorithm for distributed hybrid flow-shop scheduling. *IEEE Transactions on Emerging Topics in Computational Intelligence*. https://doi.org/10.1109/TETCI.2020.3022372

Wang, L., & Peng, Z. P. (2020). Solving energy-efficient distributed job shop scheduling via multi-objective evolutionary algorithm with decomposition. *Swarm and Evolutionary Computation, 58*(5), 100745.

Wang, D. J., Zhu, J. Q., Wei, X. W., Cheng, T. C. E., Yin, Y. Q., & Wang, Y. Z. (2019). Integrated production and multiple trips vehicle routing with time windows and uncertain travel times. *Computers & Operations Research, 103*(3), 1–12.

Yağmur, E., & Kesen, S. E. (2020). A memetic algorithm for joint production and distribution scheduling with due dates. *Computers & Industrial Engineering, 142*, 106342. https://doi.org/10.1016/j.cie.2020.106342

Yan, C., Banerjee, A., & Yang, L. (2011). An integrated production–distribution model for a deteriorating inventory item. *International Journal of Production Economics, 133*(1), 228–232.

Zhang, X., Li, X. T., & Yin, M. H. (2020). An enhanced genetic algorithm for the distributed assembly permutation flowshop scheduling problem. *International Journal of Bio-Inspired Computation, 15*(2), 113–124.

Zhang, G. T., Zhang, X., Sun, H., & Zhao, X. Y. (2021). Three-echelon closed-loop supply chain network equilibrium under cap-and-trade regulation. *Sustainability, 13*(11), 6472.

Zheng, J., Wang, L., & Wang, J.-J. (2020). A cooperative coevolution algorithm for multi-objective fuzzy distributed hybrid flow shop. *Knowledge-Based Systems, 194*, 105536. https://doi.org/10.1016/j.knosys.2020.105536

Zhao, F., Zhao, L., Wang, L., & Song, H. (2020). An ensemble discrete differential evolution for the distributed blocking flowshop scheduling with minimizing makespan criterion. *Expert Systems with Applications, 160*, 113678. https://doi.org/10.1016/j.eswa.2020.113678