Computers & Industrial Engineering 105 (2017) 158–173

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

# Improved discrete cuckoo optimization algorithm for the three-stage assembly flowshop scheduling problem

G.M. Komaki [a,*], Ehsan Teymourian [b], Vahid Kayvanfar [c], Zahra Booyavi [d]

[a] Department of Electrical Engineering and Computer Science, Case Western Reserve University, 10900 Euclid Ave., Cleveland, OH 44106, USA
[b] Department of Management Science and Information Systems, Rutgers Business School Newark and New Brunswick, 1 Washington Park, Newark, NJ 07102, USA
[c] Department of Industrial Engineering, Amirkabir University of Technology, 424 Hafez Ave., 15875-4413 Tehran, Iran
[d] Department of Industrial Engineering, University of Science & Culture, Ashrafie Esfahani Ave., Tehran, Iran

ARTICLE INFO

*Article history:*
Received 10 June 2016
Received in revised form 25 December 2016
Accepted 5 January 2017
Available online 7 January 2017

*Keywords:*
Scheduling
Discrete Cuckoo Optimization Algorithm
Three-stage assembly flowshop
Makespan

ABSTRACT

The three-stage assembly flow shop scheduling problem, where the first stage has parallel machines and the second and the third stages have a single machine, is addressed in this study. Each product has made of several components that after processing at the first stage are collected and transferred to the third stage to assemble them as the product. The goal is to find products' sequence to minimize completion time of the last product, makespan. Since the problem is NP-hard, an improved version of Cuckoo Optimization Algorithm (COA), a bio-inspired meta-heuristic, is proposed which incorporates new adjustments such as clustering, egg laying and immigration of the cuckoos based on a discrete representation scheme. These novel features result in an Improved Discrete version of COA, called IDCOA, which works efficiently. Also, for the addressed problem, a lower bound and some dispatching rules are proposed. The performance of the employed algorithms through randomly generated instances is evaluated which endorses the capability of the proposed IDCOA algorithm.



## 1. Introduction

Today, final products are complex and have several components, each of which needs to be processed on several machines before assembling them together as final products. Moreover, global competition forces enterprises to design and produce modular products which allow them to meet the increasing demand of customers in terms of volume and variety, and also reduce their production cost. Practically, parts of modular products are processed by several machines and are fed into an assembly machine to join the parts. Some examples of these production systems are fire engine assembly (Lee, Cheng, & Lin, 1993), computer manufacturing (Potts, Sevast'Janov, Strusevich, Van Wassenhove, & Zwaneveld, 1995), TV production (Mahdavi, Komaki, & Kayvanfar, 2011), and distributed database systems (Allahverdi & Al-Anzi, 2006a, 2006b). In these studies, the systems are modeled as the two-stage assembly flow shop scheduling problem, which is a relatively good approximation. However, it is still far from the reality because, for example, collecting the components and transferring them to the assembly machine is ignored (Koulamas & Kyparisis, 2001).

In this study, we address the three-stage assembly flow shop where the first stage has $m$ identical parallel machines followed by transportation and assembly stages each of which has a single machine. The second stage is the transportation stage in which an Automated Guided Vehicle (AGV) collects the processed parts from the first stage and transfers them to the third stage, which is the assembly stage. Each product has $m$ components where the first component needs to be processed by the first machine, the second one by the second machine, and so on. The objective is to find the products' sequence to minimize the completion time of the last processed product at the assembly stage, i.e., makespan. Using three-field problem classification $\alpha|\beta|\gamma$ of Graham, Lawler, Lenstra, and Rinnooy (1979), the addressed problem in this study can be represented as AF3(m,1,1)||$C_{max}$. This problem has already been studied by Koulamas and Kyparisis (2001) and it is an extension of the two-stage assembly flow shop. Koulamas and Kyparisis (2001) showed that this problem is NP-hard. Note that in the two-stage assembly flow shop scheduling problem, AF2(m,1)||$C_{max}$, the first stage has $m$ identical machines, and the second stage is the assembly stage. In the addressed problem in this paper, AF3 (m,1,1)||$C_{max}$, we have considered a *transportation stage* between the first stage and the assembly stage.

The two-stage assembly flow shop scheduling problem has been addressed by many researchers with different optimization

* Corresponding author.
*E-mail address:* gxk152@case.edu (G.M. Komaki).

http://dx.doi.org/10.1016/j.cie.2017.01.006
0360-8352/

criteria so far. Lee et al. (1993) investigated this problem where the first stage has two machines and proposed a Branch and Bound (B&B) algorithm. Hariri and Potts (1997) proposed a new B&B method for this problem and showed the superiority of their B&B algorithm over Lee et al. (1993)'s. Also, a B&B approach is proposed for this problem by Haouari and Daouas (1999). Later, Sun, Morizawa, and Nagasawa (2003) proposed B&B as well as several heuristic algorithms, termed $SMN_1$-$SMN_{14}$ and concluded that $SMN_{13}$ and $SMN_{14}$ outperform the other employed algorithms. Potts, Sevast'Janov, Strusevich, Van Wassenhove, and Zwaneveld (1995) studied a general case of the two-stage assembly flow shop, where the first stage has $m$ identical parallel machines and they proved that the optimal solution is based on the permutation of products. Also, they proposed a heuristic algorithm based on Johnson's algorithm as well as heuristics with absolute performance guarantees. For the same problem, Tozkapan, Kırca, and Chung (2003) presented a B&B method to minimize the total completion time of the jobs. Sung and Kim (2008) investigated minimizing the total completion time of jobs in the two-stage assembly flow shop where both stages have two machines. They proposed a B&B algorithm and a heuristic algorithm to solve the problem.

The general case of the two-stage assembly flow shop, where the first stage has $m$ (identical) parallel machines with different optimization criterion has been widely investigated. Previous studies have addressed makespan (Allahverdi & Al-Anzi, 2006b; Marichelvam, 2012), total completion time (Allahverdi & Al-Anzi, 2009), and tardiness (Allahverdi & Aydilek, 2015). Allahverdi and Al-Anzi (2006b) studied the general case of the two-stage assembly flow shop with setup times and suggested Particle Swarm Optimization (PSO) and Tabu Search (TS) technique showing that the PSO outperforms TS. Later, Allahverdi and Al-Anzi (2008) investigated the bi-objective two-stage assembly flow shop and proposed Simulated Annealing (SA), Ant Colony Optimization (ACO), and Self-adaptive Differential Evolution (SDE). Torabzadeh and Zandieh (2010) also developed Cloud theory based Simulated Annealing (CSA) for the same problem and showed that CSA yields better consequences than SA as proposed by Allahverdi and Al-Anzi (2008). Recently, Al-Anzi and Allahverdi (2013) proposed Artificial Immune System (AIS) for the two-stage assembly flow shop.

Koulamas and Kyparisis (2001) extended the two-stage assembly flows shop to the three-stage assembly flow shop and proposed two heuristic algorithms based on Johnson's algorithm. Also, they showed that the performance of any arbitrary algorithm that produces a "busy schedule", in which at a time at least one machine is processing a job, is not worse than 3. Hatami, Ebrahimnejad, Tavakkoli-Moghaddam, and Maboudian (2010) studied the bi-objective approach to the same problem with setup time and proposed SA and TS concluding that SA outperforms TS. Maleki-Darounkolaei, Modiri, Tavakkoli-Moghaddam, and Seyyedi (2012) studied the three-stage assembly flow shop with limited buffer and setup times and proposed SA for solving such a problem. Later, Maleki-Daronkolaei and Seyedi (2013) addressed minimizing the weighted sum of makespan and mean completion time of jobs where there is sequence dependent setup times between jobs and blocking times between stages. For this problem, they proposed Variable Neighborhood Search (VNS) algorithm and SA and the computational result showed that VNS outperforms SA while SA has lower computational time.

Most of the scheduling problems are NP-hard and exact methods such as B&B and Dynamic Programming (DP) are not able to find the optimal solution in a reasonable computational time. Therefore, heuristic and meta-heuristic algorithms have been applied to find good and near optimal solutions in a reasonable computational time. Recently, basic local search such as TS and SA (Shahvari & Logendran, 2015; Shahvari, Salmasi, Logendran, & Abbasi, 2012) or population-based meta-heuristic algorithms such as GA and PSO (Shahdi-Pashaki, Teymourian, & Tavakkoli-Moghaddam, 2016) along with the robust version of basic meta-heuristic algorithms such as stage-based TS (Shahvari & Logendran, 2016, 2017) have been successfully applied. Besides, a combination of local search and population-based structures as integrated meta-heuristic algorithms such as Tabu Search/Path-Relinking (Shahvari & Logendran, 2016) along with some other bio-inspired meta-heuristic algorithms such as the Intelligent Water Drops Algorithm (Shah-Hosseini, 2009), Bat Algorithm (Yang, 2010), Cuckoo Search (CS) with levy flight (Yang & Deb, 2010), and Cuckoo Optimization Algorithm (COA) (Rajabioun, 2011), have been developed for scheduling problems. According to Rajabioun (2011), COA has a very fast convergence rate and thus one could expect that it requires less computational time in comparison to other meta-heuristic algorithms.

The COA is inspired from the lifestyle of cuckoos which lay their eggs in the nests of other bird spices. Some of the cuckoo eggs will be identified by the host and will be abandoned or killed, and the rest will be fed by the host. After hatching from the eggs, the cuckoo chicks grow up very fast and surprisingly the mature birds follow their ancestors' pattern, i.e., they lay eggs in other birds' nests. Also, they have an immigration process by which they select nests which have access to sources of food. This immigration toward the best regions, which can be considered as a global search, leads to the exploration of new habitats and the birds then randomly lay their eggs in a limited area around the found habitat. This mechanism of immigration (global search), as well as egg laying (local search) in a new habitat, controls the balance between the diversification and intensification of the search ability of the algorithm to find the global optimal solution of problems. This algorithm has been successfully applied to many optimization problems and it is shown that it has a fast convergence rate (Rajabioun, 2011). For instance, the cuckoo search has been applied to satellite image segmentation (Bhandari, Singh, Kumar, & Singh, 2014), satellite signals recognition (Azarbad, Ebrahimzadeh, & Addeh, 2015), timetabling (Teoh, Wibowo, & Ngadiman, 2014), predicting permeability (Kaydani & Mohebbi, 2013), and clustering (Ameryan, Akbarzadeh Totonchi, & Seyyed Mahdavi, 2014).

Bhandari et al. (2014) combined the ideas behind the cuckoo search (Yang & Deb, 2010) and COA (Rajabioun, 2011) to tackle the satellite image segmentation. Azarbad et al. (2015) studied digital communication signals and applied COA for this problem. Their computational results showed that this algorithm has high accuracy. Teoh et al. (2014) applied the COA to university course timetabling and compared it with GA. They showed that the COA outperforms GA in terms of both quality and convergence speed. Moreover, applying COA to permeability prediction by Kaydani and Mohebbi (2013) showed that the COA has a faster convergence rate than PSO. Ameryan et al. (2014) investigated clustering dataset to optimize intra-cluster distance summation and proposed COA for solving the considered problem. Based on the results they obtained from experiments on standard datasets, they concluded that COA outperforms PSO, Imperialist Competitive Algorithm (ICA), and K-means clustering.

In addition to the COA (Rajabioun, 2011), the CS algorithm (Yang & Deb, 2010), an algorithm with similar inspiration in nature, has recently attracted much attentions in solving optimization problems (Hanoun, Nahavandi, Creighton, & Kull, 2012; Li & Yin, 2013; Liu & Ye, 2013; Marichelvam, 2012; Marichelvam, Prabaharan, & Yang, 2014; Nie, Liu, Xie, Liu, & Yang, 2014; Ouaarab, Ahiod, & Yang, 2014). The CS algorithm is inspired by the obligate brood parasitic behavior of some cuckoo species, combined with Lévy flights which describes the foraging patterns adopted by many animals and insects. Despite this similarity, the COA and CS have some significant unmatched properties in their

procedures like lévy flight, ELR, egg-laying mechanism, immigration, etc. which makes them obviously different. In a recently published study, Teoh et al. (2014), the authors combined the COA and CS main elements so as to tackle the university course timetabling problem.

In this paper, since the studied problem belongs to the NP-hard class of optimization problems with discrete solution space, we present an improved discrete version of COA, namely IDCOA, to find the products' sequence. The continuous nature of the COA makes it necessary to apply some adjustments to deal with a discrete environment like the considered scheduling problem. Accordingly, some enhancements are done on the main procedures of the basic COA such as egg laying, clustering of new young cuckoos and immigration of cuckoos, based on discrete scheme representation.

Therefore, the main contributions of this work can be enumerated as follows:

- Proposing a new meta-heuristic algorithm for the three-stage assembly flow shop scheduling problem that outperforms the current best approximate methods.
- A successful development of COA to a difficult combinatorial optimization problem, i.e. the three-stage assembly scheduling. This causes an enhancement in our understanding of the COA's behavior as a search method, especially in a discrete fashion.
- Presenting a discrete version of COA, IDCOA, incorporating new definitions, features, and procedures which can be used for the optimization of similar integer optimization problems. These adaptations including egg-laying, clustering of new young cuckoos, immigration of cuckoos, etc., are based on the discrete scheme representation.
- Some specific novel features incorporated in most steps of IDCOA to make it more enhanced; e.g., dynamic egg laying radius, cuckoos grouping, group evaluation, elimination, and local search mechanisms.
- Proposing a lower bound (LB) as well as some simple dispatching rules to tackle the considering problem where based on the experimental results the proposed LB and dispatching rules are fairly good.

The remainder of the paper is organized as follows. In Section 2, a formal description of the studied problem is presented. Section 3 is devoted to the existing heuristic algorithms details. Also, a lower bound of the addressed problem is developed in Section 3. In Section 4, the proposed IDCOA is presented and the computational results are discussed in Section 5. Finally, conclusions and future works are presented in Section 6.

