Journal of Optimization in Industrial Engineering Vol.12, Issue 1, Winter and Spring 2019, 103- 117
DOI: 10.22094/JOIE.2018.542997.1510

Economic Lot Sizing and Scheduling in Distributed Permutation Flow Shops

Mohammad Alaghebandha [a], Bahman Naderi [a,*], Mohammad Mohammadi[a]

[a] *Department of Industrial Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran*

Received 25 August 2017; Revised 29 June 2018; Accepted 02 July 2018

Abstract

This paper addresses a new mixed integer nonlinear and linear mathematical programming economic lot sizing and scheduling problem in distributed permutation flow shop problem with number of identical factories and machines. Different products must be distributed between the factories and then assignment of products to factories and sequencing of the products assigned to each factory has to be derived. The objective is to minimize the sum of setup costs, work-in-process inventory costs and finished products inventory costs per unit of time. Since the proposed model is NP-hard, an efficient Water Cycle Algorithm is proposed to solve the model. To justify proposed WCA, Monarch Butterfly Optimization (MBO), Genetic Algorithm (GA) and combination of GA and simplex are utilized. In order to determine the best value of algorithms parameters that result in a better solution, a fine-tuning procedure according to Response Surface Methodology is executed.

Keywords: Lot sizing; Distributed permutation flow shops; Linearization; Water Cycle Algorithm; Monarch butterfly optimization.

1. Introduction

Economic Lot Sizing Problems (or ELSP) is one of the well-recognized production planning problems belonging to the medium-term decision making. It attracts many attentions in the literature after the pioneer papers of Rogers (1958) and Elion (1959). In classic ELSP, there are a set of n products need to be processed on a single machine. The machine can process at most one product at a time. Both demand and production of each product follow constant rates and are known in advance. All the demands must be satisfied; that is, no shortage is allowed. Some setup is carried out before the production can commence. This setup could be influential regarding both its magnitude of time and its cost. Moreover, the production horizon is assumed to be continuous and infinite. The objective is to specify a production schedule so as to minimize long-run average total cost, i.e. sum of setup and inventory holding costs. The cyclic schedule is frequently assumed to be common cycle; that is, the production cycle times of all the products are the same. The production schedule includes two decision dimensions, sequence and quantity in which products are processed (Maxwell, 1964). Karimi et al. (2003) provide a review of models and algorithms for different lot sizing problems.

In practice, there are shops in which a series of operations have to be carried out to turn raw materials into finished products. One of the commonly-happening environments is to have the same route for all the products to pass through machines. In this case, the shop is called a flow shop. If it is assumed that sequences of products on all machines are the same, the shop is called a permutation flow shop. To approach realistic industrial settings, the classic problem is further developed to consider Economic Lot Sizing Problems in Permutation Flow shops (or ELSP in PFS). It seems that multi-stage ELSP was initiated by El-Najdawi (1989) through considering a two-stage problem. Later, the problem is developed to the case of arbitrary number of stages by Hsu and El-Najdawi (1990). Looking into the literature of this problem, is noticed that it experiences a new era after the late 90s. Before 1999, researches are centered on this idea that production sequence is negligible due to its trifling influence on the total cost. Therefore, the little heed was paid to the sequence decision. It was common place to tackle the problem by minimizing the total cost for a given sequence usually obtained by a simple heuristic. As a case for this point, the reader refers to a paper by El-Najdawi and Kleindorfer (1993). They make use of Shortest Processing Time (or SPT) to obtain a sequence. Their core contribution is on lot-sizing decision rather than sequencing decision. Dobson and Yano (1994) and El-Najdawi (1992, 1994) are among the other papers considering ELSP in PFS. But, they still more concentrate on lot-sizing decision. Ouenniche et al. (1999) studied ELSP in PFS, and explored the effect of sequencing decision on the final total cost. They concluded that sequencing decision should be taken into account as well as lot-sizing one. They present a MINLP model and some heuristics in two groups: constructive heuristics (CH) and improvement heuristics, commonly known as metaheuristics. Contrary to existing CHs in the literature of PFS usually minimizing completion time related objectives, they propose CHs that minimize the work-in-process holding cost. Afterwards, they present some local search procedures improving the solutions obtained by CHs as a metaheuristic.

*Corresponding author Email address: bahman.naderi@ aut.ac.ir

Later, Torabi et al. (2005) develop the problem considering presence of multiple identical machines at each stage; and they solve the problem by adaptation of the solution method previously presented by Ouenniche et al. (1999). Akrami et al (2006) addressed the common cycle multi-product lot sizing and scheduling problem in deterministic flexible flow shops where the planning horizon is finite and fixed by management and the production stages are in series, while separated by finite intermediate buffers. They used both genetic algorithm and tabu search methods to find an optimal or near-optimal solution for the problem. A differential evolution (DE) based memetic algorithm, named ODDE, is proposed by Li and Yin (2013) for permutation flow shop problem with minimizing makespan and maximum lateness objectives. Rahman et al (2015) considered a make -to stock production system, where three related issues must be considered: the length of a production cycle, the batch size of each product, and the order of the products in each cycle. To deal with these tasks, they proposed a Genetic Algorithm based lot scheduling approach with an objective of minimizing the sum of the setup and holding costs. Bargaoui et al (2017) addressed the Distributed Permutation Flow shop Scheduling Problem with an artificial chemical reaction metaheuristic which objective is to minimize the maximum completion time. In the proposed CRO, the effective NEH heuristic is adapted to generate the initial population of molecules. Viagas et al (2018) addressed the distributed permutation flow shop scheduling problem to minimize the total flow time. They first analyzed it and discussed several properties, theorems, assignment rules, representation of the solutions and speed-up procedures. They proposed an iterative improvement algorithm to further refine the so-obtained solutions.

Rifai et al (2016) propose a novel model of the developed distributed scheduling by supplementing the reentrant characteristic into the model of distributed reentrant permutation flow shop scheduling. This Problem is described as a given set of jobs with a number of reentrant layers are processed in the factories, which comprises a set of machines, with the same properties. The aim of the study is to minimize makespan, total cost and average tardiness.

Recently, Naderi and Ruiz (2009) have introduced a new generalization of the permutation flow shop originated from today companies' structure in which factories are merged to build a common enterprise. The purpose is to obtain higher productive, less production cost and so on. More details could be found in Wang (1997). In these enterprises, production planners are to handle more complicated decision making processes. In single-factory problems, there are still two above-mentioned decisions, whereas in the distributed problem another decision appears: the assignment of the products to suitable factories. Consequently, three decisions have to be taken; product allocation to factories, product scheduling at each factory as well as lot sizing decision.

Based on the best reviewed, this paper is the first study on the Economic Lot Sizing Problem in Distributed Permutation Flow Shop (or ELSP in DPFS). This problem is a well-known scheduling problem in many industries, such as steel, pharmaceutical, automobile, and food processing (Naderi and Ruiz, 2010). The paper contributes by developing two different alternative Mixed Integer Non-Linear Programming (or MINLP) models according to previous paper Ouenniche et al. (1999) in Distributed manner. This allows for a precise characterization of the ELSP in DPFS. The models' specifications are precisely compared. Apart from the MIP models, four metaheuristics based on Monarch Butterfly Optimization, Water Cycle Algorithm, Genetic Algorithm and Genetic Algorithm with Simplex are presented. The metaheuristic's performance is evaluated by comparing against the optimal solutions obtained by the linear models in small, medium and large sized problems.

The rest of the paper is organized as follows. Section 2 develops two mathematical models and linearization of the proposed models through discussion. Section 3 introduces four presented metaheuristics. Section 4 evaluates the performance of the models and the algorithms. Section 5 concludes the paper and clarifies some directions for future studies.

2. Mathematical Models

Mathematical models are known to be the best way to precisely define all the characteristics of a problem. Actually, by mathematical models, it is possible to turn the implicit explanations of a novel problem into the explicit and detailed ones. Moreover, mathematical models could be a starting point for many solution methods such as problem-specific branch-and-bound methods, approximation algorithms or even metaheuristics. Apart from being a starting point in many different algorithms, recently they could also be treated as solution methods because of available specialized software and high capacity computers. Considering all these together encourage researchers to develop effective mathematical models for their corresponding problems.

More formally, the problem of ELSP in DPFS could be described as follows: a set of n different customer demands with different quantities are received. To fulfill them, raw materials of each demand could be operated in each of g available different factories each of which has the same set of m machines deposed in series. There is no restriction on allocation of customer demands to the factories; however, all of a demand must be manufactured in one factory. When a demand is assigned to a factory, it could not be transferred to another factory. It is also considered that the production rate of each demand is not changed from factory to factory. Before the production of a product can begin on a machine, some anticipatory setup must be performed, meaning that, the setup of a product j on a machine i can start when the machine finishes the process of the previous product (even if product j is processed on machine i-1). A product cannot be processed by more than one machine at a time; and a machine cannot process more than one product at a time. There is no maintenance or breakdown, i.e. machines are always available. Customer demands are only for finished products which continuously delivered. Demand and production rates, setup times and costs, and inventory holding costs are deterministic and constant over an infinite planning horizon. Each product has a unique production rates on different machines. Inventory levels and holding time directly determine the inventory holding costs. Shortages are not allowed. When the operation of a product starts, it cannot be interrupted, i.e. products are not preemptive. There is unlimited buffer between machines, i.e. products can wait unlimitedly for the next machine. It is additionally assumed that the production cycle times of all the factories are equal.

In this section, the problem by two different MIP models are formulated, since it is not clear which model has best performance. Notice that the objective function of the models includes three parts: setup cost, inventory holding cost of finished and semi-finished products. A continuous variable *T* denoting the common cycle time is employed in two models. Before presenting the models, the parameters and indices, shown in Table 1, are defined.

