Computers and Operations Research 104 (2019) 127–138

Contents lists available at ScienceDirect

Computers and Operations Research

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

Reformulation, linearization, and a hybrid iterated local search algorithm for economic lot-sizing and sequencing in hybrid flow shop problems

Hassan Zohali[a], Bahman Naderi[a,b,*], Mohammad Mohammadi[a], Vahid Roshanaei[c]

[a] *Department of Industrial Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran*
[b] *Department of Mechanical, Automotive & Materials Engineering, Faculty of Engineering, University of Windsor, Windsor, Canada*
[c] *Department of Operations Management & Statistics, Rotman School of Management, University of Toronto, Ontario, Canada*

A R T I C L E   I N F O

Article history:
Received 10 October 2017
Revised 7 November 2018
Accepted 6 December 2018
Available online 8 December 2018

Keywords:
Lot-sizing
Sequencing
Hybrid flow shop
Mixed-integer nonlinear program
Linearization
Mixed-integer linear program
Hybrid iterated local search

A B S T R A C T

We study an integrated economic lot-sizing and sequencing problem (ELSP) in the hybrid flow shop manufacturing setting with unlimited intermediate buffers in a finite planning horizon. The ELSP entails making two simultaneous decisions regarding (i) the manufacturing sequences of products, and (ii) their production quantity. The objective is to minimize the total cost, consisting of inventory holding and set-up costs. To solve this problem, we first develop a novel mixed-integer nonlinear programming (MINLP) model that improves an existing MINLP model in the literature. We then present a novel linearization technique that transforms these two MINLP models into effective mixed-integer linear programming (MILP) models. Additionally, we develop an effective algorithm that hybridizes the iterated local search algorithm with an approximate function. We conduct comprehensive experiments to compare the performance of MILPs+CPLEX with that of MINLPs+BARON. Additionally, our proposed algorithm is compared with four existing metaheuristic algorithms in the literature. Computational results demonstrate that our novel MINLP formulation and its linearized variant significantly improve the solvability and optimality gap of an existing MINLP formulation and its linearized variant. We also show that our new hybrid iterated local search algorithm substantially improves computational performance and optimality gap of the mathematical models and the existing algorithms in the literature, on large-size instances of the problem.

1. Introduction

The economic lot-sizing and sequencing problem (ELSP) is one of the well-recognized production planning problems belonging to the medium-term, or say tactical, decision making. The ELSP has attracted significant attention after the pioneering work of Rogers (1958). The ELSP entails making two simultaneous decisions regarding (i) the manufacturing sequences of products and (ii) their production quantity. The classic ELSP literature assumes there is only a single machine that processes all products. In practice, however, we often encounter shops in which a series of operations must be carried out on different machines in different processing stages of a production line. The flow shop as a more realistic variant of a single machine environment is one of the widely studied problems in the literature of production management. In a flow shop environment, machines are disposed serially in different stages of a production line, and all jobs must be processed on all machines (there is only one machine in each stage). Hsu and El-Najdawi (1990) were the first ones to extend the classic single-machine ELSP to the flow shop ELSP (or ELSP-FS). Note that the ELSP differs from the lot-streaming problems wherein each job can be split into multiple lots (Sang et al., 2014; Zhang et al., 2017; Sang et al., 2018). Lot-streaming problems are scheduling problems with time-oriented objective functions (e.g., makespan), while the ELSP is a cost-oriented inventory model.

Having scrutinized the literature of the ELSP, we noticed a paradigm shift after 1999 in which sequences of products are recognized to impact the total cost (Xiao et al., 2015). As such, before 1999, most researchers solved the ELSP by minimizing the total cost for a given sequence of products, usually obtained by a simple heuristic (El-Najdawi and Kleindorfer (1993)). The authors make use of the shortest processing time (SPT) dispatching rule to obtain a sequence of products. Their core focus is on the lot-sizing decisions rather than on the sequencing decisions. The work of Dobson and Yano (1994) and El-Najdawi (1994) also ignores the

* Corresponding author.
E-mail address: bahman.naderi@khu.ac.ir (B. Naderi).

https://doi.org/10.1016/j.cor.2018.12.008
0305-0548/© 2018 Elsevier Ltd. All rights reserved.

influence of sequencing on lot-sizing decisions. Therefore, before 1999, all studies ignore sequencing decisions.

Ouenniche et al. (1999) study the ELSP-FS and explore the effect of sequencing decisions on the total cost. The authors show that the manufacturing sequences of parts significantly impact the total cost, concluding that it must be integrated with lot-sizing decisions. Thus, they present the first integrated mixed-integer nonlinear programming (MINLP) model, and also develop multiple heuristics, and two metaheuristics: simulated annealing and tabu search.

While ELSP has been studied in different production systems, its applications to practical production systems are very sparse. In fact, a recent survey (Santander-Mercado and Jubiz-Diaz (2016)) demonstrates that the primary focus of the ELSP literature has been on single-machine settings (Kayvanfar and Zandieh, 2012; Yan et al., 2013; Bulut and Tasgetiren, 2014; Mohammadi et al., 2015; Beck and Glock, 2016; Sağlam and Banerjee, 2018). This survey also highlights the need for studying more practical variants of the ELSP including hybrid flow shops (HFS): a flow shop setting with duplicated machines at each stage.

There are few applications of ELSP in HFS settings. Torabi et al. (2006) are the first ones who develop a novel MINLP model for the ELSP-HFS. Their MINLP model extends the MINLP model of Ouenniche et al. (1999) by allowing jobs to be flexibly assigned to machines at each processing stage. Due to the intractability of their MINLP model, the authors develop a hybrid genetic algorithm. Later, Jenabi et al. (2007) extend their MINLP model to ELSP-HFS with unrelated parallel machines. Similarly, the authors develop two meta-heuristics: a simulated annealing and a genetic algorithm for this problem. As a more realistic variant of the ELSP, Akrami et al. (2006) consider a constraint on the maximum size of the buffer for the ELSP-HFS, and develop a new MINLP model. Their MINLP model is also based on the MINLP model proposed by Torabi et al. (2006). Additionally, the authors develop two metaheuristics based on the genetic algorithm and tabu search for their problem.

In this paper, similar to Torabi et al. (2006), we focus on the ELSP in the HFS setting and aim to improve the methodologies developed for this problem. To the best of our knowledge, there is no fully linear model for the ELSP-HFS setting that can solve the realistically-sized instances of the problem to a reasonable optimality gap. To bridge these gaps, this research offers the following contributions to the literature:

- We first develop a novel MINLP model that improves the existing MINLP model of Torabi et al. (2006) that has been developed for the ELSP-HFS setting. We then present a novel linearization technique that transforms the previous two MINLP models into two tractable mixed-integer linear programming (MILP) models. We demonstrate that our MILPs have much simpler structures and hence easier to communicate to production managers. The linearization concept used in our MILP models can be generalized to ELSPs in different shop floors.
- To solve realistically-sized instances of the problem, we develop an efficient hybrid iterated local search (HILS) algorithm and compare its performance with our MINLP and MILP models and four meta-heuristic algorithms from the literature: two algorithms from Akrami et al. (2006), and Torabi et al. (2006), and two other recent metaheuristics from the HFS literature: iterated greedy with iteration jumping algorithm by Tasgetiren et al. (2017) and discrete artificial bee colony (DABC) by Pan et al. (2017a, 2017b).

We show that our new MINLP and MILP models solved via existing optimization solvers outperform their counterparts in terms of solvability and optimality gap. Since the solvability of the mathematical models is limited to small-to-medium instances of the problem, we solve the problem using our HILS algorithm that significantly outperforms the mathematical models and other existing metaheuristic algorithms on large instances of the problem.

2. Problem formulation

Consider a set of N products (jobs) that needs to be processed on M work stages. Each work stage j has o_j identical machines. The processing route for all these N products are the same from stage 1 to stage M. Each machine in each stage can process at most one product at a time. Both demand and production of each product follow constant rates and are known in advance. External demand is only for final products. All demands must be satisfied, and no shortage is allowed. Some setup is carried out before the production can start. The setup could be influential due to its time and cost magnitude. We assume a non-preemptive production, i.e., once the production of a lot starts at a stage it cannot be interrupted until it is fully completed. The process on the next stage can start only if the entire lot is completed at the current stage. The lots of each product are of equal sizes at different stages. Moreover, the production horizon is assumed to be continuous and finite. The objective is to specify a production schedule that minimizes long-run average total cost, i.e., the sum of setup and inventory holding costs. The cyclic schedule is frequently assumed to be the common cycle: the production cycle times of all products are the same. Table 1 summarizes the notation used in mathematical models.

