Cuckoo Search Algorithm for Hybrid Flow Shop Scheduling Problem with Multi-layer Assembly Operations

Mohamed Komaki
Department of Electrical Engineering and Computer Science
Case Western Reserve University, USA
Cleveland, Ohio 44106, USA
gxk152@case.edu

S. Sheikh
Management Science Department
New York Institute of Technology
1855 Broadway, New York, NY 10023
shaya.sheikh@case.edu

Ehsan Teymourian
Department of Industrial Engineering
Mazandaran University of Science and Technology
Babol, Iran
ehsan.teymorian@gmail.com

Behnam Malakooti
Department of Electrical Engineering and Computer Science
Case Western Reserve University, USA
Cleveland, Ohio 44106, USA
bxm4@case.edu

Abstract

Most of studies assume that jobs are simple string type and independent, while in the real-world situations almost all of the final products have several components and they are produced by assembling components together in different layers of subassemblies. In spite this fact, the studies addressing assembly operations are relatively less in the literature. This study considered the hybrid flowshop scheduling problem followed by assembly stage with identical parallel machines. Frist, components of products are processed in the hybrid flowshop, and then they go under assembly operations based on the predefined products assembly structure. Each product may have several assembly operations and final product is obtained by completion of its last assembly operation. The goal is to find a schedule, sequence of products and parts, which minimizes completion time of the last product, i.e., makespan. For this end, a new mathematical model is proposed; on the other hand, since the studied problem is NP-hard, a nature inspired meta-heuristic method, Cuckoo Search (CS) algorithm, is then employed such that it can handle the precedence constraints among parts and assembly operations. To evaluate the performance of the proposed algorithm, the computational results are demonstrated.

Keywords
Hybrid flow shop, multi-level assembly operations, complex products, Cuckoo Search Algorithm.

1. Introduction

In this paper Hybrid Flow Shop (HFS) scheduling problem followed by assembly stage is addressed. In this study, final products have several components which should be processed at the HFS then the completed components based on the hierarchy assembly structure of each product will go under assembly operation. HFS is well defined production environment and numerous studies have addressed it. In this paper, HFS has S stages where at each stage there are M_S parallel identical machines which never break down and are available through scheduling horizon. The last stage, stage S+1 or assembly stage, has M_{S+1} identical parallel machines which assemble the parts based on the assembly structure of the products. The completion time of each product is equal to the completion time of its last assembly operation at the assembly stage. The goal is finding the sequence of products and their components such that the makespan is minimized. This type of production system is suitable to produce high volume of customized products such as personal computers (Potts *et al.*, 1995), TV (Mahdavi et al., 2011), food, and so on. This type of production will reduce production cost due to ability to produce high volume of products in short time.

The simple version of this problem is as following, the first stage which is the fabrication stage has m parallel identical machines and the second stage is the assembly stage where there is a single assembly machine. All products have only one assembly operation. This type of problem for the first time is introduced by Lee *et al.* (1993) where the first stage has two machines and they showed minimizing the makespan of this problem is NP-Hard. They for this problem proposed Branch and Bound (B&B) algorithm as well as heuristic algorithms based on the extension of Johnson's algorithm. Later, Potts *et al.* (1995) extended this problem to the general case where the fabrication stage has m parallel identical machines and developed a heuristic algorithm. Since then, this type of problem has been investigated by many researchers and some of the proposed methods are Tabu Search and Simulated Annealing (Al-Anzi and Allahverdi, 2006), and so on.

Another version of assembly scheduling problems is agile manufacturing systems where there are two stages and similar to the above system the first stage is fabrication stage and the second stage is assembly stage, but in agile manufacturing system the products are complex and they have several assembly operations. Minimizing makespan of agile manufacturing system where both stages have single machine introduced by Kusiak (1989) and showed that MLDF (Maximum Level of Depth First) rule provide the optimal solution for this system. Then He et al. (2001) extend this problem to the case that the first stage has m identical parallel machines and developed a lower bound and a heuristic algorithm based on extension of the MLDF rule. Later, He and Babayan (2001) investigated another version of agile manufacturing system where the first stage has single machine and the assembly stage has m parallel identical machines and they showed that MLDF still can provide optimal solution for this problem. Since then other investigated agile manufacturing systems and proposed Genetic Algorithm (Gaafar and Masoud, 2005), Particle Swarm-based GA (Gaafar et al., 2009), and Ant Colony Optimization (Liao and Liao, 2008).