## 2. Problem description

In the addressed problem, AF3(m,1,1)||$C_{max}$, there are three stages where the first stage has $m$ parallel identical machines followed by transportation and assembly stages each of which has a single machine. Each product has $m$ components and $m+2$ operations where the first $m$ operations should be processed at the first stage, the $m+1$th operation at the second stage and finally the last operation should be processed at the assembly stage. Also, at the first stage, the first operation of the product should be processed on the first machine and the second operation should be processed on the second machine and so on. After completing the first $m$ operations of a product at the first stage, the transportation machine collects all components of the product and moves to the assembly stage. Thereafter, the components are assembled together at the assembly stage where this process is the last operation of the product. The goal is to find a sequence of products such that the completion time of the last processed product at the assembly stage, makespan, is minimized. As already pointed out, the considered problem in this research is NP-hard. The other assumptions are:

- All components of products are available at time zero and processing time of them are known in advance.
- Preemption is not allowed, i.e., once processing a product (or component of a product) has started, the machine has to complete the process without any interruption.
- All machines are available throughout the scheduling horizon with no breakdown.
- The processing of available components/products starts as soon as a machine becomes idle.
- All machines can process one operation/product at a time and a part/product can be processed by a machine at a time.

Let $J_i$, $i = 1,2,\ldots,n$ be the product index and $t_{ij}$ be the processing time of operation $j = 1,2,\ldots,m$ of product $i$ at the first stage. Also, suppose that $t_{iT}$ and $t_{iA}$ are the transportation and assembly times of product $i$, respectively.

It is shown that for this type of problem, the permutation of products is dominant, i.e., the optimal solution is based on the permutation of products; therefore, we restrict ourselves to the permutation of products. Koulamas and Kyparisis (2001) showed that for a given sequence of products $\pi = (i_1, i_2, \ldots, i_n)$ the makespan is as follows:

$$C_{max} = \max_{1\leqslant k_1\leqslant k_2\leqslant n}\left\{\max_{1\leqslant j\leqslant m}\left\{\sum_{k=1}^{k_1} t_{i_k j}\right\} + \sum_{k=k_1}^{k_2} t_{i_k T} + \sum_{k=k_2}^{n} t_{i_k A}\right\} \quad (1)$$

where $i_k$ represents the product in position $k$. Note that the above equation is based on the grid graph representation (for example, see Nowicki and Smutnicki (1996)) of the addressed problem and the critical path connecting the first node to the last node is based on the products in position 1 to $k_1$ on the first stage and the products in position $k_1$ to $k_2$ on the second stage and products in positions $k_2$ to $n$ on the third stage. The goal is finding sequence of products that minimizes the makespan, min($C_{max}$).

## 3. Existing algorithms and proposed lower bound

In this section, the proposed heuristic algorithms by Koulamas and Kyparisis (2001) are described, and then a lower bound, as well as some dispatching rules for the considered problem are proposed.

### 3.1. Existing heuristic algorithms

As already mentioned, Koulamas and Kyparisis (2001) proposed two heuristic algorithms, $H_0$ and $H_3$, for this problem where their worst performance ratios are 2 and $(2-1/m+\varepsilon)$, respectively. These algorithms are presented in the following.

*Heuristic $H_0$*:

Step 1. Convert the AF3(m,1,1)||$C_{max}$ problem to the artificial F2||$C_{max}$ problem using $p_{iT}$ as processing time of the artificial two-stage problem and $p_{iA}$ as processing time of products at the second stage. Then apply Johnson's algorithm to the F2||$C_{max}$ problem to find the products sequence $\pi_0 = (j_1, j_2, \ldots, j_n)$.
Step 2. Use the permutation $\pi_0$ to find the makespan of the original problem, AF3(m,1,1)||$C_{max}$.

*Heuristic $H_3$*:

This heuristic algorithm converts the problem into F3||$C_{max}$ and then any heuristic algorithm can be applied to find the sequence of

products such as Rock and Schmidt (1983) algorithm. They called this algorithm $H_{3R}$. The steps of this algorithm are as follows:

Step 1. Convert the AF3(m,1,1)||$C_{max}$ problem into F3||$C_{max}$ where the processing times of the products at the first stage is $(p_{i1} + p_{i2} + \ldots + p_{im})/m$ and the processing times of the second and the third stages are $p_{iT}$ and $p_{iA}$, respsectively.
Step 2. Apply Rock and Schmidt (1983) algorithm to F3||$C_{max}$ and find the products sequence $\pi_3 = (j_1, j_2, \ldots, j_n)$.
Step 3. Use the permutation $\pi_3$ to find the makespan of the original problem, AF3(m,1,1)||$C_{max}$.

### 3.2. The proposed lower bound

A lower bound for the considered problem is proposed in this subsection. This LB is based on bottleneck stage. The bottleneck stage has the highest average workload. One can consider three special cases based on the workload of stages and for each one, simplification of Eq. (1) provides makespan of the schedule as discussed in the following.

a. ***The first stage is the bottleneck:***

$$LB_1 = \max_{j=1,\ldots,m}\left\{\sum_{i=1}^{n} t_{ij}\right\} + \min_{i=1,\ldots,n}\{t_{iT} + t_{iA}\} \quad (2)$$

When the first stage is the bottleneck, the LB is composed of two terms; the first term computes the completion time of the last processed product at the first stage which equals to the maximum of the total processing time of products at the first stage, the second term is the earliest completion time of the last product at second and third stages. Note that $LB_1$ represents a special case of Eq. (1) where $k_1 = k_2 = n$.

b. ***The second stage is the bottleneck:***

$$LB_2 = \min_{i=1,\ldots,n}\{\max_{j=1,\ldots,m}\{t_{ij}\}\} + \sum_{i=1}^{n} t_{iT} + \min_{i=1,\ldots,n}\{t_{iA}\} \quad (3)$$

When the second stage is the bottleneck, the machine in the second stage will be idle until the arrival of the first product with the shortest processing time at the first stage which is calculated by the first term of $LB_2$. Then, the second stage will be busy until all products are processed; this time period is computed using the second term. Finally, after completion of the last product at the second stage, the earliest completion time of the last product at the assembly stage is presented by the third term in $LB_2$. Note that $LB_2$ represents a special case of Eq. (1) where $k_1 = 1$ and $k_2 = n$.

c. ***The third stage is the bottleneck:***

$$LB_3 = \min_{i=1,\ldots,n}\{\max_{j=1,\ldots,m}\{t_{ij}\} + t_{iT}\} + \sum_{i=1}^{n} t_{iA} \quad (4)$$

When the assembly stage is the bottleneck, the machine in this stage will be ideal until the arrival of the first product. The arrival time of the earliest product is computed by the first term of the $LB_3$. Subsequently, after processing of the first product is begun, it will be busy until all products are processed. The latter term is shown via the second term of $LB_3$. Note that $LB_3$ represents a special case of Eq. (1) where $k_1 = k_2 = 1$.

According to the above-mentioned three special cases, the LB is computed as the follows.

$$LB = \text{Max}\{LB_1, LB_2, LB_3\} \quad (5)$$

### 3.3. The proposed dispatching rules

In order to find the sequence of products, the following dispatching rules (DRs) are presented. Theses DRs are inspired from the lower bound proposed in the previous subsection. Each of them defines an index for each product. To find the products sequence, these indexes should be sorted in a non-decreasing order.

$$DR_1 = \max_{j=1,\ldots,m}\{t_{ij}\} \quad (6)$$

$$DR_2 = t_{iT} \quad (7)$$

$$DR_3 = t_{iA} \quad (8)$$

$$DR_4 = \max_{j=1,\ldots,m}\{t_{ij}\} + t_{iT} + t_{iA} \quad (9)$$

$DR_1$, $DR_2$, and $DR_3$ are based on the products processing time at the first, second and third stages, respectively while $DR_4$ is the summation of all processing times of each product. The performance of the proposed LB and dispatching rules are investigated using the following examples.

**Example 1.** Consider the presented example by Koulamas and Kyparisis (2001), page 693, where there are $m$ machines at the first stage and three products with the following processing times:

Product 1: $p_{1j} = 1$ for $j = 1,\ldots,m$; $p_{1T} = 1$, $p_{1A} = k$
Product 2: $p_{2j} = 1$ for $j = 1,\ldots,m$; $p_{2T} = k$, $p_{2A} = 1$
Product 3: $p_{3j} = k$ for $j = 1,\ldots,m$; $p_{3T} = 1$, $p_{3A} = 1$

where $k > 1$. The optimal schedule of this problem is 1–2–3 and its makespan is $(k + 4)$ (Koulamas & Kyparisis, 2001).

Now, we investigate the performance of the proposed LB and dispatching rules:

$$LB_1 = \text{Max}_{j=1,\ldots,m}\{1 + 1 + k\} + \text{Min}\{1 + k, k + 1, 1 + 1\} = k + 2 + 2 = k + 4$$

$$LB_2 = \text{Min}\{1, 1, k\} + (1 + k + 1) + \text{Min}\{k, 1, 1\} = 1 + (k + 2) + 1 = k + 4$$

$$LB_3 = \text{Min}\{1 + 1, 1 + k, k + 1\} + (k + 1 + 1) = 2 + (k + 2) = k + 4$$

and LB = Max{k + 4, k + 4, k + 4} = k + 4 which is the same as the optimal solution.

Now, we investigate the dispatching rules. Note that $\max_{j=1,\ldots,m}\{t_{1j}\} = 1$, $\max_{j=1,\ldots,m}\{t_{2j}\} = 1$, and $\max_{j=1,\ldots,m}\{t_{3j}\} = k$, therefore, the sequence of products based on $DR_1$ could be either 1–2–3 or 2–1–3, where their makespan are $k + 4$ and $2k + 3$, respectively.

Since $p_{1T} = p_{3T} = 1$, and $p_{2T} = k$, the possible sequence of products based on $DR_2$ is 1–3–2 or 3–1–2, and the makespan of both sequences is $2k + 3$. Also, $p_{2A} = p_{3A} = 1$, and $p_{1A} = k$, the sequence of products based on $DR_3$ would be either 2–3–1 or 3–2–1, where the makespan of the former sequence is $2k + 3$ while the makespan of the latter one is $3k + 2$. Finally, $DR_4$ assigns $k + 2$ to all products, all of which have the same priority.

**Example 2.** Consider the following example from Koulamas and Kyparisis (2001), page 694, where there are $m$ machines at the first stage and two products with the following processing times:

Product 1: $p_{1j} = 1$ for $j = 1,\ldots,m$; $p_{1T} = k$, $p_{1A} = 1/k$
Product 2: $p_{2j} = k$ for $j = 1,\ldots,m$; $p_{2T} = 1$, $p_{2A} = 2/k$

where $k > 2$ and the optimal schedule is 1–2 with $C_{max} = k + 2 + 2/k$ (Koulamas & Kyparisis, 2001).

In this case,

$$LB_1 = \max_{j=1,\ldots,m}\{1+k\} + \min\{k+1/k, 1+2/k\} = 1+k+1+2/k = k+2+2/k$$

$$LB_2 = \min\{1,k\} + k + 1 + \min\{1/k, 2/k\} = 1+k+1+1/k = k+2+1/k$$

$$LB_3 = \min\{1+k, k+1\} + (1/k + 2/k) = k+1+3/k$$

and consequently LB = Max{$LB_1, LB_2, LB_3$} = Max{k + 2 + 2/k, k + 2 + 1/k, k + 1 + 3/k} = k + 2 + 2/k.

Also, $DR_1$ assigns 1 to product 1 and $k$ to product 2. Therefore, the products sequence is 1–2, which is the same as the optimal solution. The sequence of products based on $DR_2$ is 2–1 with a makespan of $2k + 1 + 1/k$ (Koulamas & Kyparisis, 2001). Also, the product sequence based on $DR_3$ and $DR_4$ is 1–2, which is the same as the optimal solution.

According to these examples, one could expect that $LB_1$ has better performance with respect to the other LBs. It is also expected that $DR_1$ outperforms the other proposed dispatching rules.

## 4. Improved Discrete Cuckoo Optimization Algorithm (IDCOA)

The cuckoo optimization algorithm (COA) as a bio-inspired population-based meta-heuristic has recently been proposed for continuous problems. To apply it for a scheduling problem with a discrete solution space, it needs to be adjusted into the discrete environment. To do so, a suitable solution representation scheme, including egg laying and immigration of the cuckoos, is proposed in an improved discrete COA version, called IDCOA. Also, we incorporate some other features to improve the quality of the proposed IDCOA in dealing with the considering scheduling problem. In this section, the basics of the COA are first explained and the main components of the IDCOA are then presented.

### 4.1. Cuckoo optimization algorithm background

Cuckoo Optimization Algorithm (COA), proposed by Rajabioun (2011), is inspired from lifestyle of cuckoo, and its efficiency via benchmark problems has already been shown. Cuckoos are brood parasites which never build their own nest, instead they lay their eggs in nests of other species. The cuckoos are experts in imitating the color and size of the eggs of the host birds. When they want to lay an egg, they remove one egg of the host bird in the nest and replace it with their own. If the host bird does not identify the egg as unfamiliar, it will then take care of it. The cuckoo's egg hatches sooner than the host's egg and the cuckoo chick may eat or throw out the eggs of the host nest; also the cuckoo chick grows faster than the host's chicks. An important factor in laying eggs in the host nests is the availability of food sources and accordingly, the cuckoos migrate to find habitats in areas with sufficient foods. Then, they lay their eggs in the nests of host birds nearby. The steps of the COA is presented as the follows.

#### 4.1.1. Generate initial cuckoo habitat

In COA, each habitat represents a solution of the at hand problem and it is the same as a chromosome in the Genetic Algorithm (GA). Also, the quality of each habitat is presented by a profit where for the minimization problems can be presented as follows.

$$\text{Profit} = -\text{Cost}(\text{habitat}) \tag{10}$$

The COA is a population-based algorithm that starts with $N_{pop}$ habitats. It is assumed that each cuckoo can lay from 5 to 20 eggs. Another interesting behavior of the cuckoos is that they lay eggs in habitats which are in range of their current habitat. This range or maximum distance is called "*Egg laying Radius* (*ELR*)" which depends on the number of cuckoo's eggs ($N_{egg}$,), total number of eggs ($TN_{egg}$), and the upper and lower bounds of the variables of the problem ($U_{var}$ and $L_{var}$) as shown below:

$$ELR = \alpha \times \frac{N_{egg}}{TN_{egg}}(U_{var} - L_{var}) \tag{11}$$

where $\alpha$ is an integer.