Table 1
The parameters and indices used in the models

Notation	Description
n	Number of products
m	Number of machines
g	Number of factories
j, k	Index for products, $j, k \in \{1, 2, \ldots, n\}$
i	Index for machines, $i \in \{1, 2, \ldots, m\}$
f	Index for factories, $f \in \{1, 2, \ldots, g\}$
$p_{j,i}$	Production rate of product j on machine i
d_j	Demand rate of end product j
$st_{j,i}$	Setup time of product j on machine i
sc_j	Total setup cost of product j over all machines
$h_{j,i}$	Inventory holding cost per unit of product j per time unit between machines i and i+1
h_j	Inventory holding cost per unit of finished product j per time
M	Big positive number

2.a. Model 1

In this model, Binary Variables (or BV) just represent the position of products in sequence. Contrary to the next model, the two decisions of product allocation to factories and product sequence in each factory are together determined by one BV type. Notice that notation k is an index to denote positions and $ss_{j,i}$ is start time of product j on machine i that works in a similar $S_{j,i}$ in model 2. The following variables are defined in this model:

$X_{j,k,f}$: Binary variable that takes value 1 if product j occupies position k in factory f, and 0 otherwise. (1)

$S_{k,i,f}$: Continuous variable for the starting time of the product in position k on machine i in factory f. (2)

Model 1 characterizes the ELSP in DPFS as follow:

$$\text{Minimize } Z = \sum_{j=1}^{n}\left(\frac{sc_j}{T}\right) + \sum_{j=1}^{n}\left(\left[h_j \frac{d_j}{2}\left(1-\frac{d_j}{p_{j,m}}\right) + \sum_{i=2}^{m} h_{j,i-1}\frac{d_j^2}{2}\left(\frac{1}{p_{j,i}}-\frac{1}{p_{j,i-1}}\right)\right]\cdot T\right) \quad (3)$$

$$+\sum_{f=1}^{g}\sum_{k=1}^{n}\sum_{i=2}^{m}\left(\left[\sum_{j=1}^{n} X_{j,k,f}\left(h_{j,i-1}\cdot d_j\right)\right]\left[S_{k,i,f}-S_{k,i-1,f}\right]\right)$$

Subject to:

$$\sum_{k=1}^{n}\sum_{f=1}^{g} X_{j,k,f} = 1 \qquad \forall_j \qquad (4)$$

$$\sum_{j=1}^{n} X_{j,k,f} \le 1 \qquad \forall_{k,f} \qquad (5)$$

$$SS_{j,i} \ge SS_{j,i-1} + \frac{d_j\cdot T}{p_{j,i-1}} \qquad \forall_{j,i>1} \qquad (6)$$

$$SS_{j,i} \ge SS_{j',i} + \frac{d_{j'}\cdot T}{p_{j',i}} + st_{j,i} - M\left(1-X_{j,k,f}\right) - M\left(1-X_{j',k',f}\right) \qquad \forall_{j,j'\neq j,i,f,k>1,k'<k} \qquad (7)$$

$$SS_{j,i} \ge st_{j,i} \qquad \forall_{j,i} \qquad (8)$$

$$T \ge SS_{j,i} + \frac{d_j\cdot T}{p_{j,i}} + st_{j',i} - SS_{j',i} - M\left(1-X_{j,k,f}\right) - M\left(1-X_{j',k',f}\right) \qquad \forall_{j,j'\neq j,i,f,k>1,k'<k} \qquad (9)$$

$$T \ge \sum_{j=1}^{n}\sum_{k=1}^{n} X_{j,k,f}\cdot\left(st_{j,i} + \frac{d_j\cdot T}{p_{j,i}}\right) \qquad \forall_{f,i} \qquad (10)$$

$$SS_{j,i} = \sum_{k=1}^{n}\sum_{f=1}^{g} X_{j,k,f}\cdot S_{k,i,f} \qquad \forall_{j,i} \qquad (11)$$

$$T, S_{k,i,f} \ge 0 \qquad \forall_{k,i,f} \qquad (12)$$

$$X_{j,k,f} \in \{0,1\} \qquad \forall_{j,k,f} \qquad (13)$$

The objective function (3) is to minimize the sum of setup costs, work-in-process inventory costs and finished products inventory costs per unit of time. Constraint set (4) assures that every product must be assigned once. Constraint set (5) ensures that every position must be occupied once. Constraint set (6) states that the process of a product j on machine i cannot begin before its process on machine i-1 completes. Constraint sets (7) is the dichotomous pairs of constraints relating each possible production. Constraint set (8) ensures that the first product in each factory starts after its setup. Constraint set (9) and (10) ensure the minimum possible cycle time is satisfied. This minimum is obtained through capacity-constrained of each factory. Constraint set (11) ensures that each product in every position must be occupied in one factory. Constraint sets (12) and (13) define the decision variables.

2.b. Model 2

In this model there are two types of Binary Variables. The first one is to show relative sequence of every pair of products, meaning that, whether a product precedes/succeeds another product or not. The other BV is to represent product allocation to factories. The variables employed in this model are:

$X_{j,k}$: Binary variable that takes value 1 if product j follows product k, and 0 otherwise. (14)

$Y_{j,f}$: Binary variable that takes value 1 if product j is processed in factory f, and 0 otherwise. (15)

$S_{j,i}$: Continuous variable for the starting time of product j on machine$i$. (16)

Model 2 formulates the ELSP in DPFS as follow:

$$\text{Minimize } Z = \sum_{j=1}^{n}\left(\frac{sc_j}{T}\right) + \sum_{j=1}^{n}\left(\left[h_j \frac{d_j}{2}\left(1-\frac{d_j}{p_{j,m}}\right) + +\sum_{i=2}^{m} h_{j,i-1}\frac{d_j^2}{2}\left(\frac{1}{p_{j,i}}-\frac{1}{p_{j,i-1}}\right)\right]\cdot T\right) \quad (17)$$

$$+\sum_{k=1}^{n}\sum_{i=2}^{m}\left(\left[h_{j,i-1}\cdot d_j\right]\left[S_{j,i}-S_{j,i-1}\right]\right)$$

Subject to:

$$\sum_{f=1}^{g} Y_{j,f} = 1 \qquad \forall_j \qquad (18)$$

$$S_{j,i} \geq S_{j,i-1} + \frac{d_j \cdot T}{p_{j,i-1}} \qquad \forall_{j,i>1} \qquad (19)$$

$$S_{j,i} \geq S_{k,i} + \frac{d_k \cdot T}{p_{k,i}} + st_{j,i} - M(1 - X_{j,k}) - M(1 - Y_{j,f}) - M(1 - Y_{k,f}) \qquad \forall_{j,k\neq j,i,f} \qquad (20)$$

$$S_{k,i} \geq S_{j,i} + \frac{d_j \cdot T}{p_{j,i}} + st_{k,i} - M \cdot X_{j,k} - M(1 - Y_{j,f}) - M(1 - Y_{k,f}) \qquad \forall_{j,k\neq j,i,f} \qquad (21)$$

$$T \geq S_{j,i} + \frac{d_j \cdot T}{p_{j,i}} + st_{k,i} - S_{k,i} - M(1 - X_{j,k}) - M(1 - Y_{j,f}) - M(1 - Y_{k,f}) \qquad \forall_{j,k\neq j,i,f} \qquad (22)$$

$$T \geq S_{k,i} + \frac{d_k \cdot T}{p_{k,i}} + st_{j,i} - S_{j,i} - (M \cdot X_{j,k}) - M(1 - Y_{j,f}) - M(1 - Y_{k,f}) \qquad \forall_{j,k\neq j,i,f} \qquad (23)$$

$$T \geq \sum_{j=1}^{n} Y_{j,f} \cdot \left(st_{j,i} + \frac{d_j \cdot T}{p_{j,i}}\right) \qquad \forall_{f,i} \qquad (24)$$

$$S_{j,i} \geq st_{j,i} \qquad \forall_{j,i} \qquad (25)$$

$$X_{j,k} + X_{k,j} = 1 \qquad \forall_{j,k,k\neq j} \qquad (26)$$

$$T, S_{j,i} \geq 0 \qquad \forall_{j,i} \qquad (27)$$

$$X_{j,k} \in \{0, 1\} \qquad \forall_{j,k} \qquad (28)$$

$$Y_{j,f} \in \{0, 1\} \qquad \forall_{j,f} \qquad (29)$$

The objective function (17) is to minimize the sum of setup costs, work-in-process inventory costs and finished products inventory costs per unit of time. Constraint set (18) ensures that every product is allocated exactly to one factory. Constraint set (19) states that the process of a product *j* on machine *i* cannot begin before its process on machine i-1 completes. Constraint sets (20) and (21) are the dichotomous pairs of constraints relating each possible production pair. In other words, they assure that a machine processes at most one product at a time. Constraint sets (22) and (23) calculate the common cycle time, whereas Constraint set (24) specifies the capacity-constraints of each factory to manufacture assigned products. Constraint set (25) ensures that the first product in each factory starts after its setup. Constraint set (26) avoids the occurrence of cross-precedence, meaning that a product cannot be at the same time both a predecessor and a successor of another product. Constraint sets (27), (28) and (29) define the decision variables.

