Computers & Industrial Engineering 158 (2021) 107378

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Flowshop scheduling with sequence dependent setup times and batch delivery in supply chain

Humyun Fuad Rahman [a,*], Mukund Nilakantan Janardhanan [b], Liam Poon Chuen [c], S. G. Ponnambalam [d]

[a] School of Engineering and IT, University of New South Wales, Canberra, ACT 2620, Australia
[b] Mechanics of Materials Research Group, School of Engineering, University of Leicester, Leicester LE1 7RH, UK
[c] Faculty of Mechanical and Manufacturing Engineering, University Malaysia Pahang, Pahang, Malaysia
[d] School of Mechanical Engineering, Vellore Institute of Technology, Vellore 632014, India

ARTICLE INFO

Keywords:
Flow shop scheduling
Supply chain
Differential evolution
Moth flame optimization
Lévy-flight moth flame optimization

ABSTRACT

With the emergence of advanced manufacturing and Industry 4.0 technologies, there is a growing interest in coordinating the production and distribution in supply chain management. This paper addresses the production and distribution problems with sequence dependent setup time for multiple customers in flow shop environments. In this complex decision-making problem, an efficient scheduling approach is required to optimize the trade-off between the total cost of tardiness and batch delivery. To achieve this, three new metaheuristic algorithms such as Differential Evolution with different mutation strategy variation and a Moth Flame Optimization, and Lévy-Flight Moth Flame Optimization algorithm are proposed and presented. In addition, a design-of-experiment method is used to identify the best possible parameters for the proposed approaches for the problem under study. The proposed algorithms are validated on a set of problem instances. The variants of differential evolution performed better than the other compared algorithms and this demonstrates the effectiveness of the proposed approach. The algorithms are also validated using an industrial case study.

1. Introduction

Due to globalization of supply chains and advancement in information systems, the modern manufacturing industry has been experiencing pressure to adapt globalization and improve customer satisfaction to maintain their competence in the market (Cheng, Leung, & Li, 2015; Choi, 2015; Rahman & Nielsen, 2019). In order to maintain this competitive edge, manufacturers are shifting their production systems from make-to-stock production system to make-to-order (MTO) production system, where customers have the flexibility to order customized products. More specifically, automobile, computers and electronics, telecommunications, automobile, machinery, furniture, food processing and textile industries are following this trend (Chen, Huang, Luo, & Wang, 2015; Rahman, Sarker, & Essam, 2017; Wang, Luo, Liu, & Yue, 2017). With the development of e-commerce business, customers can order their products using MTO production system for products (e.g., furniture, clothes, cars) of their choice from different locations and the manufacturers will be required to manufacture and deliver the products to the customers in a timely fashion. This problem is more realistic under difficult situations (e.g., COVID-19) in which customers were unable to go for shopping physically. So, in such scenarios having an efficient MTO based-manufacturing supply chain is essential.

In MTO industries, production process starts after receiving orders from the customers and distribute the products to the customers (Wang et al., 2017). A customer sets a due date for order(s) and expects to receive the products on or before that time. To maintain a desired customer satisfaction level, manufacturers are recommended to deliver the products as soon as they are completed in a production line to avoid any tardiness (i.e., deliver the product after due date). However, it is often found that it is more economical to dispatch some products in batches and to minimize the costs associated to the customer delivery although it affects the tardiness in practice (Liu, Zhou, & Yang, 2017; Mazdeh, Rostami, & Namaki, 2013). Hence, the production managers have to coordinate both production scheduling and delivering the final product to the customers in a timely and cost-effective manner (Chen et al., 2015). In other words, there should be a tradeoff between the

* Corresponding author.
E-mail addresses: humyun.fuad@adfa.edu.au (H.F. Rahman), mukund.janardhanan@leicester.ac.uk (M.N. Janardhanan), pc.liam12@gmail.com (L. Poon Chuen), ponnambalam.g@vit.ac.in (S.G. Ponnambalam).

https://doi.org/10.1016/j.cie.2021.107378
Received 23 April 2020; Received in revised form 23 March 2021; Accepted 27 April 2021
Available online 30 April 2021

tardiness cost and delivery costs when making managerial decisions. However, most of the scheduling literature for MTO systems for supply chain only concerns about scheduling of products and avoids the decision-making process of customer delivery in the supply chain. As a result, the solution generated by those approaches may result in high delivery costs and the production business may run with suboptimal solutions.

The permutation flow shop scheduling problem (PFSP) is a well-known scheduling problem in many manufacturing industries (Rahman, Janardhanan, & Nielsen, 2019; Rahman, Sarker, & Essam, 2015a; Rahman, Sarker, & Essam, 2018). Furthermore, a realistic and interesting extension of PFSPs is the PFSPs with sequence dependent setup times (SDSTs), where each product needs additional setup time in each machine before processing another product in the same machine (Ruiz, Maroto, & Alcaraz, 2005).

Notably, these studies considered a restricted assumption that cost associated with product delivery is negligible (Ruiz & Maroto, 2005). Therefore, like other production layout (e.g., single machine or multi-machine production line), existing scheduling approaches ignores the impact of delivery costs in flow shop based MTO industries. Although there are some studies available which considers integrating production with distribution in PFSPs for MTO industries, however to the authors' best knowledge, studies with the same problem with SDST is not reported from the literature.

In order to address the academic research gap and practical challenges, this paper presents the following contributions to the field of flow shop scheduling problems: firstly, a mixed integer non-linear programming model (MINLP) is developed for the considered problem, which is an extension of the model proposed by Wang et al. (2017) by considering SDST to the problem. Secondly, three meta-heuristic-based solution approaches have been proposed to solve the proposed problem. The proposed algorithms are tested on a real-life case study to demonstrate the effectiveness of the algorithms.

Since classical PFSP has proven to be strongly NP-hard (Ruiz & Maroto, 2005), the problem under study, is considerably more difficult to solve. Over the last few decades, exact techniques, such as branch and bound algorithm (Bansal, 1977) and integer programming approach (Selen & Hott, 1986), can only solve classical PFSPs with small number of machines effectively. Thus, researchers have focused on developing meta-heuristic techniques for solving classical PFSPs. The detailed literature review of the meta-heuristic algorithms presented in (Ruiz & Maroto, 2005) for solving classical PFSPs and its variants reveals two main concerns in the literature.

- No single algorithm is suitable for solving all type of PFSPs problems with reasonable solution quality and computational times.
- Even though there are some algorithms that has shown enhanced performance, their solutions may be far from the optimal or lower bound solutions (if known).

The first point relates to the no-free-lunch theorem for optimization, which is defined as the performance of all search algorithms perform exactly the same when they search for an extremum of a cost function and when averaged over all possible cost functions (Ho & Pepyne, 2002; Wolpert & Macready, 1997). The second point is the consequence of NP-hard problems. As neither of these concerns can be satisfied, there is a need to develop meta-heuristic techniques that may be able to solve PFSPs effectively, since it has an outstanding track record in solving complex scheduling problems (Li, Janardhanan, Tang, & Ponnambalam, 2019; Rahman & Nielsen, 2019; Rahman, Sarker, & Essam, 2015b). Furthermore, there are no alternative algorithms available in the literature to solve the problem under study. Therefore, major contribution of this paper is to utilize metaheuristic algorithms such as Differential Evolution (DE), Moth Flame Optimization (MFO) and Lévy-Flight Moth Flame Optimization (LFMFO) algorithms to solve the problem under study. The proposed algorithms aim to optimize the total cost by scheduling products while considering SDST in flow shop so that the products are delivered with optimized total costs. As there are no benchmark data available for the problem under study, the performance of the proposed algorithms is evaluated using a set of realistic problem instances and a case study from a sanitaryware industry.

The reminder of the paper is organized as follows. The next section presents the overview of the background study of the research topics related to this research. Section 3 formulates the studied problem. The solution approaches are discussed in Section 4 for solving the model, while Section 5 presents the numerical studies and Section 6 presents a case study. Finally, in Section 7, the research findings are concluded along with future research directions are discussed.

2. Related works

Despite their widespread applications in manufacturing, until recently, integrated production scheduling with distribution problems in MTO industries for supply materials have not been widely studied. However, motivated by the importance of integrating this decision-making process, some studies have been dedicated to focus on production scheduling problems with batch delivery for different production layout (e.g., single machine and multi-machine) in recent years. In this section, a review on production scheduling approaches for MTO based supply chain is presented, with a special emphasis on PFSPs.

Most of the related works on production and batch delivery scheduling problems has focused on the single-machine and the parallel-machine environments. For the first time, Cheng and Kahlbacher (1993) introduced the single machine production and delivery scheduling problem in supply chain to the scientific community. They made a computational study on that problem and proved that minimization of the total cost of holding and distribution is NP-hard, but polynomial solvable for equal weights. Yang (2000) studied the extension of this problem with generalized due dates. Hamidinia, Khakabimamaghani, Mazdeh, and Jafari (2012) developed an integer programming approach and a genetic algorithm (GA) with the objective of minimization of the sum of earliness, tardiness, inventory holding, and distribution costs. The solutions outperformed the traditional algorithm. More recently, Ahmadizar and Farhadi (2015) extended this problem by considering product release dates and due date time windows, they presented a non-linear mathematical model and also provided the mathematical model by linearizing the objective functions and constraints. Mazdeh et al. (2013) proposed a branch and bound algorithm with a local search approach to solve the single-machine batch delivery scheduling problem and proposed a non-linear programming model. For parallel machine environments, Wang and Cheng (2000) developed a dynamic programming (DP) algorithm to optimized the total cost of total flow time and distribution cost. Hybridized DP with shortest processing time rule for solving identical parallel machine scheduling problem with batch delivery was proposed later (Wan & Zhang, 2014). Liu and Lu (2016) studied two identical-parallel-machine scheduling problems with batch delivery with machine availability constraints. The objective of minimizing the time by which all the jobs gets delivered was considered. Guo, Zhang, Leung, and Shi (2016) proposed a bi-level evolutionary algorithm to integrating production and distribution decisions for unrelated parallel-machine environment and also proposed a bi-level mixed integer non-linear program. The proposed approach is superior to other algorithms used for comparisons. To simplify the problem, most of the research studies on single machine and parallel machine production and distribution scheduling problems assumed a single customer in supply chain. Considering the reality, this is however a restrictive assumption and therefore some recent studies considered the production and delivery scheduling problems with multiple customers in supply chain, such as (Mazdeh et al., 2013) (Ahmadizar & Farhadi, 2015; Cakici, Mason, & Kurz, 2012; Hamidinia et al., 2012; Selvarajah & Zhang, 2014).

The classical PFSPs have been widely studied in the literature. In

1954, Johnson proposed an optimal algorithm for solving two machine and special three machine classical PFSPs (Johnson, 1954). It is proven that more than three machine classical PFSPs is NP-hard (Garey, Johnson, & Sethi, 1976). To solve large problems, researchers have proposed heuristic and metaheuristics algorithms (Kizilay, Tasgetiren, Pan, & Gao, 2019; Ruiz & Maroto, 2005; Ruiz, Maroto, & Alcaraz, 2006; Zobolas, Tarantilis, & Ioannou, 2009). In classical PFSPs, n products (i.e., jobs) are processed by m machines, where each machine has to follow the same operational order of products and each product. Over the past few decades, most of the studies about PFSPs focus on scheduling a set of products on a set of machines while minimizing the makespan as the objective (Öztop, Tasgetiren, Eliiyi, Pan, & Kandiller, 2020). Makespan is the time difference between the start time of the first product in the first machine and the finishing time of the last product in the last machine.

SDST is an interesting extension of PFSPs, which is studied as an extension of classical PFPS or as a hybrid PFSPs (each stage of production line has multiple machines) (Allahverdi, 2015; Li, Li, Gao, & Meng, 2020). The research of SDST-PFSP scheduling is widely at the academic level, since it has also applied in several engineering fields, e.g., steel industries (Han, Tang, Zhang, & Cao, 2020; Sbihi, Bellabdaoui, & Teghem, 2014; Sbihi & Chemangui, 2018), electronics industries (Hidri & Gharbi, 2017), digital dental laboratory (Valizadeh, Fatahi Valilai, & Houshmand, 2019), paper industry (Ruiz & Maroto, 2006), etc. Although these studies consider solving production scheduling problem with SDSTs in flow shop environment, they do not consider the delivery problem in their study.

As discussed in the previous section, these studies assume that the time or cost of delivery is zero in MTO systems. To address this gap, recently some researchers have focused on addressing production scheduling with batch delivery problem in flow shop environments. Mazdeh et al. (2013) studied an integrated production and distribution scheduling problem for two machine flow shops to minimize the total sum of weighted number of tardy products and delivery cost. Chen et al. (2015) proposed genetic algorithm for solving a synchronized scheduling problem of production simultaneity and shipment punctuality in a two-stage assembly flowshop system. Kazemi, Mazdeh, and Rostami (2017) proposed a hybrid imperialist competitive algorithm for solving two stage assembly scheduling problem within batch delivery systems. In their research, they considered variable processing and assembly times for all products for two customers. Wang et al. (2017) proposed two heuristics, namely Earliest Due Date (EDD), minimum slack-based heuristic, and a GA hybridized by variable neighborhood search and teaching–learning algorithm to determine the minimal tardiness cost and minimal distribution cost for batch delivery to multiple customers. Mishra and Shrivastava (2018) proposed a Jaya algorithm and Teaching-Learning Based Optimization algorithm for a flow shop scheduling problem where penalty cost and inventory cost are taken into consideration. Other than that, they consider the machine sequence independent batch set up time within their model to simulate the automotive industry.