#### 4.1.2. Laying eggs

Each cuckoo starts laying eggs in the nest of host birds which are in ELR range of the current habitat. After completion of the egg laying process, some cuckoo eggs which are less similar to the host's eggs are identified and destroyed by the host bird, either throwing out the strange egg or abandoning the nest. Usually, $p$% of the eggs of lower quality will be identified and killed by the host birds. The rest of the eggs will hatch and will be fed by the host birds. It is worth mentioning that at each nest only one cuckoo chick can grow up, because when one of cuckoo egg hatches, it will throw out all other eggs in the nest including both the host's eggs and other eggs of cuckoos. Also, cuckoo chicks have a fascinating growth rate such that they push the host's chicks and eat all the food. As a result, the host's chicks will eventually die due to starvation.

#### 4.1.3. Immigration of cuckoos

As the young chicks grow up and become mature, they live in their society for a while but in egg laying season, they migrate to new and better habitats in which they can lay their eggs. Also, the new habitats are intended to have greater sources of food. Before immigration, cuckoos form groups in different areas, and the group with the best profit is the goal point to immigrate. After cuckoos become mature, it is difficult to recognize which cuckoo belongs to which group; therefore, cuckoos are grouped using the $k$-mean clustering method, and usually $k$ in range 3–5 is sufficient. Then, the mean profit of each group is calculated and the group with the highest mean profit is the destination/goal point to migrate.

In the migration process, cuckoos do not fly entire path to the destination, but they fly some part of the way and have some deviation from the direct path from the current habitat to the destination. Let say, a cuckoo flies $\lambda$% of the direct path from the current habitat to the destination and it has $\varphi$ radian deviation from the direct path. So, using $\lambda$%, $\lambda \sim U(0,1)$ where U is uniform distribution between 0 and 1, and $\varphi$, $\varphi \sim U(-\omega,\omega)$, where $\omega = \pi/6$ seems reasonable deviation, we can model the immigration process of cuckoos.

After the immigration process is completed and new habitats are identified, cuckoos can start laying eggs randomly based on the ELR rule

#### 4.1.4. Eliminating cuckoos with low quality

The number of cuckoos is limited to $NC_{max}$, because when the number of cuckoos is high, they will be killed by predators and also finding food and nests to lay their eggs in will be difficult for them. In order to keep a limited number of cuckoos, keep $NC_{max}$ cuckoos with the high quality and kill the rest of cuckoos.

#### 4.1.5. Convergence

As explained in the earlier steps, after some iteration, all cuckoos will opt to move toward the destination which presents maximum similarly of eggs and food sources. Consequently, they will form a group with the maximum profit. The COA algorithm terminates when more than 95% of cuckoos converge to the same habitat. The pseudo code of COA is presented in Fig. 1.

*Cuckoo optimization algorithm's main steps*

Step I. Initialize cuckoos with eggs
Step II. Lay eggs in the hosts' nest
Step III. Detect and kill some eggs of bad quality
Step IV. Check the population size is less than max value
If not, kill some cuckoos in worst area
Step V. Evaluate profit value (fitness) of eggs
Step VI. Check the stopping criteria
If satisfied, stop; otherwise
i) Let eggs hatch and chicks grow
ii) Find the nests with the best profit values
iii) Cluster cuckoos and determine the goal habitat
iv) Immigrate all cuckoos toward goal habitat
v) Determine *ELR* for each cuckoo
vi) Go to Step I

**Fig. 1.** Pseudo-code of the basic COA.

### 4.2. The proposed IDCOA for the three-stage assembly flow shop

Since the standard COA cannot be directly applied to an optimization problem in discrete solution space, of course it is not suitable for a permutation-based problem. Thus, it is necessary to make some specific adjustments to the standard continuous COA to deal with the optimization of the three-stage assembly flow shop scheduling problem. To this end, we propose a new discrete version of the COA with some enhancements, namely IDCOA, which introduces new adjustments, e.g., egg laying, clustering of new young cuckoos and immigration of cuckoos based on discrete scheme representation. Also, specific novel features are incorporated in most steps of the IDCOA to improve it further. The IDCOA performs global and local searches efficiently using proposed migration and egg laying mechanisms which are enhanced by both directed and random moves, respectively. In other words, IDCOA manages the balance between diversification and intensification of the search.

#### 4.2.1. Solution representation

The effectiveness of operators in meta-heuristic algorithms depends greatly on the solution representation scheme used. Each individual (cuckoo or solution), in the IDCOA for the studying discrete problem, uses a product permutation scheme which is simple and effective in most of the meta-heuristic optimization algorithms.

Since an egg can present a new cuckoo and one egg -if there is- in each nest of a habitat survives, we can represent a solution by a nest, habitat, cuckoo, chick or an egg, i.e., all of them are equivalent.

#### 4.2.2. Generating initial cuckoo habitats

To start the proposed optimization algorithm, an initial population of candidate habitats in the form of a matrix of size $N_{pop} \times n$ is generated, where $N_{pop}$ and $n$ are the size of the initial population and the number of products, respectively. As mentioned by several studies including Naderi, Zandieh, and Roshanaei (2009), the quality of final solution of metaheuristic algorithms highly depends on the quality of initial solutions. In this study, in order to generate high quality initial solutions, we suggest using the proposed dispatching rules in Section 3.3 and the rest of the sequences are then generated randomly.

#### 4.2.3. Habitat evaluation

The goal of the profit (fitness) evaluation is to calculate the goodness of habitat/candidate individuals regarding the objective function, i.e., measuring the suitability rate of a solution for raising cuckoo eggs (Shahdi-Pashaki, Teymourian, Kayvanfar, Komaki, & Sajadi, 2015). Since the cuckoo optimization algorithm tries to maximize a profit function, and here the objective, makespan, is a cost criterion which should be minimized; the profit function has the reverse relation to its corresponding objective function. Hence, the profit value for each habitat/solution can be calculated as follows:

$$\text{Profit (habitat)} = -\text{makespan (habitat)} \tag{12}$$

where the makespan of each solution is computed as described in Section 3. Note that the higher the profit value, the better the performance of the habitat. Therefore, candidate habitats with higher fitness values have more chances to remain.

#### 4.2.4. Egg laying of cuckoos

As stated in Rajabioun (2011), after cuckoos identify new (or initial) habitats, each one starts laying $N_{egg}$ eggs randomly in some other host birds' nests within an *Egg Laying Radius* (*ELR*) from her habitat. This mechanism brings the local search operation to the mind. After introducing the computations of required parameters of the egg laying strategy, neighborhood structures for applying a local search (egg laying) on the problem's solutions (habitats) are presented.

Each cuckoo based on its profit value can lay a number of eggs ($N_{egg}$) which is a random number as follows.

$$N_{egg} = w_i \left( \frac{f_i}{f_{best}} \right) \tag{13}$$

where $w_i \sim U(L_{egg}, U_{egg})$ and $L_{egg}$ and $U_{egg}$ are used as the upper and lower limits of egg dedication to each cuckoo at different iterations, respectively. Besides, $w_i(f_i/f_{best})$ handles the effectiveness of the habitat's goodness to generate $N_{egg}$ for each cuckoo $i$. This calculation approach provides more chances for the better habitats to be more explored.

Another habit of real cuckoos is that they lay eggs within a maximum distance from their habitat (*ELR*) which is here redefined by how it could be compatible with a discrete scheme of solution space and fitness of habitat. Also, it diminishes through the algorithm evolutions. Thereby, we introduce *Dynamic ELR* (*DELR*) which is proportional to the size of solution in the considered problem (number of products, $n$, in this problem), the total number of eggs, number of considering cuckoo's eggs and a new dynamic controlling parameter $\alpha_{itr}$ as follows:

$$DELR = \alpha_{itr} \cdot n \cdot \frac{N_{egg}}{TN_{egg}} \tag{14}$$

where $\alpha_{itr}$ is a parameter to control the value of *DELR*. The larger $\alpha_{itr}$ is, the better the global optimization capability will be. But if so, the computational effort of IDCOA will be raised. Thus, in order to master the problem, we introduce a dynamic $\alpha_{itr}$ value rather than a constant value. This means $\alpha_{itr}$ will be dynamically reduced over the time to get better convergence or intensification of IDCOA in last evolutions and more diversification in early ones. To do so, we define a linear changing mechanism like the cooling function in Simulated Annealing (SA), i.e., $\alpha_{itr}$ is obtained in each evolution (iteration of IDCOA) as shown in Eq. (15):

$$\alpha_{itr} = \alpha_{max} - N_{itr} \frac{(\alpha_{max} - \alpha_{min})}{TN_{itr}} \tag{15}$$

where $\alpha_{min}$ and $\alpha_{max}$ are defined as the minimum and maximum limits of $\alpha_{itr}$ during the IDCOA, respectively. Also, $N_{itr}$ and $TN_{itr}$ identify the iteration number whose $\alpha_{itr}$ is computed and the total number of iterations, respectively.

*4.2.4.1. Neighborhood structures.* To perform the egg laying strategy for each of the new reached habitats, we should use neighborhood structures of the solution space. As mentioned earlier, the solution

representation used in this paper is based on the permutation of products and accordingly, there are mainly two types of moves for such a representation: *Insert* and *Swap*. It is shown that swap moves are not better than insert moves (Grabowski & Pempera, 2005). Therefore, only insert moves are considered in our neighborhood structure. In this paper, we suggest using the *k*-insert moves, $k \leqslant DELR$ which *k* products are randomly selected and reinserted in random positions. In this way, using *k*-insert operator, the incumbent habitat evolves to obtain new solutions interpreted as new eggs laid by a current cuckoo within its *DELR*.

*4.2.5. Growing up the new chicks*

After all the cuckoos lay their eggs in host birds' nests, some eggs less similar to host birds' own eggs will be detected by the host birds. These eggs are thrown out of the nest or abandoned. Some others may be hunted and eaten as a good prey by other animals. Thus, *p*% of all eggs (usually 10%), with less profit values, will be killed; other ones grow up and become new young cuckoos. They have no chance to survive and grow. Rest of the eggs grow in host nests, hatch and are fed by host birds. However, due to the aforementioned details about what happens between undetected cuckoo eggs and host birds' eggs, only one egg in a nest will have the chance to survive and grow.

*4.2.6. Grouping, evaluating and immigration of cuckoos*

*4.2.6.1. Grouping (clustering of cuckoos).* According to the statement in Rajabioun (2011), each type of cuckoo is specifically suited to infiltrate its eggs into the nests of a certain type of host birds. Therefore, cuckoos are form various groups in different areas. In order to group (cluster) the new young cuckoos, Rajabioun (2011) suggested a *K*-means clustering approach with a *k* of 3–5. Since this method is a time consuming optimization algorithm to be embedded in COA, especially in a discrete fashion, we adopt an efficient alternative, similar to one presented for the clustering of parts and related machines in the Cell Formation Problem (CFP) based on the Jaccard similarity coefficient. This similarity coefficient is simple and easy to calculate, so it has been used widely by many clustering algorithms (see Yin & Yasuda, 2006).

In this paper, a new clustering method using a novel similarity metric for grouping new cuckoos is proposed based on the idea of the Jaccard approach. Once this coefficient has been computed for new cuckoo pairs, an algorithm called Cuckoos Clustering Algorithm (CCA) groups the cuckoos with the highest similarity. So, we define a new similarity coefficient between each pair of cuckoos *a* and *b*, namely the Cuckoo Similarity Coefficient, $CSC(a,b)$, as follows.

$$CSC(a,b) = \frac{s_b^a + 2z_b^a}{n} \tag{16}$$

where *n* is the number of products. Also, we define the $s_b^a$ as the number of similar pair parts (edge) between solutions *a* and *b*, see Eq. (17), and $z_b^a$ is the number of same elements in sequencing string between the two solutions or the number of $s_b^a$ in the same position of both solutions.

$$s_b^a = \sum_{i,j \in \{1,\ldots,n\}} l_{ij} \tag{17}$$

where

$$l_{ij} = \begin{cases} 1 & \text{if (i, j) is an edge of both solutions a and b} \\ 0 & \text{Otherwise} \end{cases}$$

After computation of *CSC* for all cuckoo pairs, the CCA containing following steps is applied as follows (Fig. 2):

Utilizing the proposed technique instead of the K-means clustering approach, not only reduces the computational efforts during the IDCOA but also is carried out in an easier and simpler way.

*4.2.6.2. Group evaluation.* Once the cuckoos' chicks grow and become mature, they tend to immigrate to new and better habitats for egg laying, where their eggs are more similar to those of host birds and also more food resources are available for their chicks. The society with the best attractiveness (profit value) is selected as the goal point (the best habitat in this group) for other cuckoos to immigrate. Each group's profit is evaluated according to Eq. (18) as follows.

$$\begin{aligned} \textit{Group profit} = \alpha & \\ & \times \text{profit value of the best solution in group} \\ & + \beta \times \text{mean profit of cuckoos in group} \end{aligned} \tag{18}$$

The maximum value of these profits identifies the goal group and consequently its habitat is the new destination for immigrant cuckoos.

*4.2.6.3. Immigration operator.* When the goal point habitat is identified, the mature cuckoos fly to immigrate toward it. The aim of immigration or the moving toward phase is to introduce attributes of the goal point solution into solutions obtained by moving away from the current solution. They do not, however, fly the entire path to the destination point. They only fly a part of the way based on the neighborhood structure defined for this movement. Identical parts of the two (current and goal) solutions should remain unchanged during the moving. Fig. 3 illustrates this moving and potential intermediate solutions 1–5, where a new solution with step length of 3 is presented as the result of immigration.

As a case in point, in Fig. 4 $x_{current}$ and $x_{goal}$ represent the current solution/habitat and goal point respectively, it is necessary to define a set of swaps *D* with 4 elements to show the potential moves between two solutions as calculated in Eq. (19).

$$D = x_{goal} - x_{current} + \{(2,3),(3,9),(9,6),(2,9)\} \tag{19}$$

Besides, $\lambda \in (0,1)$ the length of step is supposed to be 0.72, so new position $x_{new}$ could be determined by Eqs. (20) and (21) as follows:

$$x_{new} = x_{current} + \lambda \times D \tag{20}$$

$$x_{new} = x_{current} + \{(2,3),(3,9),(9,6)\} \tag{21}$$

Considering the three first moves (swaps) in *D* yield the new solution. An important fact which says there is not a unique path *D* between each pair of habitats, may result in various ways and consequently different intermediate solutions as figured out in Fig. 3.