2.1 Linearization of the proposed model

The proposed models are MINLP because of the nonlinear term in the objective function and also constraints. In order to decrease the number of nonlinear terms, linearization in a similar way as in Rodriguez et al (2014), You and Grossmann (2008) and Pakzad-Moghaddam et al. (2014) is used. In the model 1, the bilinear terms between the continuous variable T and the binary variables $X_{j,k,f}$ in objective function, also constraints (10 and 11), are linearized as follows. Notice that constraint set (11) have been replaced with constraint sets (30) – (33), and constraint (34) of the first derivative of the objective function is obtained.

$$\text{Minimize } Z = \sum_{j=1}^{n} \frac{sc_j}{\sqrt{\dfrac{\sum_{j=1}^{n} sc_j}{\sum_{j=1}^{n}\left(\left[h_j \frac{d_j}{2}\left(1-\frac{d_j}{p_{j,m}}\right)+\sum_{i=2}^{m} h_{j,i-1}\frac{d_j^2}{2}\left(\frac{1}{p_{j,i}}-\frac{1}{p_{j,i-1}}\right)\right]\right)}}}$$
$$+ \sqrt{\sum_{j=1}^{n} sc_j \cdot \left\{\sum_{j=1}^{n}\left(\begin{bmatrix} h_j \frac{d_j}{2}\left(1-\frac{d_j}{p_{j,m}}\right) + \\ \sum_{i=2}^{m} h_{j,i-1}\frac{d_j^2}{2}\left(\frac{1}{p_{j,i}}-\frac{1}{p_{j,i-1}}\right)\end{bmatrix}\right)\right\}}$$
$$+ \sum_{f=1}^{g}\sum_{k=1}^{n}\sum_{i=2}^{m}\left(\begin{bmatrix}\sum_{j=1}^{n}(G_{j,i,k,f} - G_{j,i-1,k,f}). \\ (h_{j,i-1}\cdot d_j)\end{bmatrix}\right)$$

Subject to:

Eq. (4) – Eq. (10) and Eq. (12) – Eq. (13)

$$G_{j,i,k,f} \leq X_{j,k,f}.M \qquad \forall_{j,k,i,f} \qquad (30)$$

$$G_{j,i,k,f} \geq S_{k,i,f} - (1 - X_{j,k,f}).M \qquad \forall_{j,k,i,f} \qquad (31)$$

$$G_{j,i,k,f} \leq S_{k,i,f} + (1 - X_{j,k,f}).M \qquad \forall_{j,k,i,f} \qquad (32)$$

$$G_{j,i,k,f} \geq 0 \qquad \forall_{j,k,i,f} \qquad (33)$$

$$\text{Minimize } Z = \sum_{j=1}^{n} \frac{sc_j}{\sqrt{\dfrac{\sum_{j=1}^{n} sc_j}{\sum_{j=1}^{n}\left(\left[h_j \frac{d_j}{2}\left(1-\frac{d_j}{p_{j,m}}\right)+\sum_{i=2}^{m} h_{j,i-1}\frac{d_j^2}{2}\left(\frac{1}{p_{j,i}}-\frac{1}{p_{j,i-1}}\right)\right]\right)}}} +$$
$$\sum_{j=1}^{n}\left(\left[h_j \frac{d_j}{2}\left(1-\frac{d_j}{p_{j,m}}\right)+\sum_{i=2}^{m} h_{j,i-1}\frac{d_j^2}{2}\left(\frac{1}{p_{j,i}}-\frac{1}{p_{j,i-1}}\right)\right]\cdot T\right)$$
$$+\sum_{j=1}^{n}\sum_{i=2}^{m}\left([h_{j,i-1}\cdot d_j][S_{j,i} - S_{j,i-1}]\right)$$

- Subject to:
- Eq. (18) – Eq. (24) and Eq. (25) – Eq. (29)

$$\sqrt{\frac{\sum_{j=1}^{n} sc_j}{\sum_{j=1}^{n}\left(\begin{bmatrix} h_j \frac{d_j}{2}\left(1-\frac{d_j}{p_{j,m}}\right) + \\ \sum_{i=2}^{m} h_{j,i-1}\frac{d_j^2}{2}\left(\frac{1}{p_{j,i}}-\frac{1}{p_{j,i-1}}\right)\end{bmatrix}\right)}} \qquad (34)$$

In a similar way in the model 2, one term in objective function is divided by continuous variable (T) and constraint (24), the bilinear terms consisting of a continuous variable (T) product a binary variable $Y_{j,f}$ are must be linearized as follows.

3. Complexity and Test Problems

The performance of the models and method is studied through its solutions for a total of 180 problems in small, medium and large sized. Different problem sets with 3, 5 or 10 products and 3, 4, 5, 8, 10 or 15 machines and 2, 4, 5, 6 and 10 factories in three sized are considered. Each problem set is composed of 20 randomly generated problems as follows in Table 2 and 3. There are 9 combinations of *n*, *m* and *g*. Twenty instances for each combination for a total of 180 instances are generated.

Table 2
Generate of the test problems

Parameter	Description
n	Small sized (3 and 5) , Medium sized (5) and Large sized (10)
m	Small sized (3, 4 and 5) , Medium sized (8, 10 and 15) and Large sized (5, 10 and 15)
g	Small sized (2 and 4) , Medium sized (4, 5 and 6) and Large sized (5 and 10)
$p_{j,i}$	uniform distribution (300,9000)
d_j	uniform distribution (50,500)
$st_{j,i}$	uniform distribution (0.01,0.5)
sc_j	$sc_j = \left(15000 \times \sum_{i=1}^{m} st_{j,i}\right) + 1000 \times$ uniform distribution (0,1)
$h_{j,i}$	$h_{j,i} = h_{j,i-1}$+uniform distribution (1,7) and $h_{j,1}$ = uniform distribution (1,10)
h_j	uniform distribution (10,170)×10%
M	10000

Table 3
Models comparison on the size complexity

Problem size	number of variables		number of constraints	
	model 1	model 2	model 1	model 2
3×3×2	99	24	490	103
3×4×2	164	27	1045	135
5×5×4	725	70	10216	891
5×8×5	3300	100	100751	2171
5×10×6	3950	105	120871	2581
5×15×4	6021	120	271247	2631
10×5×5	1675	200	50326	4716
10×10×10	12436	300	851502	18391
10×15×10	26886	350	2928002	27541

To verify efficiency of the proposed algorithms, all of the experimental tests have been implemented on a personal computer with a core i7 processor (2.5 GHz) and four GB RAM. The models and algorithms were coded by LINGO 16 and MATLAB (Version R2016a) language.

4. Metaheuristic Algorithms

In this section, two metaheuristic algorithms are explained in brief. For solving the models, use of metaheuristic algorithms have been considered for obtaining optimum values for the model 1 and 2. In this paper, metaheuristic algorithms including the Water Cycle Algorithm (WCA), Monarch Butterfly Optimization (MBO) Genetic Algorithm (GA) and combination of GA and simplex are used to find an approximate solution for the considered models.

Few researchers have considered methods with metaheuristic algorithms to support scheduling in distribution systems. Generally, distributed scheduling problems deal with the assignment of products to suitable factories and determine their production scheduling accordingly (Chan, et al. (2005)). The design of a suitable chromosome is the first step for a successful metaheuristic implementation because it applies probabilistic rule on each chromosome to create a population of chromosomes, representing a good candidate solution. In this approach, each chromosome for model 1 represents a solution corresponding to:
(i) The allocation of products to factories,
(ii) The production priority of each product in each machine.

A chromosome is composed of genes. Each gene consists of three parameters (i.e. *fij*), representing:
Factory number (*f*),
Machine number (*i*),
Product number (*j*),

Fig.1 shows a sample coding of a chromosome for the allocation and scheduling of three products to two factories, in which each factory has three machines. The scheduling result is shown in Fig.2.

fij	→	**111**	**123**	**121**	**131**	**222**	**212**	**232**	**113**	**133**

Fig.1. A sample coding of chromosome A

In Fig.1, the first gene (111) shows that j1 (Product 1) is allocated to *f*1's *i*1 (Factory 1's Machine 1).

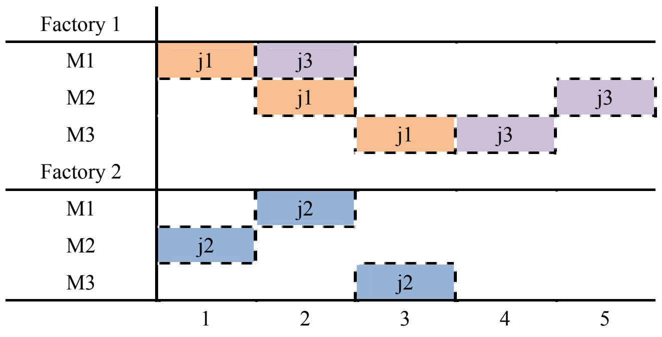

Fig. 2. Scheduling result of sample chromosome

The second step is to copy the first and the second positions from the gene of the chromosome A into the gene of the chromosome B, and change the last position of the chromosome A into last gene's position of the chromosome B by generated integer random number between one and the product number. If the new two neighbor genes in chromosome B are similar exactly to chromosome A, then change these two gene's positions and generate another integer random number. Fig.3 shows a sample coding of a chromosome B.

113	122	121	132	223	212	233	111	131

Fig. 3. A sample coding of chromosome B

The objective is to minimize the total cost of the system. It corresponds to the main objective function as model 1. Two types of chromosomes for model 2 are used. Chromosome type 1 is a linear and has the virtue of showing that every product must be done in one factory and in what sequence. For instance, consider the case with 2 products and 5 factories that should create 6 permutations as follows (Fig.4):

5	3	6	1	2	4

Fig. 4. A sample coding of chromosome

To determine which products should be produced in the first factory, must started from the left side of the chromosome and stop whenever to reach number 6, so numbers after 6 are related to the second factory. In other words, products 3 and 5 in the first factory and products 1, 2 and 4 must be produced in the second factory.
Generally, if the number of products is equal to n_p and the number of factories is equal to n_f, then the number of permutations will be n_p+n_f-1. If the objective function is makespan, to find the start time, knowing sequence is enough, and in fact, earliest time is the same start time. Otherwise, earliest time instead of start time could not found. Because in this model differences between start times in objective function is obtained, independent of the sequence, the delay time should be calculated. For this purpose, matrix of Chromosome type 2 with $n_p \times n_m$ dimension ($n_{p:}$ number of product, $n_{m:}$ number of machine) on the interval [0, 1] must be provided. The numbers in the matrix should be added as a delay time to the start time on the machine, then, earliest time with respect to the constraints must be earned.
In this paper, corrected one-point crossover for variable x and uniform crossover for variable y are used. Also, swap, insertion and inversion mutation for variable x and uniform mutation for variable y are applied.
Because of the complexity of exact methods due to the presence of discrete variables and to get closer answer via metaheuristic algorithms, in this paper for comparison between GA, WCA and MBO, simplex and GA (SGA) combined method is used. In SGA method, at first, x and y variables are calculated with GA and chromosome type 1. So, optimum sequence is used, then with respect to related constraints the model must be solved by simplex. In other words, value of S variable for each optimum sequence is obtained till the third part of the objective function is minimized.