In summary, from the related works on scheduling literature on MTO industries, it can be seen that the above-mentioned studies have limitations and restrictive assumptions. These assumptions are as follows: (1) scheduling approaches on classical PFSPs ignores the delivery costs in their models, and (2) research works on single machine, parallel machine, and flow shop environments did not consider SDSTs in the models. As a result, these assumptions isolate the studies on MTO based production systems from real-life flow-shop based production environments, and therefore in this study, the main aim is to address these limitations by considering the above-mentioned factors. From the literature, it can be also seen that authors have applied both non-linear as well as linear programming models to solve the problem of this type. Since non-linear and linear models are reported to be very time consuming, use of heuristics and metaheuristic strategy is very common in the literature. Based on these above-mentioned reasons, metaheuristics have been applied in this study and details of these are presented in the following sections.

3. Problem description and mathematical model

In this section, the production scheduling and batch delivery problem in flow shop considering SDSTs are described and necessary assumptions are discussed.

3.1. Problem statement

SDST-PFSP with batch delivery in supply chain is a process that involves integrated decision making for both production and delivery in supply chain. As discussed in Section 2, several attempts have been made to model the PFSPs without considering the impact of distribution aspect in the model. However, one may argue that the effectiveness of a production scheduling model depends on simplicity in validity of the model and universality of the model itself (Sbihi et al., 2014). Thus, there is no universal methodology that represents that SDST-PFSP with batch delivery in supply chain since each problem depends on the considered problem case. Moreover, the existing models developed for solving PFSPs and SDST-PFSP are more restrictive with respect to the real-life manufacturing supply chain environments. Therefore, in this study, authors have developed a generic production planning model for the SDST-PFSP with batch delivery in supply chain, where: (1) production scheduling part of the problem can be seen as PFSP for scheduling products based on customer order while considering the setup times (both initial and sequence dependent setup times), and (2) distribution part of the problem deals with the decision making on how to form batches (for same or different customers) based on due date of customer and their distance from the factory and other customers in the system. Both parts of the problem should be dealt in an integrated manner to ensure that total costs in supply chain in minimized.

3.2. Problem Definition

PFSPs with batch delivery to multiple customers' in supply chain addresses to the integration of production and distribution cost within the permutation flow shop environment. The proposed algorithms will solve two interdependent sub problems that are the permutation flow shop scheduling in the production stage and the multiple customers batch delivery system in the distribution stage with the minimum cost. To define the permutation flow shop problem with batch delivery to multiple customers, the constraints and assumption made are as follow:

- **Within production stage:** All products from customers are released simultaneously to the permutation flow shop at time, $t = 0$. Each machine can only process one product at a time and have identical product processing sequence for every subsequent machine. Every product must be processed on every machine before considered as completed. Pre-emptive production is not allowed. The production and delivery operations run under ideal conditions, i.e. there is no interruption during production.
- **Within distribution stage:** There are enough vehicles to dispatch the completed products to the customers. Each batch delivery does not have a quantity or distance limit. Each delivery is sent to a particular customer only. Customers are geographically located in different places and the delivery time and cost from the production floor to a customer are fixed.

In this research, the sequencing of products and the grouping of products to be distributed to the customer through batch delivery are the two operational decisions that are to be integrated to obtain the optimal or best results of minimizing the total cost.

3.3. Mathematical model of the problem

To further describe and explain the system to study the effect of each parameters to the objective function of the algorithm, a mixed integer non-linear programming (MINLP) model including the parameters and decision variables of the studied PFSP are defined in Table 1.

Based on the presented indices, parameters and decision variables in Table 1, the objective function to minimize the total cost of tardiness and delivery cost is shown in Equation (1). This helps to obtain the optimal production and distribution decision schedule.

$$TotalCost = min \sum_{b=1}^{l} \sum_{i=1}^{f} \sum_{j=1}^{n} \alpha_{ij} \times max\{0, y_{ib} \times x_{\pi_j b} \times (AT_{ib} - d_{ij})\} + \sum_{b=1}^{l} \times \sum_{i=1}^{f} (y_{ib} \times DC_{ib}) \tag{1}$$

subject to:

$$C_{\pi_1 1} \geq P_{\pi_1 1} \tag{2}$$

$$C_{\pi_j 1} \geq C_{\pi_{j-1} 1} + Q_{\pi_1 jk} + S_{\pi_j jhk} + P_{\pi_j 1}, j = 2, 3, ..., n \tag{3}$$

$$C_{\pi_1 k} \geq C_{\pi_1 (k-1)} + P_{\pi_1 k} + Q_{\pi_1 jk} + S_{\pi_j jhk}, k = 2, 3, ..., m \tag{4}$$

$$C_{\pi_j k} \geq \max\{C_{\pi_{j-1} k}, C_{\pi_j (k-1)}\} + P_{\pi_j k} + Q_{\pi_1 jk} + S_{\pi_j jhk}, j, h = 2, 3, ..., n; k = 2, 3, ..., m \tag{5}$$

$$\sum_{b=1}^{l} x_{\pi_j b} = 1, \forall j = 1, 2, \cdots, n. \tag{6}$$

$$\sum_{b=1}^{l} y_{ib} = 1, \forall i = 1, 2, \cdots, f. \tag{7}$$

$$\sum_{i=1}^{f} y_{ib} = 1, \forall b = 1, 2, \cdots, l. \tag{8}$$

Table 1
Indices, Parameters and Variables of the permutation flow shop problem.

Indices	
i	Customer index,$i = 1, 2, \cdots, f$.
j	Product index,$j = 1, 2, \cdots, n$.
k	Machine Index,$k = 1, 2, \cdots, m$.
b	Batch index,$b = 1, 2, \cdots, l$.
Parameters	
π_j	j*th*product in the sequence$\pi = \pi_1, \pi_2, \cdots, \pi_j$
$P_{\pi jk}$	Processing time of the product$j$ on machine k,$j = 1, 2, \cdots, n; k = 1, 2, \cdots m$.
$Q_{\pi_1 jk}$	Initial setup time of j*th* product in machine k, given that j*th* product is placed in the first position of the sequence, i.e.,π_1
$S_{\pi_j jhk}$	Sequence dependent setup time of the j*th* product, if it is processed after h*th* product ($j \neq h$) on machine k, $j = 1, 2, \cdots, n; k = 1, 2, \cdots, m$. If j*th* product is placed in π_j*th* position, then h*th* product must be placed in π_{j-1}*th* position in the sequence. Also, j*th* product cannot be placed in the first position of the sequence, i.e., $\pi_{j \neq 1}$.
DT_i	Delivery time from the factory to the customer$i$.
DC_{ib}	Delivery cost to customer i per batch.
d_{ij}	Due date of product j of customer i.
α_{ij}	Tardiness cost per unit of product j for customer$i$.
Decision variables	
$C_{\pi jk}$	Completion time of product π_j on machine$k$
$x_{\pi_j b}$	1 if the j*th* product of the π*th*sequence assigned to batch$b$; 0 otherwise.
y_{ib}	1 if batch b contains the products of customer i, 0 otherwise.
AT_{ib}	Arrival time of batch b at customer$i$.

Table 2
Nomenclature and definition for MFO.

Nomenclature	Definition
Moth	Candidate solutions
Moth Position	Problem's variable (such as combinations)
Moth Fitness	Fitness of the candidate solution in the current iteration.
Flame	Best moth positions obtained from previous iterations.
Flame Fitness	Best fitness obtained from previous iterations.

Table 3
Selected parameters.

Algorithm	Parameter	Level
DE	Population Size	200
	Iteration	200
	Crossover Factor	0.8
	Mutation Scale	0.4
MFO	Population Size	200
	Iteration	200
LFMFO	Population Size	200
	Iteration	200
	Moth- Lévy factor	0.2
GA	Population Size	200
	Iteration	200
	Crossover Factor	0.8
	Mutation Scale	0.4

$$AT_{ib} \geq y_{ib} \times \left\{ \max_{j=1,2,\cdots,n} \{x_{\pi_j b} \times C_{\pi_j m}\} + DT_i \right\}, \forall i = 1, 2, \cdots, f; b = 1, 2, \cdots l. \tag{9}$$

$$C_{\pi_j k}, AT_{ib} > 0 i = 1, 2, \cdots, f; b = 1, 2, \cdots l; j = 1, 2, \cdots n. \tag{10}$$

Constraint 2 (expressed by Equation (2)) express the equation for the completion time of product π_1 in machine $k = 1, 2, \cdots, m$. Constraint 3 states that the completion time of the subsequent $(n-1)$ product on the machine after adding the initial setup time and sequence dependent setup time are the first sets of constraints that are crucial to the proper calculation of the makespan calculation for Equation (1). Constraint 4 and Constraint 5 works similarly by providing the completion time of the first product and the remaining $(n-1)$ products respectively for all other machines. In the distribution stage, Constraint 6 shows that every finished product received can be allocated to a single batch only, this means that one product cannot be delivered twice to the customer. The constraint also indicates that it is possible to have the maximum batch delivered as the number of products received. Subsequently, Constraint 7 guarantees that for each customer, there is at least one batch delivered in the distribution stage. While Constraint 8 ensures that all the products in the same batch belong to one customer only. Constraint 9 provides the constraint of batch arrival time to the customer, where it is the summation of latest completion time of the last product in the same batch and the delivery time. Finally, Constraint 10 ensures that the completion time and arrival time will always be a non-negative value.

3.4. An example problem

An example problem for the problem under study is discussed in this section. Fig. 1 presents a Gantt chart of a schedule for this problem. In this example, two customers order eight products, where customer 1 places 3 products (i.e., product 1–3) and customer 2 places 5 products (product 4 to product 8). All these products are processed in a two-machine flow shop. As shown in Fig. 1, the sequence of product in this schedule is 6–4-2–7-1–8-5–3. Therefore, each machine follows this sequence to process all products. The set of input data for the example problem are listed in Table A1 to Table A4 (in Appendix). Table A1 consists of the input data relevant to the production stage, which

Table 4
Rank measurement of different algorithms using Friedman test.

Dataset Group	DE						MFO	LFMFO	GA
	rand/1	rand/2	best/1	best/2	current-to-best/1	current-to-pbest/1			
Small	4.042	4.000	3.625	4.875	**3.333**	4.250	7.125	6.792	6.958
Medium	5.958	5.667	4.250	**3.625**	4.958	4.375	5.750	4.708	5.708
Large	4.083	4.833	5.250	4.750	**3.750**	4.500	5.500	5.750	6.583
Average	4.694	4.833	4.375	4.417	**4.014**	4.375	6.125	5.750	6.417

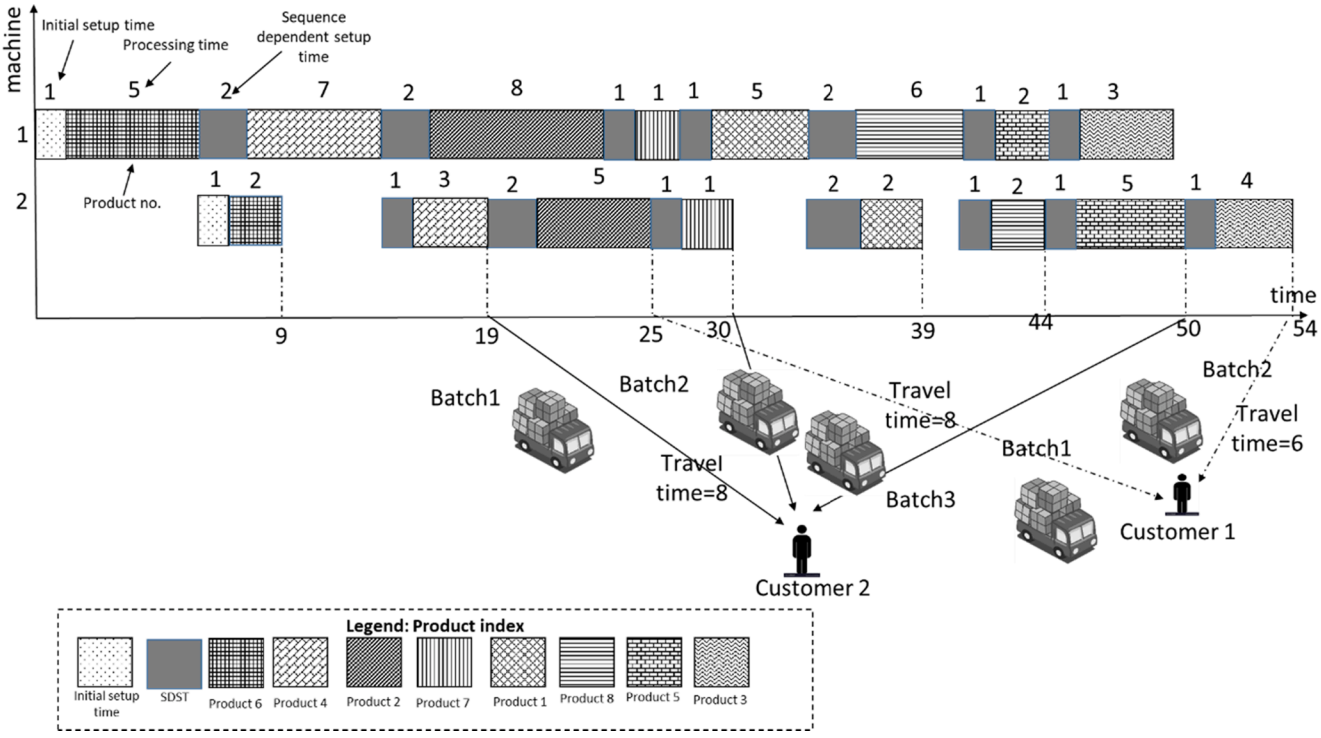

Fig. 1. A Gantt chart of the example problem.

includes processing time and due dates of the products in each machine; while Table A2 shows the input data relevant to the distribution stage, which includes the delivery cost and delivery time to each customer. Based on Table A3 and Table A4 shows the setup time for the first product on each machine and sequence dependent setup time for different product sequence. Fig. 1 shows the legend of the initial setup time, sequence dependent setup time and processing time of each product. Note that product 2, 5, and 8 are ordered by customer 1 and rest of the products are ordered by customer 2. Note that initial setup is defined as the time to setup for a product at the beginning of production.