It is commonly assumed that the inventory holding cost is a non-decreasing function of the stage number: $H_{i,j-1} \leq H_{i,j},\ H_{i,j} < H_{i,M};\ \ i=1, \ldots, N,\ j=2, \ldots, M-1.$

The objective is to minimize the average total cost (TC) per time unit, including the inventory holding and setup costs. The average setup cost for all products per time unit is as follow:

$$cost_{setup} = \sum_{i=1}^{N} \frac{SC_i}{t}. \tag{1}$$

We now calculate the inventory holding costs. We consider two types of inventories: (i) the *work-in-process*, and (ii) *the final stage*. The inventory level of product i at different stages is shown in Fig. 1 (Ouenniche et al., 1999).

The inventory between stages j and $j+1$ ($j<M$) is the work-in-process (Fig. 1(a)). After starting the process of product i at stage j, the stock level of this product gradually increases to q_i (i.e., the determined lot size). When the production of this lot is completed, products of the lot can be gradually moved to the next stage (the inventory level decreases) if any of the processors at the next stage is available. Otherwise, they must wait for a processor to become available (Fig. 1(a)). Note that the maximum inventory level of product i at the final stage is less than q_i, because the delivery of final products to customers is continuous (Fig. 1(b)). Note that the production of each product in each cycle starts when its stock level reaches zero. Therefore, the difference between the starting and ending times of each product at each stage is equal to the cycle time.

According to Fig. 1(a), the average work-in-process inventory holding cost for all products per unit of time is:

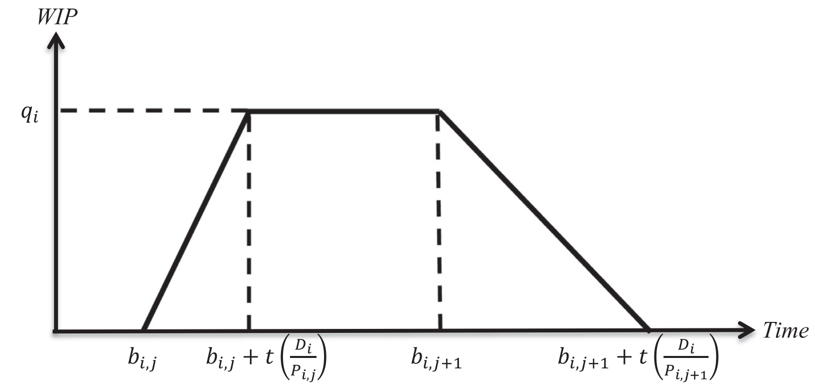
$$cost_{WIP} = \sum_{i=1}^{N} \sum_{j=1}^{M-1} H_{i,j} \cdot D_i \left(\left(b_{i,j+1}\right) + t\left(\frac{D_i}{2P_{i,j+1}}\right) - \left(b_{i,j}\right) - t\left(\frac{D_i}{2P_{i,j}}\right) \right). \tag{2}$$

Also, from Fig. 1(b), the average inventory holding cost for all final products per time unit is:

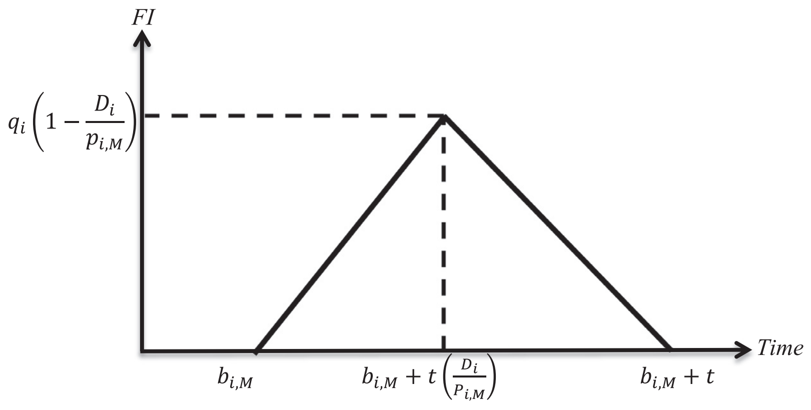
$$cost_{FP} = \sum_{i=1}^{N} H_{i,M} \cdot \frac{D_i}{2}\left(1 - \frac{D_i}{P_{i,M}}\right)t. \tag{3}$$

Table 1
Notations.

Parameters and indexes	
N	Number of products
M	Number of stages
i, k	Indices for products ($i = 1, \ldots, N$; $k = 1, \ldots N$)
j	Index for stages ($j = 1, \ldots, M$)
O_j	Number of identical parallel machines in stage j
r	Index for machines in each stage ($r = 1, \ldots, O_j$)
D_i	Demand rate of product i
$P_{i,j}$	Production rate of product i in stage j
$H_{i,j}$	Inventory holding cost per unit of product i per time unit between stages j and $j+1$
$S_{i,j}$	Setup time of product i in stage j
SC_i	Sum of setup costs of product i over all stages
PH	Planning horizon length
L	A large positive number
Decision variables	
t	Common cycle length
f	Number of common cycles upon the planning horizon
q_i	Production lot-size of product i at different stages ($q_i = t \bullet D_i$)
$b_{i,j}$	Starting time of product i at stage j
$c_{i,j}$	Completion time of product i at stage j
$z_{i,j,r}$	$\begin{cases} 1 \; if \; product \; i \; is \; processed \; on \; the \; rth \; machine \; in \; stage \; j \\ 0 \; otherwise \end{cases}$
$x_{i,j,k,r}$	$\begin{cases} 1 \; if \; product \; i \; is \; processed \; after \; product \; k \; on \; machine \; r \; in \; stage \; j \\ 0 \; otherwise \end{cases}$



(a) The work-in-process inventory of product i between stages j and $j+1$



(b) The final product inventory for product i

Fig. 1. The inventory levels, (a) The work-in-process inventory of product i between stages j and $j+1$, (b) The final product inventory for product i.

Therefore, the objective function (i.e., the total cost per time unit) is as follow:

$$TC_1 = \sum_{i=1}^{N} \frac{SC_i}{t} + \sum_{i=1}^{N} \left[H_{i,M} \cdot \frac{D_i}{2} \left(1 - \frac{D_i}{P_{i,M}} \right) + \sum_{j=1}^{M-1} H_{i,j} \cdot \frac{D_i^2}{2} \times \left(\frac{1}{P_{i,j+1}} - \frac{1}{P_{i,j}} \right) \right] t + \sum_{i=1}^{N} \sum_{j=1}^{M-1} H_{i,j} \cdot D_i \left(b_{i,j+1} - b_{i,j} \right). \quad (4)$$

3. Mathematical modeling

In this section, we first develop a new MINLP model, which is a reformulation of the MINLP model of Torabi et al. (2006) for the ELSP-HFS problems. The authors use a position-based modeling paradigm, in which the sequencing decision variable assigns activities to priority positions on each machine at each stage of the production line. Such a product sequence modeling paradigm generates an excessive number of constraints for hybrid flowshops. This paradigm is designed for flowshop that all products are processed by the same machine at each stage. In the flowshop problems, the product in the *kth* position can be processed only when the product in the preceding position $k-1$ is completed. In the hybrid flowshop problems, each product needs to visit a machine in each stage. Therefore, there may be no precedence relationship between products in two consecutive positions on different machines in each stage. As such, the start of the product in position k must be checked for all preceding positions from the beginning to $k-1$. Another sequence modeling paradigm has been proposed in Manne (1960) and it commonly performs better for sequencing problems. The Manne-based sequence modeling paradigm does not require the determination of the direct successor or predecessor of an activity. Thus, we first develop a Manne-based MINLP model and then we introduce a linearization procedure to transform the MINLP model into a MILP model. Since the source of non-linearity for both the *position-based* (Torabi et al. (2006)) and *Manne-based* models are the same, we adopt a common approach to linearize them.