4.1 Water cycle algorithm

4.1.1 Basic concepts

he idea of water cycle algorithm (WCA) was firstly originated from one of the natural phenomena by Eskandar et al. (2012) (Khodabakhshian et al. (2016)) and that is inspired by nature and is based on the observation of the water cycle process and how rivers and streams flow downhill towards the sea in the real world (sadollah et al. (2015)). To understand this further, an explanation on the basics of how rivers are created and water travels down to the sea is given as follows. A river, or a stream, is formed whenever water moves downhill from one place to another. This means that most rivers are formed high up in the mountains, where snow or ancient glaciers melt. The rivers always flow downhill. On their downhill journey and eventually ending up to a sea, water is collected from rain and other streams (Eskandar et al. (2012).
In the real world, as snow melts and rain falls, most of water enters the aquifer. There are vast fields of water reserves underground. The aquifer is sometimes called groundwater. The water in the aquifer then flows beneath the land the same way water would flow on the ground surface (downward). The underground water may be discharged into a stream (marsh or lake). Water evaporates from the streams and rivers, in addition to being transpired from the trees and other greenery, hence, bringing more clouds and thus more rain as this cycle counties (David, (1993) and Elhameed (2017).

4.1.2 Create the initial population

The WCA is a population-based algorithm; therefore, an initial population of designs variables (i.e., streams) is randomly generated between upper (UB) and lower (LB) bounds. In the proposed method terminologies such array is called ''Raindrop'' for a single solution (Eskandari et al. (2012)). The best individual, classified in terms of having the minimum cost function (or maximum fitness), is chosen as the sea. Then, a number of good individuals (i.e., cost function values close to the current best solution) are chosen as rivers, while all other streams are called streams which flow to rivers and sea. In an N dimensional optimization problem, a stream is an array of 1×N. This array is defined as follows (Sadollah et al. (2015)):

$$\text{Raindrop} = [x_1, x_2, x_3, \cdots, x_N] \qquad (35)$$

Where N is the number of design variables. To start the optimization algorithm, a candidate representing a matrix of raindrops of size N_{pop} N_{var} is generated (i.e. population of raindrops). Hence, the matrix X which is generated randomly is given as Eq. (36) (rows and column are the number of population and the number of design variables, respectively) (Eskandar et al. 2012).

$$Population\ of\ raindrops = \begin{bmatrix} Raindrop_1 \\ Raindrop_2 \\ Raindrop_3 \\ \vdots \\ Raindrop_{N_{pop}} \end{bmatrix} = \begin{bmatrix} x_1^1 & x_2^1 & x_3^1 & \cdots & x_{N_{var}}^1 \\ x_1^1 & x_2^1 & x_3^1 & \cdots & x_{N_{var}}^1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_1^{N_{pop}} & x_2^{N_{pop}} & x_3^{N_{pop}} & \cdots & x_{N_{var}}^{N_{pop}} \end{bmatrix} \quad (36)$$

Then, a number of good streams are chosen as rivers and the rest of the streams flow to the rivers and sea. Depending on the magnitude of their flow, rivers and sea absorb water from the streams. In fact, the amount of water in a stream entering a river and/or sea varies from the other streams (depends on their objective function value). In order to designate/assign streams to the rivers and sea depending on the intensity of the flow, the following equation is given as follows (Sadollah et al. (2015)):

$$C_n = Cost_n - Cost_{N_{sr+1}} \qquad n=1, 2, 3, ..., N_{sr} \quad (37)$$

$$NS_n = Round\left\{\left|\frac{C_n}{\sum_{i=1}^{N_{sr}} C_i}\right| \times N_{Stream}\right\}, \quad n = 1,2,\dots,N_{sr} \quad (38)$$

Where NS_n and N_{stream} are the number of streams that flow to the specific rivers and number of sea and rivers, respectively. Indeed, in coding WCA, the costs of sea and each river have been deducted by the cost of an individual (i.e., N_{sr+1}) in stream population as can be seen in Eq. (37). Streams tend to move toward sea and rivers based on their magnitude of flow (intensity of flow). It means more streams flow into the sea than rivers. Hence, one of the best approaches to hand out the streams among sea and rivers in proportional way is to use the cost functions (fitness functions) of the sea and rivers. Since the solution of the equation must be obtained in the form of integer value, round operator is used.

4.1.3 Flow of the stream to the river or sea

According to process for the WCA, rivers and sea are affected by the flow of streams and are selected as better and best solutions respectively. Therefore, sea receives more streams than rivers do. For a minimization problem, the lowest cost is obtained for streams which flow to the sea. Also, the lower cost is considered for other streams which flow to rivers. According to the nature of water cycle process, streams are made of the raindrops and gather to create new rivers. Some streams may flow directly to the sea. Also, all rivers and streams will finally arrive into the sea which is the best solution. For a set of N_{pop} streams, it is assumed that NS_{n-1} numbers of streams are considered as rivers and one of them is chosen as the sea. The relation between a stream and a significant river is depicted in Fig. 5. In this figure, circle and star are shown as stream and river respectively. Also, parameter X is defined as the distance between a stream and the associated river. X is randomly calculated as follows (Sadollah et al. (2015)):

$$\square\ (0, C\times d), \quad C > 1 \quad (39)$$

Where C is an arbitrary value between 1 and 2. The best selection for the value of C is 2. This parameter is set more than 1 because this selection lets streams flow in different directions into rivers. Therefore, as the exploitation phase in the WCA, the new positions for streams and rivers can be given in the following equations (Sadollah et al. (2015)):

$$X\ Stream(t+1) = X\ Stream(t) + rand \times C \times (X\ River(t) - X\ Stream(t)) \quad (40)$$

$$X\ Stream(t+1) = X\ Stream(t) + rand \times C \times (X\ Sea(t) - X\ Stream(t)) \quad (41)$$

$$X\ River(t+1) = X\ River(t) + rand \times C \times (X\ Sea(t) - X\ River(t)) \quad (42)$$

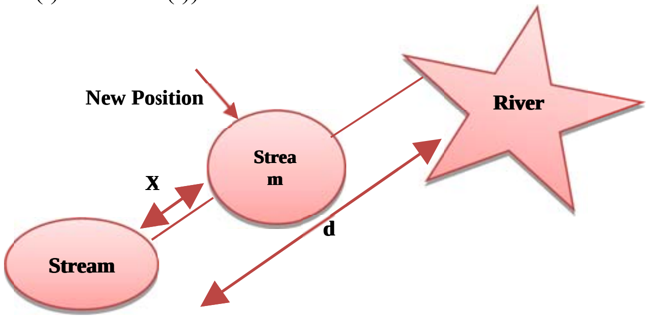

Fig. 5. Schematic view of the stream flowing into a significant river (Khodabakhshian et al. (2016))

Where *rand* is an uniformly distributed random number ([0,1]). Eqs. (40)-(41) are considered as updating equations for new positions of streams which flow to rivers and sea, respectively, while Eq. (42) is the updated equation for the rivers which pour to the sea. Notations having vector signs correspond vector values, otherwise the rest of notations and parameters considered as scalar values. To prevent an imperfect convergence, the evaporation process operator is considered in the proposed algorithm. In a real world, this process means that the evaporation of the water of sea occurs and the water vapor goes to the sky for next precipitations. This stage of the water cycle is considered for the river/stream that is close enough to the sea (Khodabakhshian et al. (2016)). Therefore, the occurrence of the evaporation process is verified by using a criterion as follows:

$$\begin{cases} if\ \mid X\ Sea(t) - X\ River(t) \mid < d_{max}\ or\ rand < 0.1 \\ \qquad i = 1,2,3,\dots,N_{sr} \\ then\ evaporation\ and\ raining\ process\ is\ occured \end{cases} \quad (43)$$

Where d_{max} refers to a small number of distance near zero. Therefore, if the distance between a river and sea is less than d_{max}, it indicates that the river has reached the sea and controls the search intensity near the optimum solution (the sea). After satisfying the evaporation process, the raining process is applied. In the raining process, new streams form in different locations (acting similar to the mutation operator in the GAs). In fact, the evaporation condition is responsible for the exploration phase in the WCA. For termination criteria, the WCA proceeds until the maximum number of iterations as a convergence criterion is satisfied. The schematic view of the WCA is illustrated in Fig. 6 where circles, stars, and the diamond correspond to streams, rivers, and sea, respectively. From Fig. 6, the white (empty) shapes refer to the new positions found by streams and rivers (Sadollah et al. (2015)).