Table A1
Input data relevant to production stage.

Customer number, F	Product number, n	Unit tardiness cost, α_{ij}	Due date of each products, d_{ij}	Processing Time in different machine of flowshop, $PT_{\pi jk}$	
				Machine 1	Machine 2
1	1	6	15	7	2
1	2	3	20	8	5
1	3	3	8	3	4
2	4	2	19	7	3
2	5	14	25	2	5
2	6	9	9	5	2
2	7	3	21	1	1
2	8	8	12	6	2

Table A2
Input data relevant to distribution stage.

Customer number, F	Delivery cost per batch to the ith customer, DC_i	Transportation time to deliver a batch to a customer, DT_i
1	8	6
2	5	8

Table A3
Input data relevant to initial setup time for first product in the flow shop.

Setup time for the first product in a sequence in a machine		
Product	**Machine 1**	**Machine 2**
1	11	3
2	8	3
3	10	14
4	2	13
5	10	15
6	1	1
7	13	2
8	8	7

Fig. 1 also suggests that the products are transferred in batches.

As described in the introduction section (Section 1), PFSPs with production and batch delivery to multiple customers has two integrated

Table A4
Sequence dependent setup time for the two machine flowshop.

Sequence Dependent Setup Time in Machine 1								
	Product 1	**Product 2**	**Product 3**	**Product 4**	**Product 5**	**Product 6**	**Product 7**	**Product 8**
Product 1	0	6	5	10	5	4	1	5
Product 2	10	0	8	2	2	3	4	9
Product 3	7	8	0	10	1	8	5	1
Product 4	10	6	10	0	1	2	10	2
Product 5	1	7	7	6	0	6	8	1
Product 6	3	4	10	6	10	0	4	9
Product 7	10	1	3	4	9	7	0	7
Product 8	2	10	7	2	5	9	3	0
Sequence Dependent Setup Time in Machine 2								
	Product 1	**Product 2**	**Product 3**	**Product 4**	**Product 5**	**Product 6**	**Product 7**	**Product 8**
Product 1	0	3	7	6	10	3	2	10
Product 2	10	0	5	2	5	3	1	5
Product 3	3	10	0	9	4	4	10	8
Product 4	1	3	10	0	1	1	7	6
Product 5	1	8	4	1	0	9	10	1
Product 6	8	3	9	6	5	0	9	2
Product 7	4	1	7	4	8	9	0	5
Product 8	1	1	4	4	6	6	3	0

decision-making process, namely sequencing the products and grouping products into batches for delivery. Therefore, as shown in Fig. 2, the solution of the problem can be represented by a $(3 \times n)$ matrix. The top row in the matrix is a permutation of n products, indicating the product processing sequence. The products need to be split into suitable number of batches, and therefore the second row is an array of batch indices, determining the assigned batch for each product. The bottom row presents the customer indices, to whom the products (in batches) are delivered. According to assumption (3), a batch contains the products for the same customer. Considering equation (1) and relevant data for this problem, the objective function (total cost) calculated for this schedule is$1531.

Total cost calculation for sequence dependent setup time (following equation (1)

$$\begin{aligned} Totalcost, \$ = {} & [(9 \times max\{0, (19+8)-9\}) + (2 \\ & \times max\{0, (19+8)-19\})] + 5 + 3 \\ & \times max\{0, (19+6)-20\}) + 8 + 3 \\ & \times max\{0, (30+8)-21\}) + 5 + [(6 \\ & \times max\{0, (54+6)-15\}) + (3 \\ & \times max\{0, (54+6)-8\})] + 8 + [(8 \\ & \times max\{0, (50+8)-12\}) + (14 \times max\{0, (50+8)-25\})] + 5 \end{aligned}$$

$$= 162 + 16 + 5 + 51 + 5 + 15 + 8 + 270 + 156 + 8 + 368 + 462 + 5 = \$1531$$

4. Methodology

This section discusses the methodology and details of the proposed algorithms such as Differential Evolution (DE), Moth Flame Optimization (MFO) and Lévy-Flight Moth Flame Optimization (LFMFO) algorithm for single objective optimization applied to solve the problem under study.

4.1. Differential evolution algorithm

Differential evolution (DE) algorithm is an efficient and robust evolutionary algorithm introduced by Storn and Price (1997). DE has been extensively applied to solve real world problems that are combinatorial in nature such as traveling salesman problem, job shop scheduling and machine layout design problem (Nearchou & Omirou, 2006; Onwubolu & Davendra, 2006). DE uses operators similar to genetic algorithm. Due to the simplicity of implementation, faster convergence and few parameters to fine tune, this algorithm has been selected for solving the considered problem.

DE algorithm proposed in this research uses similar framework as conventional DE algorithm while adapted to process PFSPs, the pseudocode used during the coding process is as follows:

Algorithm 1: Differential Evolution for Single Objective Optimization

Input: Vector product processing time for each machine, delivery time and distribution cost for customers, tardiness cost.
Output: Schedule with near optimal Total Cost.
begin
Step 1(read files)
retrieve input data;

(*continued on next page*)

Product number	**6**	**4**	**2**	**7**	**1**	**8**	**5**	**3**
Batch number	1	2	2	1	1	1	2	1
Customer number	2	2	1	2	2	1	1	2

Fig. 2. An example of feasible schedule.

(*continued*)

Algorithm 1: Differential Evolution for Single Objective Optimization

Step 2 (Initialization)
generate individuals with randomized schedules in the form of matrix;
Step 3 (Evaluate)
evaluate the initialized individual based on the objective functions;
Step 4 (Stopping rule)
stop function = false;
If Current Iteration ≤Maximum Iteration, **then**
go to step 5
Elseif Current Iteration >Maximum Iteration, **then**
Stop the algorithm;
end
Step 5 (Mutation)
Generate mutant vectors based on chosen mutation strategy.
Step 6 (Crossover)
Randomize floating point for every individual, then compare to crossover factor;
If individual number ≤crossover factor, **then**
pair the individuals into sets of parent individuals;
perform two-point crossover;
Elseif individual number >crossover factor, **then**
do not perform crossover on the schedule;
end
Step 7 (Evaluation)
evaluate the trial individuals based on the objective functions;
Step 8 (Selection)
Compare and choose the trial individuals against parent individuals;
update the iteration count and go to step 4;
end
report best schedule and the Total Cost.

In Differential Evolution (DE), the mutant vectors $v_i = \{v_i^1, v_i^2, \ldots, v_i^D\}$ are generated through the mutation operation. There are many mutation strategies and operators developed and studied by other researchers with varying results. In this research work, the six most popular mutation strategies are identified are used for comparison and based on the variants proposed by Vincent, Ponnambalam, and Kanagaraj (2014) and Nearchou and Omirou (2006).

$$\mathrm{DE/rand/1}v = x_{r1}^{(G)} + F \cdot \left(x_{r2}^{(G)} - x_{r3}^{(G)}\right) \tag{11}$$

$$\mathrm{DE/rand/2}v = x_{r5}^{(G)} + F \cdot \left(x_{r1}^{(G)} + x_{r2}^{(G)} - x_{r3}^{(G)} - x_{r4}^{(G)}\right) \tag{12}$$

$$\mathrm{DE/best/1}v = x_{best}^{(G)} + F \cdot \left(x_{r2}^{(G)} - x_{r3}^{(G)}\right) \tag{13}$$

$$\mathrm{DE/best/2}v = x_{best}^{(G)} + F \cdot \left(x_{r1}^{(G)} + x_{r2}^{(G)} - x_{r3}^{(G)} - x_{r4}^{(G)}\right) \tag{14}$$

$$\mathrm{DE/current-to-best/1}v = x_i^{(G)} + F \cdot (x_{best}^{(G)} - x_i^{(G)}) + F \cdot (x_{r1}^{(G)} - x_{r2}^{(G)}) \tag{15}$$

$$\mathrm{DE/current-to-pbest/1}v = x_i^{(G)} + F \cdot \left(x_{best}^{(P)} - x_i^{(G)}\right) + F \cdot \left(x_{r1}^{(G)} - x_{r2}^{(G)}\right) \tag{16}$$

In the six variants of popular mutation strategies described in Equation (11) to Equation (16), *r1, r2, r3, r4,* and *r5* are the schedules of different individuals within the current populations, *F* refers to the scale factor for the magnitude of the vectors.$x_{best}^{(P)}$, $x_{best}^{(G)}$ shows the best vector in the current and global population. In this work, the process used for encoding the vectors in DE is based on the work reported in (Nearchou & Omirou, 2006; Vincent et al., 2014). In this research, all six variants described are implemented and their performance are compared with the other optimization algorithms. The process used for encoding the vectors in DE is based on the work reported in (Vincent et al., 2014). After mutation, the crossover operation is performed between two parent vectors to form trial vectors. The working principle for the selection of the parent for crossover operation is similar to the method used in mutation process. A randomly generated floating number between the constraint [0, 1] is assigned to each individual in every iteration and determines whether the individual ($u_{i,G}^j$) is chosen for recombination operation as shown in the logic below in Equation (17).

$$u_i^G = \begin{cases} u_{i,G}^j, if \;\; rand_j(0,1) \le CR \\ v_i^{(G)}, else \end{cases} \tag{17}$$

where $i = 1, 2 \ldots NP$; $j = 1, 2 \ldots D$; and Crossover Factor, CR ∈ [0, 1] are the parameters for the crossover rate. *NP* denotes the number of population, *randj* [0, 1] is a uniformly distributed random number, which is called for each *j*th component,$v_i^{(G)}$ is the best mutation individual in the population, u_i^G is the trial vector for the crossover operation. When multiple pairs of parent individuals are chosen, they are assigned randomly in pairs to form couples. In this research, the single point crossover is chosen as the crossover method. This is done by randomly picking a point on the parent individuals' chromosomes as the designated crossover point. Part of the parent chromosomes are swapped between the partner's chromosomes. Forming two sets of newly formed trial vectors inheriting part of the chromosomes from both parents. Fig. 3 gives an example of single-point crossover.

After the parent chromosomes are paired, a crossover point is determined randomly to be swapped between the parents. The swapping process ensures that the offspring chromosomes retain some information from both parents. After the crossover operation is completed, the newly formed trial vectors are then evaluated and the fitness of the current population is compared to the previous population. The encoded vectors which violate the constraints are repaired using the procedure reported in (Vincent et al., 2014).

Selection is the mechanism used in DE algorithm to determine whether the trial vector replaces the parent vector according to the criterion from the objective function. The working principle of the selection mechanism within a DE algorithm remains the same regardless of the different DE algorithms proposed in most research works as shown in (Nearchou & Omirou, 2006; Santucci, Baioletti, & Milani, 2016; Zhou & Wu, 2019). This is done by comparing the fitness of individuals from current trial vector and target vector from previous iterations, the individuals with the greatest fitness are chosen as the parents for the next iteration. The selection operator is performed as shown in Equation (18).

$$x_i^{G+1} = \begin{cases} x_i^G, iff\left(x_i^G\right) \le f(u_i^G) \\ u_i^G, else \end{cases} \tag{18}$$

4.2. Moth flame optimization algorithm

Motivated by the flight pattern of moths towards artificial lights such as flame at night, Mirjalili (Mirjalili, 2015) proposed a swarm intelligence metaheuristic algorithm known as Moth Flame Optimization (MFO). This algorithm is inspired by the flight pattern of moths towards light sources at night. Conventionally, the moths maintain a fixed perpendicular angle with respect to the light source from the moon, which is also known as 'traverse orientation'. The large distance between the moon and moth ensures the moth can efficiently navigate long distance during the night. However due to urbanization, the moths often mistaken the artificial light source as the moon, prompting the moths to fly in a traverse orientation based on the percieved light source. This causes a spiral flight pattern, slowly converging towards the artificial light source due to the short distance between the moth and the light source. MFO has been implemented to solve various real-world optimization problems such as optimizing power flow (Buch, Trivedi, & Jangir, 2017), and optimizing economic order quantity (Khalilpourazari & Khalilpourazary, 2019). Eventhough few researchers have reported using MFO in solving combinatorial optimizational problems (Ghobaei-Arani et al., 2020; Mohseni, Brent, & Burmester, 2019; Rashid, Rose, Mohamed, & Romlay, 2019), there is lot of scope for implementing MFO for such problems. In this paper, the encoding pocedure used is based on the procedure utilized for DE. To the authors knowledge, research done