***The cuckoos clustering algorithm***

Step I. *Make* a similarity matrix which stores all values of *CSC(a,b)*;

Step II. *Determine* number of *k* cuckoos (solutions) with the highest difference or minimum pair *CSC*, as the *k* clusters;

Step III. *Add* new cuckoo *i* to the cluster *A* ($A \in \{1,2,..,k\}$) so that maximize $\sum_{j \in A} CSC(i,j)$;

Step IV. *If* there is any unassigned cuckoo *i, go to* Step III; otherwise, *return* the resulting clusters

**Fig. 2.** The main steps of CCA.

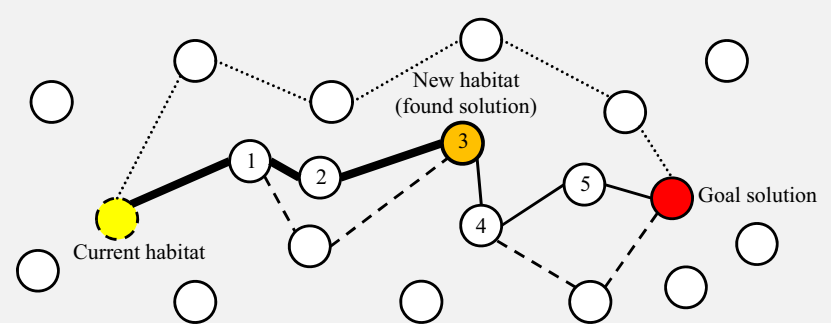



**Fig. 3.** Cuckoo immigration in a discrete fashion.

$x_{current}$= | 5 | 3 | 6 | 7 | 1 | 2 | 9 | 8 | 4 |

$x_{goal}$ = | 5 | 9 | 2 | 7 | 1 | 6 | 3 | 8 | 4 |

**Fig. 4.** An exemplary current and goal habitat for the considering problem.

Once all cuckoos have immigrated toward a goal point and new habitats are specified, new *DELR*s are calculated for cuckoos considering the number of eggs dedicated to them. Then, a new egg laying process restarts.

Fig. 5 shows the main concepts of IDCOA; where the extent of each habitat is related to its profit and determined by *DELR*. The immigration step and finding a new habitat through one of the possible paths is also illustrated. The discrete version of COA may use *DELR* for various neighborhood structures based on the nature of the problem, i.e., one can use different local search operators to search and map new solutions/nests for laying eggs regarding this *DELR* as a maximum step length.

*4.2.7. Eliminating cuckoos in worst habitats*

Because of natural limitations and threats like food availability and predators, there is always an equilibrium in live cuckoos similar to other birds' population. In COA, $NC_{max}$ is a parameter which controls and limits the maximum number of live cuckoos in the environment (solution space). Rajabioun (2011) proposed a model in which only those $NC_{max}$ number of cuckoos survive that have better profit values and others demise. But, we present a probability $P_e$ based on cuckoo's profit value to choose cuckoos which are not elite in their populations and must perish. Ignoring a set of $N_e$ of the best cuckoos marked as elites set $S_e$, other ones could survive if not selected for elimination by probability $P_e$.

$$P_e(x_i) = \frac{Profit(1/x_i)}{\sum_{j \notin S_b} Profit(1/x_j)} \tag{22}$$

This function not only prevents the premature convergence of the proposed algorithm in earlier iterations but also preserves randomly exploring of solution space to some extent; where it brings the chance to the few potential solutions –while they have worst profit values now- to be improved.

*4.2.7.1. Local search on elite solutions.* In each iteration of the algorithm, we apply $\eta$-insertion moves on the elite solutions and accept the obtained solution if it improves the initial sequence. In this algorithm, $\eta$ is randomly chosen from Allahverdi and Al-Anzi (2006b), Mahdavi et al. (2011), Koulamas and Kyparisis (2001), Hariri and Potts (1997).

Ultimately, before resuming these steps, the algorithm checks the stop criteria defined here as either the maximum number of

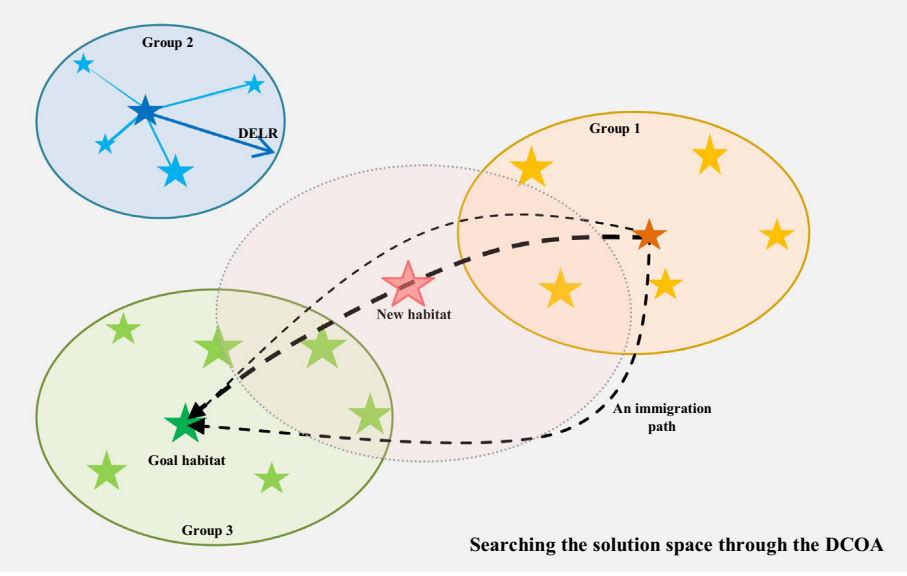



**Fig. 5.** Schematic illustration of the steps 4 & 6 of the IDCOA.

| *Improved Discrete Cuckoo Optimization Algorithm* | |
|---|---|
| Step I. | *Initialize* the parameters of the considering problem and IDCOA; |
| Step II. | *Generate* initial population of cuckoo habitats & *Evaluate* them #Sections 4.2.2 & 4.2.3# |
| Step III. | *Dedicate* some eggs to each cuckoo #Section 4.2.4# |
| Step IV. | *Define DELR* for each cuckoo #Section 4.2.4# |
| Step V. | Let cuckoos to *lay* eggs inside their corresponding *DELR* #Section 4.2.4# |
| Step VI. | *Kill* those eggs that are recognized by host birds (***p***% of all cuckoos with less profit value) #Section 4.2.5# |
| Step VII. | Let unrecognized eggs hatch and chicks *grow* #Section 4.2.5# |
| Step VIII. | *Evaluate* the habitat of each newly grown cuckoo #Section 4.2.3# |
| Step IX. | *Cluster* cuckoos by CCA; *Evaluate* and *find* the goal habitat #Section 4.2.6# |
| Step X. | Let new cuckoo groups *immigrate* toward the goal habitat #Section 4.2.6# |
| Step XI. | *Limit* cuckoos' maximum number ($NC_{max}$) in environment; so, kill those live in worst habitats with a probability distribution of $P_e$. Also *save* the elite set $S_b$ and perform *local search* around its solutions #Section 4.2.7# |
| Step XII. | If termination criterion ($NI_{itr}$ or $TN_{itr}$) is satisfied, Stop; otherwise Go to Step III |

**Fig. 6.** The main steps of the IDCOA.

evolutions (iterations) without any improvement ($NI_{itr}$) or the maximum number of iterations $TN_{itr}$. The pseudo code of the whole procedure of IDCOA, described above in detail, is presented in Fig. 6.

## 5. Experimental results

Values of the parameters for the proposed meta-heuristics have been calibrated based on Taguchi method. This method is designed to determine the optimal level of controllable factors or parameters. The parameters of the developed algorithms, COA and IDCOA, are listed in Table 1. Note that $N_e$ and $NI_{itr}$ are applicable only for IDCOA. Also, $L_{egg}$ and $U_{egg}$, p, and $TN_{itr}$ are set as suggested by Rajabioun (2011). For other parameters, applicable for both COA and IDCOA, we have defined possible values as presents in Table 1. Recall that in COA algorithm, ($\alpha_{min}$,$\alpha_{max}$) are the same and fixed while their values in IDCOA is dynamic.

For both methods, basic COA and IDCOA, the initial number of cuckoos or initial population size is set to 5, while the maximum number of allowed cuckoos ($NC_{max}$) which could be alive after each iteration, is regarded as 40. Also, as previously reported in Rajabioun (2011), each cuckoo in nature may lay from 5 to 20 eggs. These values are used as the upper ($U_{egg}$) and lower ($L_{egg}$) bounds of egg dedication to each cuckoo at different iterations. Besides, after egg laying procedure, *p*% of all eggs (usually 10%), with less profit values, will demise.

Specifically, the parameter $N_e$ in IDCOA identifies the number of elite solutions in current population, which should be saved for the next generation, is defined as 5% of the population. The controlling parameters of *DELR*, $\alpha_{min}$ and $\alpha_{max}$, are set to 0.5 and 2, respectively. Finally, $NI_{itr}$ and $TN_{itr}$, the allowed number of iterations without any improvement and the maximum number of iterations of the IDCOA, are tuned to 5 and 400, to terminate the algorithm.

As identified before, the common parameters of COA is the same as for the IDCOA, the other ones are as follows. The controlling parameter of egg laying, $\alpha$, can be defined in [0.5,2] based on the size of problem, e.g., it would be equal to 1.5 for large size cases; and the COA terminates when more than 95% of cuckoos converge to the same habitat or maximum number of iterations, 400, is met.

The performance of the algorithms is tested based on randomly generated instances. The processing times of products are presented in Table 2 where sets 1, 2 and 3 are from Koulamas and Kyparisis (2001), and in set 4 the processing times of all stages fit into the same range. Also, the number of machines at the first stage is defined as {2, 4, 6 and 8}, while the number of products is {20, 40, 60 and 80}. Altogether, we run 64 instances, each of which includes 100 generated test problems.

The proposed algorithms are compared to SA (Maleki-Darounkolaei et al., 2012), VNS (Maleki-Daronkolaei & Seyedi, 2013) and heuristic algorithms proposed by Koulamas and Kyparisis (2001). All algorithms are programmed in MATLAB 7.5.0 and run on a PC with a 2.66 GHz Intel Core 2 Duo processor and 3 GB RAM memory. Due to the random nature of meta-heuristic algorithms, each case has been run 25 times. The deviation of the proposed LB from the best solution of the algorithms, DVL, is measured as follows:

$$DVL = \frac{Alg^*_{sol} - LB_i}{Alg^*_{sol}} \times 100 \quad (23)$$

where $LB_i$ is the proposed LB, i.e., $LB_1$, $LB_2$, $LB_3$ and LB, and $Alg_{sol}^*$ is the best obtained objective value by the proposed algorithms. The obtained results are given in Table 3.

**Table 1**
Parameters of the algorithms and their best values.

| Parameter (or factor) | Range | COA | IDCOA |
|---|---|---|---|
| Initial number of cuckoo | (2,4,5,6) | 5 | 5 |
| The maximum number of allowed cuckoos ($NC_{max}$) | (20,40,60) | 40 | 40 |
| Lower limit of egg lying ($L_{egg}$) | – | 5 | 5 |
| Upper limit of egg lying ($U_{egg}$) | – | 20 | 20 |
| % of demised eggs (*p*) | – | 10% | 10% |
| The number of elite solutions in current population ($N_e$) | (3%,5%,7%) | – | 5% |
| Lower limit of $\alpha_{itr}$ ($\alpha_{min}$) | (0.3,0.4,0.5) | 1.5 | 0.5 |
| Upper limit of $\alpha_{itr}$ ($\alpha_{max}$) | (1.7,2,2.2) | | 2 |
| Number of iterations without any improvement ($NI_{itr}$) | (3,5,7) | – | 5 |
| The maximum number of iterations ($TN_{itr}$) | – | 400 | 400 |

**Table 2**
Processing times of products at different stage.

| Problem set | Range of $t_{ij}$ (processing times) | Range of $t_{iT}$ (transportation times) | Range of $t_{iA}$ (assembly times) |
|---|---|---|---|
| Set 1 | U[0,100] | U[0,10] | U[0,100] |
| Set 2 | U[0,100] | U[0,50] | U[100,200] |
| Set 3 | U[100,200] | U[0,10] | U[0,100] |
| Set 4 | U[0,100] | U[0,100] | U[0,100] |

**Table 3**
LB deviation from the best solutions based on the number of machines and products.

| a. Based on number of products | | | | | b. Based on number of machines | | | | |
|---|---|---|---|---|---|---|---|---|---|
| n | $LB_1$ | $LB_2$ | $LB_3$ | LB | m | $LB_1$ | $LB_2$ | $LB_3$ | LB |
| 20 | 16.98 | 67.94 | 22.10 | 1.06 | 2 | 17.96 | 68.64 | 20.32 | 0.77 |
| 40 | 16.92 | 68.63 | 21.32 | 0.71 | 4 | 17.05 | 68.78 | 20.99 | 0.71 |
| 60 | 16.96 | 68.91 | 20.50 | 0.54 | 6 | 16.54 | 68.36 | 21.27 | 0.67 |
| 80 | 16.93 | 68.77 | 20.07 | 0.44 | 8 | 16.24 | 68.45 | 21.42 | 0.60 |
| Average | 16.95 | 68.56 | 21.00 | 0.69 | | | | | |

According to Table 3a, the performance of the LB in overall is 0.69% and indicates that the proposed LB is fairly good. Also, $LB_1$ yields the best consequences, as we already expected. $LB_3$ and the $LB_2$ have worse performance. According to Table 3b, $LB_1$ has the best performance based on the number of machines in all cases while $LB_2$ has the worst performance. Note that the average of lower bounds in Table 3b are the same as the presented averages in Table 3a.

According to Table 4 for the sets 1 and 4, $LB_1$ has the best performance. Moreover, since the assembly stage and the first stage have the higher workloads in set 2 and 3; $LB_3$ and $LB_1$ have the better performance for these sets, respectively.

In order to compare the efficiency of the algorithms, Relative Percentage Deviations (RPD) are used as follows:

$$RPD = \frac{Alg_{sol} - LB^*}{LB^*} \times 100 \tag{24}$$

$$LB^* = \max\{LB_1, LB_2, LB_3\} \tag{25}$$

In which $Alg_{sol}$ is the obtained objective value by the selected technique and $LB^*$ signifies the lower bound as defined above. The lower values of *RPD* obviously indicate the better performance of the algorithm. Table 5 shows the calculated RPDs for the algorithms while Table 6 reports the average Frequency of resulting in the Best Solution (FBS). Note that since each metaheuristic algorithm has been run 25 times, the (rounded) average of results is reported. IDCOA has the highest FBS while $DR_3$ has the lowest, which verifies our conclusion about the performance of the algorithms. Also, the FBS of COA, SA and VNS are relatively close to the FBS of IDCOA where VNS has a better FBS than COA and SA.