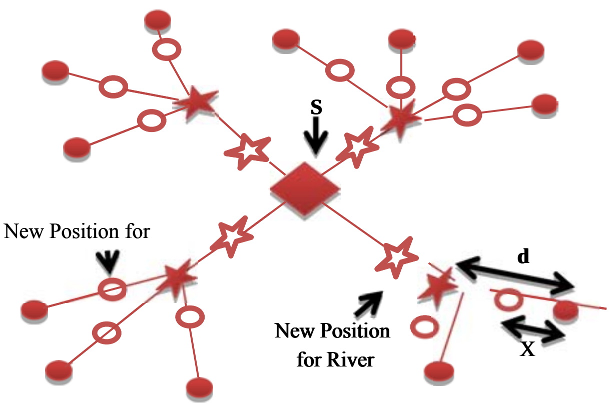

Fig. 6. Schematic view of the WCA's movement towards the best optimum point (sea) (Sadollah et al. 2015)

4.2. Monarch butterfly optimization

Recently, by examining and simulating the migration behavior of monarch butterflies in nature, Wang et al. (2016) proposed a new swarm intelligence-based metaheuristic algorithm, called monarch butterfly optimization (MBO), for addressing various global optimization tasks. The foundation of this algorithm is based on the study of the migration characteristics of monarch butterflies. In MBO method, the butterflies in Land 1 and Land 2 are updated by migration operator and butterfly adjusting operator, respectively. These optimization processes are repeated till any pre-fixed stop condition is satisfied.

In order to make the migration behavior of monarch butterflies address various optimization problems, the migration behavior of monarch butterflies can be idealized into the following rules (Wang et al. (2015)).

1. All the monarch butterflies are only located in Land 1 or Land 2. That is to say, monarch butterflies in Land 1 and Land 2 make up the whole monarch butterfly population.
2. Each child monarch butterfly individual is generated by migration operator from monarch butterfly in Land 1 or in Land 2.
3. In order to keep the population unchanged, an old monarch butterfly will pass away once a child is generated. In the MBO method, this can be performed by replacing its parent with newly generated one if it has better fitness as compared to its parent. On the other hand, the newly generated one is liable to be discarded if it does not exhibit better fitness with respect to its parent. Under this scenario, the parent is kept intact and undestroyed.
4. The monarch butterfly individuals with the best fitness move automatically to the next generation, and they cannot be changed by any operators. This can guarantee that the quality or the effectiveness of the monarch butterfly population will never deteriorate with the increment of generations

4.2.1 Migration operator

According to the time when monarch butterflies stay at Land 1 and Land 2, their numbers in Land 1 and Land 2 can be considered as ceil(p*NP) (NP1, Subpopulation 1) and NP-NP1 (NP2, Subpopulation 2), respectively. Here, ceil(x) rounds x to the nearest integer greater than or equal to x; NP is the population size; p is the ratio of monarch butterflies in Land 1. This migration process can be formulated below (Wang et al. (2015)):

$$x_{i,k}^{t+1} = x_{r_1,k}^{t} \tag{44}$$

Where $x_{i,k}^{t+1}$ indicates the kth element of x_i at generation t+1. Similarly, $x_{r_1,k}^{t}$ indicates the kth element of x_{r_1}. t is the current generation number. Butterfly r1 is randomly selected from Subpopulation 1. When $r \le p$, $x_{i,k}^{t+1}$ is generated by Eq. (44). Here, r can be calculated as

$$r = rand * peri \tag{45}$$

$peri$ indicates migration period and is set to 1.2 in the basic MBO method. *rand* is a random number. If r>p, $x_{i,k}^{t+1}$ is generated by

$$x_{i,k}^{t+1} = x_{r_2,k}^{t} \tag{46}$$

Where $x_{r_2,k}^{t}$ indicates the kth element of x_{r_2} , and butterfly r2 is randomly selected from Subpopulation 2. Accordingly, the migration operator can be represented in Algorithm 1.

Algorithm 1 Migration operator (Wang et al. (2015))

```
Begin
    for i= 1 to $NP_1$ (for all monarch butterflies in Subpopulation 1) do
        for k=1 to D (all the elements in ith monarch butterfly) do
            Randomly generate a number rand by uniform distribution;
            r=rand*peri;
            if $r \le p$ then
                Randomly select a monarch butterfly in Subpopulation 1 (say $r_1$);
                Generate $x_{i,k}^{t+1}$ by Eq. (44).
            else
                Randomly select a monarch butterfly in Subpopulation 2 (say $r_2$);
                Generate $x_{i,k}^{t+1}$ by Eq. (46).
            end if
        end for k
    end for i
End.
```

4.2.2 Butterfly adjusting operator

For all the elements in butterfly j, if rand ≤ p, it can be updated as

$$x^{t+1}_{j,k} = x^{t}_{best,k} \tag{47}$$

Where $x^{t+1}_{j,k}$ indicates the kth element of x_j at generation t+1. Similarly, $x^{t}_{best,k}$ indicates the kth element of the fittest butterfly x_{best}. If rand>p, it can be updated as:

$$x^{t+1}_{j,k} = x^{t}_{r_3,k} \tag{48}$$

where $x^{t}_{r_3,k}$ indicates the kth element of x_{r_3}. Here, r3 □ {1, 2, . . ., NP_2}.

Under this condition, if rand>BAR, it can be further updated as follows:

$$x^{t+1}_{j,k} = x^{t+1}_{j,k} + \alpha \times (dx_k - 0.5) \tag{49}$$

Where BAR indicates butterfly adjusting rate. *dx* is the walk step of butterfly j and which can be calculated by Levy flight.

$$dx = Levy(x^{t}_{j}) \tag{50}$$

Lévy flights, named after the French mathematician Paul Lévy, are a class of random walks in which the step lengths are drawn from a probability distribution with a power law tail. These probability distributions are known as Lévy distributions or stable distributions (Brown, C. et al. (2007)).

In Eq. (49), α is the weighting factor as shown in Eq. (51).

$$\alpha = {S_{max}}/{t^2} \tag{51}$$

where S_{max} is max walk step.

The main steps of the butterfly adjusting operator can be given in Algorithm 2 (Wang et al. (2016)).

Algorithm 2 Butterfly adjusting operator

```
Begin
    for j= 1 to NP_2 (for all butterflies in Subpopulation 2) do
        Calculate the walk step dx by Eq. (50);
        Calculate the weighting factor by Eq. (51);
        for k=1 to D (all the elements in jth butterfly) do
            Randomly generate rand;
            if rand ≤ p then
                Generate $x^{t+1}_{j,k}$ by Eq. (47).
            else
                Randomly select a butterfly in Subpopulation 2 (say r_3);
                Generate $x^{t+1}_{j,k}$ by Eq. (48).
                if rand>BAR then
                    $x^{t+1}_{j,k} = x^{t+1}_{j,k} + \alpha \times (dx_k - 0.5)$;
                end if
            end if
        end for k
    end for j
End
```

4.2.3 Schematic presentation of MBO algorithm

By idealizing the migration behavior of the monarch butterfly individuals, MBO method can be formed, and its schematic description can be given as shown in Algorithm 3. A brief presentation of the MBO algorithm is shown in Fig. 7.

According to Algorithm 3, firstly, all the parameters are initialized followed by the generation of initial population and evaluation of the same by means of its fitness function. Subsequently, the positions of all monarch butterflies are updated step by step until certain conditions are satisfied. It should be mentioned that, in order to make the population fixed and reduce fitness evaluations, the number of monarch butterflies, generated by migration operator and butterfly adjusting operator, are NP1 and NP2, respectively (Wang et al. 2015).

Algorithm 3 Monarch Butterfly Optimization algorithm

```
Begin
    Step 1: Initialization. Set the generation counter
            t = 1; initialize the population P of NP monarch butterfly
            individuals randomly; set the maximum generation
            MaxGen, monarch butterfly number NP1 in Land 1 and
            monarch butterfly number NP2 in Land 2, max step SMax,
            butterfly adjusting rate BAR, migration period peri, and the
            migration ratio p.
    Step 2: Fitness evaluation. Evaluate each monarch butterfly
              according to its position.
    Step 3: While the best solution is not found or t < MaxGen do
              Sort all the monarch butterfly individuals according to
              their fitness.
              Divide monarch butterfly individuals into two
              subpopulations (Land 1 and Land 2);
              for i= 1 to NP1 (for all monarch butterflies in
              Subpopulation 1) do
                  Generate new Subpopulation 1 according to
              Algorithm 1.
              end for i
              for j= 1 to NP2 (for all monarch butterflies in
              Subpopulation 2) do
                  Generate new Subpopulation 2 according to
              Algorithm 2.
              end for j
              Combine the two newly-generated subpopulations into
              one whole population;
              Evaluate the population according to the newly updated
              positions;  t = t+1.
    Step 4: end while
    Step 5: Output the best solution.
End.
```

4.3 Parameter tuning

In this section, tuning the input parameters of four algorithms is focused. Since all meta-heuristic algorithms are severely depends on their parameters. So, response surface methodology (RSM) is applied to tune the algorithms parameters. RSM is a collection of statistical and mathematical techniques useful for optimization particularly in situations where several input variables potentially influence some performance measure or quality characteristic called response [28]. Here the aim is to find the levels of the algorithms' parameters (as input variables) so that response variable (objective function) is optimized. For instance, Population size, number of iteration, crossover probability and mutation probability are considered in GA that each given the values of -1, 0, and 1 for their low, medium and high levels, respectively in MINITAB software.