3.1. Mixed-integer nonlinear programming model

Our Manne-based MINLP model is as follows: minimize TC_1 subject to:

$$\sum_{r=1}^{O_j} z_{i,j,r} = 1 \quad \forall_{i,j} \tag{5}$$

$$x_{i,j,k,r} + x_{k,j,i,r} \geq z_{i,j,r} + z_{k,j,r} - 1 \quad \forall_{i \neq k,j,r} \tag{6}$$

$$b_{i,j} \geq c_{k,j} + S_{i,j} - L\left(1 - x_{i,j,k,r}\right) \quad \forall_{i \neq k,j,r} \tag{7}$$

$$b_{i,j} + t\left(\frac{D_i}{P_{i,j}}\right) = c_{i,j} \quad \forall_{i,j} \tag{8}$$

$$c_{i,j} \leq b_{i,j+1} \quad \forall_{i,j<M} \tag{9}$$

$$c_{i,M} \leq t \quad \forall_i \tag{10}$$

$$b_{i,j} \geq S_{i,j} \quad \forall_{i,j} \tag{11}$$

$$f \cdot t = PH \tag{12}$$

$$b_{i,j}, c_{i,j}, \ t \ \geq 0 \quad \forall_{i,j} \tag{13}$$

$$x_{i,j,k,r} \ \in \{0,1\} \quad \forall_{i \neq k,j,r} \tag{14}$$

$$z_{i,j,r} \ \in \{0,1\} \quad \forall_{i,j,r} \tag{15}$$

$$f \geq 1 \textit{ and integer} \tag{16}$$

Constraint set (5) ensures products are uniquely assigned to one of the machines available at each stage. Constraint set (6) specifies that if two products are assigned to the same machine in a certain stage, they must be sequenced. Constraint set (7) ensures that on each machine, a product cannot be processed before the departure of its predecessor in the corresponding production sequence. Constraint set (8) guarantees the difference between the starting and the completion times of each product in each stage is equal to the corresponding processing time. Constraint set (9) enforces that no product can be processed before it is departed from the previous stage. Constraint set (10) assures that the cyclic time has been met. Constraint set (11) achieves that the processing of each product at each stage cannot start before its setup is done at the corresponding stage. Constraint (12) shows that the planning horizon is divisible by the common cycle. Finally, the domains of decision variables are defined by Constraint sets (13)–(16).

Our MINLP model requires few continuous variables by using two two-indexed continuous variables b_{ij} and c_{ij} to determine the beginning and completion times of each product in each stage.

3.2. Model linearization

The MINLP model presented in Section 3.1 has two nonlinear terms: (i) the objective function (*TC1*), and (ii) constraint (12). In the objective function, the first term is nonlinear ($\sum_{i=1}^{N} \frac{SC_i}{t}$). To linearize the objective function, the average setup cost is calculated using a novel approach. *PH* represents the number of time units in the planning horizon and f represents the number of production setups for each product. Therefore, the average number of production setups in time units for product i is equal to f/PH. Also, the production setup cost of product i for each setup operation is equal to SC_i. As a result, the average setup cost of product i per time unit is equal to $SC_i \times f/PH$. Finally, the average setup cost per time unit for all products is:

$$cost_{setup} = \frac{f}{ph} \sum_{(i=1)}^{N} SC_i. \tag{17}$$

Therefore, the linearized objective function of the problem is as follow:

$$TC_2 = \frac{f}{PH}\sum_{i=1}^{N} SC_i + \sum_{i=1}^{N}\left[H_{i,M}\cdot\frac{D_i}{2}\left(1-\frac{D_i}{P_{i,M}}\right)+\sum_{j=1}^{M-1}H_{i,j} \cdot\frac{D_i^2}{2}\left(\frac{1}{P_{i,j+1}}-\frac{1}{P_{i,j}}\right)\right]t+\sum_{i=1}^{N}\sum_{j=1}^{M-1}H_{i,j}\cdot D_i\left(b_{i,j+1}-b_{i,j}\right). \tag{18}$$

We now linearize Constraint (12). In Constraint (12), we multiply an integer variable (f) by a continuous variable (t) to generate common cycles for a finite planning horizon (*PH*). This constraint is to divide *PH* into integer number of common cycles. To linearize Constraint (12), we perform the following steps:

Step 1. Calculate the upper bound on the optimal number of common cycles: F_{max}.

Step 2. Replace variable f in the objective function (18) with $\sum_{l=1}^{F_{max}} l \cdot y_l$, where y_l is an auxiliary binary variable taking value 1 if the number of common cycles in the planning horizon is equal to l, and; 0 otherwise.

Step 3. Replace the nonlinear Constraint (12) with the linear Constraint sets (19)–(22):

$$\sum_{l=1}^{F_{\max}} y_l = 1 \tag{19}$$

$$PH \cdot y_l \leq l \cdot t \quad \forall_{l=1,2,\ldots,F_{\max}} \tag{20}$$

$$l \cdot t \leq PH + PH(l-1)\cdot(1-y_l) \quad \forall_{l=1,2,\ldots,F_{max}} \tag{21}$$

$$y_l \ \in \{0,1\} \quad \forall_{l=1,2,\ldots,F_{\max}} \tag{22}$$

In step 1, to calculate $F_{\max}$, we need to have the lower bound on the optimal common cycle length, $T_{\min}$. Then, from the definition, we have

$$F_{\max} = \frac{PH}{T_{\min}}, \tag{23}$$

where $\frac{PH}{T_{\min}}$ is the smallest integer that is equal to or greater than $\frac{PH}{T_{\min}}$.

To calculate $T_{\min}$, we use an approach similar to Ouenniche and Bertrand (2001). In the proposed MINLP model, we relax Constraint sets (5)–(7), (10)–(12), and (14)–(16) and assume the products are scheduled independently of each other and the processing start time of each product at each stage is assumed to be equal to its completion time at the previous stage:

$$c_{i,j} = b_{i,j+1} \text{ or } b_{i,j+1} - b_{i,j} = t\left(\frac{D_i}{P_{i,j}}\right) \quad \forall_{i,j<M}. \tag{24}$$

Thus, by replacing Eq. (24) in TC_1, the lower bound on the total cost is computed as follows:

$$TC_{LB} = \sum_{i=1}^{N}\frac{SC_i}{t} + \sum_{i=1}^{N}\left[H_{i,m}.\frac{D_i}{2}\left(1-\frac{D_i}{P_{i,M}}\right) + \sum_{j=1}^{M-1} H_{i,j}.\frac{D_i^2}{2}\left(\frac{1}{P_{i,j+1}}+\frac{1}{P_{i,j}}\right)\right]t. \tag{25}$$

The cycle time is the only variable in Eq. (25) and TC_{LB} is a convex function because

$$\frac{\partial^2 TC_{LB}}{\partial t^2} = \frac{2\sum_{i=1}^{N} SC_i}{t^3} > 0. \tag{26}$$

Therefore, the global minimum of TC_{LB} is obtained by Eq. (25). To this end, we take the derivative with respect to variable t as follows:

$$\frac{\partial TC_{LB}}{\partial t} = -\frac{\sum_{i=1}^{n} SC_i}{t^2} + \sum_{i=1}^{N}\left[H_{i,M}.\frac{D_i}{2}\left(1-\frac{D_i}{P_{i,M}}\right) + \sum_{j=1}^{M-1} H_{i,j}.\frac{D_i^2}{2}\left(\frac{1}{P_{i,j+1}}+\frac{1}{P_{i,j}}\right)\right] = 0. \tag{27}$$

Therefore, the minimum cycle time that provides the lower bound on the total cost is as follow:

$$T_{\min} = \sqrt{\frac{2\sum_{i=1}^{n} SC_i}{\sum_{i=1}^{N}\left[H_{i,M}.d_i\left(1-\frac{D_i}{P_{i,M}}\right) + \sum_{j=1}^{M-1} H_{i,j}.D_i^2\left(\frac{1}{P_{i,j+1}}+\frac{1}{P_{i,j}}\right)\right]}}. \tag{28}$$

Obviously, $T_{\min}$ is the minimum cycle time value of t disregarding the constraints of the problem. Since we have a multi-product model where each product has a setup time $S_{i,j}$, and a machine can process at most one product at a time, $T_{\min}$, is a lower bound for t. Given the value of $T_{\min}$, we are now able to compute $F_{\max}$ using (23).

4. Hybrid iterated local search

We develop an HILS to solve realistically-sized instances of the ELSP-HFS problem. The iterated local search is an effective technique that has been used in a variety of scheduling research topics. For instance, Naderi et al. (2010) use the ILS for makespan minimization in a realistic variant of the flow shop scheduling problem. Ribas et al. (2013) develop an ILS algorithm for the tardiness minimization in the flow shop problem. Dong et al. (2015) present an ILS for the flow shop problem with flow time criterion. Pan et al. (2017b) present an ILS for earliness and tardiness minimization for the HFS problem. Due to the successful application of the ILS algorithm in scheduling problems and also the NP-hardness of ELPS-HFS (Torabi et al., 2006), we develop an iterated local search (ILS) algorithm hybridized with an expediting mechanism. The ILS generates many solutions during search and in our case, it is ineffective computationally to calculate the objective function of all these new solutions. We instead use an approximate selective procedure to quickly discard inferior solutions.