As can be seen in Table 5, the proposed IDCOA clearly outperforms the other employed algorithms including classic COA, with an average RPD of 0.48%. COA has the second best performance with an average RPD of 0.62. Also, VNS is the third best algorithm. Moreover, $H_{3R}$ with an average RPD of 3.25%, outperforms $H_0$ and all other dispatching rules. $DR_1$ with an average RPD of 3.33% has the best performance among dispatching rules, $DR_1$-$DR_4$, and it also outperforms $H_0$. The average RPD of $DR_3$ is 12.75%, which indicates that this dispatching rule yields the worst consequences among all other employed algorithms.

Generally speaking, it can be said that although few studies employ statistical procedures to compare results, their use is recently growing and it appears to be required by a large number of reviewers. Statistical analyses are usually based on the average and variance by using a parametric test such as Analysis of Variance (ANOVA) (Ozcelik & Erzurumlu, 2006). Lately, non-parametric statistical procedures have been considered for employment in analyses of obtained results by algorithms (García, Molina, Lozano, & Herrera, 2007). Based on this, we are interested in employing parametric or non-parametric statistical tests to analyze the behavior of the applied algorithms over the optimization problem.

**Table 4**
LB deviation from the best solutions based on used sets.

| Set | $LB_1$ | $LB_2$ | $LB_3$ | LB |
|---|---|---|---|---|
| 1 | 1.752 | 88.184 | 8.202 | 0.334 |
| 2 | 61.343 | 80.561 | 0.004 | 0.003 |
| 3 | 0.005 | 94.004 | 64.94 | 0.004 |
| 4 | 4.702 | 11.492 | 10.863 | 2.421 |

Now, we are going to compare the results of the employed techniques. To check whether the differences observed between employed algorithms are statistically significant or not, ANOVA should be carried out for the obtained RPD outcomes. By doing so, one could confirm the statistical validity of the attained consequences and it is possible to determine which algorithm is the best. In these calculations, the different techniques are considered as factors and the RPD is regarded as a response variable.

In order to use the parametric test, it is necessary to check three conditions: independence of residuals, normality and homoscedasticity. To test the normality of obtained results, we apply Kolmogorov-Smirnov method. The P-value (sig.) of this test is obtained zero and obviously smaller than the significance level (0.05), accordingly, the attained data are not normally distributed. In such a situation, we cannot use parametric tests such as ANOVA, and consequently the equivalent non-parametric tests should be applied. Non-parametric tests can be employed to compare algorithms whose outcomes signify the average values for each problem, despite the inexistence of relationships among them. They do not require explicit conditions related to the sample of data to be conducted.

As a non-parametric technique for testing whether samples originate from the same distribution or not, we apply the Kruskal–Wallis one-way analysis of variance. This test is employed to compare two or more samples which are independent and may have different sample sizes. Unlike the analogous one-way ANOVA, this test does not assume a normal distribution of the residuals. When the null hypothesis of the Kruskal-Wallis test is rejected, it means that at least one of cases stochastically dominates at least one other case. Table 7 shows the mean rank of groups.

The null hypothesis of the Kruskal-Wallis test assumes that the samples are from identical populations, while alternative hypothesis assumes that the samples belong to different populations/society. The null hypothesis could be stated in another way; the mean ranks of samples from the populations are expected to be the same. The smallest value gets the rank of 1, the next smallest gets the rank of 2, and so on. Fig. 7 depicts the mean rank of employed algorithms.

Since the Kruskal-Wallis is a non-parametric statistical test, there is no possibility for checking whether, for example, the COA, VNS and SA algorithms are significantly different or not, in a statistical way. In better words, this test only considers all groups altogether and it says that at least one of employed algorithms stochastically dominates at least one other algorithm or not. But, if we were allowed to use parametric tests (i.e., ANOVA), one could apply the Tukey test. The Tukey test, probably the most

**Table 5**
Average RPD of algorithms.

| ID | n | m | Set | IDCOA | COA | SA | VNS | $DR_1$ | $DR_2$ | $DR_3$ | $DR_4$ | $H_0$ | $H_{3R}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 20 | 2 | 1 | 0.114 | 0.150 | 1.186 | 1.182 | 3.95 | 8.66 | 22.32 | 13.71 | 6.74 | 2.91 |
| 2 | 20 | 2 | 2 | 0.026 | 0.050 | 0.062 | 0.051 | 0.23 | 1.15 | 1.71 | 0.36 | 1.15 | 1.15 |
| 3 | 20 | 2 | 3 | 0.017 | 0.040 | 0.042 | 0.042 | 1.53 | 1.84 | 2.95 | 2.77 | 0.70 | 1.32 |
| 4 | 20 | 2 | 4 | 2.409 | 3.870 | 5.600 | 5.507 | 8.90 | 23.92 | 28.14 | 18.61 | 15.87 | 10.16 |
| 5 | 20 | 4 | 1 | 0.494 | 0.640 | 0.744 | 0.610 | 6.18 | 9.20 | 21.79 | 17.36 | 7.46 | 2.67 |
| 6 | 20 | 4 | 2 | 0.004 | 0.010 | 0.008 | 0.009 | 0.26 | 0.75 | 1.44 | 0.47 | 0.75 | 0.75 |
| 7 | 20 | 4 | 3 | 0.005 | 0.010 | 0.009 | 0.011 | 1.53 | 1.51 | 2.88 | 2.81 | 0.55 | 1.31 |
| 8 | 20 | 4 | 4 | 2.551 | 3.930 | 4.377 | 4.203 | 19.64 | 22.84 | 27.40 | 21.41 | 14.98 | 10.22 |
| 9 | 20 | 6 | 1 | 0.637 | 0.650 | 0.710 | 0.684 | 5.18 | 8.47 | 21.86 | 19.30 | 6.68 | 2.14 |
| 10 | 20 | 6 | 2 | 0.002 | 0.010 | 0.002 | 0.008 | 0.27 | 0.53 | 1.25 | 0.35 | 0.53 | 0.53 |
| 11 | 20 | 6 | 3 | 0.001 | 0.008 | 0.009 | 0.009 | 1.51 | 1.47 | 2.86 | 2.78 | 0.57 | 1.11 |
| 12 | 20 | 6 | 4 | 2.162 | 2.340 | 3.920 | 3.450 | 10.58 | 23.32 | 28.01 | 23.08 | 14.79 | 10.45 |
| 13 | 20 | 8 | 1 | 0.143 | 0.250 | 0.464 | 0.370 | 6.09 | 7.52 | 20.93 | 19.44 | 5.64 | 1.79 |
| 14 | 20 | 8 | 2 | 0.002 | 0.008 | 0.009 | 0.008 | 0.36 | 0.34 | 1.18 | 0.32 | 0.34 | 0.34 |
| 15 | 20 | 8 | 3 | 0.002 | 0.007 | 0.007 | 0.006 | 1.47 | 1.45 | 2.83 | 2.78 | 0.51 | 1.12 |
| 16 | 20 | 8 | 4 | 1.219 | 3.140 | 3.512 | 3.200 | 11.41 | 23.24 | 27.43 | 23.07 | 15.46 | 10.77 |
| 17 | 40 | 2 | 1 | 0.247 | 0.720 | 0.773 | 0.710 | 1.81 | 6.10 | 23.77 | 13.48 | 4.91 | 1.71 |
| 18 | 40 | 2 | 2 | 0.003 | 0.020 | 0.023 | 0.010 | 0.16 | 0.64 | 0.99 | 0.19 | 0.64 | 0.64 |
| 19 | 40 | 2 | 3 | 0.002 | 0.010 | 0.005 | 0.004 | 0.80 | 0.83 | 1.56 | 1.48 | 0.18 | 0.76 |
| 20 | 40 | 2 | 4 | 2.228 | 2.590 | 3.334 | 3.050 | 4.41 | 22.76 | 27.31 | 15.20 | 15.72 | 9.78 |
| 21 | 40 | 4 | 1 | 0.266 | 0.420 | 0.584 | 0.470 | 3.01 | 6.31 | 22.53 | 18.07 | 4.48 | 1.56 |
| 22 | 40 | 4 | 2 | 0.004 | 0.007 | 0.003 | 0.004 | 0.11 | 0.48 | 0.87 | 0.19 | 0.48 | 0.48 |
| 23 | 40 | 4 | 3 | 0.004 | 0.006 | 0.001 | 0.003 | 0.78 | 0.87 | 1.53 | 1.49 | 0.26 | 0.69 |
| 24 | 40 | 4 | 4 | 1.589 | 2.090 | 2.822 | 2.150 | 6.19 | 22.47 | 26.52 | 18.73 | 15.56 | 10.42 |
| 25 | 40 | 6 | 1 | 0.293 | 0.350 | 0.371 | 0.240 | 3.88 | 6.03 | 22.62 | 20.04 | 3.92 | 1.23 |
| 26 | 40 | 6 | 2 | 0.008 | 0.010 | 0.000 | 0.010 | 0.12 | 0.38 | 0.77 | 0.21 | 0.38 | 0.38 |
| 27 | 40 | 6 | 3 | 0.006 | 0.010 | 0.000 | 0.010 | 0.82 | 0.90 | 1.54 | 1.50 | 0.10 | 0.70 |
| 28 | 40 | 6 | 4 | 1.836 | 2.200 | 2.714 | 2.530 | 7.17 | 23.09 | 26.34 | 20.98 | 15.84 | 10.78 |
| 29 | 40 | 8 | 1 | 0.160 | 0.290 | 0.348 | 0.320 | 4.07 | 6.02 | 21.59 | 19.98 | 3.49 | 1.19 |
| 30 | 40 | 8 | 2 | 0.001 | 0.001 | 0.001 | 0.001 | 0.12 | 0.27 | 0.68 | 0.21 | 0.27 | 0.27 |
| 31 | 40 | 8 | 3 | 0.001 | 0.001 | 0.001 | 0.001 | 0.68 | 0.92 | 1.52 | 1.51 | 0.15 | 0.64 |
| 32 | 40 | 8 | 4 | 1.732 | 2.060 | 2.353 | 2.150 | 9.09 | 22.27 | 25.62 | 20.71 | 15.41 | 10.47 |
| 33 | 60 | 2 | 1 | 0.159 | 0.400 | 0.496 | 0.420 | 1.23 | 5.08 | 23.00 | 11.91 | 3.51 | 1.21 |
| 34 | 60 | 2 | 2 | 0.007 | 0.010 | 0.013 | 0.010 | 0.11 | 0.51 | 0.71 | 0.11 | 0.51 | 0.51 |
| 35 | 60 | 2 | 3 | 0.008 | 0.009 | 0.008 | 0.008 | 0.53 | 0.59 | 1.07 | 1.04 | 0.06 | 0.45 |
| 36 | 60 | 2 | 4 | 1.789 | 2.630 | 2.570 | 2.040 | 3.75 | 22.83 | 27.30 | 14.28 | 17.52 | 10.94 |
| 37 | 60 | 4 | 1 | 0.186 | 0.250 | 0.380 | 0.260 | 2.53 | 5.04 | 22.56 | 17.54 | 3.15 | 0.95 |
| 38 | 60 | 4 | 2 | 0.002 | 0.000 | 0.000 | 0.000 | 0.09 | 0.38 | 0.61 | 0.13 | 0.38 | 0.38 |
| 39 | 60 | 4 | 3 | 0.002 | 0.000 | 0.000 | 0.000 | 0.51 | 0.55 | 1.06 | 1.04 | 0.04 | 0.47 |
| 40 | 60 | 4 | 4 | 2.079 | 2.110 | 2.261 | 2.090 | 4.47 | 22.07 | 25.07 | 17.83 | 16.09 | 10.33 |
| 41 | 60 | 6 | 1 | 0.203 | 0.250 | 0.339 | 0.280 | 6.81 | 4.61 | 22.54 | 20.13 | 3.06 | 1.00 |
| 42 | 60 | 6 | 2 | 0.001 | 0.004 | 0.002 | 0.003 | 0.10 | 0.26 | 0.54 | 0.13 | 0.26 | 0.26 |
| 43 | 60 | 6 | 3 | 0.001 | 0.003 | 0.002 | 0.002 | 0.52 | 0.56 | 1.06 | 1.05 | 0.03 | 0.54 |
| 44 | 60 | 6 | 4 | 0.789 | 0.880 | 1.980 | 0.850 | 15.01 | 22.44 | 24.46 | 18.71 | 16.03 | 10.90 |
| 45 | 60 | 8 | 1 | 0.182 | 0.190 | 0.220 | 0.190 | 2.83 | 4.33 | 21.27 | 19.42 | 2.70 | 0.79 |
| 46 | 60 | 8 | 2 | 0.001 | 0.002 | 0.004 | 0.002 | 0.10 | 0.19 | 0.51 | 0.14 | 0.19 | 0.19 |
| 47 | 60 | 8 | 3 | 0.001 | 0.012 | 0.004 | 0.003 | 0.51 | 0.54 | 1.05 | 1.04 | 0.03 | 0.46 |
| 48 | 60 | 8 | 4 | 0.708 | 1.020 | 1.763 | 1.140 | 5.08 | 22.01 | 24.88 | 19.12 | 15.85 | 10.41 |
| 49 | 80 | 2 | 1 | 0.202 | 0.210 | 0.358 | 0.240 | 2.27 | 4.00 | 23.11 | 11.48 | 2.70 | 0.86 |
| 50 | 80 | 2 | 2 | 0.004 | 0.010 | 0.012 | 0.004 | 0.09 | 0.39 | 0.55 | 0.10 | 0.39 | 0.39 |
| 51 | 80 | 2 | 3 | 0.003 | 0.004 | 0.003 | 0.003 | 0.39 | 0.50 | 0.81 | 0.78 | 0.02 | 0.38 |
| 52 | 80 | 2 | 4 | 1.030 | 1.670 | 2.114 | 1.420 | 3.30 | 23.12 | 26.09 | 13.35 | 17.43 | 10.59 |
| 53 | 80 | 4 | 1 | 0.150 | 0.190 | 0.310 | 0.220 | 1.71 | 4.27 | 23.20 | 18.25 | 2.64 | 0.75 |
| 54 | 80 | 4 | 2 | 0.002 | 0.003 | 0.004 | 0.003 | 0.09 | 0.31 | 0.53 | 0.11 | 0.31 | 0.31 |
| 55 | 80 | 4 | 3 | 0.001 | 0.002 | 0.002 | 0.002 | 0.39 | 0.42 | 0.81 | 0.79 | 0.03 | 0.36 |
| 56 | 80 | 4 | 4 | 1.015 | 1.340 | 1.653 | 1.370 | 13.65 | 21.31 | 25.41 | 16.21 | 15.89 | 10.25 |
| 57 | 80 | 6 | 1 | 0.030 | 0.070 | 0.258 | 0.090 | 2.15 | 4.22 | 22.64 | 19.92 | 2.75 | 0.73 |
| 58 | 80 | 6 | 2 | 0.003 | 0.004 | 0.004 | 0.004 | 0.08 | 0.20 | 0.40 | 0.11 | 0.20 | 0.20 |
| 59 | 80 | 6 | 3 | 0.004 | 0.010 | 0.004 | 0.003 | 0.47 | 0.42 | 0.80 | 0.79 | 0.02 | 0.34 |
| 60 | 80 | 6 | 4 | 1.170 | 1.280 | 1.617 | 1.380 | 4.03 | 21.86 | 24.43 | 17.66 | 15.75 | 10.45 |
| 61 | 80 | 8 | 1 | 0.090 | 0.120 | 0.176 | 0.110 | 2.22 | 3.98 | 22.40 | 20.53 | 2.33 | 0.71 |
| 62 | 80 | 8 | 2 | 0.003 | 0.004 | 0.003 | 0.004 | 0.08 | 0.17 | 0.42 | 0.11 | 0.17 | 0.17 |
| 63 | 80 | 8 | 3 | 0.003 | 0.004 | 0.004 | 0.003 | 0.37 | 0.47 | 0.80 | 0.79 | 0.02 | 0.33 |
| 64 | 80 | 8 | 4 | 0.840 | 1.050 | 1.567 | 1.070 | 15.16 | 22.55 | 24.98 | 19.07 | 16.47 | 11.07 |
| Average | | | | 0.450 | 0.619 | 0.814 | 0.690 | 3.33 | 7.45 | 12.75 | 9.47 | 5.17 | 3.25 |