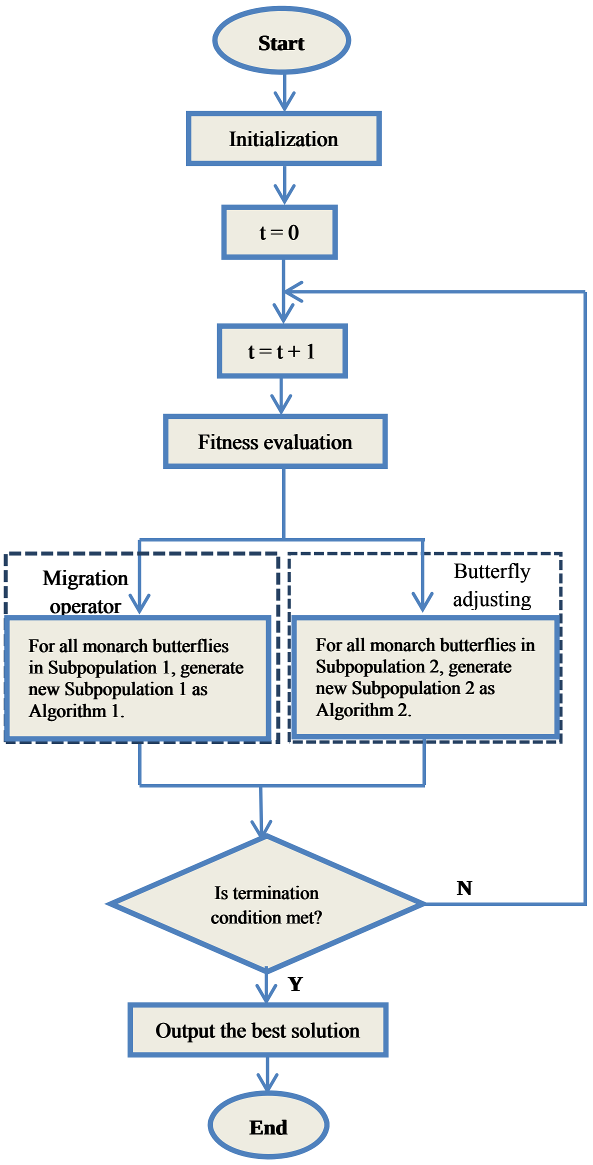
Start
Initialization
t = 0
t = t + 1
Fitness evaluation
Migration operator
Butterfly adjusting
For all monarch butterflies in Subpopulation 1, generate new Subpopulation 1 as Algorithm 1.
For all monarch butterflies in Subpopulation 2, generate new Subpopulation 2 as Algorithm 2.
Is termination condition met?
N
Y
Output the best solution
End

Fig. 7. Schematic flowchart of MBO method (Wang et al. (2015))

In this method, central composite design with 4 factors for 3 sizes of problems as an example with 3×3×2, 3×4×2 and 5×5×4 for small size, 5×8×5, 5×10×6 and 5×15×4 for medium size and 10×5×5, 10×10×10 and 10×15×10 for large size is used, that these numbers indicate n×m×g. According to RSM, values of lack of fit for all sizes are larger than 0.05 (i.e 0.081, 0.067 and 0.958, respectively), so indicates that the fitted regression functions with 95% confidence level are true. Based on the solution obtained using regression functions by LINGO, the optimum values of the parameters are finally obtained that are presented in Table 4.

Table 4
Optimum value of input parameters in GA

problem	Npop	Niteration	Pc	Pm
Small	20	420	0.95	0.5
Medium	30	1200	0.8	0.5
Large	62	12000	0.7	0.3

For other metaheuristic this method is duplicated. Then, the following Tables 5-7 are obtained for these problems. Also, Fig 8 shows convergence plot as an example just for large size. SGA could not find optimum objective in 3600 second for large size (see Table5 for large problem).

Table 5
Optimum value of input parameters in SGA

problem	Npop	Niteration	Pc	Pm
Small	3	4	0.7	0.3
Medium	4	7	0.7	0.3
Large	?	?	?	?

Table 6
Optimum value of input parameters in WCA

problem	Npop	Niteration	Nsr	dmax
Small	30	100	2	0.01
Medium	50	150	3	0.01
Large	100	500	4	0.05

Table 7
Optimum value of input parameters in MBO

problem	Npop	Niteration	NP1	P
Small	30	100	28	7/12
Medium	30	200	38	5/12
Large	50	400	48	7/12

5. Computational Results and Models Evaluation

In this section, nine different problems with three size numerical examples are applied (i.e. small, medium and large); for each problem size, 20 problem instances have been randomly generated then, to demonstrate performance of the proposed algorithms, the results are analyzed statistically and graphically. For each numerical example, 10 independent runs have been performed by the LINGO, GA, SGA, WCA and MBO to decrease uncertainty of generated runs. Then, the reported value is the algorithms outputs in these 10 runs which are shown in Table 8-10 for CPU TIME, MIN, and RPD (Relative percentage deviation) criteria, respectively.
The algorithms outputs are compared with each other in following terms:

(I) RPD: This criterion is one of the well-developed approaches for measuring the efficiency in mathematical programming models. RPD is obtained as Eq. (52):

$$RPD=[(MIN_{stage}-MIN_{total})/MIN_{total}]100\% \qquad (52)$$

where MIN_{stage} and MIN_{total} are the best cost of algorithm in each stage and best cost that earned up to now, respectively. Obviously, the algorithms with lowest RPD are the best one.

(II) Best cost (Minimum): The algorithms with better objective function are the best one.

(III) CPU TIME: The computational time of running the algorithms to reach best solutions.

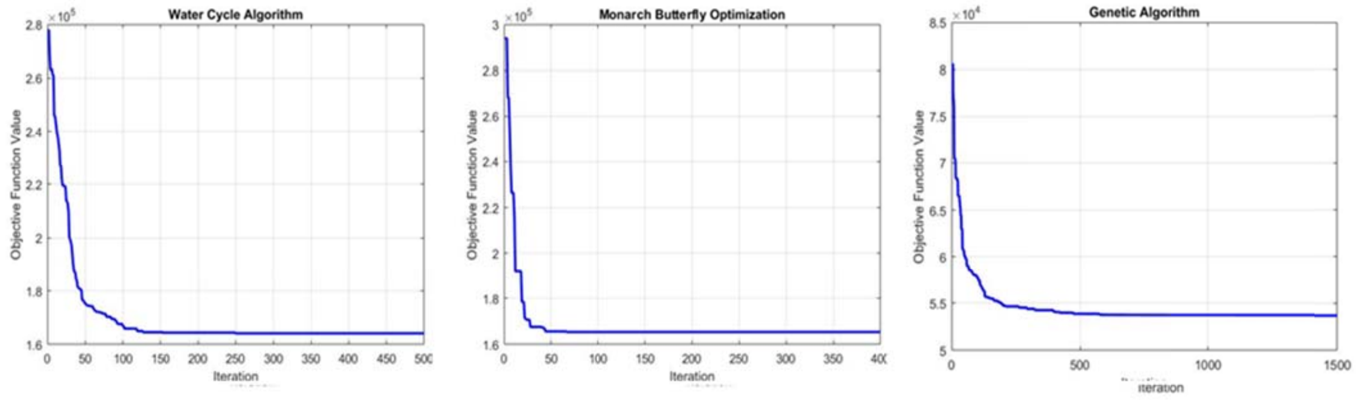

Fig. 8. Convergence plot for GA, MBO and WCA in large size

In general based on the outputs of Table 8-10 and Fig 9-11, it is clear that model 2 shows better performances in terms of all criteria rather than model 1, so model 2 is chosen and results of this model are analyzed. From Table 8 and Fig 9 it becomes clear that the model 2 is the better model for solving the proposed problem in compare to this criteria for model 1. This is due to low average CPU T me for small, medium and large sizes. In addition, the WCA was able to find solution for small and medium size and MBO for large size in better time between other algorithms except LINGO for model 2. Based on the results provided by Table 9 and Fig 10 the WCA performs ell in minimum criteria for small and large sizes but the erformance of the GA is better than other algorithms in medium size, except LINGO for all instances in model 2. The GA and WCA are able to provide lower RPD in each t st problem but WCA is the best between all metaheuristic algorithms and SGA is the worst according to Table 10 and Fig 11.

For all size problem instances, the solutions of algorithms have been compared with the LINGO's solution. In summary, the following observations can be made from t e numerical experiments for average of all sizes in model 2 (Table 11).

Table 8
CPU time of all problems for LINGO, GA, SGA, WCA and MBO

size	problem	CPU Time for model 1					CPU Time for model 2				
	n×m×g	LINGO	GA	SGA	WCA	MBO	LINGO	GA	SGA	WCA	MBO
small	3×3×2	0.2	4.29	1.09	1	1.31	0.07	1.5	0.38	0.25	0.46
	3×4×2	0.28	6.41	1.12	1.12	1.58	0.08	1.83	0.32	0.22	0.45
	5×5×4	1.35	21.71	164.93	5.4	12.26	0.12	1.93	14.66	0.43	1.09
Average		0.61	10.80	55.71	2.51	5.05	0.09	1.75	5.12	0.3	0.66
medium	5×8×5	78.52	1748.85	2227.77	392.6	874.43	0.22	4.9	62.39	1.45	2.45
	5×10×6	112.49	1713.6	3301.94	449.96	926.17	0.3	4.57	88.04	1.39	2.47
	5×15×4	53.25	1122.17	1244.32	213	394.61	0.95	20.02	221.94	5.15	7.04
Average		81.42	1528.21	2258.01	351.85	731.73	0.49	9.83	124.12	2.66	3.98
large	10×5×5	83.71	712.14	3452.32	418.55	562.92	0.69	5.87	3669.68	3.9	4.64
	10×10×10	*	2798.66	3600	3600	2530.97	1.47	28.02	3600	45.61	25.34
	10×15×10	*	3600	3600	3600	787.33	1.61	49.1	3600	71.44	34.57
Average		?	2370.27	3550.77	2539.52	2231.29	1.25	27.66	3623.22	40.31	21.51