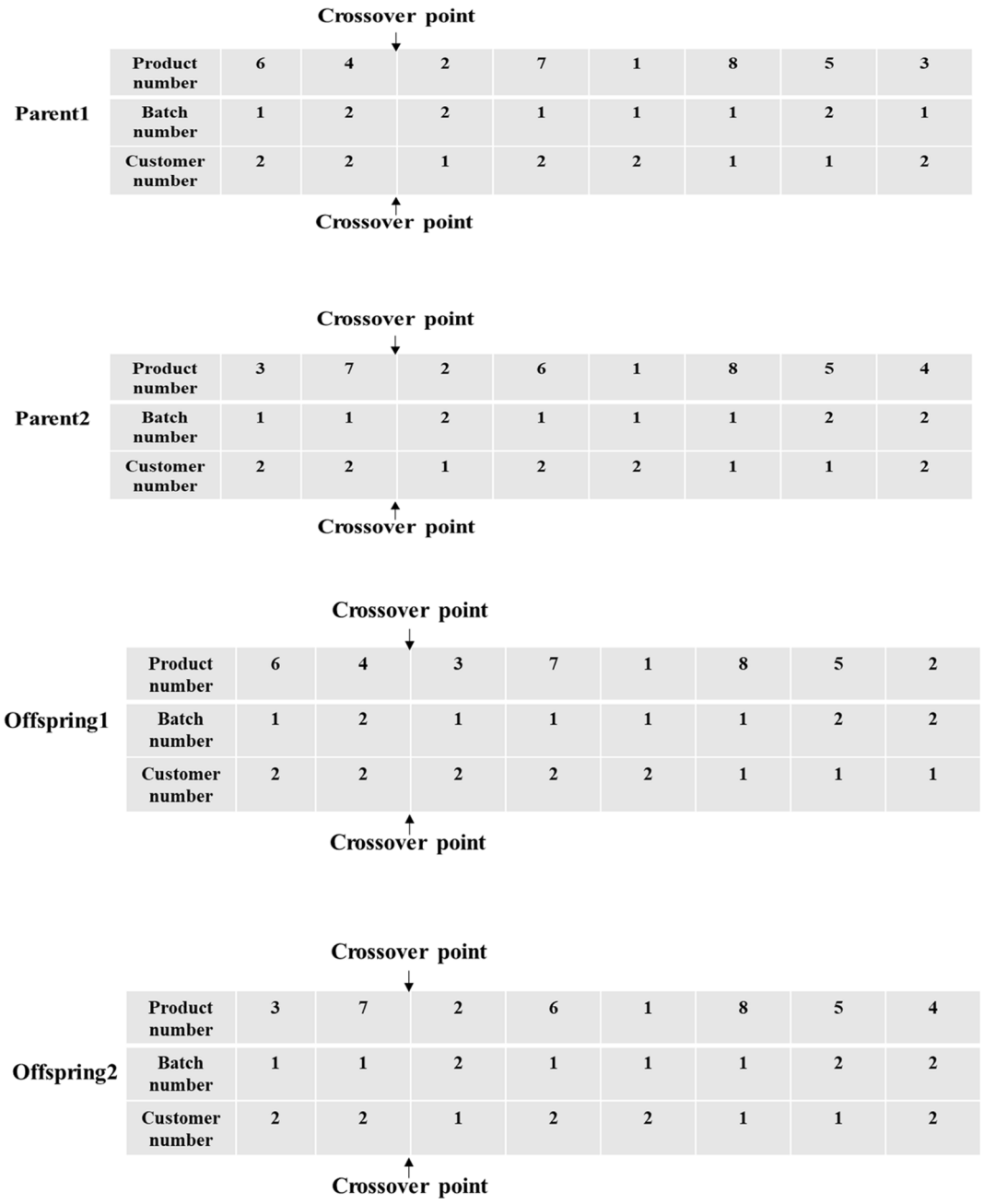

Fig. 3. Illustration of Single Point Crossover Method.

by AbdElhamid, Helmi, and Ziedan (2018) is one of the few research directed towards implementing MFO to solve a well known Travelling Salesman Problem. The nomenclature used in MFO are provided as shown in Table 2.

Similar to DE algorithm, the solutions from the set of moths are represented in a matrix as shown in Equation (19), while the corresponding fitness values of each moths are stored in the array as shown in Equation (20).

$$Moth, M = \begin{bmatrix} m_1^1 & m_1^2 & \cdots & \cdots & m_1^D \\ m_2^1 & m_2^2 & \cdots & \cdots & m_2^D \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ m_n^1 & m_n^2 & \cdots & \cdots & m_n^D \end{bmatrix} \tag{19}$$

$$MothFitness, OM = \begin{bmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_n \end{bmatrix} \tag{20}$$

where n is the number of moths and D refers to the number of bits within the solution vector. Each moth, $m = 1,2, \ldots n$ with dimension D produces a fitness value returned from the objective function. The moth fitness, OM are arranged according to the moth position vectors. Note that the flame, F and flame fitness, OF are similar to the moth and moth fitness as shown in Equation (21) and Equation (22).

$$Flame, F = \begin{bmatrix} F_1^1 & F_1^2 & \cdots & \cdots & F_1^D \\ F_2^1 & F_2^2 & \cdots & \cdots & F_2^D \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ F_n^1 & F_n^2 & \cdots & \cdots & F_n^D \end{bmatrix} \quad (21)$$

$$Flame\ \ Fitness, OF = \begin{bmatrix} OF_1 \\ OF_2 \\ \vdots \\ OF_n \end{bmatrix} \quad (22)$$

The differences between the moths and flames are that moths act as the active search agents that moves within the solution space; while the flames are the best positions obtained by the moths over iterations. As the moths discover a position with better fitness than the current flame, the flame is updated to the position of the newly found best flames. This process repeats until all of the moth either converges into a single true best result or the iteration reached the maximum limit. The pseudo-code of MFO is presented to illustrate the processes in MFO algorithm.

Algorithm 2: Moth Flame Optimization for Single Objective Optimization

Input: Vector product processing time for each machine, delivery time and distribution cost for customers, tardiness cost.
Output: Schedule with near optimal Total Cost.
begin
  Step 1(read files)
    retrieve input data;
  Step 2 (Initialization)
  generate initial moths with randomized schedules in the form of matrix;
  Step 3 (Evaluate)
    evaluate the initialized moths based on the objective function;
  Step 4 (New Moth Position Calculation)
    Perform calculation for number of flames, t and r;
    Update number of flames, t and r;
    Perform calculation for new moth positions;
  Step 5 (Evaluation)
    evaluate the new moth position based on the objective function;
  Step 6 (Selection)
    compare the new moth position against flame;
    update best positions as flames for next iteration;
    Iteration = Iteration + 1;
  Step 7 (Stopping rule)
    stop function = false;
      If Current Iteration ≤Maximum Iteration, **then**
      go to step 4
      Elseif Current Iteration >Maximum Iteration, **then**
      Stop function toggle true; return;
      end
end
report best schedule and the Total Cost.

An adaptive mechanism of reducing the number of flames over the course of iterations is used to ensure the exploitative quality of MFO algorithm. This is done by utilizing the Equation (23), where the number of flames reduces gradually as shown in Fig. 4.

$$Number\ \ of\ \ flames = round\left(PopSize - 1 \cdot \frac{PopSize - 1}{MaximumIteration}\right) \quad (23)$$

Note that as the iteration approaches maximum iteration, the flame converges into the single best flame found. This allows the trend of gradual decrement in number of flames over iteration allow a balance in exploration and exploitation of the search space, where the moths move towards the position with the greatest fitness. The update motion of the moth as the active search agent is described as a logarithmic spiral with the conditions:

1. The initial point of the spiral motion starts from the moth.
2. The destination of the spiral motion is the position of the flame.
3. The fluctuation of the range of spiral should not go beyond the search space.

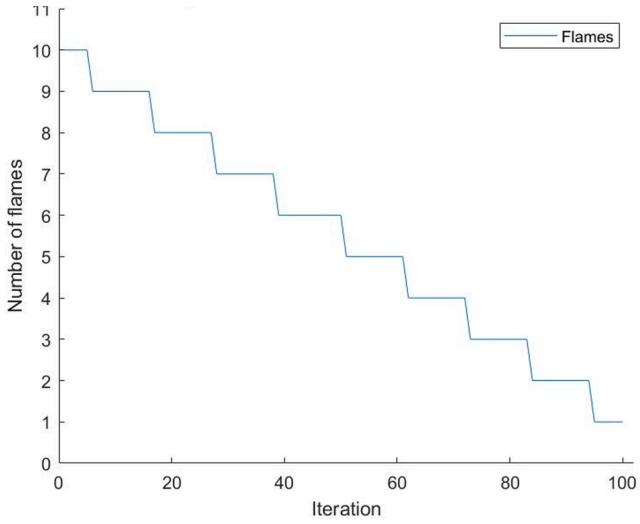

Fig. 4. Trend of number of flames over iterations in Moth Flame Optimization.

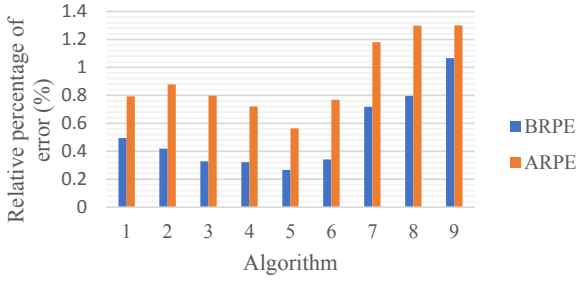

Fig. 5. Average BRPE and ARPE for all problem sizes: (1) DE/rand/1, (2) DE/rand/2, (3) DE/best/1, (4) DE/best/2, (5) DE/current-to-best/1, (6) DE/current-to-pbest/1, (7) MFO, (8) LFMFO, (9) GA.

With these conditions as consideration, the equation of logarithmic spiral used in MFO algorithm is defined as Equation (23).

$$Spiral, S(M_i, F_j) = D_i \cdot e^{bt} \cdot \cos(2\pi t) + F_j \quad (24)$$

$$MothtoFlameDistance, D_i = |F_i - M_i| \quad (25)$$

$$t = (a-1) \cdot \left(\frac{-1}{MaximumIteration}\right) \quad (26)$$

$$a = -1 + Iteration \cdot \left(\frac{-1}{MaximumIteration}\right) \quad (27)$$

$$b = Unif(0.0, 1.0) \quad (28)$$

Equation (24) presents the equation of logarithmic spiral of the moth positional update with the constraints and information from Equation (25) to Equation (28). The distance between the assigned moth flame pairs are calculated using Equation (25); while Equation (26) to Equation (28) describes the adaptive variables that provide slight randomness as the moths approaches the flame.

4.3. Lévy-flight moth flame optimization

Lévy-Flight Moth Flame Optimization Algorithm (LFMFO) is the implementation of the random motion of the Lévy-Flight function to improve the exploration and convergence rate of the conventional Moth Flame Optimization Algorithm (Mirjalili, 2015). This is because Lévy-Flight has the prominent properties of increasing the diversity of population, effectively allowing the algorithm to jump out of the local

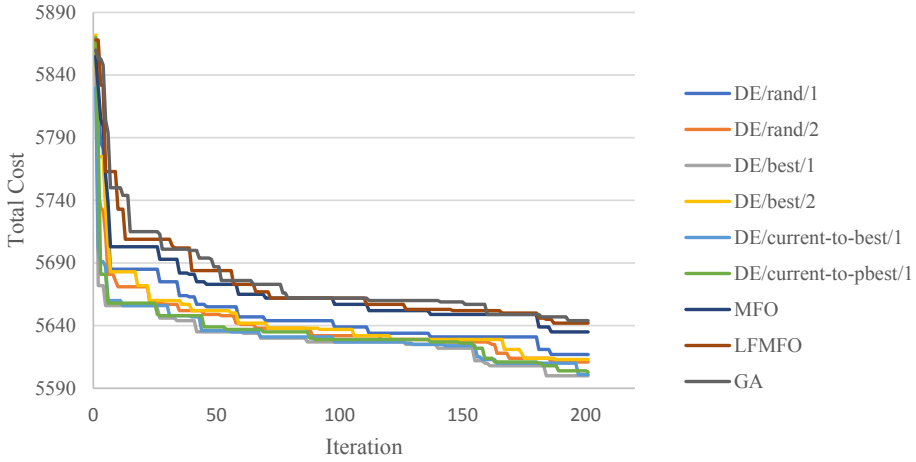

Fig. 6. Graph of convergence curves for algorithms for a problem instance.