"conservative" multiple comparison test, is used along with an ANOVA so as to find which algorithms are significantly different from each other.

According to Fig. 7, the IDCOA gets the first rank among all other algorithms while $DR_4$ yields the worst results among them. IDCOA, COA, VNS as well as SA have considerably a lower mean rank with respect to the others. Also, $H_{3R}$, $DR_1$, and $H_0$ to some extent have the same mean rank. Note that IDCOA presents the best solution quality among all considered algorithms, since it uses novel features and adjustments which improve its performance. The proposed migration and egg laying mechanisms, enhanced by directed and random moves, help IDCOA to search locally and

**Table 6**
Average FBS of algorithms.

| ID | n | m | Set | IDCOA | COA | SA | VNS | $DR_1$ | $DR_2$ | $DR_3$ | $DR_4$ | $H_0$ | $H_{3R}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 20 | 2 | 1 | 92[a] | 53 | 27 | 38 | 19 | 0 | 0 | 0 | 10 | 3 |
| 2 | 20 | 2 | 2 | 99 | 91 | 89 | 95 | 56 | 12 | 2 | 41 | 12 | 12 |
| 3 | 20 | 2 | 3 | 92 | 82 | 83 | 85 | 8 | 2 | 0 | 0 | 47 | 8 |
| 4 | 20 | 2 | 4 | 100 | 74 | 45 | 51 | 2 | 0 | 0 | 0 | 1 | 6 |
| 5 | 20 | 4 | 1 | 85 | 45 | 36 | 38 | 7 | 0 | 0 | 0 | 9 | 7 |
| 6 | 20 | 4 | 2 | 100 | 99 | 99 | 100 | 50 | 18 | 5 | 29 | 18 | 18 |
| 7 | 20 | 4 | 3 | 99 | 96 | 97 | 99 | 2 | 3 | 0 | 0 | 49 | 9 |
| 8 | 20 | 4 | 4 | 93 | 69 | 65 | 72 | 1 | 0 | 0 | 0 | 2 | 5 |
| 9 | 20 | 6 | 1 | 87 | 82 | 53 | 81 | 10 | 0 | 0 | 0 | 9 | 9 |
| 10 | 20 | 6 | 2 | 100 | 99 | 99 | 100 | 47 | 16 | 4 | 36 | 16 | 16 |
| 11 | 20 | 6 | 3 | 100 | 99 | 100 | 100 | 7 | 2 | 0 | 0 | 42 | 16 |
| 12 | 20 | 6 | 4 | 95 | 87 | 86 | 92 | 0 | 0 | 0 | 0 | 4 | 7 |
| 13 | 20 | 8 | 1 | 82 | 77 | 57 | 73 | 4 | 0 | 0 | 0 | 12 | 18 |
| 14 | 20 | 8 | 2 | 100 | 100 | 100 | 100 | 33 | 27 | 7 | 34 | 27 | 27 |
| 15 | 20 | 8 | 3 | 100 | 99 | 100 | 100 | 0 | 2 | 0 | 0 | 50 | 8 |
| 16 | 20 | 8 | 4 | 89 | 75 | 68 | 71 | 0 | 0 | 0 | 0 | 0 | 4 |
| 17 | 40 | 2 | 1 | 97 | 79 | 75 | 79 | 21 | 1 | 0 | 0 | 8 | 4 |
| 18 | 40 | 2 | 2 | 94 | 92 | 89 | 91 | 42 | 9 | 5 | 40 | 9 | 9 |
| 19 | 40 | 2 | 3 | 96 | 92 | 91 | 94 | 4 | 0 | 0 | 0 | 46 | 10 |
| 20 | 40 | 2 | 4 | 93 | 88 | 82 | 87 | 1 | 0 | 0 | 0 | 0 | 1 |
| 21 | 40 | 4 | 1 | 94 | 82 | 80 | 84 | 4 | 0 | 0 | 0 | 17 | 9 |
| 22 | 40 | 4 | 2 | 100 | 100 | 99 | 100 | 54 | 11 | 2 | 41 | 11 | 11 |
| 23 | 40 | 4 | 3 | 99 | 96 | 98 | 98 | 2 | 0 | 0 | 0 | 45 | 8 |
| 24 | 40 | 4 | 4 | 88 | 76 | 84 | 86 | 0 | 0 | 0 | 0 | 0 | 0 |
| 25 | 40 | 6 | 1 | 93 | 79 | 45 | 76 | 4 | 0 | 0 | 0 | 22 | 10 |
| 26 | 40 | 6 | 2 | 100 | 100 | 100 | 100 | 55 | 7 | 3 | 32 | 7 | 7 |
| 27 | 40 | 6 | 3 | 100 | 100 | 100 | 100 | 2 | 0 | 0 | 0 | 61 | 4 |
| 28 | 40 | 6 | 4 | 91 | 79 | 74 | 86 | 0 | 0 | 0 | 0 | 0 | 1 |
| 29 | 40 | 8 | 1 | 99 | 95 | 87 | 91 | 2 | 2 | 0 | 0 | 19 | 6 |
| 30 | 40 | 8 | 2 | 100 | 99 | 100 | 100 | 50 | 13 | 2 | 27 | 13 | 13 |
| 31 | 40 | 8 | 3 | 100 | 100 | 100 | 100 | 6 | 0 | 0 | 0 | 53 | 8 |
| 32 | 40 | 8 | 4 | 94 | 79 | 68 | 73 | 0 | 0 | 0 | 0 | 0 | 1 |
| 33 | 60 | 2 | 1 | 95 | 91 | 84 | 88 | 22 | 0 | 0 | 0 | 13 | 5 |
| 34 | 60 | 2 | 2 | 96 | 95 | 95 | 96 | 44 | 4 | 1 | 31 | 4 | 4 |
| 35 | 60 | 2 | 3 | 100 | 99 | 100 | 100 | 1 | 0 | 0 | 0 | 51 | 4 |
| 36 | 60 | 2 | 4 | 87 | 68 | 74 | 79 | 0 | 0 | 0 | 0 | 0 | 0 |
| 37 | 60 | 4 | 1 | 100 | 92 | 76 | 91 | 2 | 0 | 0 | 0 | 16 | 2 |
| 38 | 60 | 4 | 2 | 100 | 99 | 99 | 100 | 44 | 6 | 2 | 33 | 6 | 6 |
| 39 | 60 | 4 | 3 | 100 | 100 | 100 | 100 | 2 | 0 | 0 | 0 | 45 | 5 |
| 40 | 60 | 4 | 4 | 88 | 81 | 72 | 85 | 0 | 0 | 0 | 0 | 0 | 0 |
| 41 | 60 | 6 | 1 | 93 | 89 | 81 | 90 | 4 | 0 | 0 | 0 | 14 | 2 |
| 42 | 60 | 6 | 2 | 100 | 99 | 99 | 100 | 34 | 5 | 0 | 27 | 5 | 5 |
| 43 | 60 | 6 | 3 | 100 | 100 | 100 | 100 | 2 | 0 | 0 | 0 | 57 | 5 |
| 44 | 60 | 6 | 4 | 83 | 60 | 62 | 76 | 0 | 0 | 0 | 0 | 0 | 1 |
| 45 | 60 | 8 | 1 | 94 | 79 | 85 | 94 | 2 | 0 | 0 | 0 | 21 | 6 |
| 46 | 60 | 8 | 2 | 100 | 100 | 100 | 100 | 43 | 15 | 1 | 29 | 15 | 15 |
| 47 | 60 | 8 | 3 | 100 | 99 | 100 | 100 | 1 | 0 | 0 | 0 | 55 | 4 |
| 48 | 60 | 8 | 4 | 86 | 84 | 71 | 77 | 0 | 0 | 0 | 0 | 0 | 0 |
| 49 | 80 | 2 | 1 | 98 | 97 | 90 | 94 | 0 | 0 | 0 | 0 | 0 | 0 |
| 50 | 80 | 2 | 2 | 100 | 99 | 97 | 100 | 38 | 7 | 0 | 30 | 7 | 7 |
| 51 | 80 | 2 | 3 | 100 | 100 | 99 | 100 | 2 | 0 | 0 | 0 | 49 | 0 |
| 52 | 80 | 2 | 4 | 81 | 66 | 60 | 75 | 0 | 0 | 0 | 0 | 0 | 0 |
| 53 | 80 | 4 | 1 | 92 | 84 | 78 | 86 | 12 | 0 | 0 | 0 | 17 | 4 |
| 54 | 80 | 4 | 2 | 100 | 98 | 98 | 100 | 34 | 5 | 0 | 32 | 5 | 5 |
| 55 | 80 | 4 | 3 | 100 | 100 | 98 | 100 | 2 | 0 | 0 | 0 | 59 | 5 |
| 56 | 80 | 4 | 4 | 87 | 69 | 64 | 78 | 1 | 0 | 0 | 0 | 0 | 1 |
| 57 | 80 | 6 | 1 | 97 | 93 | 85 | 91 | 3 | 0 | 0 | 0 | 14 | 3 |
| 58 | 80 | 6 | 2 | 100 | 100 | 100 | 100 | 43 | 9 | 1 | 25 | 9 | 9 |
| 59 | 80 | 6 | 3 | 98 | 84 | 100 | 100 | 0 | 0 | 0 | 0 | 50 | 4 |
| 60 | 80 | 6 | 4 | 89 | 83 | 76 | 81 | 1 | 0 | 0 | 0 | 0 | 0 |
| 61 | 80 | 8 | 1 | 92 | 78 | 75 | 84 | 3 | 0 | 0 | 0 | 22 | 4 |
| 62 | 80 | 8 | 2 | 100 | 100 | 99 | 100 | 43 | 11 | 1 | 22 | 11 | 11 |
| 63 | 80 | 8 | 3 | 100 | 99 | 100 | 100 | 5 | 0 | 0 | 0 | 56 | 7 |
| 64 | 80 | 8 | 4 | 96 | 89 | 71 | 82 | 0 | 0 | 0 | 0 | 0 | 0 |
| Average | | | | 95.36 | 88.08 | 83.81 | 88.86 | 13.77 | 2.92 | 0.56 | 7.95 | 19.06 | 6.23 |

[a] The (rounded) average of FBS (= [total no. of FBS/total run (=100 × 25)] where [x] represents the smallest integer greater to real number $x$).

globally more efficient, this is beside local search mechanisms which provides more opportunity to improve the obtained solutions.

If the result of Kruskal–Wallis test is not significant, there is no evidence of stochastic dominance between the samples. After applying Kruskal-Wallis test, the P-value (sig.) is obtained zero and consequently the null hypothesis is rejected meaning there is a statistically significant difference between performances of the employed algorithms. In other words, one could reject the idea which the difference is owing to random sampling and it could be concluded that the populations have different distributions.

**Table 7**
The ranks of groups in Kruskal–Wallis test.

| | Group | N | Mean rank |
|---|---|---|---|
| RPD | IDCOA | 64 | 175.24 |
| | COA | 64 | 194.55 |
| | VNS | 64 | 209.42 |
| | SA | 64 | 195.91 |
| | H3R | 64 | 364.13 |
| | DR1 | 64 | 420.85 |
| | H0 | 64 | 487.23 |
| | DR2 | 64 | 428.73 |
| | DR3 | 64 | 365.98 |
| | DR4 | 64 | 362.96 |
| | Total | 640 | |

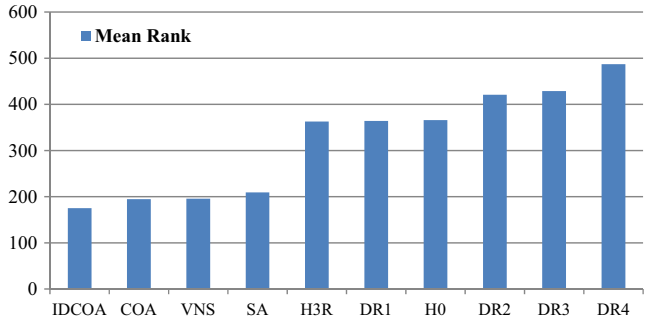



**Fig. 7.** Mean rank of employed algorithms.

In Figs. 8–10, the performance of applied techniques is presented based on number of machines, products, and sets. As can be implied from these figures, IDCOA and $DR_3$ have the best and worst performance among all other algorithms, respectively. Also, VNS and SA have the second and the third ranks in term of the best outcomes, respectively.