The ILS algorithm is a local search metaheuristic. The basic idea of the ILS is to carry out randomized walks in the solution space. The search in the ILS starts from a single solution, commonly generated by a heuristic. Then, it repeatedly utilizes the following three mechanisms to improve this solution. The first mechanism is to apply a local search engine to the current solution until the search is stuck in a local optimum. The second mechanism is an acceptance criterion, determining whether the new solution must replace the current solution or not. The third mechanism is to escape from the current local optimum by perturbing the solution. The procedure of the proposed HILS is shown in Fig. 2.

4.1. Solution encoding and fitness function

The ELSP-HFS involves two parts: (i) the *discrete part* in which the assignment of products to machines and their sequences on each machine are determined, and (ii) the *continuous part* in which the lot sizes and production starting times for each product at different stages are determined. The actual complexity of our model stems from the discrete part. Therefore, the function of the metaheuristic algorithm is to obtain a good solution for the discrete part of the problem. The discrete part involves variables $x_{i,j,\ k,\ r}$ and $z_{i,j,\ r}$.

The solution representation format relates to the permutation vector of size N (i.e., the number of products) in which the processing order of the given set of products at the first stage is determined. Such a vector by itself does not specify the complete solution for the discrete part of the problem. Thus, to obtain a complete solution for a given permutation vector, we use the "First Available Machine" rule by which we assign products to machines at each stage (Torabi et al., 2006). The product sequence for subsequent stages is based on the "First In First Out" rule by which a product with the earliest completion time in the previous stage is scheduled first in the next stage. A solution of the HILS is not a complete solution to the problem and only represents a partial solution.

The assignment and sequence of products on machines are obtained from the metaheuristic algorithm, and their values are represented by $\bar{z}_{i,j,r}$ and $\bar{x}_{i,j,k,r}$, respectively. These fixed values are used as the inputs to the continuous part of the problem which is formulated as the following linear programming model:

Restricted model: minimize TC_2 subject to:

Constraint sets (8)–(11), (13) and (19)–(22).

$$b_{i,j} \geq c_{k,j} + S_{i,j} - M\cdot\left(1-\bar{x}_{i,j,k,r}\right) \quad \forall_{i\neq k,j,r}. \tag{29}$$

This model is, in fact, the linear version of the proposed model while fixing variables $z_{i,j,\ r}$ and $x_{i,j,\ k,\ r}$. In this case, the model is a pure linear program.

4.2. Initial solution

To generate the initial population, we use five simple constructive heuristics proposed in Torabi et al. (2006). These heuristics include Shortest Processing Time (SPT), Longest Processing Time (LPT), Weighted SPT (WSPT), HY rule of Hahm and Yano (1995), and CDS heuristic by Brah and Loo (1999). By the SPT (LPT) heuristics, at most M different permutation vectors (solutions) are generated where in the jth permutation, the products are sorted according to a non-decreasing (non-increasing) order of their processing time at stage j. The WSPT (HY) heuristic is similar to the SPT (LPT) heuristic with the exception that it arranges the products in an ascending (descending) order of $PT_{i,j}/H_{i,j}$ values where $PT_{i,j} = t \bullet D_i/P_{i,j}$.

CDS generates $M-1$ dummy two-stage flowshops from an M-stage flow shop and builds one solution from each one. The other

```
Procedure: Hybrid iterated local search
Input: $\pi_{int}$(initial solution), $\alpha, T_{min}$
Output: $\pi_{best}$ (best solution)

$\pi := \pi_{int}$
$\pi_{best} := \pi$
While stopping criterion has not been met do
    $\pi :=$ local search $(\pi, \pi_{best}, \alpha, T_{min})$        % local search
    If $obj(\pi) < obj(\pi_{best})$ do
        $\pi_{best} := \pi$
    Elseif
        If $rand < exp\left(\frac{obj(\pi_{best})-obj(\pi)}{obj(\pi_{best})} 100\right)$ do    % acceptance criterion
            $\pi := perturbation(\pi)$        % perturbation
        Elseif
            $\pi := perturbation(\pi_{best})$        % perturbation
        Endif
    Endif
Endwhile
```

Fig. 2. The procedures of the proposed HILS.

four heuristics are for the M-stage flowshop and generate 4 solutions in total. Then, we can solve each of these dummy $M-1$ two-stage flowshops by these heuristics. Therefore, these five heuristics generate a total of $5M-1$ solutions. To have a two-stage flowshop, we require processing times of products for two machines. The processing times of rth dummy two-stage flowshop in the CDS is as follows:

$$t^r_{i,1} = \sum_{j=1}^{r} PT_{i,j},\ t^r_{i,2} = \sum_{j=M+1-r}^{M} PT_{i,j};\ i = 1, \ldots, N, r = 1, \ldots, M-1.$$

where $t^r_{i,1}$and $t^r_{i,2}$ are the processing times of products on dummy machines 1 and 2.

Since the value of variable t is unknown, we consider $t=1$ to calculate the processing times in the mentioned heuristics. Note that if we require less than $5M-1$ solutions as initial solutions, we select the best. If we require more than $5M-1$ solutions, we generate the rest randomly.

4.3. Local search

The local search that we use is a variable neighborhood search (Mladenović and Hansen, 1997; Sang et al., 2016) that uses two neighborhood structures based on insertion moves. The first neighborhood structure (NS1) is as follows: for each product in the sequence, neighbors are generated by removing one product from its position and inserting it in all other possible positions. This neighborhood structure can generate $(N-1)^2$ neighbors for each solution. In the second neighborhood structure (NS2) each neighbor is obtained by inserting a pair of successive products i and i' after every possible product in the sequence. This neighborhood structure can generate $(N-2)^2$ neighbors for each solution.

4.4. Approximate objective function

To calculate the fitness function of each solution, we solve a large linear program (*LP*). It takes a significant amount of time for checking all the solutions. Therefore, one idea is to select better neighbors in each neighborhood structure and calculate the fitness function of these neighbors. The main decision variable in the total cost (see Eq. (4)) is the common cycle length (t). Also, the lower bound on the total cost and related common cycle (T_{min}) are calculated in Section 2 (see Eqs. (26) and (29)). This means, each solution (assignment and sequence of products) that makes the cycle time closer to the T_{min} value has a higher chance to be a better solution. Also, the cycle time of each solution is equal to the

```
Procedure: Local search
Input: $\pi$ (current solution), $\pi_{best}$ (best solution as so far), $\alpha, T_{min}$
Output: $\pi$

$k = 1$
While $k \le 2$ do
    $\pi' := NS_k(\pi, \alpha, T_{min})$
    If $obj(\pi') < obj(\pi)$ do
        $\pi := \pi'$
        $k := 1$
    Elseif
        $k := k + 1$
    Endif
    If $obj(\pi) < obj(\pi_{best})$ do
        break
    Endif
Endwhile
```

Fig. 3. The procedure of the local search.

maximum completion time (C_{max}) of the solution. The C_{max} value for each new solution is calculated, using the "First Available Machine" rule and by considering setup and processing times. Note that products are scheduled on machines at the earliest possible time (as soon as possible). The processing time of product i at stage j is $PT_{i,j} = t^* \bullet D_i/P_{i,j}$, where t is the value of the optimal cycle time related to the best solution in the search history. Then, the absolute difference between $T_{\min}$ and $C_{\max}$ for each solution is calculated and then the solutions are sorted in an ascending manner according to these distance values. Subsequently, the α neighbors at the top of the sorted list are selected for the exact objective calculation. Note that α is the parameter of the algorithm that needs to be calibrated.

The selected neighbors via approximate objective function are evaluated (by solving the related MILP model) one by one until the first improvement is reached. This new neighbor is replaced with the current solution. The procedure of the local search is shown in Fig. 3.

4.5. Acceptance criterion and perturbation mechanism

A new local optimum is accepted according to its fitness function. First, the acceptance probability of the new solution π' is calculated as follows:

$$\text{Acceptance Probability (AP)} = e^{\left(\frac{TC(\pi_{best})-TC(\pi')}{TC(\pi_{best})} \times 100\right)}.$$

Then, a random value r is generated from a uniform distribution between 0 and 1. If $r < AP$, the new solution replaces the current solution. By this mechanism, each new solution that has better fitness function is accepted with a higher probability.