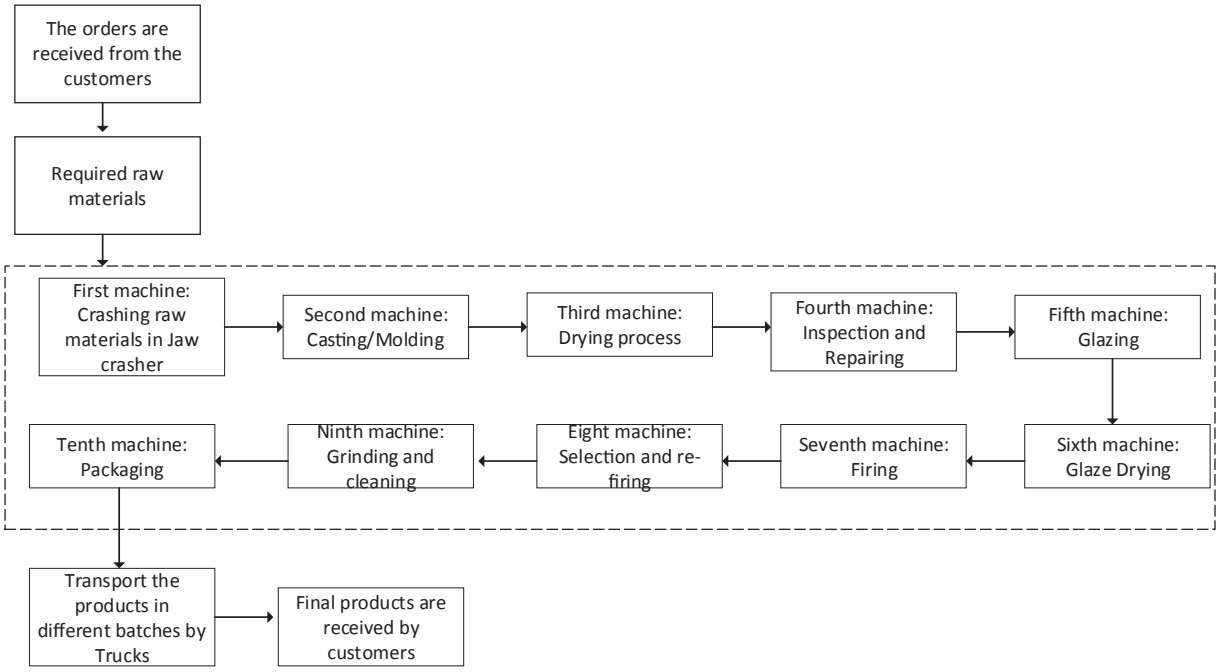

Fig. 7. Production flow for a sanitaryware production and distribution in supply chain.

optimum point (Li, Janardhanan, Tang, & Nielsen, 2016). The randomized motion of the Lévy-Flight motion plays an important role in the exploration and exploitation properties of the LFMFO algorithm to optimize complex problems (Kamaruzaman, Zain, Yusuf, & Udin, 2013; Trivedi, Kumar, Ranpariya, & Jangir, 2016), more effectively. The hybridization of Lévy-Flight function into Moth Flame Optimization is performed by inserting a probability of a moth individual to perform Lévy-Flight motion or the spiral motion from the conventional MFO (Mirjalili, 2015) (Trivedi et al., 2016) (Kamaruzaman et al., 2013), where the probability randomly chooses moth individual to perform the Lévy-Flight function after the position is updated to further enhance the global search ability to provide more successful results for multimodal functions (Li et al., 2016). The Lévy-Flight function follows the Lévy distribution as shown in Equation (29).

$$\hat{\text{Lévy}}(\beta)\ \mu = t^{-1-\beta}, (0 \le \beta < 2) \tag{29}$$

$$\phi = \left[\frac{\Gamma(1+\beta) \times \sin(\pi \times \beta/2)}{\Gamma(((1+\beta)/2) \times \beta \times 2^{(\beta-1)/2})}\right]^{1/\beta} \tag{30}$$

Equation (30) presents the calculation for the new position of the moth that follows Lévy-Flight function, where $\beta = 1.5$ and Γ represents a standard Gamma function. The global search ability of LFMFO is expected to be improved from the hybridization of the Lévy-Flight function to MFO, utilizing the strength of both search mechanism to explore the solution space. The pseudo-code for LFMFO is presented below:

Algorithm 3: Lévy-Flight Moth Flame for Single Objective Optimization

Input: Vector product processing time for each machine, delivery time and distribution cost for customers, tardiness cost.
Output: Schedule with near optimal Total Cost.
begin
Step 1(read files)
retrieve input data;
Step 2 (Initialization)
generate initial moths with randomized schedules in the form of matrix;
Step 3 (Evaluate)
evaluate the initialized moths based on the objective function;
Step 4 (New Moth Position Calculation)
Perform calculation for number of flames, t and r;

(*continued on next page*)

(*continued*)

Algorithm 3: Lévy-Flight Moth Flame for Single Objective Optimization

```
    Update number of flames, t and r;
      If random number between 0 &1 ≤ Moth Levy factor, then
      Implement Lévy-Flight function on moth position;
      Elseif random number between 0 &1 >Moth Levy factor, then
      Implement logarithmic spiral function on moth position;
      end
  Step 5 (Evaluation)
    evaluate the moth position based on the objective function;
  Step 6 (Selection)
    compare the new moth position against flame;
    update best positions as flames for next iteration;
    Iteration = Iteration + 1; go to step 4;
  Step 7 (Stopping rule)
    stop function = false;
      If Current Iteration ≤Maximum Iteration, then
      go to step 4
      Elseif Current Iteration >Maximum Iteration, then
      Stop function toggle true; return;
      end
end
report best schedule and the Total Cost.
```

5. Results and discussion

This section discusses the results and data collected using the proposed metaheuristic algorithms, the optimal parameters through tuning and the comparison of the results obtained from the proposed metaheuristic algorithms. The software used in this research are MATLAB R2018b and Minitab 15. The metaheuristic algorithms are coded in MATLAB R2018b. Meanwhile, Minitab is used for Design of Experiment and statistical analysis in this research. The PFSPs were solved using total cost as objective functions. A total of 36 problem instances are generated and solved. The datasets will be available on request and also available in public domain like ResearchGate.

Since genetic algorithm (GA) is one of the metaheuristic algorithms which has a successful track record of solving complex scheduling problems (Abreu, Cunha, Prata, & Framinan, 2020; Ruiz et al., 2006). Therefore, to evaluate the performance of the proposed algorithms, proposed algorithms are compared with GA as well. GA starts with a set of random solutions and evolve by tournament selection, similar job order crossover, and shift mutation operators. The reason to select these operators is that they are successful in solving classical PFSPs. Note that, due to space constraints details of GA are not presented. In this section, the output of the proposed algorithms from parameter tuning and the experimental result of the proposed algorithms are discussed.

5.1. Problem set generation

As described in Section 1 and 2, authors study the PFSPs with batch delivery to the PFSP with batch delivery to multiple customers with SDST for the first time and there are no benchmark data sets available in the literature. Therefore, in this paper, three different groups of test problems, ranging from small, medium and large-sized problems are generated randomly to validate the effectiveness of the proposed algorithms. For each of the group, there is a total of 12 test problems sets produced for each size, with a total of$36(=12+12+12)$ problem instances were generated. For the small-sized problem, the product sizes, n has two levels: {20, 35} and machine sizes, m has two levels:$\{5,10\}$; As for medium-sized problems, the product sizes, $n=\{50,75\}$ and machine sizes, $m=\{10,15\}$; Similarly, the product sizes, $n=\{100,125\}$ and machine sizes, $m=\{15,20\}$ are assigned for large-sized problems. With this assignment, the test problems can be increased by assigning different number of customers, f in the distribution stage ranging from 2 to 4. To ensure there are no bias in the problem sets developed, it is important that the problem instances are generated randomly through multiple sets of constraints and different uniform distributions. Each time-related and cost-related parameter is generated using different uniform distributions and the range/constraints of uniform distributions for each of these parameters are discussed from Equation (31) to Equation (42).

$$productsize \in [20,125] \tag{31}$$

$$Numberofmachines, m \in [5,20] \tag{32}$$

$$Numberofcustomer, f \in [2,4] \tag{33}$$

Equation (34) to equation (41) define the constraints for the combinations of problem set to be generated for small, medium and large sized problems. Equation (34) to equation (36) describes the range of value used for the generation of unit tardiness cost for each product, delivery time and delivery cost to each customer. Equation (37) and equation (38) describe the constraint on setup time for the first product for every machines and subsequent products respectively. Equation (39) shows the constraint of processing time for each product to be a uniform distribution between [1, 30], while the due date assigned for each product are also uniformly distributed between [*LB*, *UB*] as shown in Equation (40), Equation (41) and Equation (42).

$$UnitTardinessCost, \alpha_{ij} = Unif(1,20) \tag{34}$$

$$DeliveryTime, DT_i = Unif(5,15) \tag{35}$$

$$DeliveryCost, DC_{ib} = Unif(1,99) \tag{36}$$

$$Firstjobsetuptimeoneachmachines, Q_{\pi_1 j,k} = Unif(1,20), \forall k \tag{37}$$

$$SubsequentSetupTime, S_{j,k} = Unif(1,20) \tag{38}$$

$$ProcessingTime, P_{\pi jk} = Unif(1,30) \tag{39}$$

$$DueDate, d_{ij} = Unif(LB_{ij}, UB_{ij}) \tag{40}$$

$$LowerBoundary, LB_{ij} = max\{Q_{\pi_1 jk} + max\{S_{\pi_j jhk}\} + \max\{P_{\pi jk}\} + DT_i, \forall i \tag{41}$$

$$UpperBoundary, UB_{ij} = \sum_{k=1}^{m} Q_{\pi_1 jk} + \sum_{k=1}^{m} S_{\pi_j jhk} + \sum_{k=1}^{m} P_{\pi jk} + DT_i, \forall i \tag{42}$$

5.2. Performance metrics

This section explains the methodology used to evaluate the performance of the single objective optimization algorithm. The most crucial part in this section is to establish a measurement for performance of the proposed algorithms through computational experiments, the obtained performance measurement for each algorithm is then compared with each other to identify the best performing algorithm. The methods used to measure the performance of single objective optimization are quite simple and straightforward. The performance metric known as relative percentage error (RPE) is calculated as shown in Equation (43).

$$RPE = \frac{|Method_{sol} - Best_{sol}|}{|Best_{sol}|} \times 100 \tag{43}$$

where $Method_{sol}$ refers to the solution obtained by the method of the studied algorithms, and $Best_{sol}$ indicates the best solution generated by all the considered algorithms. Equation (43) is considered to be reliable for small and moderate sized problem set. However, the inherent randomness of metaheuristic algorithms may result in different solutions for every run. Therefore, the best RPE (BRPE) and average RPE (ARPE) are implemented to study the upper bound and average of the result spread. The equation for BRPE and ARPE are presented in Equation (44) and Equation (45) respectively. To utilize the BRPE and ARPE, the metaheuristic algorithms are run for ten times on the problems

generated to obtain the results of the run. Generally, RPE refers to the distance of variation of result obtain from the global best result. Therefore, it is desirable to obtain lower BRPE and ARPE value.

$$BRPE = \frac{|Method_{Bestsol} - Best_{sol}|}{|Best_{sol}|} \times 100\% \quad (44)$$

$$ARPE = \frac{1}{10}\sum_{r=1}^{10} \frac{|Method_{sol} - Best_{sol}|}{|Best_{sol}|} \times 100\% \quad (45)$$

5.3. Parameter tuning

The purpose of parameter tuning is to identify the metaheuristic algorithm parameters to produce the most optimal results. The tuning process is very important because the quality of performance by the metaheuristic algorithm is very sensitive to the parameter used. The method proposed for the parameter tuning by Taguchi's design of experiment (DOE) in order to reduce the number of experiments needed to obtain similar result. Taguchi design method details are not presented due to space constraints. However, they are available upon request and will be uploaded in platforms such as ResearchGate for readers' benefit. To determine the appropriate parameters to be utilized through Taguchi design, the parameter tuning is done on a medium sized problem with the following details (75 products, 10 machines and 3 customers). Each parameter for each algorithm has four levels. Therefore, an L_{16} orthogonal array was used to tune the parameters of the proposed algorithm. To ensure the reliability of the results, each run is repeated for 10 times to obtain the Average Relative Percentage Error (ARPE), which is calculated by Equation (45). Fig. A1. to Fig. A3. (in Appendix) represents the main effect plots of ARPE of the DE, MFO, and LFMF0 algorithm, respectively. To make the comparison fair, the parameters of GA are also tuned by Taguchi's DOE and the results of the parameter tuning is presented in Fig. A4 (in appendix).

Based on the Figs. A1, A2, and A3, the chosen parameter levels for each algorithm are shown in Table 3. From the chosen parameter settings, there is an obvious trend of maximizing population size and iterations to reduce the ARPE. This is because the increased population size and iterations improved the chance of finding the optimal results with a trade-off of increased computation resources used.

5.4. Experimental results

This section discusses the results obtained from the experiments on problem instances generated with different sizing for single objective optimization with total cost as the objective function. All parameters used for the algorithms are based on the chosen parameters in Table 3. Each combination is repeated 25 times to ensure the reliability of the ARPE obtained by recording the average relative percentage error of the final results from the algorithms, while the BRPE observed from the best results obtain throughout the 25 repetition in this research. IBM CPLEX Optimization studio version 12.6.0.0 is used for solving the problems. It is observed that the solver was not able to generate optimal solutions for all the problems and only the first three small instances were solved by CPLEX. CPLEX returned an error message as 'Search Space Exceeded' for other instances in the small category. The proposed algorithms were able to achieve the optimal solution obtained by the CPLEX solver.

To evaluate the performance of each of the algorithms, the ARPE and BRPE are recorded and the average for each algorithm in different sizes of problem instances are obtained. Subsequently, bar charts are constructed to illustrate the average performance of each algorithm and its variants. Table A5, Table A6, and Table A7 (in appendix) presents the results obtained from the experiments conducted on small, medium, and large sized problem sets respectively. Based on the results obtained, average BRPE and ARPE values are plotted in Fig. 5, the average BRPE and ARPE indicates that Differential Evolution outperforms Moth Flame Optimization, Lévy-Flight Moth Flame Optimization and GA. The DE/current-to-best/1 shows the best performance with 0.446 BRPE (average value of all the BRPEs), followed by DE/best/2 and DE/best/1. This indicates that these three variants of DE algorithms are able to achieve the closest to best found results and this is evident when comparing the average ARPE values of these algorithms. While both MFO and LFMFO indicates the least BRPE results with 0.989 and 1.096 respectively. This is probably due to the mechanism of MFO and LFMFO which are more suited for position-based optimization problems. Furthermore, average value of both BRPE and ARPE shows that DE, MFO, and LFMFO algorithms outperforms GA as well.