In Fig. 10 the performances of the algorithms based on sets are shown. The performance of the algorithms for sets 2 and 3 are prominent; however, for sets 1 and 4 their performance dramatically changes.

Also, in terms of the computational time, it should be mentioned that all of the heuristic algorithms and dispatching rules perform very similar to each other and actually there is not a tangible difference among them. However, it can be concluded that the IDCOA in general spends more computational time than SA and VNS as presented in Fig. 11.

In order to investigate the effect of each of the incorporated improvements in the proposed IDCOA algorithm, experiments are conducted on sample problems with n = 60, i.e., problems 33–48 from Tables 5 and 6, where these problems are large enough in terms of dimension to distinguish the effect of each improvement in the proposed IDCOA. Recall that each problem has 100 instances, and the performance of IDCOA with and without the proposed feature is tested based on 1600 instances which are fairly extensive benchmark set and it allows us to extend the conclusion to other benchmark sets.

In this set of experiments, in order to test the contribution of each feature, experiments are conducted with and without that feature, the experiment with the addressed feature is the same as the results reported in Table 5 while the experiments without that feature are based on only dropping the addressed feature from the proposed IDCOA. In other words, each section/group column of Table 8 is devoted to examine one feature of the proposed IDCOA, where the left column represents results using the addressed feature while the other column(s) in the right side of the section represents the result using classical feature as proposed by Rajabioun (2011) in the original COA algorithm. The results in Table 8 are reported in "x/y" format where "x" represents the average RPD and "y" represents the computational time. The features (improvements) of the proposed IDCOA are "Grouping Technique", "DELR", "Elimination", and "Local Search". The last row of Table 8 represents the performance in terms of average RPD and computational times. It shows that using K-means in the grouping technique has better performance comparing CCA while it increases dramatically the computational time of the algorithm. Using dynamic controlling parameter $\alpha$ and regarding the quality of cuckoo to define the radius/step length for egg laying as well as the number of eggs in that region (DELR component) is also investigated; i.e., considering these features will enhance search capability in terms of effectiveness and efficiency (quality and time). In terms of the elimination strategy, dropping the best one is faster but it deteriorates the performance of the algorithm. Finally, local search improves the performance of the IDCOA considerably while its computational burden is acceptable.

In order to investigate that whether there is statistically difference between using a feature in the proposed IDCOA or not, the ANOVA is conducted at significance level of 0.05. Table 9 presents the result of the ANOVA test where "Y" and "N" indicates "Yes" and "No", respectively. In better words, "Y" means there is significant

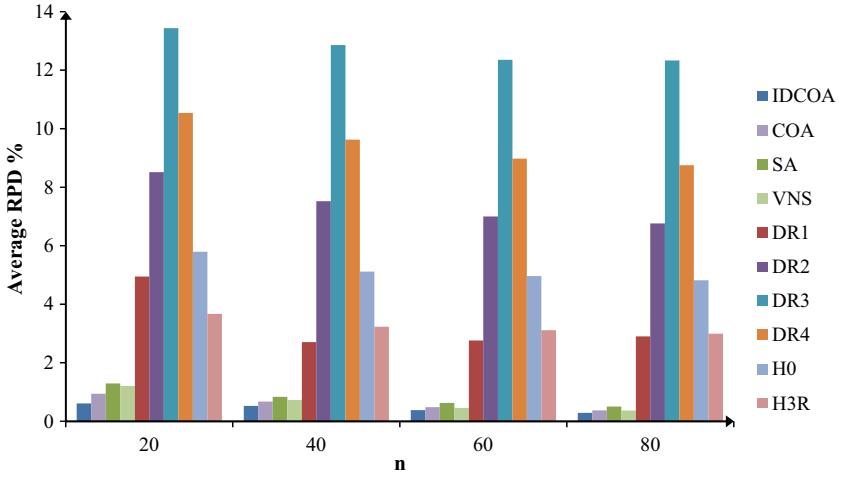



**Fig. 8.** RPD of algorithms based on the number of products.

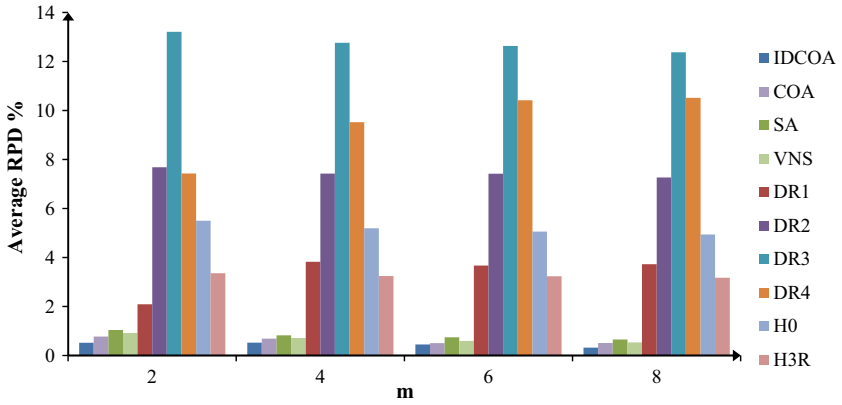



**Fig. 9.** RPD of algorithms based on the number of machines.

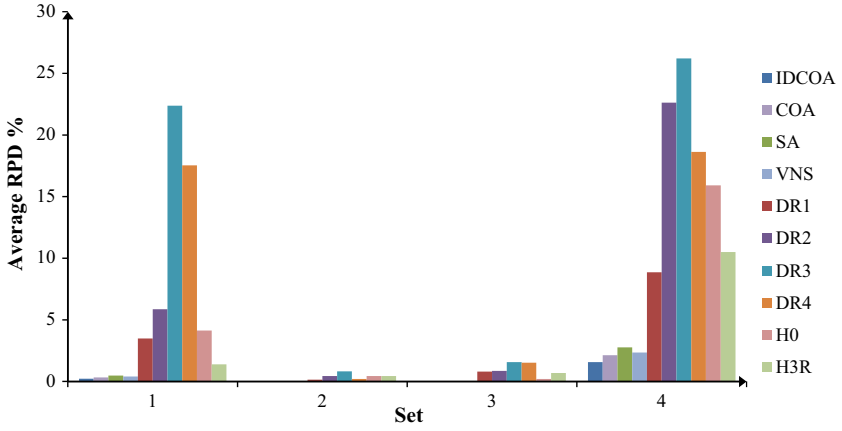
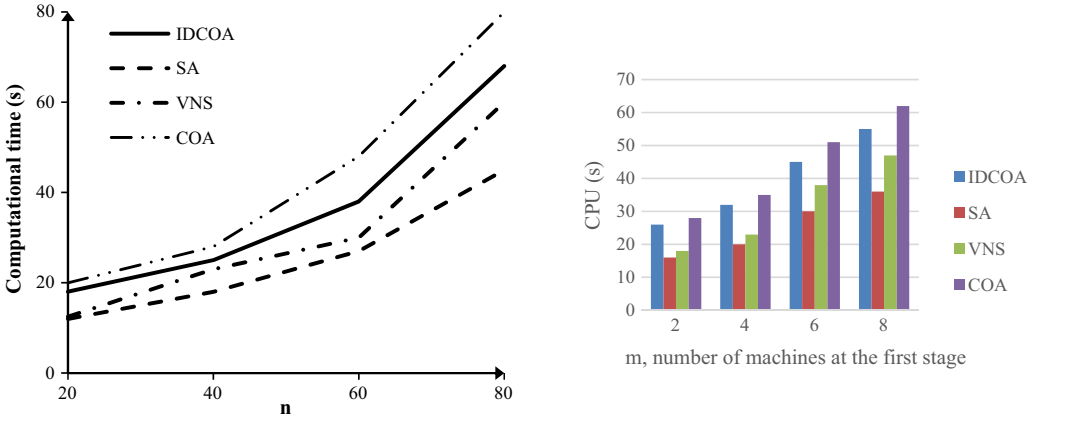



**Fig. 10.** RPD of algorithms based on sets.




(a) based on n, the number of products (b) based on n, the number of machines at the first stage

**Fig. 11.** Computational time of the algorithms based on number of products.

difference in using the given feature while "N" signifies no significant difference could be observed by using that given feature. According to the results presented in Table 9, the grouping techniques are not significantly different from each other in most of the problems while their computational time are considerably significant from each other. It should be pointed out that if the overall result of DELR is significant, at least one of the (1) & (2), (1) & (3) or (2) & (3) features has significant difference. Inversely, if the overall result of DELR is not significant, none of the above-mentioned pairwise of features are significant. In this case, the result is presented by "-". Generally speaking, one should mention that in doing ANOVA, the null hypothesis signifies that the population means for all conditions are the same. Accordingly, when the null hypothesis is rejected, it could be concluded that at least one of the population means differs from at least one other population mean. In overall result of DELR in Table 9, the left element of Y/N, i.e., Y, corresponds to "quality" or average RPD and the right element of Y/N, i.e., N, corresponds to "computational time." As it could be

**Table 8**
Effect of each feature of IDCO based on average RPD and time (Ave. RPD/T).[a]

| Problem | | | | Grouping tech. | | DELR (dynamic $\alpha$ & $Q_f$) | | | Elimination | | Local Search | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ID | Structure | | | CCA | K-Means | Dynamic $\alpha$ | | (3) Basic | $P_e$ | Best ones | Yes | No |
| | n | m | Set | | | (1) With $Q_f$ | (2) Without $Q_f$ | | | | | |
| 33 | 60 | 2 | 1 | 0.16/25 | 0.16/27 | 0.16/25 | 0.17/26 | 0.22/27 | 0.16/25 | 0.16/25 | 0.16/25 | 0.33/23 |
| 34 | 60 | 2 | 2 | 0.01/27 | 0.01/30 | 0.01/27 | 0.01/28 | 0.01/28 | 0.01/27 | 0.01/26 | 0.01/27 | 0.01/26 |
| 35 | 60 | 2 | 3 | 0.00/35 | 0.00/38 | 0.00/35 | 0.00/36 | 0.00/36 | 0.00/35 | 0.00/33 | 0.00/35 | 0.00/32 |
| 36 | 60 | 2 | 4 | 1.79/42 | 1.66/45 | 1.79/42 | 1.94/44 | 2.15/47 | 1.79/42 | 1.82/40 | 1.79/42 | 2.04/38 |
| 37 | 60 | 4 | 1 | 0.19/46 | 0.19/49 | 0.19/46 | 0.22/46 | 0.22/49 | 0.19/46 | 0.19/43 | 0.19/46 | 0.23/43 |
| 38 | 60 | 4 | 2 | 0.00/46 | 0.00/51 | 0.00/46 | 0.00/47 | 0.00/50 | 0.00/46 | 0.00/43 | 0.00/46 | 0.00/44 |
| 39 | 60 | 4 | 3 | 0.00/45 | 0.00/52 | 0.00/45 | 0.00/49 | 0.00/50 | 0.00/45 | 0.00/44 | 0.00/45 | 0.00/46 |
| 40 | 60 | 4 | 4 | 2.08/48 | 2.01/55 | 2.08/48 | 2.09/48 | 2.09/55 | 2.08/48 | 2.10/46 | 2.08/48 | 2.09/51 |
| 41 | 60 | 6 | 1 | 0.20/34 | 0.19/35 | 0.20/34 | 0.21/34 | 0.21/37 | 0.20/34 | 0.23/32 | 0.20/34 | 0.22/32 |
| 42 | 60 | 6 | 2 | 0.00/36 | 0.00/44 | 0.00/36 | 0.00/37 | 0.00/39 | 0.00/36 | 0.00/35 | 0.00/36 | 0.00/33 |
| 43 | 60 | 6 | 3 | 0.00/38 | 0.00/45 | 0.00/38 | 0.00/40 | 0.00/44 | 0.00/38 | 0.00/36 | 0.00/38 | 0.00/34 |
| 44 | 60 | 6 | 4 | 0.79/30 | 0.77/37 | 0.79/30 | 0.83/37 | 0.84/39 | 0.79/30 | 0.81/26 | 0.79/30 | 0.80/35 |
| 45 | 60 | 8 | 1 | 0.18/36 | 0.18/39 | 0.18/36 | 0.18/38 | 0.18/40 | 0.18/36 | 0.18/34 | 0.18/36 | 0.19/34 |
| 46 | 60 | 8 | 2 | 0.00/38 | 0.00/46 | 0.00/38 | 0.00/41 | 0.00/47 | 0.00/38 | 0.00/36 | 0.00/38 | 0.00/35 |
| 47 | 60 | 8 | 3 | 0.00/40 | 0.00/51 | 0.00/40 | 0.01/45 | 0.01/49 | 0.00/40 | 0.01/36 | 0.00/40 | 0.01/37 |
| 48 | 60 | 8 | 4 | 0.71/42 | 0.67/60 | 0.71/42 | 0.79/46 | 0.93/57 | 0.71/42 | 0.77/37 | 0.71/42 | 0.96/38 |
| Average | | | | 0.38/38 | 0.37/44 | 0.38/38 | 0.40/40 | 0.43/43 | 0.38/38 | 0.39/36 | 0.38/38 | 0.43/36 |

[a] RPDs are rounded into two digits.