The incorporation of a weak perturbation likely yields an algorithm that gets rapidly stuck in a local optimum, whereas the incorporation of an effective perturbation prevents an algorithm from experiencing a premature convergence to a local optimum. The perturbation mechanism used in most ILSs developed for other flow shop problems consists of several random insertion moves. We use a novel double-insertion move instead of the several insertion moves for the perturbation of the given solution in our HILS. The double-insertion move removes two consecutive products from positions j and $j+1$ (j is randomly selected between 1 to $N-1$) and reinserts them together into positions k and $k+1$ (k is randomly selected between 1 to $N-1$, and $k > j+1$ or $k < j-1$) in the same order.

5. Experimental evaluation

We evaluate the performances of our models and the HILS. First, we compare the performances of the mathematical models, including two MINLPs and two MILPs. We reiterate that three of these models are novel contributions of this paper and only the nonlinear variant of the problem is due to Torabi et al. (2006). One of the MILP models is the linearized variant of the MINLP that we have developed for our problem and the other MILP is a linearized variant of the existing MINLP model in literature. There exist numerous metaheuristics in the literature that can be used for our comparison purposes. However, we choose some of the recently published *population-based* and *local-search-based* meta-heuristics for our problem. These algorithms include the iterated greedy with iteration jumping (IG-IJ) algorithm by Tasgetiren et al. (2017) (*local search*) and discrete artificial bee colony (DABC) by Pan et al. (2017a, 2017b) (*population-based*). We also compare our HILS with the hybrid genetic algorithm (HGA) of Torabi et al. (2006), tabu search (TS) of Akrami et al. (2006) that have been directly developed for the same problem. Therefore, we compare our HILS with four meta-heuristic algorithms in the literature.

Note that the mathematical models are coded into GAMS 24.1.2, and all the algorithms are programmed in MATLAB 2012a. Experiments are implemented in a notepad with 2.50 GHz Intel Core i5 Duo and 4 GB of RAM memory. The experimental instances of Torabi et al. (2006) include up to 10 products and 10 stages. We generate two new sets of experimental instances as follows: Set 1 includes $N=\{4,\ 6,\ 8,\ 10\}$ and $M=\{3,\ 5\}$ as small- and medium-sized instances, respectively, and Set 2 includes $N=\{15,\ 20,\ 25\}$ and $M=\{10,\ 15\}$ as large-sized instances. The time unit is set to one working day. The rest of the parameters are generated in accordance with Torabi et al. (2006) and Ouenniche et al. (1999) as follows:

$$D_i \sim U(100, 1000)$$

$$P_{i,j} = Nd_i\theta;\ \ \theta \sim U(1.5, 2.5)$$

$$S_{i,j} \sim U(0.01, 0.25)$$

$$SC_i = 10000\left(\sum_{j=1}^{M} s_{i,j}\right) + 1000\rho;\ \ \rho \sim (0,1)$$

$$H_{i,j} \sim U(1,10);\ H_{i,j+1} = H_{i,j} + \beta;\ \ \beta \sim (1,5)$$

$$O_j = \begin{cases} 1, & j \text{ is odd number} \\ 2, & j \text{ is even number} \end{cases}$$

$$PH = 500 \text{ working days}$$

Note that $U(a,\ b)$ is a uniform distribution taking a value between a and b. We generate 5 and 10 random instances for each problem size of Sets 1 and 2, respectively, leading to a total of 40 and 60 random instances for Sets 1 and 2, respectively. The stopping criterion for all algorithms is set to NM seconds. The performance measure to compare the algorithms is based on the relative percentage deviation (RPD), calculated as follows:

$$\text{Relative percentage deviation (RPD)} = \frac{Some_{sol} - Best_{sol}}{Best_{sol}} \times 100,$$

where $Some_{sol}$ is the objective function value obtained by a given algorithm on a given instance and $Best_{sol}$ is the lowest objective function value obtained for that instance.

5.1. Models evaluation

In this section, we compare the performances of the four mathematical models. We denote them as follows:

Model1: our developed nonlinear model,
Model2: our linearized version of *Model1*,
Model3: the existing MINLP model of Torabi et al. (2006),
Model4: our linearized version of the *Model3*.

The proposed mathematical models are assessed in terms of both their size and computational complexities. The size of mathematical models indicates the numbers of variables and constraints that each of these models require to solve any instance of the problem. The computational complexity of models alludes to the amount of time they need to solve the same instance of the problem. The number of continuous variables (NCV), the number of binary variables (NBV), and the number of constraints (NC) required by the four tested models to formulate the same problem instance with N products and M stages are shown in Table 2.

Although both nonlinear models (*Model*1 and *Model*3) possess the same NBVs, the *Model*1 establishes a tighter interconnection between binary variables of each pair. That is to say, $x_{i,j,\ k,\ r} = 1 - x_{k,j,\ i,\ r}$. Constraint (7) ensures $b_{i,j} > c_{k,j}$ if $x_{i,j,\ k,\ r} = 1$. In this case, $b_{k,j} \ngtr c_{i,j}$, and this holds only if $x_{k,j,\ i,\ r} = 0$. This interconnection between variables culminates in a more effective model that helps commercial solvers solve *Model*1 faster. Our nonlinear model, *Model*1, requires fewer constraints than the existing model in the literature, *Model*3. The linear version of each model has

Table 2
Size of mathematical models.

Model	NBV	NCV	NC
*Model*1	$N^2 \sum_{j=1}^{M} O_j$	$2NM+1$	$N(4M+2(N-1)\sum_{j=1}^{M} O_j)+1$
*Model*2	$N^2 \sum_{j=1}^{M} O_j + F_{max}$	$2NM+2$	$N(4M+2(N-1)\sum_{j=1}^{M} O_j)+2F_{max}+2$
*Model*3	$N^2 \sum_{j=1}^{M} O_j$	$NM+1$	$N(3M+((N-1)^2+1)\sum_{j=1}^{M} O_j)+1+(N-1)\sum_{j=1}^{M} O_j$
*Model*4	$N^2 \sum_{j=1}^{M} O_j + F_{max}$	$NM+2$	$N(3M+((N-1)^2+1)\sum_{j=1}^{M} O_j)+(N-1)\sum_{j=1}^{M} O_j+2F_{max}+2$

more NBVs, NCVs, and NCs than the nonlinear version since we add the auxiliary variable y_l and replace Constraint (12) with Constraints (19)–(22). We will later show that achieving linearity for models at the expense of adding more variables and constraints is worthwhile from a computational standpoint.

To compare the computational complexity of the models, we use the instances of Set 1 using CPLEX solver for MILP models and BARON solver for MINLP models (Table 3). The column "Time" shows the average computational time for that instance. The column "Gap" shows the optimality gap reported by GAMS after 3600 s of computation time. "–" means that the corresponding model is not able to find any integer solution for that instance within the allotted time. The results demonstrate that in most instances, *Model*3 cannot even find a feasible integer solution, but *Model*1 obtains either optimal or near-optimal solutions in most cases. While *Model*1 and *Model*3 can achieve optimal solutions for 13 (32%) and 5 (12%) instances out of 40 available instances, respectively, this number for *Model*2 and *Model*4 is 30 (75%) and 12 (30%), respectively, clearly demonstrating the effectiveness of the linearization schemes used in *Model*2 and *Model*4. The existing MINLP model in the literature (*Model*3) fails to find integer feasible solutions for 23 instances, but *Model*2 can find optimal solutions for all instances up to 8 products and feasible integer solutions for the rest of instances within a reasonable computational time.

5.2. Algorithm evaluation

As mentioned before, we compare our HILS algorithm with four existing metaheuristics in the literature:

(1) HGA by Torabi et al. (2006) for the same problem.
(2) TS by Akrami et al. (2006) for ELSP-HFS with limited buffers.
(3) IG-IJ by Tasgetiren et al. (2017) for blocking flow shop scheduling with makespan criterion.
(4) DABC by Pan et al. (2017a, 2017b) for minimization the makespan in hybrid flow shop scheduling with setup times.