To validate the solutions, a lower bound (LB) is proposed, which is expressed with equation (46). In deriving the LB, we have made with the following assumptions, (1) all products for a customer f is delivered in single batch, i.e., each batch is formed only with the products from the same customer, and (2) the start time of the first job in each batch, b in

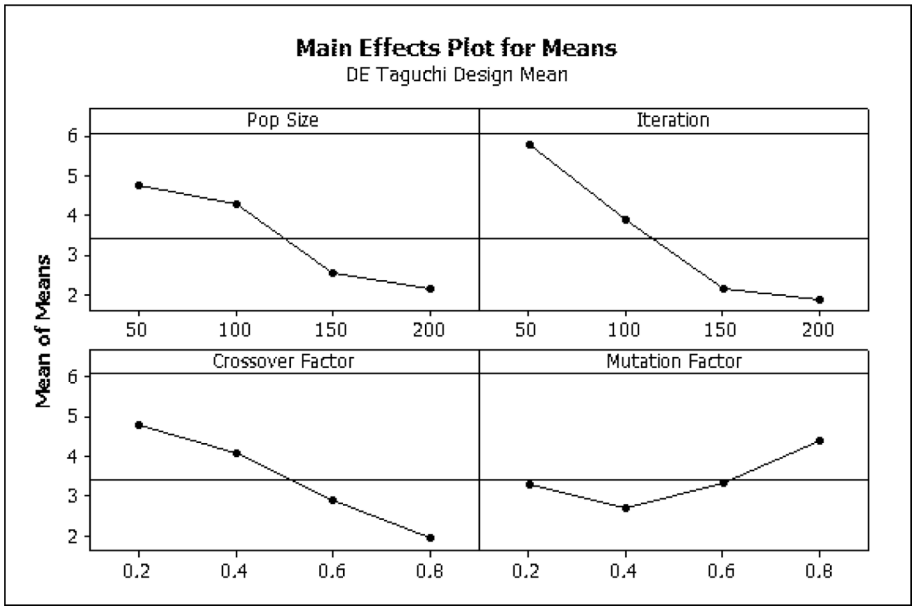

Fig. A1. Main Effect Plot of ARPE for each parameter levels of DE algorithm.

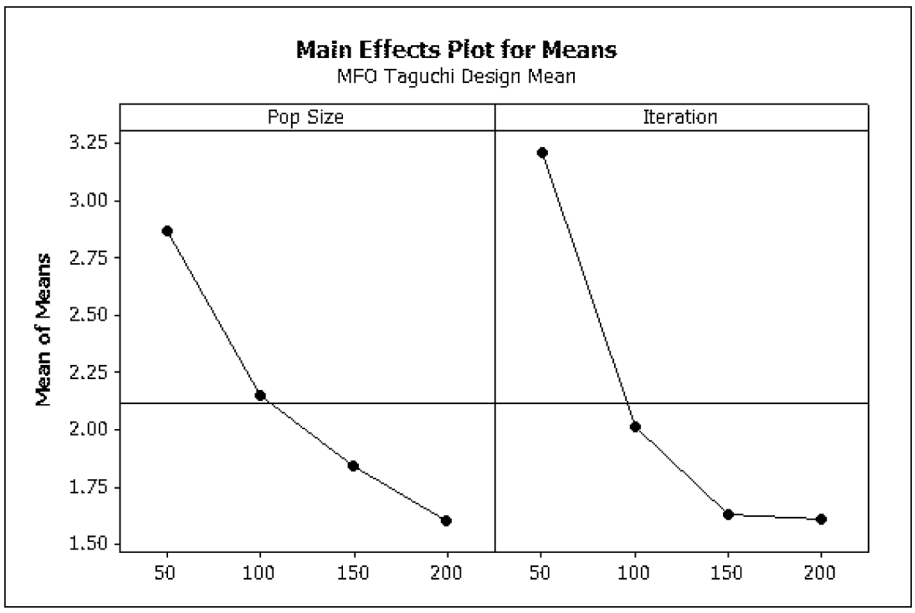

Fig. A2. Main Effect Plot of ARPE for parameter each levels of MFO algorithm.

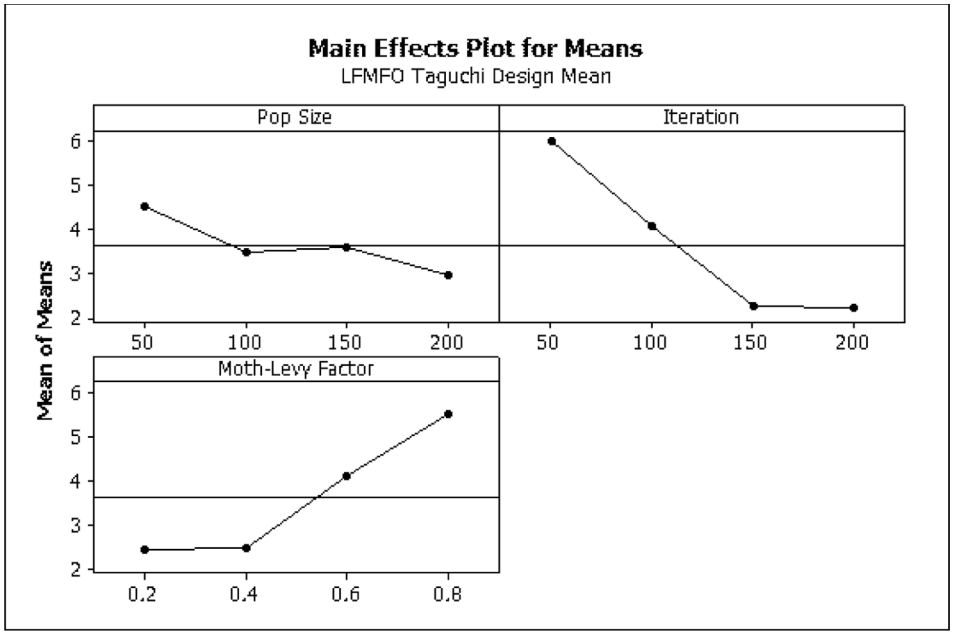

Fig. A3. Main Effect Plot of ARPE for each levels of LFMFO algorithm.

the first machine for customer i , H_b is zero. (3) the products in for a customer (i.e., within the batch are sequenced based on their ascending order of due dates) Based on the assumptions, this bound could be different. However, a reasonable LB is proposed for comparing the results. To further evaluate the performances of the proposed algorithms, we have proposed a simple lower bound (LB) which is expressed with equation (46). For evaluating the performance, the results obtained for the proposed algorithms are compared with respect to the LB by using equation (47) and (48). The results for small, medium, and large problem instances are tabulated in Table A8, A9, and A10 respectively. Based on the comparison of the calculated average LBRPE and LARPE values, the variants of DE performed better than the other compared algorithms. Among the variants DE/current-to-best/1 is the best performer and this is consistent with the performance comparison previously discussed in this section.

$$LB = \sum_{b=1}^{l}\sum_{i=1}^{f}\sum_{j=1}^{n}\alpha_{ij} \times max\{0, y_{ib} \times x_{\pi_{jb}} \times (AT_{ib} - d_{ij} + H_{ib})\} + \sum_{b=1}^{n}\sum_{i=1}^{f}(y_{ib} \times DC_{ib}), where Q_b = 0 \tag{46}$$

$$LBRPE = \frac{|Method_{Bestsol} - LB|}{LB} \times 100\% \tag{47}$$

$$LARPE = \frac{1}{25}\sum_{r=1}^{10}\frac{|Method_{sol} - LB|}{LB} \times 100\% \tag{48}$$

According to the trend of the cost as shown in Fig. 6, it is found that the convergence trends of differential evolution variants are very

Fig. A4. Main Effect Plot of ARPE for each levels of GA.

competitive and similar to each other, where there are no significant difference in the quality of solution found at the maximum iteration. On average, DE/best/1 has proven to reach the best-found result compared to the other variants at iteration = 143. Meanwhile, LFMFO is the worst performing algorithm in this research work. This is due to the random walk in Lévy flight function affecting the quality of solution by randomly changing the candidate solutions' schedule or 'moth position', trading off the exploitation for the chance of improving the exploration of the original moth flame optimization. Similarly, the Moth Flame Optimization is the second worst performing algorithm in the single objective optimization despite the steep average convergence rate before iteration = 50. This is because of original Moth Flame Optimization's tendency to being entrapped into local best solution and the lack of sufficient exploration performed from the spiral motion of the candidate solutions or 'moth', as described in (Li et al., 2016; Sayed & Hassanien, 2018).

In order to have a better observation of the performance of algorithms and to ascertain that the difference among the algorithms is statistically significant, this research conducts the Friedman test. The statistical analysis is conducted on all the three dataset groups. Under each instance type, for each of the algorithms, relative ranks are calculated by considering the ARPE values. All relative ranks are addressed in the Table 4. As evident, DE/current to-best/1 performs better than other algorithms except for the medium datasets followed by DE/best/1 and DE/best 2. DE variants have performed better overall. MFO, LFMFO and GA are the poor performers among the tested algorithms for all dataset groups. Hence it can be claimed that performance of DE algorithms is highly competitive and consistent among the test datasets.

5.5. Experimental result validation

Since there are no standard benchmark data for solving the problem under study, authors have solved the proposed algorithms by realistic problem instances. Hence, it is important to validate the solutions generated by the proposed algorithms and GA. To do this, authors have validated the solutions generated by a lower bound (LB) (equation (46), 47, and 48). For each problem instance, the value of the developed LB will remain unchanged, and therefore, provide a baseline for future comparisons. These values will be released publicly for future research. The deviation from LB for all the tested algorithms in small, medium, and large instances are summarized in Table A8, A9 and A10 (in appendix). Moreover, for further validation, authors have utilized the IBM CPLEX Optimization solver and it was able find optimal solutions for the first three small instances, which was also same as the solutions achieved by some variants of the proposed DE algorithms.

6. Case study

In order to test the effectiveness of the proposed approaches in solving real-life problems, authors collected data and solved a production problem from a sanitaryware production system in Bangladesh. Fig. 7 shows the production flow diagram for the production system under study. As shown in the figure, the manufacturer receives the orders from the customers (i.e., MTO system) and the products are processed by 10 machines/stages (from jaw crasher to packaging) before they are transferred to the customers by trucks. All the products need setups in the molds, glazing lines, drying process in kilns, grinding and cleaning process during production. These setup times are sequence dependent. At the beginning of production, the production manager generates a schedule for the products to minimize the total cost of the supply chain. Therefore, scheduling decision is crucial for ensuring profitability of that production company. To demonstrate the effectiveness of the proposed algorithms, 50 production-5 customer problem has been experimented, and the results with best and average performances of the proposed algorithms are presented in Table A11. As seen from the table, the proposed algorithms show promising results in minimizing total costs of the supply chain. Moreover, different variants of DE performed better than MFO, LFMFO and GA, which is also consistent with the results obtained by solving the developed problem instances in Section 5.

7. Conclusion and future research directions

This research considers the PFSP with batch delivery to multiple customers in supply chain. From the literature review, it is clear that the problem has been scarcely addressed by the researchers. To the authors' best knowledge, it is the first attempt to integrate production and distribution decisions in a permutation flow shop while considering SDSTs. This is a complex industrial scheduling problem that needs an efficient

Table A5
Experimental result of various metaheuristic algorithm for small sized problem sets.

Instances	DE/rand/1		DE/rand/2		DE/best/1		DE/best/2		DE/current-to-best/1		DE/current-to-pbest/1		MFO		LFMFO		GA	
	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE
S1	0.000	0.006	0.000	0.012	0.000	0.003	0.000	0.017	0.000	0.001	0.009	0.048	0.013	0.237	0.020	0.148	0.017	0.157
S2	0.000	0.050	0.005	0.048	0.000	0.005	0.003	0.022	0.004	0.019	0.011	0.024	0.029	0.103	0.041	0.113	0.041	0.116
S3	0.080	0.133	0.000	0.004	0.018	0.038	0.051	0.070	0.582	0.618	0.072	0.085	0.236	0.271	0.174	0.200	0.179	0.194
S4	0.003	0.042	0.006	0.013	0.310	0.340	0.000	0.001	0.122	0.193	0.062	0.082	0.981	1.039	0.862	0.925	0.857	0.924
S5	0.000	0.000	0.000	0.000	0.000	0.012	0.000	0.001	0.000	0.001	0.000	0.001	0.000	0.054	0.000	0.031	0.000	0.026
S6	0.103	0.175	0.167	0.260	0.114	0.154	0.692	0.738	0.271	0.309	0.635	0.736	0.276	0.318	0.352	0.484	0.352	0.485
S7	0.004	0.018	0.011	0.025	0.000	0.023	0.010	0.028	0.000	0.013	0.003	0.022	0.937	1.451	1.467	1.885	1.464	1.888
S8	0.109	0.201	0.017	0.042	0.019	0.032	0.170	0.288	0.002	0.014	0.322	0.355	1.606	1.788	0.944	1.008	0.944	1.007
S9	0.528	0.794	0.716	0.800	0.377	0.419	0.488	0.514	0.322	0.363	0.002	0.018	0.803	0.849	1.682	1.702	1.682	1.703
S10	0.000	0.000	0.000	0.001	0.000	0.001	0.000	0.001	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.000	0.000	0.000
S11	0.572	0.992	0.523	0.756	0.349	0.453	0.207	0.292	0.167	0.220	0.421	0.460	0.000	0.001	0.030	0.060	0.021	0.060
S12	0.030	0.063	0.000	0.001	0.019	0.024	0.111	0.178	0.017	0.053	0.000	0.000	0.110	0.297	0.283	0.311	0.283	0.319
Average	0.119	0.206	0.120	0.164	0.101	0.125	0.144	0.179	0.124	0.150	0.128	0.153	0.416	0.534	0.488	0.572	0.487	0.573

Table A6
Experimental result of various metaheuristic algorithm for medium sized problem sets.