**Table 9**
ANOVA test on the experiments of Table 7 at significance level of 0.05.

| Problem ID | Grouping tech. | DELR ($\alpha$ & $Q_f$) | | | | Elimination | Local search |
|---|---|---|---|---|---|---|---|
| | | Overall | (1)[a]&(2) | (1)&(3) | (2)&(3) | | |
| 33 | N/N | Y/N | N/– | Y/– | N/– | N/N | Y/N |
| 34 | N/N | N/N | –/– | –/– | –/– | N/N | N/N |
| 35 | N/N | N/N | –/– | –/– | –/– | N/N | N/N |
| 36 | Y/Y | Y/Y | Y/N | Y/Y | Y/Y | N/N | Y/Y |
| 37 | N/N | Y/N | Y/– | Y/– | N/– | N/N | Y/N |
| 38 | N/Y | N/N | –/– | –/– | –/– | N/N | N/N |
| 39 | N/Y | N/N | –/– | –/– | –/– | N/N | N/N |
| 40 | N/Y | N/Y | –/N | –/Y | –/Y | N/N | Y/Y |
| 41 | N/N | N/N | –/– | –/– | –/– | Y/N | Y/N |
| 42 | N/Y | N/N | –/– | –/– | –/– | N/N | N/Y |
| 43 | N/Y | N/Y | –/N | –/Y | –/Y | N/N | N/Y |
| 44 | N/Y | Y/Y | Y/Y | Y/Y | N/N | N/Y | N/Y |
| 45 | Y/Y | N/N | –/– | –/– | –/– | N/N | Y/N |
| 46 | N/Y | N/Y | –/N | –/Y | –/Y | N/N | N/Y |
| 47 | N/Y | Y/Y | Y/Y | Y/Y | N/N | N/Y | N/Y |
| 48 | N/Y | Y/Y | N/N | Y/Y | Y/Y | Y/Y | Y/Y |

[a] (1): Dynamic $\alpha$ with $Q_f$, (2): dynamic $\alpha$ without $Q_f$, (3): basic.

observed from Table 9, by increasing the size of the problem, adding the mentioned features in IDCOA play an important role in yielding higher quality solutions, while control the computational times in a desirable way. This table clearly shows the usefulness of the used features in IDCOA, separately.

## 6. Conclusions and future works

Minimizing the makespan of a three-stage assembly flow shop problem is addressed in this study where the first stage has $m$ identical parallel machines and the other stages have single machine. There are $n$ products where each product has $m$ components that should be processed at the first stage. The second stage collects the processed parts and transfers them to the assembly stage, i.e. the third stage; finally, at the assembly stage these components go under an assembly operation to produce the final product.

For this problem we proposed a lower bound (LB) and some dispatching rules called $DR_1$, $DR_2$, $DR_3$ and $DR_4$. The experiments showed that our LB is fairly good and $DR_1$ outperforms other proposed dispatching rules. Also, we proposed the Improved Discrete Cuckoo Optimization Algorithm (IDCOA) with new considerations to cope with the scheduling problem. The IDCOA incorporates novel mechanisms and computations to better explore the solution space. The adjusted mechanisms for new immigration, *DELR*, egg laying and grouping of cuckoos based on the discrete fashion of solution representation, generating initial population using the proposed dispatching rules and embedded local search strategy applied to elite solutions enable our algorithm to solve the considered problem more efficiently. Regarding the random parameters of step length and *DELR* in moving toward the best group region and egg laying, besides focusing around the elite solutions as in each iteration of IDCOA, guarantee the diversification and intensification aspects of the search. On the other hand, dynamic controlling of the *DELR* balances these aspects more effectively. Our experiments showed that IDCOA outperforms the other proposed meta-heuristic algorithms for the three-stage assembly flow shop problem. Although its computational time is a little bit more than the compared simple algorithms, it saves considerable computational time due to its improved clustering procedure.

As a direction of future study, one could adopt and apply the proposed algorithm, i.e. IDCOA, to the other scheduling problems,

for instance, flow shop and job shop. Also, investigating the studied problem with different criteria such as minimizing total tardiness could be an interesting research topic. Furthermore, we assumed parallel machines of the first stage are identical but in reality they could be non-identical including either uniform or unrelated machines. Extending the problem to non-identical parallel machines at the first stage could also be investigated.

## References


Al-Anzi, F., & Allahverdi, A. (2013). An artificial immune system heuristic for two-stage multi-machine assembly scheduling problem to minimize total completion time. *Journal of Manufacturing Systems, 32*(4), 825–830.

Allahverdi, A., & Al-Anzi, F. S. (2006a). A PSO and a tabu search heuristics for assembly scheduling problem of the two-stage distributed database application. *Computers and Operations Research, 33*, 1056–1080.

Allahverdi, A., & Al-Anzi, F. S. (2006b). Evolutionary heuristics and an algorithm for the two-stage assembly scheduling problem to minimize makespan with setup times. *International Journal of Production Research, 44*(22), 4713–4735.

Allahverdi, A., & Al-Anzi, F. S. (2008). The two-stage assembly flowshop scheduling problem with bicriteria of makespan and mean completion time. *International Journal of Advanced Manufacturing Technology, 37*(1–2), 166–177.

Allahverdi, A., & Al-Anzi, F. S. (2009). The two-stage assembly scheduling problem to minimize total completion time with setup times. *Computers and Operations Research, 36*, 2740–2747.

Allahverdi, A., & Aydilek, H. (2015). The two stage assembly flowshop scheduling problem to minimize total tardiness. *Journal of Intelligent Manufacturing, 26*(2), 225–237.

Ameryan, M., Akbarzadeh Totonchi, M. R., & Seyyed Mahdavi, S. J. (2014). Clustering based on cuckoo optimization algorithm. In *Intelligent systems (ICIS), 2014 Iranian conference on* (pp. 1–6). IEEE.

Azarbad, M., Ebrahimzadeh, A., & Addeh, J. (2015). A new intelligent approach for recognition of digital satellite signals. *Journal of Signal Processing Systems, 79*(1), 1–14 (2013).

Bhandari, A. K., Singh, V. K., Kumar, A., & Singh, G. K. (2014). Cuckoo search algorithm and wind driven optimization based study of satellite image segmentation for multilevel thresholding using Kapur's entropy. *Expert Systems with Applications, 41*(7), 3538–3560.

García, S., Molina, D., Lozano, M., & Herrera, F. (2007). An experimental study on the use of non-parametric tests for analyzing the behaviour of evolutionary algorithms in optimization problems. In *Proceedings of the Spanish congress on metaheuristics, evolutionary and bioinspired algorithms (MAEB'2007)* (pp. 275–285) (in Spanish).

Grabowski, J., & Pempera, J. (2005). Some local search algorithms for no-wait flow-shop problem with makespan criterion. *Computers & Operations Research, 32*(8), 2197–2212.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Rinnooy, A. H. G. (1979). Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annal Discrete Math, 5*(1), 287–326.

Hanoun, S., Nahavandi, S., Creighton, D., & Kull, H. (2012). Solving a multiobjective job shop scheduling problem using Pareto Archived Cuckoo Search. In *IEEE international conference on emerging technologies and factory automation, ETFA. .* Art. no. 6489617.

Haouari, M., & Daouas, T. (1999). Optimal scheduling of the 3-machine assembly-type flow shop. *RAIRO-Operations Research, 33*(04), 439–445.

Hariri, A. M. A., & Potts, C. N. (1997). A branch and bound algorithm for the two-stage assembly scheduling problem. *European Journal of Operational Research, 103*(3), 547–556.

Hatami, S., Ebrahimnejad, S., Tavakkoli-Moghaddam, R., & Maboudian, Y. (2010). Two meta-heuristics for three-stage assembly flowshop scheduling with sequence-dependent setup times. *International Journal of Advanced Manufacturing Technology, 50*(9–12), 1153–1164.

Kaydani, H., & Mohebbi, A. (2013). A comparison study of using optimization algorithms and artificial neural networks for predicting permeability. *Journal of Petroleum Science and Engineering, 112*, 17–23.

Koulamas, C., & Kyparisis, G. J. (2001). The three-stage assembly flowshop scheduling problem. *Computers and Operations Research, 28*, 689–704.

Lee, C. Y., Cheng, T. C. E., & Lin, B. M. T. (1993). Minimizing the makespan in the 3-machine assembly-type flowshop scheduling problem. *Management Science, 39*, 616–625.

Li, X., & Yin, M. (2013). A hybrid cuckoo search via Lévy flights for the permutation flow shop scheduling problem. *International Journal of Production Research, 51*(16), 4732–4754.

Liu, C., & Ye, C. (2013). Cuckoo search algorithm for the problem of permutation flow shop scheduling, Shanghai Ligong Daxue Xuebao. *Journal of University of Shanghai for Science and Technology, 35*(1), 17–20.

Mahdavi, I., Komaki, G. M., & Kayvanfar, V. M. (2011). Aggregate hybrid flowshop scheduling with assembly operations. In *Proceedings of the Spanish congress on metaheuristics, evolutionary and bioinspired algorithms (MAEB' 2007)* (pp. 663–667). IEEE.

Maleki-Daronkolaei, A., & Seyedi, I. (2013). Taguchi method for three-stage assembly flow shop scheduling problem with blocking and sequence-dependent set up times. *Journal of Engineering Science and Technology, 8*(5), 603–622.

Maleki-Darounkolaei, A., Modiri, M., Tavakkoli-Moghaddam, R., & Seyyedi, I. (2012). A three-stage assembly flow shop scheduling problem with blocking and sequence-dependent set up times. *Journal of Industrial Engineering International, 8*(1), 1–7.

Marichelvam, M. K. (2012). An improved hybrid Cuckoo Search (IHCS) metaheuristics algorithm for permutation flow shop scheduling problems. *International Journal of Bio-Inspired Computation, 4*(4), 200–205.

Marichelvam, M. K., Prabaharan, T., & Yang, X. S. (2014). Improved cuckoo search algorithm for hybrid flow shop scheduling problems to minimize makespan. *Applied Soft Computing Journal, 19*, 93–101.

Naderi, B., Zandieh, M., & Roshanaei, V. (2009). Scheduling hybrid flowshops with sequence dependent setup times to minimize makespan and maximum tardiness. *International Journal of Advanced Manufacturing Technology, 41*(11–12), 1186–1198.

Nie, H., Liu, B., Xie, P., Liu, Z., & Yang, H. (2014). A cuckoo search algorithm for scheduling multi skilled workforce. *Journal of Networks, 9*(5), 1346–1353.

Nowicki, E., & Smutnicki, C. (1996). A fast tabu search algorithm for the permutation flow-shop problem. *European Journal of Operational Research, 91*(1), 160–175.

Ouaarab, A., Ahiod, B., & Yang, X.-S. (2014). Improved and discrete cuckoo search for solving the travelling salesman problem. *Studies in Computational Intelligence, 516*, 63–84.

Ozcelik, B., & Erzurumlu, T. (2006). Comparison of the warpage optimization in the plastic injection molding using ANOVA, neural network model and genetic algorithm. *Journal of Materials Processing Technology, 171*(3), 437–445.

Potts, C. N., Sevast'Janov, S. V., Strusevich, V. A., Van Wassenhove, L. N., & Zwaneveld, C. M. (1995). The two-stage assembly scheduling problem: Complexity and approximation. *Operations Research, 43*, 346–355.

Rajabioun, R. (2011). Cuckoo optimization algorithm. *Applied Soft Computing Journal, 11*(8), 5508–5518.

Rock, H., & Schmidt, G. (1983). Machine aggregation heuristics in shop scheduling. *Mathematical Methods of Operations Research, 45*, 303–314.

Shahdi-Pashaki, S., Teymourian, E., Kayvanfar, V., Komaki, G. H. M., & Sajadi, A. (2015). Group technology-based model and cuckoo optimization algorithm for resource allocation in cloud computing. *IFAC-PapersOnLine, 48*(3), 1140–1145.

Shahdi-Pashaki, S., Teymourian, E., & Tavakkoli-Moghaddam, R. (2016). New approach based on group technology for the consolidation problem in cloud computing-mathematical model and genetic algorithm. *Computational and Applied Mathematics*, 1–26. http://dx.doi.org/10.1007/s40314-016-0362-4.

Shah-Hosseini, H. (2009). The intelligent water drops algorithm: A nature-inspired swarm-based optimization algorithm. *International Journal of Bio-Inspired Computation, 1*(1), 71–79.

Shahvari, O., & Logendran, R. (2015). Bi-criteria batch scheduling on unrelated-parallel machines. In *Proceedings of the 2015 industrial and systems engineering research conference (ISERC2015), Tennessee, USA.*

Shahvari, O., & Logendran, R. (2016). Bi-criteria batch scheduling in hybrid flow shop. In *Proceedings of the 2016 industrial and systems engineering research conference (ISERC2016), California, USA.*

Shahvari, O., & Logendran, R. (2016). Hybrid flow shop batching and scheduling with a bi-criteria objective. *International Journal of Production Economics, 179*, 239–258.

Shahvari, O., & Logendran, R. (2017). An enhanced tabu search algorithm to minimize a bi-criteria objective in batching and scheduling problems on unrelated-parallel machines with desired lower bounds on batch sizes. *Computers and Operations Research, 77*, 154–176.

Shahvari, O., Salmasi, N., Logendran, R., & Abbasi, B. (2012). An efficient tabu search algorithm for flexible flow shop sequence-dependent group scheduling problems. *International Journal of Production Research, 50*(15), 4237–4254.

Sun, X., Morizawa, K., & Nagasawa, H. (2003). Powerful heuristics to minimize makespan in fixed, 3-machine, assembly-type flowshop scheduling. *European Journal of Operational Research, 146*, 498–516.

Sung, C. S., & Kim, A. H. (2008). A two-stage multiple-machine assembly scheduling problem forminimizing sum of completion times. *International Journal of Production Economics, 113*, 1038–1048.

Teoh, C. K., Wibowo, A., & Ngadiman, M. S. (2014). An adapted cuckoo optimization algorithm and genetic algorithm approach to the university course timetabling problem. *International Journal of Computational Intelligence and Applications, 13*(01).

Torabzadeh, E., & Zandieh, M. (2010). Cloud theory-based simulated annealing approach for scheduling in the two-stage assembly flowshop. *Advances in Engineering Software, 41*, 1238–1243.

Tozkapan, A., Kırca, Ö., & Chung, C. S. (2003). A branch and bound algorithm to minimize the total weighted flowtime for the two-stage assembly scheduling problem. *Computers and Operations Research, 30*(2), 309–320.

Yang, X. S. (2010). A new metaheuristic bat-inspired algorithm. In C. Cruz, J. R. Gonźalez, D. A. Pelta, & G. Terrazas (Eds.), *Nature inspired cooperative strategies for optimization (NISCO 20 10). Studies in computational intelligence* (Vol. 284, pp. 65–74). Berlin: Springer.

Yang, X. S., & Deb, S. (2010). Engineering optimisation by cuckoo search. *International Journal of Mathematical Modelling and Numerical Optimisation, 1*(4), 330–343.

Yin, Y., & Yasuda, K. (2006). Similarity coefficient methods applied to the cell formation problem: A taxonomy and review. *International Journal of Production Economics, 101*(2), 329–352.