The IG-IJ and DABC have been developed for other variants of flow shop problems, but their permutation-based encoding scheme allows us to easily adapt them to our problem. To this end, we only need to use the decoding process in Section 3.1 for adjustment. First, we calibrate our HILS algorithm. Then, we investigate the general performance of the algorithms against the optimal solutions obtained by mathematical models. Finally, we compare the HILS algorithm with other algorithms. We reiterate that the metaheuristic algorithms determine only the discrete part of the problem. To optimally solve the continuous part of the problem, we call CPLEX 12.4 within our code in MATLAB to solve the *restricted model* as described in Section 4.1.

5.2.1. Parameter tuning of the proposed HILS

Parameter calibration significantly impacts the performance of metaheuristics. The HILS algorithm has only one parameter to be calibrated. We consider 5 levels for this parameter:

$$\alpha = \left\{1, \frac{N}{4}, \frac{N}{2}, N, 2N\right\}.$$

Thus, we will have five different variants of the HILS. A full factorial experiment is designed to perform the experiments. In this regard, 10 random instances for each problem size of Set 2 are generated. Fig. 4 shows the average RPD as well as the least significant difference (LSD) intervals. It is evident from Fig. 4 that α value equal to N provides the best result.

5.2.2. Small- and medium-sized instances

We evaluate the general performance of the five algorithms on 20 instances obtained from Set 1 generated for models' evaluation.

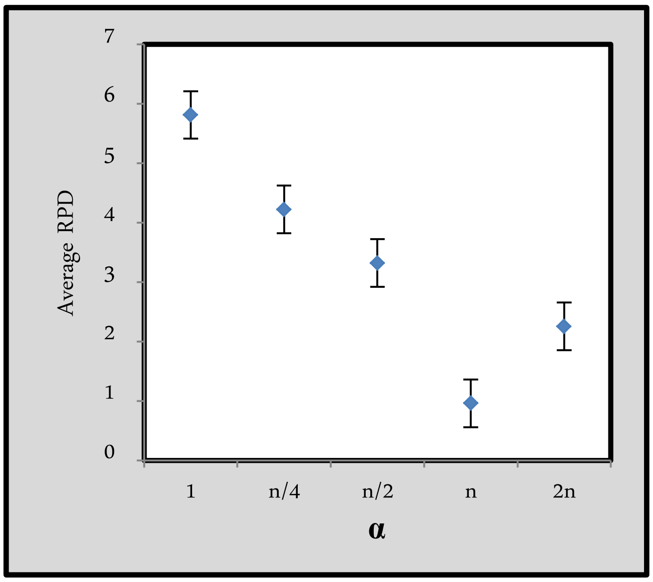


Fig. 4. The average RPD and LSD intervals for the parameter tuning.

We compare the algorithms' solutions against the best integer solution of mathematical models (Table 4). As can be seen, all algorithms yield optimality gaps that are very close to zero. On average, the HILS algorithm is slightly better than the other algorithms in terms of the achieved optimality gap.

5.2.3. Large size instances

The performance of algorithms is also examined on the 60 instances of Set 2, each of which is solved 5 times by each algorithm to verify algorithms' robustness. Table 5 shows the average RPD obtained for different instances of the problem. There are three rows for each problem size. The columns "Best" and "Worst" show the best and worst RPD, respectively, that is gained by each algorithm for 10 different instances. Also, the column "Average" shows the average RPD obtained by each algorithm for the 10 different instances. To conduct a performance evaluation of algorithms, the results of the row "Average" are used. The HILS algorithm obtains the best results with an average RPD of 0.413%, whereas the DABC achieves the second rank with an average RPD of 3.032%. The worst-performing algorithm is the HGA with an average RPD of 4.628%. We implement a one-way ANOVA test with $\alpha = 0.05$ (significance level) on the results. The ANOVA test results reveal a statistically significant difference among these three algorithms with a p-value equal to zero (Table 6). Fig. 5 shows the average RPD as well as the least significant difference (LSD) intervals. The HILS algorithm statistically outperforms the other four algorithms. The TS and DABC have similar performances and they outperform the other two algorithms. Figs. 6 and 7 also show the average RPD of the tested algorithms versus the number of products and stages, respectively. The HILS algorithm shows a robust performance across all the problem sizes.

This HILS's outperformance is likely attributable to the fact that HILS strikes a suitable balance between diversification (perturbation) and intensification (greedy local search) mechanisms. According to the literature, the greedy search for the permutation-like encoding scheme performs well in most applications, but it suffers from being slow as it evaluates all the possible combinations. We overcame this shortcoming by expediting the search and selectively discarding several combinations.

Table 3
Comparison of mathematical models in terms of optimality gaps and computation times.

No.	*N*	*M*	Model	Time (s)	Gap (%)	No.	*N*	*M*	Model	Time (s)	Gap (%)
1	4	3	*Model*1	15.3	0	21	4	5	*Model*1	50.4	0
			*Model*2	4.6	0				*Model*2	5.6	0
			*Model*3	10.9	0				*Model*3	3600	15.3
			*Model*4	10.4	0				*Model*4	14.8	0
2			*Model*1	21.6	0	22			*Model*1	103	0
			*Model*2	4.9	0				*Model*2	2.8	0
			*Model*3	17.3	0				*Model*3	–	
			*Model*4	6.4	0				*Model*4	12.5	0
3			*Model*1	17.5	0	23			*Model*1	395	0
			*Model*2	3.8	0				*Model*2	6.4	0
			*Model*3	18.0	0				*Model*3	–	
			*Model*4	12.2	0				*Model*4	22.7	0
4			*Model*1	10.9	0	24			*Model*1	18.7	0
			*Model*2	4.1	0				*Model*2	6.1	0
			*Model*3	10.2	0				*Model*3	3600	43.9
			*Model*4	9.4	0				*Model*4	17.8	0
5			*Model*1	8.3	0	25			*Model*1	62.5	0
			*Model*2	4.5	0				*Model*2	4.9	0
			*Model*3	14.1	0				*Model*3	–	
			*Model*4	13.6	0				*Model*4	14.2	0
6	6	3	*Model*1	3357	0	26	6	5	*Model*1	3600	76
			*Model*2	34.9	0				*Model*2	25.4	0
			*Model*3	3600	60.7				*Model*3	–	
			*Model*4	1928	0				*Model*4	3600	74.5
7			*Model*1	3600	1.1	27			*Model*1	3600	59.3
			*Model*2	31.5	0				*Model*2	49.7	0
			*Model*3	3600	43.6				*Model*3	–	
			*Model*4	3600	24.1				*Model*4	3600	88.5
8			*Model*1	3600	3.1	28			*Model*1	3600	14.8
			*Model*2	55.9	0				*Model*2	70.4	0
			*Model*3	3600	119				*Model*3	–	
			*Model*4	3600	22.7				*Model*4	3600	68.9
9			*Model*1	2872	0	29			*Model*1	3600	2.7
			*Model*2	20.7	0				*Model*2	112	0
			*Model*3	3600	83.5				*Model*3	–	
			*Model*4	1395	0				*Model*4	3600	48.3
10			*Model*1	3045	0	30			*Model*1	3600	31.0
			*Model*2	39.4	0				*Model*2	17.5	0
			*Model*3	3600	32.8				*Model*3	–	
			*Model*4	3600	17.3				*Model*4	3600	69.5
11	8	3	*Model*1	3600	43.1	31	8	5	*Model*1	3600	108
			*Model*2	3407	0				*Model*2	3561	0
			*Model*3	3600	68.1				*Model*3	–	
			*Model*4	3600	33.2				*Model*4	3600	60.6
12			*Model*1	3600	90.8	32			*Model*1	3600	71.1
			*Model*2	3411	0				*Model*2	3004	0
			*Model*3	3600	295				*Model*3	–	
			*Model*4	3600	87.1				*Model*4	3600	54.2
13			*Model*1	3600	81.7	33			*Model*1	3600	79.2
			*Model*2	1239	0				*Model*2	3281	0
			*Model*3	3600	341				*Model*3	–	
			*Model*4	3600	55.4				*Model*4	3600	62.8
14			*Model*1	3600	74.2	34			*Model*1	3600	60.7
			*Model*2	1508	0				*Model*2	2944	0
			*Model*3	3600	237				*Model*3	–	
			*Model*4	3600	49.6				*Model*4	3600	71.1
15			*Model*1	3600	99.4	35			*Model*1	3600	34.2
			*Model*2	1125	0				*Model*2	3219	0
			*Model*3	3600	614				*Model*3	–	
			*Model*4	3600	62.5				*Model*4	3600	37.2
16	10	3	*Model*1	3600	32.7	36	10	5	*Model*1	–	
			*Model*2	3600	11.3				*Model*2	3600	10.7
			*Model*3	–					*Model*3	–	
			*Model*4	3600	88.3				*Model*4	3600	157
17			*Model*1	3600	69.1	37			*Model*1	–	
			*Model*2	3600	16.3				*Model*2	3600	24.9
			*Model*3	–					*Model*3	–	
			*Model*4	3600	44.1				*Model*4	3600	77.3