Instances	DE/rand/1		DE/rand/2		DE/best/1		DE/best/2		DE/current-to-best/1		DE/current-to-pbest/1		MFO		LFMFO		GA	
	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE
M1	0	0.001	0	0.000	0	0.000	0	0.000	0	0.001	0	0.000	0	0.001	0	0.102	0.006	0.101
M2	0	0.404	0.303	0.512	0	0.051	0	0.048	0.02	0.086	0.042	0.137	0.033	0.048	0.008	0.014	0.017	0.018
M3	0	0.000	0	0.001	1.811	2.196	0.762	0.973	0	0.001	0	0.001	0	0.004	0	0.001	0.000	0.008
M4	3.03	3.458	1.569	1.575	0	0.000	0.003	0.027	0	0.053	3.814	4.020	4.103	4.755	3.167	3.223	3.167	3.223
M5	0	0.021	0.8	0.818	0	0.000	0.05	0.127	0.47	0.877	0.158	0.179	0	0.070	0	0.001	0.000	0.005
M6	1.131	1.567	2.095	2.117	1.079	1.314	0	0.000	0	0.015	0	0.001	0	0.003	0.01	0.061	0.010	0.060
M7	1.103	1.391	0.158	0.418	0.418	0.582	0.12	0.264	0.01	0.189	0.231	0.443	0.451	0.579	0.312	0.489	0.312	0.497
M8	1.738	2.085	0	0.084	0.052	0.104	2.541	2.662	3.38	3.872	0.623	0.818	3.012	3.282	4.58	4.725	4.580	4.727
M9	0	0.001	0	0.014	0	0.001	0	0.000	0.005	0.022	0	0.001	0	0.001	0	0.000	0.000	0.004
M10	0.126	0.309	0.885	0.893	0.02	0.040	0	0.032	0.003	0.051	0	0.074	0.069	0.110	2.351	2.500	2.352	2.504
M11	0.87	1.197	1.002	1.500	1.921	2.044	0.34	0.402	0	0.001	0.504	0.591	2.728	2.993	0.972	1.099	1.102	1.101
M12	0.201	0.320	0	0.052	0.003	0.017	1.119	1.395	0.17	0.207	0.01	0.034	0.03	0.041	0.005	0.032	0.0.33	0.031
Average	0.683	0.896	0.568	0.665	0.442	0.529	0.411	0.494	0.338	0.448	0.449	0.525	0.869	0.990	0.950	1.020	1.050	1.023

Table A7
Experimental result of various metaheuristic algorithm for large sized problem instances.

Instances	DE/rand/1		DE/rand/2		DE/best/1		DE/best/2		DE/current-to-best/1		DE/current-to-pbest/1		MFO		LFMFO		GA	
	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE	BRPE	ARPE
L1	2.871	3.007	0.749	0.911	2.113	2.851	0.981	1.147	0.000	0.000	1.105	1.749	0.155	0.243	0.311	0.719	0.314	0.720
L2	1.005	1.130	0.810	1.058	1.330	1.413	0.276	0.312	1.199	1.485	0.317	0.511	0.000	0.004	1.044	1.216	1.044	1.218
L3	0.000	0.084	1.032	1.490	0.000	0.034	1.522	1.829	0.784	1.002	2.698	2.879	1.330	1.837	2.871	3.032	2.878	3.033
L4	0.874	0.992	4.001	5.082	4.612	4.956	0.000	0.124	0.901	1.052	2.512	3.003	4.612	5.339	4.412	4.993	4.412	4.995
L5	0.855	0.901	0.988	1.213	2.733	3.009	0.000	0.030	2.366	2.876	1.003	1.217	1.981	2.012	0.127	0.278	0.000	0.280
L6	1.101	1.218	2.113	2.275	0.000	1.566	2.735	3.483	1.275	1.412	1.274	1.896	0.777	1.154	2.733	3.113	0.010	3.113
L7	0.540	0.693	0.349	0.617	0.784	1.293	1.139	1.805	0.873	1.109	0.000	0.417	1.032	1.723	1.864	2.349	1.864	2.354
L8	0.777	0.907	1.672	1.934	1.520	2.105	1.275	2.047	0.915	1.080	0.000	0.084	1.995	2.325	0.580	1.272	0.589	1.273
L9	1.275	1.333	0.939	1.109	0.986	1.001	0.931	1.701	0.540	0.721	0.855	1.013	4.001	4.419	3.700	4.139	3.704	4.142
L10	0.045	0.394	4.612	5.144	0.000	0.308	3.712	4.561	1.032	1.585	1.606	1.892	2.951	3.451	0.780	0.912	2.352	0.924
L11	0.951	1.011	0.660	0.790	1.275	1.397	0.540	0.600	0.000	0.000	0.048	0.499	0.922	1.052	1.842	2.660	1.102	2.667
L12	3.559	3.648	0.000	0.033	0.677	0.919	0.050	0.211	0.640	0.806	4.210	4.352	0.440	0.623	1.958	2.938	0.0.33	2.938
Average	1.154	1.277	1.494	1.805	1.336	1.738	1.097	1.488	0.877	1.094	1.302	1.626	1.683	2.015	1.154	1.277	1.661	2.304

Table A8
Deviation of various metaheuristic algorithm from the LB for three small sized problem instances.

Instances	DE/rand/1		DE/rand/2		DE/best/1		DE/best/2		DE/current-to-best/1		DE/current-to-pbest/1		MFO		LFMFO		GA	
	LBRPE	LARPE	LBRPE	LARPE	LBRPE	LARPE	LBRPE	LARPE	LBRPE	LARPE	LBRPE	LARPE	LBRPE	LARPE	LBRPE	LARPE	LBRPE	LARPE
S1	3.002	3.014	3.001	3.015	3.004	3.004	3.006	3.022	3.005	3.009	3.011	3.051	3.02	3.24	3.025	3.156	3.021	3.164
S2	3.008	3.057	3.008	3.049	3.006	3.007	3.004	3.023	3.013	3.021	3.014	3.029	3.035	3.111	3.045	3.114	3.044	3.124
S3	3.087	3.142	3.006	3.008	3.022	3.047	3.056	3.075	3.585	3.622	3.076	3.093	3.238	3.273	3.183	3.206	3.185	3.197
S4	3.005	3.05	3.013	3.022	3.312	3.343	3.006	3.006	3.123	3.2	3.066	3.087	3.983	4.048	3.87	3.934	3.866	3.926
S5	3.001	3.008	3.006	3.009	3.004	3.015	3.003	3.004	3.002	3.003	3.007	3.008	3.003	3.061	3.001	3.034	3.004	3.029
S6	3.11	3.178	3.174	3.268	3.122	3.163	3.695	3.743	3.277	3.31	3.637	3.744	3.281	3.322	3.358	3.486	3.359	3.488
S7	3.012	3.023	3.013	3.032	3.002	3.027	3.016	3.033	3.008	3.018	3.012	3.025	3.943	4.454	4.473	4.893	4.472	4.896
S8	3.113	3.203	3.023	3.05	3.027	3.04	3.171	3.294	3.011	3.017	3.327	3.359	4.613	4.796	3.95	4.011	3.945	4.011
S9	3.535	3.795	3.725	3.806	3.382	3.425	3.497	3.517	3.328	3.364	3.007	3.022	3.806	3.858	4.685	4.711	4.686	4.708
S10	3.001	3.008	3.002	3.006	3.005	3.006	3.003	3.009	3.003	3.003	3.003	3.004	3.004	3.008	3.003	3.004	3.006	3.007
S11	3.58	3.995	3.524	3.76	3.355	3.459	3.208	3.299	3.17	3.229	3.424	3.462	3.008	3.009	3.036	3.069	3.025	3.069
S12	3.034	3.064	3.005	3.01	3.024	3.032	3.114	3.182	3.02	3.054	3.001	3.002	3.111	3.303	3.29	3.315	3.288	3.327
Average	3.124	3.211	3.125	3.170	3.105	3.131	3.148	3.184	3.129	3.154	3.132	3.157	3.420	3.540	3.493	3.578	3.492	3.579

Table A9
Deviation of various Metaheuristic algorithm from the LB for three medium sized problem instances.

Instances	DE/rand/1		DE/rand/2		DE/best/1		DE/best/2		DE/current-to-best/1		DE/current-to-pbest/1		MFO		LFMFO		GA	
	LBRPE	LARPE	LBRPE	LARPE	LBRPE	LARPE	LBRPE	LARPE	LBRPE	LARPE	LBRPE	LARPE	LBRPE	LARPE	LBRPE	LARPE	LBRPE	LARPE
M1	5.502	5.504	5.501	5.503	5.502	5.504	5.501	5.503	5.504	5.504	5.503	5.504	5.503	5.504	5.504	5.605	5.508	5.602
M2	5.503	5.907	5.806	6.013	5.502	5.554	5.501	5.55	5.522	5.589	5.543	5.64	5.537	5.55	5.51	5.516	5.52	5.522
M3	5.501	5.504	5.503	5.504	7.312	7.697	6.264	6.477	5.503	5.504	5.503	5.504	5.504	5.508	5.503	5.505	5.502	5.509
M4	8.533	8.961	7.07	7.078	5.504	5.504	5.507	5.53	5.501	5.556	9.318	9.521	9.605	10.257	8.67	8.725	8.67	8.726
M5	5.504	5.524	6.301	6.321	5.502	5.504	5.553	5.63	5.971	6.378	5.661	5.68	5.503	5.574	5.502	5.502	5.504	5.509
M6	6.632	7.07	7.596	7.621	6.581	6.818	5.502	5.503	5.502	5.516	5.503	5.504	5.504	5.506	5.511	5.562	5.512	5.561
M7	6.607	6.894	5.66	5.92	5.922	6.083	5.621	5.767	5.512	5.692	5.734	5.947	5.953	6.08	5.813	5.991	5.815	6.00
M8	7.24	7.589	5.503	5.586	5.555	5.608	8.043	8.163	8.883	9.373	6.124	6.322	8.515	8.784	10.082	10.229	10.082	10.228
M9	5.502	5.504	5.503	5.515	5.502	5.502	5.502	5.502	5.507	5.525	5.504	5.504	5.504	5.504	5.503	5.504	5.503	5.508
M10	5.627	5.813	6.387	6.396	5.524	5.544	5.503	5.535	5.505	5.553	5.504	5.575	5.571	5.611	7.853	8.003	7.855	8.005
M11	6.374	6.699	6.505	7.001	7.422	7.545	5.842	5.905	5.504	5.505	6.008	6.095	8.232	8.494	6.473	6.6	6.602	6.606
M12	5.704	5.821	5.504	5.554	5.504	5.521	6.62	6.899	5.672	5.71	5.512	5.536	5.534	5.542	5.508	5.534	5.502	5.832
Average	6.186	6.399	6.070	6.168	5.944	6.032	5.913	5.997	5.841	5.950	5.951	6.028	6.372	6.493	6.453	6.523	6.465	6.551

Table A10
Deviation of Various Metaheuristic algorithm from the LB for three Large Sized Problem Instances.

Instances	DE/rand/1		DE/rand/2		DE/best/1		DE/best/2		DE/current-to-best/1		DE/current-to-pbest/1		MFO		LFMFO		GA	
	LBRPE	LARPE	LBRPE	LARPE	LBRPE	LARPE	LBRPE	LARPE	LBRPEL	LARPE	LBRPEL	LARPE	LBRPEL	LARPE	LBRPEL	LARPE	LBRPEL	LARPE
L1	9.874	10.008	7.753	7.913	9.115	9.853	7.984	8.148	7.003	7.003	8.107	8.752	7.156	7.244	7.315	7.723	7.317	7.723
L2	8.008	8.131	7.811	8.059	8.334	8.417	7.279	7.313	8.2	8.486	7.32	7.513	7.002	7.006	8.047	8.22	8.045	8.221
L3	7.002	7.088	8.033	8.492	7.004	7.038	8.526	8.83	7.788	8.004	9.701	9.88	8.331	8.838	9.874	10.033	9.88	10.037
L4	7.876	7.996	11.004	12.084	11.616	11.959	7.001	7.126	7.903	8.054	9.513	10.006	11.614	12.343	11.416	11.995	11.416	11.997
L5	7.856	7.904	7.989	8.217	9.736	10.011	7.002	7.032	9.37	9.88	8.006	8.219	8.984	9.016	7.131	7.279	7.002	7.281
L6	8.105	8.221	9.116	9.279	7.001	8.568	9.739	10.486	8.278	8.414	8.278	8.898	7.781	8.157	9.737	10.114	7.013	10.116
L7	7.542	7.694	7.352	7.62	7.785	8.295	8.142	8.808	7.874	8.11	7.004	7.421	8.036	8.726	8.866	9.351	8.865	9.356
L8	7.779	7.909	8.676	8.936	8.523	9.106	8.278	9.05	7.916	8.081	7.003	7.085	8.997	9.329	7.581	8.275	7.592	8.276
L9	8.278	8.337	7.941	8.11	7.988	8.004	7.935	8.705	7.542	7.724	7.857	8.016	11.005	11.421	10.703	11.14	10.706	11.143
L10	7.049	7.398	11.616	12.148	7.002	7.309	10.714	11.563	8.036	8.589	8.607	8.896	9.953	10.454	7.783	7.915	7.927	9.353
L11	7.952	8.013	7.661	7.792	8.278	8.401	7.543	7.603	7.002	7.003	7.05	7.502	7.923	8.053	8.846	9.664	8.105	9.67
L12	10.563	10.651	7.003	7.035	7.681	7.921	7.051	7.212	7.644	7.808	11.213	11.353	7.441	7.626	8.96	9.941	7.002	7.331
Average	8.157	8.279	8.496	8.807	8.339	8.740	8.100	8.490	7.880	8.096	8.305	8.628	8.685	9.018	8.855	9.304	8.406	9.209

Table A11
Performance of all algorithms for the case study.