Another version of assembly operations scheduling problems is the general case where the production (fabrication) stage is (hybrid) flow shop and the assembly stage can have single or parallel (identical) machines, and products can be simple (only one assembly operation) or can be complex and have several assembly operation. For instance, Mahdavi et al. (2011) studied the hybrid flow shop followed by single assembly machine and complex products.

In this study, we extend Mahdavi et al.'s (2011) study where the assembly stage has parallel identical assembly machines. A mathematical model as well as a Lower Bound for this problem as presented. Since the problem is NP-hard, we propose Cuckoo Search Algorithm (CS) which is inspired from behavior of some species of cuckoos. CS developed by Yang and Deb (2009) and since then it has been applied to many combinatorial optimization problems such as Knapsack problem (Layeb, 2011), TSP (Ouaarab et al., 2014), scheduling problems such as permutation flow shop (Marichelvam, 2012), and parallel machines (Guo et al., 2013). The result was impressive, for instance, CS showed better performance than Particle Swarm Optimization (PSO) and GA on some well-known benchmark functions (Fister et al., 2014). Comparing to other metaheuristic algorithms, CS has less parameters, and also it is simple and effective. Ouaarab et al. (2011) developed discreet CS for TSP and the results showed that it outperforms the stat-of-the-are. Also, most of the meta-heuristic algorithm may premature converge to local optima while CS usually converge to the global optimum, moreover, search mechanism of CS is capable to do local and global search. Another feature of CS is Lévy flights or process which enables CS to search solution space more efficiently.

Due to simplicity and effectiveness of the CS, a lot of variations (such as discreet, discreet binary, modified, parallel and so on) and hybridization of this algorithm has been proposed.

The rest of the paper is organized as follows. The next section is devoted to the problem definition and Section 3 presents the mathematical model of the problem as well as numerical example. Then Section 4 describes the proposed Cuckoo Search and in Section 5 the computational results are presented. Section 6 is devoted to the conclusion and future work.

2. Problem Description

Before presenting the detail of the problem, the following definition in needed to be introduced.

1. Assembly level: Assembly level of the final assembly operation of each product is 1 and going downward to the leaf of the product it increases by one unit (Mahdavi et al., 2011). Note that the assembly level of each part (component) is the same as the assembly level of the assembly operation. See Fig. 1 in which five products are shown, and assembly level of Product 1 is 1, assembly level of Products 2 and 3 are 2, and assembly level of Products 3 and 5 are 3. In this study, the assembly levels closer to leaves of the products are called higher assembly level and those levels close to the final assembly operation are called lower assembly level.

2. Simple/Complex product: Simple product is a product which as single assembly operation at each assembly level and obviously the complex product is a product which is not simple (Kusiak, 1989). In Fig. 1, Products 1,2, and 3 are simple and Products 4 and 5 are complex.