(*continued on next page*)

Table 3 (*continued*)

No.	*N*	*M*	Model	Time (s)	Gap (%)	No.	*N*	*M*	Model	Time (s)	Gap (%)
18			*Model*1	3600	33.5	38			*Model*1	–	
			*Model*2	3600	8.9				*Model*2	3600	12.8
			*Model*3	–					*Model*3	–	
			*Model*4	3600	60.8				*Model*4	3600	49.3
19			*Model*1	3600	92.7	39			*Model*1	–	
			*Model*2	3600	20.1				*Model*2	3600	32.9
			*Model*3	–					*Model*3	–	
			*Model*4	3600	74.6				*Model*4	3600	109
20			*Model*1	3600	143	40			*Model*1	–	
			*Model*2	3600	31.6				*Model*2	3600	28.3
			*Model*3	–					*Model*3	–	
			*Model*4	3600	93.2				*Model*4	3600	90.4

Table 4
Comparison between the optimal solution and results of the algorithms.

No	*N*	*M*	HGA	TS	HILS	IG-IJ	DABC
6			0.067	0.000	0.043	0.000	0.000
7			0.000	0.000	0.000	0.000	0.000
8	6	3	0.000	0.000	0.000	0.000	0.000
9			0.000	0.000	0.000	0.000	0.091
10			0.000	0.000	0.000	0.000	0.000
11			0.000	0.000	0.000	0.000	0.000
12			0.000	0.000	0.000	0.000	0.000
13	6	5	0.000	0.000	0.000	0.238	0.000
14			0.000	2.412	0.000	0.000	1.346
15			0.000	0.000	0.000	0.000	0.000
26			0.074	0.000	0.025	0.000	0.418
27			0.000	0.000	0.000	0.000	0.000
28	8	3	0.000	0.000	0.000	0.000	0.000
29			0.000	0.000	0.000	0.000	0.000
30			0.000	0.000	0.000	0.057	0.023
31			0.015	0.000	0.000	0.000	0.076
32			0.010	0.131	0.000	0.152	0.227
33	8	5	0.148	0.240	0.061	0.000	0.419
34			0.007	0.013	0.000	0.381	0.085
35			0.032	0.048	0.000	0.749	0.116
Avg.			0.017	0.142	0.006	0.079	0.140

Table 5
The average RPD obtained by algorithms for large instances.

					RPD			
N	*M*		HGA	TS	HILS	IG-IJ	DABC	
15	10		0.045	0.051	0.004	0.116	0.038	Best
20	10		0.144	0.018	0.023	0.191	0.208	
25	10		0.535	0.002	0.060	1.325	0.311	
15	15		0.334	2.412	0.000	0.314	0.127	
20	15		3.638	0.193	0.034	0.381	0.229	
25	15		5.231	2.954	0.075	0.436	0.285	
		Ave.	1.654	0.938	0.032	0.460	0.199	
15	10		1.117	1.401	0.139	1.622	1.217	Average
20	10		1.546	0.877	0.248	1.214	1.384	
25	10		2.975	1.069	0.346	4.153	3.156	
15	15		4.639	5.959	0.438	3.712	2.410	
20	15		7.237	4.856	0.418	5.625	4.107	
25	15		10.256	6.017	0.899	8.259	5.918	
		Ave.	4.628	3.363	0.413	4.097	3.032	
15	10		2.477	4.328	0.788	5.117	3.824	Worst
20	10		5.434	3.472	1.792	4.729	4.251	
25	10		4.976	2.415	2.089	5.962	3.156	
15	15		8.453	12.365	2.173	7.260	5.192	
20	15		10.731	8.873	2.256	9.815	9.628	
25	15		13.537	8.502	2.624	13.050	10.523	
		Ave.	7.601	6.659	1.953	7.655	6.095	

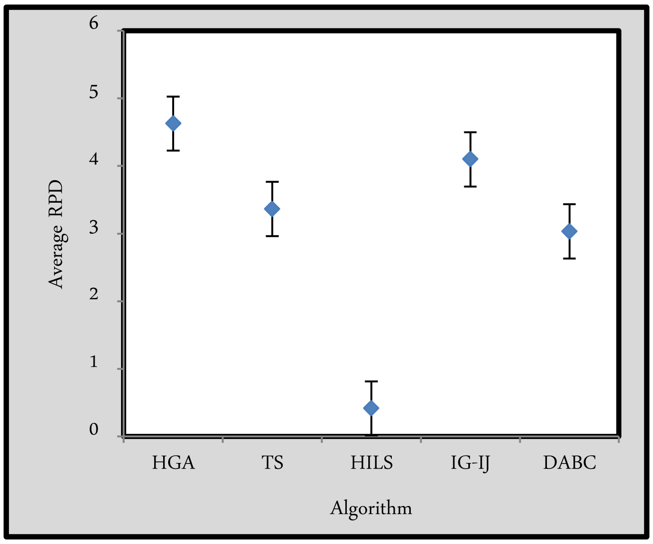


Fig. 5. The average RPD and LSD intervals for the algorithms.

Table 6
ANOVA results.

Factor	df	SS	MS	F	*P*_value
Algorithm	4	636.89	159.22	19.34	0.000
Error	295	2428.34	8.23		
Total	299	3065.23			

6. Conclusion and future research

We studied the Economic Lot-sizing and Sequencing Problem (ELSP) in a Hybrid Flow Shop setting from both mathematical modeling and algorithmic perspectives. As the primary purpose of this research, we developed a mixed-integer nonlinear programming (MINLP) model that improved an existing MINLP model in terms of size dimensionality, computational time, and solvability. Subsequently, we proposed an effective linearization scheme using auxiliary variables. The application of this linearization technique to the MINLP models led to two novel mixed-integer linear programming (MILP) models. These two MILPs are the first linear models ever presented for this problem and are generalizable to other ELSPs in different scheduling shop floors. We showed that the new MILP models could solve the small- and medium-sized instances of the problem to near optimality, but they produced large optimality gaps for large instances. Therefore, we developed a Hybrid Iterated Local Search (HILS) algorithm that solved the large instances of the problem more efficiently. We additionally compared the HILS performance with our mathematical models and four metaheuristic algorithms from the literature and demonstrated the HILS superior performance.

Future research directions include extending the problem and research findings to other ELSPs in different shop floors. We have been working on developing models and methodologies to solve a

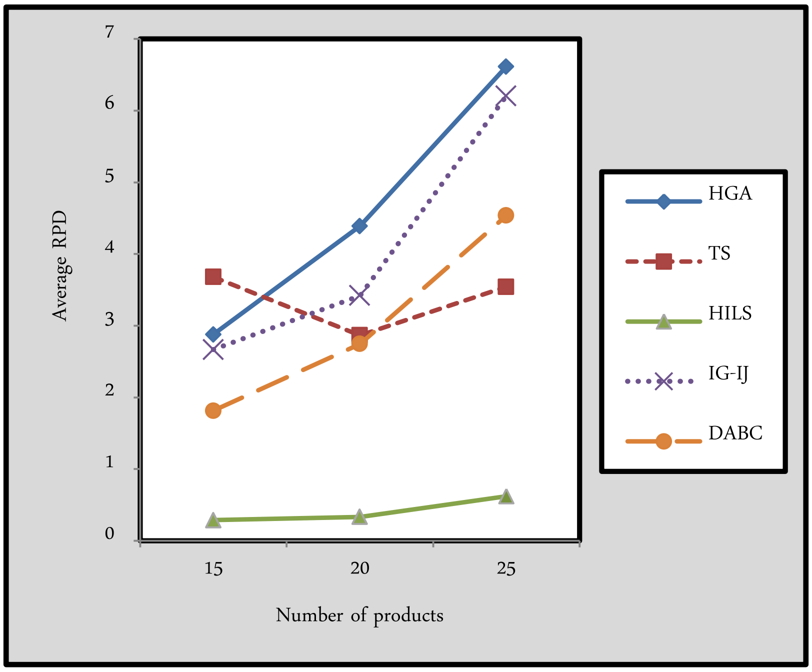


Fig. 6. The average RPD versus the number of products.