No: of products	Number of customers	Total costs ($)															
		DE/rand/1		DE/rand/2		DE/best/1		DE/best/2		DE/current-to-best/1		DE/current-to-pbest/1		MFO		LFMFO		GA	
		Best	Average	Best	Average	Best	Average	Best	Average	Best	Average	Best	Average	Best	Average	Best	Average	Best	Average
50	5	1051	1179.167	930	966.833	930	966.83	951	973.33	898	938	960	1001	1112	1151.5	1101	1102.66		
1098	1158.74																		

approach. The efficient approach should provide the best for the integrated production and distribution decisions to minimize the total costs in flow shop business. Hence, firstly, this paper develops a mathematical model to describe the problem. Secondly, three metaheuristic based algorithms have been proposed to address the problem, namely Differential Evolution (DE) algorithm with the six (6) popular mutation operators, Moth Flame Optimization (MFO) and Lévy Flight Moth Flame Optimization (LFMFO).

Experimental investigations suggest that DE algorithm variants outperform MFO and its relatively new hybrid LFMFO, this is apparent when the product size varies from small to medium sized. This is most likely due to the spiral motion of the candidate solution known as the 'moth position' in Moth Flame Optimization which means there are less exploration and exploitation potential as the iteration approaches the maximum iteration. Furthermore, although the Lévy Flight Function allows LFMO algorithm to maintain its exploration capability, the diminishing spiral motion of the 'moth position' does not allow sufficient exploitation in a multimodal environment such as PFSPs. However, this can be mitigated by increasing the population size of moth in MFO and LFMFO.

For the managerial implications, the developed model and algorithms can be employed by the production managers to improve the efficiency of production, reduce the cost of distribution, and therefore, it makes the manufacturers competitive. The developed algorithms will also help the managers obtain satisfying schedule within a reasonable computational time. Therefore, it can act as an essential decision-making tool for the managers. In future work, the study can be expanded to investigate the inclusion of other realistic settings, e.g., buffer capacity constraints, labor costs, process interruptions, inventory costs and shortage of material supply to further simulate the real manufacturing circumstances. Furthermore, since the energy consumption is also an important aspect in decision making, the proposed approaches could be implemented to further expand to the research on this topic. This will enable the wider application of multi-objective optimization algorithms to assist decision making in the manufacturing industry.

Appendix A

See Figs. A1-A4.
See Tables A1-A11.

References

AbdElhamid, H., Helmi, A. & Ziedan, I. (2018). LCMFO: An Improved Moth-Flame Algorithm for Combinatorial Optimization Problems. / International Journal of Engineering and Technology.

Abreu, L. R., Cunha, J. O., Prata, B. A., & Framinan, J. M. (2020). A genetic algorithm for scheduling open shops with sequence-dependent setup times. *Computers & Operations Research, 113*, Article 104793.

Ahmadizar, F., & Farhadi, S. (2015). Single-machine batch delivery scheduling with job release dates, due windows and earliness, tardiness, holding and delivery costs. *Computers & Operations Research, 53*, 194–205.

Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research, 246*(2), 345–378.

Bansal, S. (1977). Minimizing the sum of completion times of n jobs over m machines in a flowshop—A branch and bound approach. *AIIE Transactions, 9*(3), 306–311.

Buch, H., Trivedi, I. N., & Jangir, P. (2017). Moth flame optimization to solve optimal power flow with non-parametric statistical evaluation validation. *Cogent Engineering, 4*(1), 1286731.

Cakici, E., Mason, S. J., & Kurz, M. E. (2012). Multi-objective analysis of an integrated supply chain scheduling problem. *International Journal of Production Research, 50*(10), 2624–2638.

Chen, J., Huang, G. Q., Luo, H., & Wang, J. (2015). Synchronisation of production scheduling and shipment in an assembly flowshop. *International Journal of Production Research, 53*(9), 2787–2802.

Cheng, B.-Y., Leung, J. Y.-T., & Li, K. (2015). Integrated scheduling of production and distribution to minimize total cost using an improved ant colony optimization method. *Computers & Industrial Engineering, 83*, 217–225.

Cheng, T., & Kahlbacher, H. (1993). Scheduling with delivery and earliness penalties. *Asia-Pacific Journal of Operational Research, 10*(2), 145–152.

Choi, T.-M. (2015). Supply chain systems coordination with multiple risk sensitive retail buyers. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 46*(5), 636–645.
Garey, M. R., Johnson, D. S., & Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of operations research, 1*(2), 117–129.
Ghobaei-Arani, M., Souri, A., Safara, F., & Norouzi, M. (2020). An efficient task scheduling approach using moth-flame optimization algorithm for cyber-physical system applications in fog computing. *Transactions on Emerging Telecommunications Technologies, 31*(2), Article e3770.
Guo, Z., Zhang, D., Leung, S. Y.-S., & Shi, L. (2016). A bi-level evolutionary optimization approach for integrated production and transportation scheduling. *Applied Soft Computing, 42*, 215–228.
Hamidinia, A., Khakabimamaghani, S., Mazdeh, M. M., & Jafari, M. (2012). A genetic algorithm for minimizing total tardiness/earliness of weighted jobs in a batched delivery system. *Computers & Industrial Engineering, 62*(1), 29–38.
Han, D., Tang, Q., Zhang, Z., & Cao, J. (2020). Energy-efficient integration optimization of production scheduling and ladle dispatching in steelmaking plants. *IEEE Access*.
Hidri, L., & Gharbi, A. (2017). New efficient lower bound for the hybrid flow shop scheduling problem with multiprocessor tasks. *IEEE Access, 5*, 6121–6133.
Ho, Y.-C., & Pepyne, D. L. (2002). Simple explanation of the no-free-lunch theorem and its implications. *Journal of Optimization Theory and Applications, 115*(3), 549–570.
Johnson, S. M. (1954). Optimal two-and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly, 1*(1), 61–68.
Kamaruzaman, A.F., Zain, A.M., Yusuf, S.M. & Udin, A. (2013). Levy flight algorithm for optimization problems-a literature review. In: Applied Mechanics and Materials (Vol. 421, pp. 496-501): Trans Tech Publ.
Kazemi, H., Mazdeh, M. M., & Rostami, M. (2017). The two stage assembly flow-shop scheduling problem with batching and delivery. *Engineering Applications of Artificial Intelligence, 63*, 98–107.
Khalilpourazari, S., & Khalilpourazary, S. (2019). An efficient hybrid algorithm based on Water Cycle and Moth-Flame Optimization algorithms for solving numerical and constrained engineering optimization problems. *Soft Computing, 23*(5), 1699–1722.
Kizilay, D., Tasgetiren, M. F., Pan, Q.-K., & Gao, L. (2019). A variable block insertion heuristic for solving permutation flow shop scheduling problem with makespan criterion. *Algorithms, 12*(5), 100.
Li, Y., Li, X., Gao, L., & Meng, L. (2020). An improved artificial bee colony algorithm for distributed heterogeneous hybrid flowshop scheduling problem with sequence-dependent setup times. *Computers & Industrial Engineering, 147*, Article 106638.
Li, Z., Janardhanan, M. N., Tang, Q., & Nielsen, P. (2016). Co-evolutionary particle swarm optimization algorithm for two-sided robotic assembly line balancing problem. *Advances in Mechanical Engineering, 8*(9), 1–14.
Li, Z., Janardhanan, M. N., Tang, Q., & Ponnambalam, S. (2019). Model and metaheuristics for robotic two-sided assembly line balancing problems with setup times. *Swarm and Evolutionary Computation, 50*, Article 100567.
Liu, G.-S., Zhou, Y., & Yang, H.-D. (2017). Minimizing energy consumption and tardiness penalty for fuzzy flow shop scheduling with state-dependent setup time. *Journal of Cleaner Production, 147*, 470–484.
Liu, P., & Lu, X. (2016). Integrated production and job delivery scheduling with an availability constraint. *International Journal of Production Economics, 176*, 1–6.
Mazdeh, M. M., Rostami, M., & Namaki, M. H. (2013). Minimizing maximum tardiness and delivery costs in a batched delivery system. *Computers & Industrial Engineering, 66*(4), 675–682.
Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems, 89*, 228–249.
Mishra, A., & Shrivastava, D. (2018). A TLBO and a Jaya heuristics for permutation flow shop scheduling to minimize the sum of inventory holding and batch delay costs. *Computers & Industrial Engineering, 124*, 509–522.
Mohseni, S., Brent, A. C., & Burmester, D. (2019). A demand response-centred approach to the long-term equipment capacity planning of grid-independent micro-grids optimized by the moth-flame optimization algorithm. *Energy Conversion and Management, 200*, Article 112105.
Nearchou, A. C., & Omirou, S. L. (2006). Differential evolution for sequencing and scheduling optimization. *Journal of Heuristics, 12*(6), 395–411.
Onwubolu, G., & Davendra, D. (2006). Scheduling flow shops using differential evolution algorithm. *European Journal of Operational Research, 171*(2), 674–692.
Öztop, H., Tasgetiren, M. F., Eliiyi, D. T., Pan, Q.-K., & Kandiller, L. (2020). An energy-efficient permutation flowshop scheduling problem. *Expert Systems with Applications, 150*, Article 113279.
Rahman, H. F., Janardhanan, M. N., & Nielsen, I. E. (2019). Real-Time Order Acceptance and Scheduling Problems in a Flow Shop Environment Using Hybrid GA-PSO Algorithm. *IEEE Access, 7*, 112742–112755.
Rahman, H. F., & Nielsen, I. (2019). Scheduling automated transport vehicles for material distribution systems. *Applied Soft Computing, 105552*.
Rahman, H. F., Sarker, R., & Essam, D. (2015a). A genetic algorithm for permutation flow shop scheduling under make to stock production system. *Computers & Industrial Engineering, 90*(Supplement C), 12–24.
Rahman, H. F., Sarker, R., & Essam, D. (2015b). A real-time order acceptance and scheduling approach for permutation flow shop problems. *European Journal of Operational Research, 247*(2), 488–503.
Rahman, H. F., Sarker, R., & Essam, D. (2017). A genetic algorithm for permutation flowshop scheduling under practical make-to-order production system. *AI EDAM, 31*(1), 87–103.
Rahman, H. F., Sarker, R., & Essam, D. (2018). Multiple-order permutation flow shop scheduling under process interruptions. *The International Journal of Advanced Manufacturing Technology*, 1–28.
Rashid, M. F. F. A., Rose, A. N. M., Mohamed, N. M. Z. N., & Romlay, F. R. M. (2019). Improved moth flame optimization algorithm to optimize cost-oriented two-sided assembly line balancing. *Engineering*. Computations.
Ruiz, R., & Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research, 165*(2), 479–494.
Ruiz, R., & Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research, 169*(3), 781–800.
Ruiz, R., Maroto, C., & Alcaraz, J. (2005). Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristics. *European Journal of Operational Research, 165*(1), 34–54.
Ruiz, R., Maroto, C., & Alcaraz, J. (2006). Two new robust genetic algorithms for the flowshop scheduling problem. *Omega, 34*(5), 461–476.
Santucci, V., Baioletti, M., & Milani, A. (2016). Solving permutation flowshop scheduling problems with a discrete differential evolution algorithm. *AI Communications, 29*(2), 269–286.
Sayed, G. I., & Hassanien, A. E. (2018). A hybrid SA-MFO algorithm for function optimization and engineering design problems. *Complex & Intelligent Systems, 4*(3), 195–212.
Sbihi, A., Bellabdaoui, A., & Teghem, J. (2014). Solving a mixed integer linear program with times setup for the steel-continuous casting planning and scheduling problem. *International Journal of Production Research, 52*(24), 7276–7296.
Sbihi, A., & Chemangui, M. (2018). A genetic algorithm for the steel continuous casting with inter-sequence dependent setups and dedicated machines. *RAIRO-Operations Research, 52*(4–5), 1351–1376.
Selen, W. J., & Hott, D. D. (1986). A mixed-integer goal-programming formulation of the standard flow-shop scheduling problem. *Journal of the Operational Research Society, 37*(12), 1121–1128.
Selvarajah, E., & Zhang, R. (2014). Supply chain scheduling at the manufacturer to minimize inventory holding and delivery costs. *International Journal of Production Economics, 147*, 117–124.
Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization, 11*(4), 341–359.
Trivedi, I., Kumar, A., Ranpariya, A. H., & Jangir, P. (2016). In *Economic Load Dispatch problem with ramp rate limits and prohibited operating zones solve using Levy flight Moth-Flame optimizer* (pp. 442–447). IEEE.
Valizadeh, S., Fatahi Valilai, O., & Houshmand, M. (2019). Flexible flow line scheduling considering machine eligibility in a digital dental laboratory. *International Journal of Production Research*, 1–19.
Vincent, L. W. H., Ponnambalam, S., & Kanagaraj, G. (2014). Differential evolution variants to schedule flexible assembly lines. *Journal of Intelligent Manufacturing, 25*(4), 739–753.
Wan, L., & Zhang, A. (2014). Coordinated scheduling on parallel machines with batch delivery. *International Journal of Production Economics, 150*, 199–203.
Wang, G., & Cheng, T. E. (2000). Parallel machine scheduling with batch delivery costs. *International Journal of Production Economics, 68*(2), 177–183.
Wang, K., Luo, H., Liu, F., & Yue, X. (2017). Permutation flow shop scheduling with batch delivery to multiple customers in supply chains. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 48*(10), 1826–1837.
Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation, 1*(1), 67–82.
Yang, X. (2000). Scheduling with generalized batch delivery dates and earliness penalties. *Iie Transactions, 32*(8), 735–741.
Zhou, B., & Wu, Q. (2019). An improved immune clonal selection algorithm for bi-objective robotic assemble line balancing problems considering time and space constraints. *Engineering Computations, 36*(6), 1868–1892.
Zobolas, G., Tarantilis, C. D., & Ioannou, G. (2009). Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm. *Computers & Operations Research, 36*(4), 1249–1267.