T ble 9
Minimum of all problems for LINGO, GA, SGA, WCA and MBO

size	proble	MIN for model 1					MIN for model 2				
	n×m×g	LINGO	GA	SGA	WCA	MBO	LINGO	GA	SGA	WCA	MBO
small	3×3×2	9198.58	9198.58	9198.58	9198.58	9198.58	9198.58	9198.58	9198.58	9198.58	9198.5
	3×4×2	11867.58	11867.84	11867.58	11867.58	11867.58	11867.58	11867.84	11867.58	11867.58	11867.58
	5×5×4	20311.47	20311.47	20311.47	20311.47	20311.47	20311.47	20311.47	20311.47	20311.47	20311.47
Average		13792.54	13792.63	13792.54	13792.54	13792.54	13792.54	13792.63	13792.54	13792.54	13792.54
medium	5×8×5	47238.51	47238.51	47238.51	47238.51	47238.51	47238.51	47238.51	47238.51	47238.51	47238.51
	5×10×6	50535.19	50535.19	50535.19	50535.19	50535.19	50535.19	50535.19	50535.19	50535.19	50535.19
	5×15×4	67174.25	68778.83	69469.34	67700.31	67441.33	67174.25	67778.83	68251.81	67785.5	68441.33
Average		54982.6	55517.51	55747.68	55158.00	55071.68	54982.65	55184.18	55341.84	55186.4	55405.
large	10×5×5	53722.85	53722.85	53945.06	53722.85	53722.85	53722.85	53722.85	53745.06	53722.9	53826.63
	10×10×10	*	163971.8	*	163971.8	163971.8	163971.8	163971.8	168241.9	163971.8	163971.8
	10×15×10	*	245155.1	*	245155.1	245706.8	245155.1	245926.6	260112.3	245155.1	245706.8
Average		?	154283.3	?	154283.3	154467.1	154283.3	154540.4	160699.8	154283.3	154501.7

Table 10
RPD of all problems for LINGO, GA, SGA, WCA and MBO

size	problem	RPD for model 1					RPD for model 2				
	n×m×g	LINGO	GA	SGA	WCA	MBO	LINGO	GA	SGA	WCA	MBO
small	3×3×2	0	0	0	0	0	0	0	0	0	0
	3×4×2	0	0	0	0	0	0	0	0	0	0
	5×5×4	0	0	0	0	0	0	0	0	0	0
Average		0	0	0	0	0	0	0	0	0	0
medium	5×8×5	0	0	0	0	0	0	0	0	0	0
	5×10×6	0	0	0	0	0	0	0	0	0	0
	5×15×4	0	0.020	0.030	0.008	0.003	0	0.009	0.010	0.009	0.018
Average		0	0.007	0.010	0.003	0.001	0	0.003	0.003	0.003	0.006
large	10×5×5	0	0	0.004	0	0	0	0	0	0	0.001
	10×10×10	?	0	?	0	0	0	0	0.026	0	0
	10×15×10	?	0	?	0	0.002	0	0.003	0.061	0	0.002
Average		?	0	?	0	0.001	0	0.001	0.029	0	0.001

Table 11
Comparison of all algorithms for Model 2

Model 2	LINGO	GA	SGA	WCA	MBO
CPU Time	reference	Less than SGA, WCA	More than all	Less than SGA	Less than GA, SGA, WCA
MIN	reference	Less than SGA, MBO	More than all	Less than GA, SGA, MBO	Less than SGA
RPD	reference	Less than SGA, MBO	More than all	Less than GA, SGA, MBO	Less than SGA

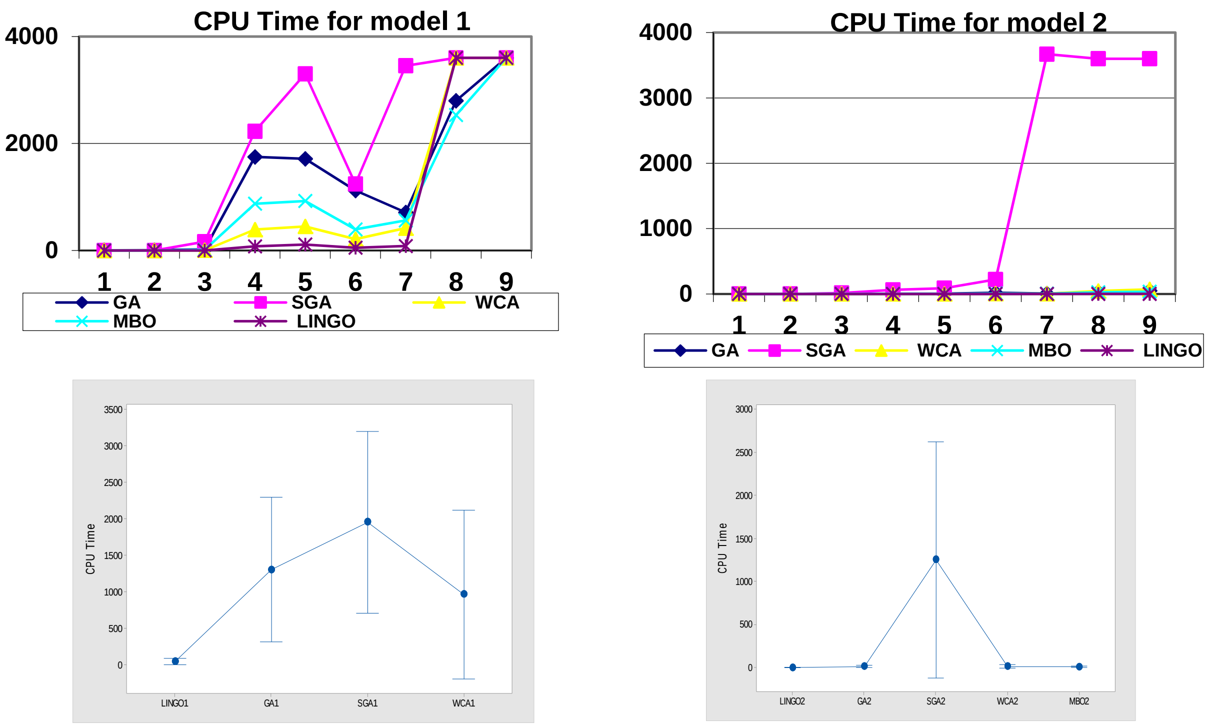

Fig. 9. CPU Time's box plot and graphical comparison of all methods for nine problems in model 1 and 2

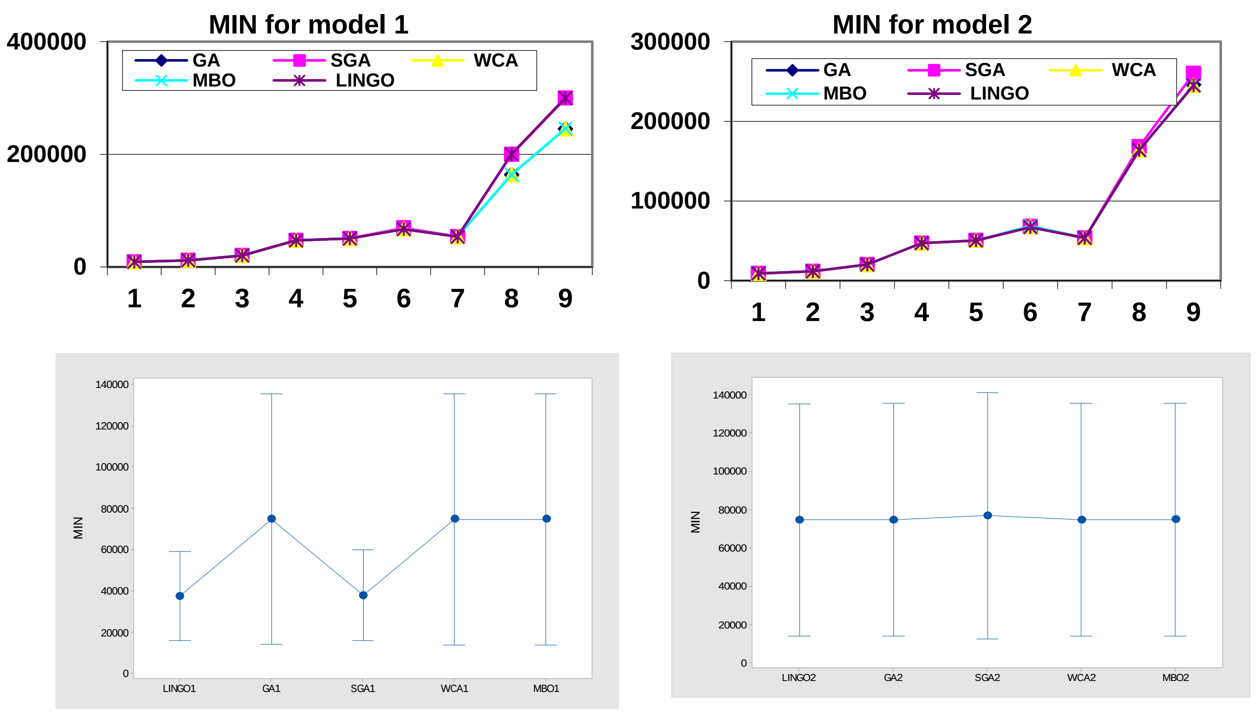

Fig. 10. MIN's box plot and graphical comparison of all methods for nine problems in model 1 and 2