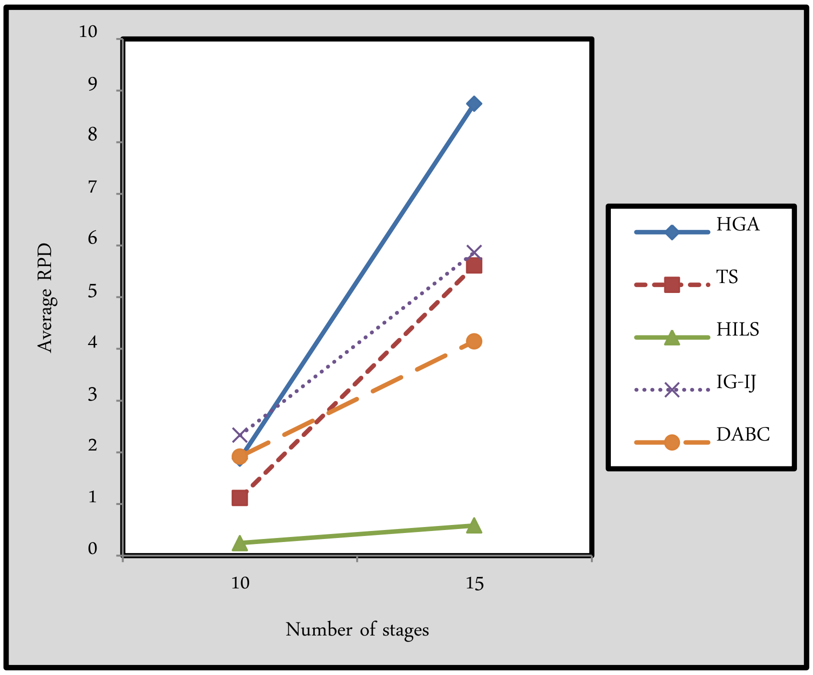


Fig. 7. The average RPD versus the number of stages.

more practical variant of the ELSP-HFS with limited intermediate buffers between stages. Our ELSP-HFS can be extended to cases where different items have different cycle times. Finally, solving the stochastic MILP version of the ELSP under processing time uncertainty using exact techniques is interesting from both practical and theoretical standpoints.

References

Akrami, B., Karimi, B., Hosseini, S.M., 2006. Two metaheuristic methods for the common cycle economic lot sizing and scheduling in flexible flow shops with limited intermediate buffers: the finite horizon case. Appl. Math. Comput. 183 (1), 634–645.
Beck, F.G., Glock, C.H., 2016. The impact of batch shipments on the Economic Lot Scheduling Problem. Comput. Indus. Eng. 96, 126–139.
Brah, S.A., Loo, L.L., 1999. Heuristics for scheduling in a flow shop with multiple processors. Eur. J. Oper. Res. 113 (1), 113–122.
Bulut, O., Tasgetiren, M.F., 2014. An artificial bee colony algorithm for the economic lot scheduling problem. Int. J. Prod. Res. 52 (4), 1150–1170.
Dobson, G., Yano, C.A., 1994. Cyclic scheduling to minimize inventory in a batch flow line. Eur. J. Oper. Res. 75 (2), 441–461.
Dong, X., Nowak, M., Chen, P., Lin, Y., 2015. Self-adaptive perturbation and multi--neighborhood search for iterated local search on the permutation flow shop problem. Comput. Indus. Eng. 87, 176–185.
El-Najdawi, M.K., 1994. A job-splitting heuristic for lot-size scheduling in multi--stage, multi-product production processes. Eur. J. Oper. Res. 75 (2), 365–377.
El-Najdawi, M.K., Kleindorfer, P.R., 1993. Common cycle lot-size scheduling for multi-product, multi-stage production. Manag. Sci. 39 (7), 872–885.

Hahm, J., Yano, C.A., 1995. The economic lot and delivery scheduling problem: the common cycle case. IIE Trans. 27 (2), 113–125.
Hsu, J.I.S, EI-Najdawi, M., 1990. Common cycle scheduling in a multistage production process. Eng. Costs Prod. Econ. 20 (1), 73–80.
Jenabi, M., Ghomi, S.F., Torabi, S.A., Karimi, B., 2007. Two hybrid meta-heuristics for the finite horizon ELSP in flexible flow lines with unrelated parallel machines. Appl. Math. Comput. 186 (1), 230–245.
Kayvanfar, V., Zandieh, M., 2012. The economic lot scheduling problem with deteriorating items and shortage: an imperialist competitive algorithm. Int. J. Adv. Manuf. Technol. 62 (5-8), 759–773.
Manne, A.S., 1960. On the job shop scheduling problem. Oper. Res. 8, 219–223.
Mladenović, N., Hansen, P., 1997. Variable neighborhood search. Comput. Oper. Res. 24 (11), 1097–1100.
Mohammadi, M., Musa, S.N., Bahreininejad, A., 2015. Optimization of economic lot scheduling problem with backordering and shelf-life considerations using calibrated metaheuristic algorithms. Appl. Math. Comput. 251, 404–422.
Naderi, B., Ruiz, R., Zandieh, M., 2010. Algorithms for a realistic variant of flow shop scheduling. Comput. Oper. Res. 37 (2), 236–246.
Ouenniche, J., Bertrand, J.W.M., 2001. The finite horizon economic lot sizing problem in job shops: the multiple cycle approach. Int. J. Prod. Econ. 74 (1-3), 49–61.
Ouenniche, J., Boctor, F.F., Martel, A., 1999. The impact of sequencing decisions on multi-item lot sizing and scheduling in flow shops. Int. J. Prod. Res. 37 (10), 2253–2270.
Pan, Q.K., Gao, L., Li, X.Y., Gao, K.Z., 2017. Effective metaheuristics for scheduling a hybrid flowshop with sequence-dependent setup times. Appl. Math. Comput. 303, 89–112.
Pan, Q.K., Ruiz, R., Alfaro-Fernández, P., 2017. Iterated search methods for earliness and tardiness minimization in hybrid flowshops with due windows. Comput. Oper. Res. 80, 50–60.
Ribas, I., Companys, R., Tort-Martorell, X., 2013. An efficient iterated local search algorithm for the total tardiness blocking flow shop problem. Int. J. Prod. Res. 51 (17), 5238–5252.
Rogers, J., 1958. A computational approach to the economic lot scheduling problem.. Manag. Sci. 4 (3), 264–291.
Sağlam, Ü., Banerjee, A., 2018. Integrated multiproduct batch production and truck shipment scheduling under different shipping policies. Omega 74, 70–81.
Sang, H., Pan, Q.K., Duan, P., Li, J., 2018. An effective discrete invasive weed optimization algorithm for lot-streaming flowshop scheduling problems. J. Intell. Manuf. 29, 1337–1349.
Sang, H., Gao, L., Li, X., 2014. An iterated local search algorithm for the lot-streaming flow shop scheduling problem. Asia-Pacific J. Oper. Res. 31 (6), 1450045-1-1450045-19.
Sang, H., Pan, Q.K., Duan, P., 2016. Self-adaptive fruit fly optimizer for global optimization. Nat. Comput. doi:10.1007/s11047-016-9604-z.
Santander-Mercado, A., Jubiz-Diaz, M., 2016. The economic lot scheduling problem: a survey. Int. J. Prod. Res. 54 (16), 4973–4992.
Tasgetiren, M.F., Kizilay, D., Pan, Q.K., Suganthan, P.N., 2017. Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion. Comput. Oper. Res. 77, 111–126.
Torabi, S.A., Ghomi, S.F., Karimi, B., 2006. A hybrid genetic algorithm for the finite horizon economic lot and delivery scheduling in supply chains. Eur. J. Oper. Res. 173 (1), 173–189.
Xiao, J., Yang, H., Zhang, C., Zheng, L., Gupta, J.N.D., 2015. A hybrid Lagrangian-simulated annealing-based heuristic for the parallel-machine capacitated lot-sizing and scheduling problem with sequence-dependent setup times. Comput. Oper. Res. 63, 72–82.
Yan, C., Liao, Y., Banerjee, A., 2013. Multi-product lot scheduling with backordering and shelf-life constraints. Omega 41 (3), 510–516.
Zhang, B., Pan, Q.K., Gao, L., Zhang, X., Sang, H., Li, J., 2017. An effective modified migrating birds optimization for hybrid flowshop scheduling problem with lot streaming. Appl. Soft Comput. 52, 14–27.