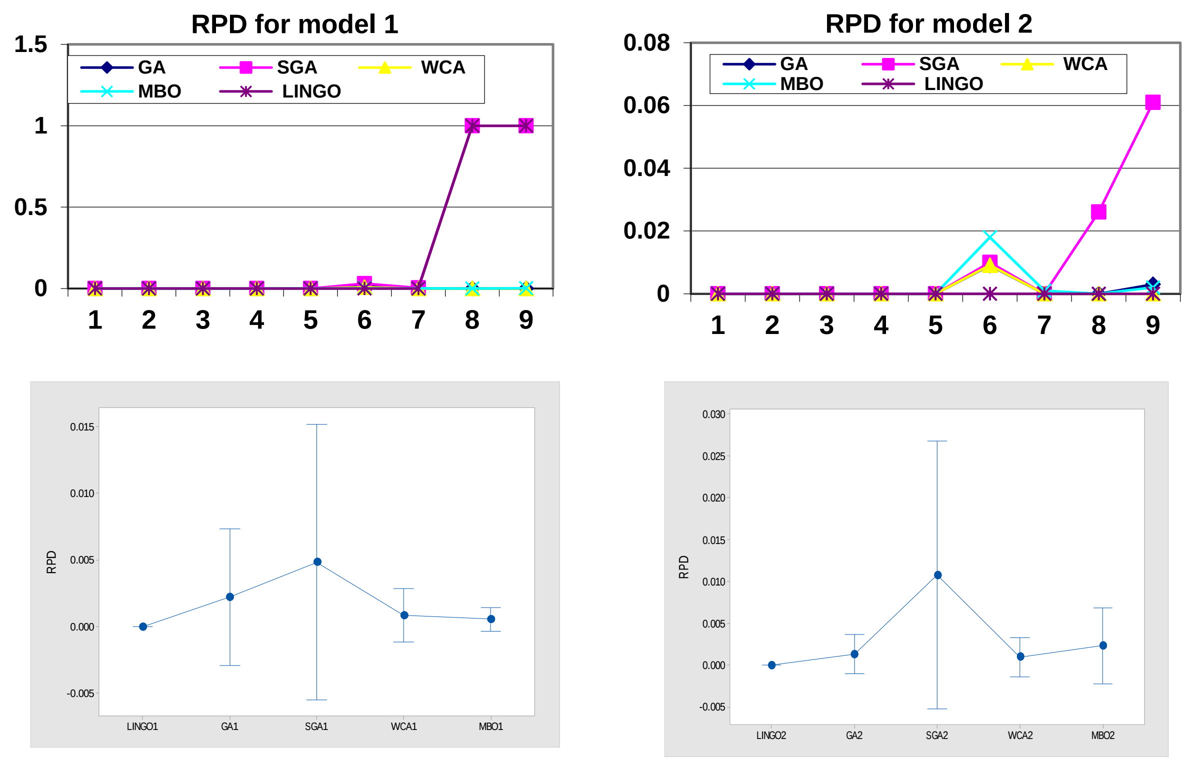

Fig. 11. RPD's box plot and graphical comparison of all methods for nine problems in model 1 and 2

6. Conclusion and Future Research Directions

This paper presented mathematical models to design economic nonlinear distributed permutation flow shop lot sizing and scheduling problem. In these problems two interconnected decisions are to be taken: assignment of products to factories and sequencing of the products assigned to each factory in sequence base and position base aspect. Linearizations for the models have been proposed and MINLP and MILP performance analyzed. Since the proposed models are NP-hard, GA, SGA, WCA and MBO were applied and compared with test problems via three metrics based on RPD, MIN and CPU Time. Among the proposed models, a sequence-based model (Model 2) has produced the best results, as shown by the careful statistical analysis and Table 3. According to parameter tuning procedure and outputs, the results show that the proposed linearization significantly works better than other meta-heuristic algorithms in terms of RPD, MIN and CPU Time criteria for small and medium sizes of Model 1 and better than other algorithms for all criteria and sizes of Model 2, while the WCA shows better performance in RPD and MIN criteria and the MBO in CPU Time.

As a direction for future studies, since this paper considers identical factories and machines, it could be interesting to develop distinct factories with different machine capacities and limited buffers. So, in order to solve the mathematical model, other algorithms and exact methods can be proposed.

References

Akrami, B., Karimi, B., Moattar 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. *Applied Mathematics and Computation*, 183, 634–645

Bargaoui, H., Driss, O.B., Ghedira, K. (2017). A Novel Chemical Reaction Optimization for the Distributed Permutation Flow shop Scheduling Problem with Makespan Criterion, *Computers & Industrial Engineering*, doi: http://dx.doi.org/10.1016/j.cie.2017. 07.020.

Brown, C., Liebovitch, L.S., & Glendon, R. (2007). Lévy flights in Dobe Ju/'hoansi foraging patterns. *Human Ecol*, 35: 129-138.

Chan F.T.S., Chung S.H., & Chan P.L.Y. (2005). An adaptive genetic algorithm with dominated genes for distributed scheduling problems. *Expert System with Applications*, Vol. 29: 364-371.

David S. (1993). The water cycle, illustrations by John Yates, *Thomson Learning*, New York.

Dobson, G., & Yano, C.A. (1994). Cyclic scheduling to minimize inventory in a batch flow line. *European Journal of Operational Research*, 75: 441–461.

Eilon, S. (1959). Economic batch-size determination for multi-product scheduling. *Operations Research*, 10(4), 217–227.

Elhameed, A., & El-Fergany, A.A. (2017). Water Cycle Algorithm-based Economic Dispatcher for Sequential and Simultaneous Objectives Including Practical Constraints. *Applied Soft Computing*. DOI.org/10.1016/j.asoc.2017.04.046.

El-Najdawi, M. (1989). Common cycle approach to lot-size scheduling for multistage, multiproduct production processes. *Unpublished PhD. Dissertation, Wharton School, University of Pennsylvania*, Philadelphia, PA.

El-Najdawi, M. (1992). A compact heuristic for common cycle lot-size scheduling in multi-stage, multi-product production processes. *International Journal of Production Economics*, 27(1), 29–41.

El-Najdawi, M., & Kleindorfer, P.R. (1993). Common cycle lot-size scheduling for multi-product, multi-stage production. *Management Science*, 39:872–885.

El-Najdawi, M. (1994). A job-splitting heuristic for lot-size scheduling in multi-stage, multi-product production processes. *European Journal of Operational Research*, 75, 365–377.

Eskandar, H., Sadollah, A., Bahreininejad, A., & Hamdi, M. (2012). Water cycle algorithm: A novel metaheuristic optimization method for solving constrained engineering optimization problems. *Computers and Structures*, 151-166

Hsu, J.I.S., & El-Najdawi, M. (1990). Common cycle scheduling in a multistage production process. *Engineering Costs and Production Economics*, 20: 73–80.

Karimi, B., Fatemi Ghomi, S.M.T., & Wilson, JM. (2003). The capacitated lot sizing problem: a review of models and algorithms. *Omega*, 31 (5), 365-378.

Khodabakhshian, A., Esmaili, M-R., & Bornapour, M. (2016). Optimal coordinated design of UPFC and PSS for improving power system performance by using multi-objective water cycle algorithm. *Electrical Power and Energy Systems*, 83, 124-133.

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

Maxwell, W.L. (1964). The scheduling of economic lot sizes. *Naval Research Logistics Quarterly*, 11, 89–124.

Naderi, B., Ruiz, R. (2009). The distributed permutation flowshop problem. *Computers and Operations Research*, doi.10.1016/j.cor. 2009.06. 019

Ouenniche, J., Boctor, F.F., & Martel, A. (1999). The impact of sequencing decisions on multi-item lot sizing and scheduling in flow shops. *International Journal of Production Research*, 37(10), 2253–2270.

Pakzad-Moghaddam, S.H., Tavakkoli-Moghaddam, R., & Mina, H. (2014). An approach for modeling a new single machine scheduling problem with deteriorating and learning effects. *Computers & Industrial Engineering*. DOI.org/10.1016/j.cie.2014.09.021.

Rahman, H.F., Sarker, R., Essam, D. (2015). A Genetic Algorithm for Permutation Flow Shop Scheduling under Make to Stock Production system. *Computers & Industrial Engineering*, doi: http://dx.doi.org/ 10.1016/j.cie.2015.08.006

Rifai, A. P., Nguyen, H.T., & Dawal, S.Z. (2016). Multi-objective adaptive large Neighborhood search for distributed reentrant permutation flow shop scheduling. *Applied Soft Computing*, 40, 42–57.

Rodriguez, M.A., Vecchietti, A.R., Harjunkoski, I., & Grossmann, I.E. (2014). Optimal supply chain design and management over a multi-period horizon under demand uncertainty. Part I: MINLP and MILP models. *Computers and Chemical Engineering,* 62, 194–210.

Rogers, J. (1958). A computational approach to the economic lot scheduling problem. *Management Science*, 4(3), 264–291.

Sadollah, A., Eskandar, H., Bahreininejad, A., & Kim, J-H. (2015). Water cycle algorithm with evaporation rate for solving constrained and unconstrained optimization problems. *Applied Soft Computing*, 30: 48-51.

Sadollah, A., Eskandar, H., Yoo, D-G., & Kim, J-H. (2015). Approximate solving of nonlinear ordinary differential equations using least square weight function and metaheuristic algorithms., *Engineering Applications of Artificial Intelligence*, 40: 117-132.

Torabi, S.A., Karimi, B., & Fatemi Ghomi, S.M.T. (2005). The common cycle economic lot scheduling in flexible job shops: The finite horizon case. *International Journal of Production Economics*, 97, 52–65.

You, F., & Grossmann, I.E. (2008). Integrated Multi-Echelon Supply Chain Design with Inventories under Uncertainty: MINLP Models. *Computational Strategies*, Carnegie Mellon University, USA.

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

Wang, B. (1997). Integrated product, process and enterprise design. *Chapman and Hall*, 1st Edition,

Wang, G-G., Deb, S., Zhao, X., & Cui, Z. (2016). A new monarch butterfly optimization with an improved crossover operator. *Oper Res Int J.* DOI 10.1007/s12351-016-0251-z

Wang, G-G., Deb, S., & Cui, Z. (2015). Monarch butterfly optimization. *Neural Comput & Applic*. DOI:10.1007/ s00521-015-1923-y.

Wang, G-G., Zhao, X., & Deb, S. (2015). A Novel Monarch Butterfly Optimization with Greedy Strategy and Self-adaptive Crossover Operator, *2nd international conference on soft computing & machine intelligence*, Hong Kong, (ISCMI 2015), Nov 23–24. DOI 10.1109/ISCMI.2015.19

This article can be cited: Alaghebandha M., Naderi B. & Mohammadi M. (2019). Economic Lot Sizing and Scheduling in Distributed Permutation Flow Shops. *Journal of Optimization in Industrial Engineering*. 12 (1), 103- 117.

http://www.qjie.ir/article_543808.html
DOI: 10.22094/JOIE.2018.542997.1510