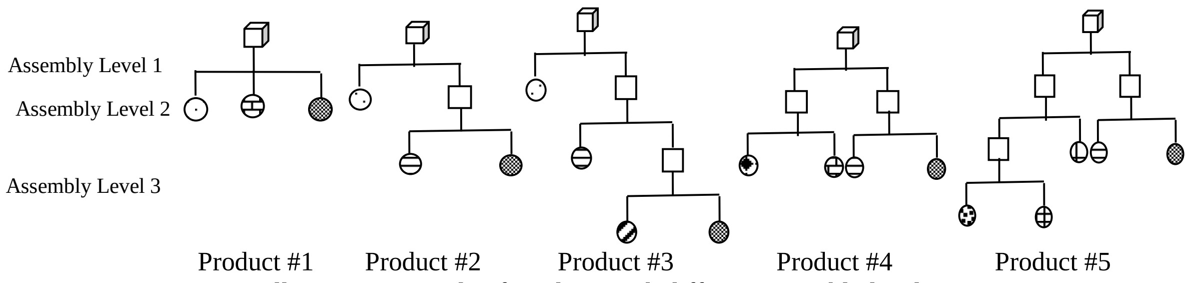

Figure 1: Illustrative example of products with different assembly levels

In this problem, Hybrid flow shop (HFS) is followed by assembly stage where the assembly stage has m parallel identical machines. HFS has S stage and at each stage $t \in \{1,2,\ldots,S\}$ there are M_t parallel identical machines. Also, there are H products where the assembly structure and bill of material (BOM) of each product is known in advance. Components of products need to be processed at the stages of HFS first, and then when all components of an assembly operation are ready, they should go under the assembly operation at the assembly stage. This process continues until all assembly operations are processed. Note that the completion time of each product is completion time of the last assembly operation which is in the assembly level 1. The goal is finding the sequence of products and their components such that the completion time of the last product is minimized.
It has been shown that minimizing makespan in the hybrid flow shop is NP-hard; therefore, the studied problem is NP-hard due to more constraints. Note that in this problem, the precedence relation among parts needs to be considered, i.e., in order to start processing of assembly operation, the corresponding parts of the assembly operation should be ready.

3. Mathematical Model

There are H products with the same BOM structure, which is known in advance. Also, number of stages of HFS (S), number of machines of each stage (M_t), and processing (assembly) times are known in advance. Decision variables are determining the components (parts) sequence in HFS stages, and assembly operations sequence in the assembly stage, in a way that minimize C_{max}.

3.1 Assumptions

Our assumptions are as following. There are S+1-stages where the first S stages are HFS, and stage S+1 is the assembly stage. Each stage has M_t parallel identical machines which never breakdown and are available throughout the scheduling period. Each machine can process one part (assembly operation) at a time and each part (assembly operation) can be processed by one machine at a time, pre-emption is also not allowed and the capacity of buffers between stages are unlimited. BOM of products are known, and the components of products are available at time zero. The completion time of each product is equals to the completion time of the assembly operation at the assembly level 1. Note that the assembly operation (SUA) of each product is numerated from maximum assembly level to the assembly level 1(see Fig. 2), e.g., SUA_{21} represents the first assembly operation of Product #2, and similarly SUA_{22} is second assembly operation of Product #2.
In this problem, the processing of parts in HFS stages as well as assembly stage will be based on first available machine (FAM) rule, i.e., when processing a part is completed, it will be processed by the FAM rule in the next stage. Note that at the assembly stage precedence relations need to be considered, i.e., when processing a part at stage S is completed, associated assembly operation cannot be started until all parts of the assembly operation as well as those assembly operations which are considered as component (child) of the current assembly operation are ready.

3.2 Notation

- Subscripts

H	Total number of products
N	Total number of parts
S+1	Total number of stages
p,*h*,*f*	Product index (p,h,f = 1,2,…H)

i,j,d	Part index (i,j,d = 1,2,…N)
t	Stage index (t = 1, 2,…S+1)
m	Machine index in HFS stages (m =1, 2, …,M_t)
k	Machine index in assembly stage (k =1, 2, …,M_{S+1})
I_P	Total number of subassembly operations for product p
g,r,e	Index for subassembly operations of product p (g,r,e = 1,2,…,I_p)
Q_{pr}	The set of parts belonged to the *r*th assembly operation of product p
J_{pr}	The set of assembly operations which are prerequisite of the *r*th assembly operation of product p

- *Input Parameters*

P_i^t	Processing time of part i on stage t, $\forall i \epsilon \{1,..,N\}$, $t \in \{1,2,\ldots,S\}$
q_{pr}	Processing time of rth subassembly of product p at assembly stage (S+1), $\forall p \epsilon \{1,..,H\}, \forall r \epsilon \{1,..,I_p\}$
M_t	Number of machines at stage t; $t \in \{1,2,\ldots,S,S+1\}$
M	A big positive arbitrary number

- *Decision variables*

C_i^t	The completion time of part i at stage t, $\forall i \epsilon \{1,..,N\}$, $t \in \{1,2,\ldots,S\}$
Co_{pr}	The completion time of rth assembly operation of product p, $\forall p \epsilon \{1,..,H\}, \forall r \epsilon \{1,..,I_p\}$
Com_p	The completion time of product p, , $\forall p \epsilon \{1,..,H\}$
x_{ijm}^t	1 if part j is processed exactly after part i on machine m at stage t, $\forall i,j \epsilon \{1,..,N\}, \forall t \epsilon \{1,\ldots,S\}, \forall m \in \{1,..,M_t\}$ (HFS); 0 otherwise
y_{ijm}^t	1 if processing of part i precedes part j on machine m at stage t, $\forall i,j \epsilon \{1,..,N\}, \forall t \epsilon \{1,\ldots,S\}, \forall m \in \{1,..,M_t\}$ (HFS); 0 otherwise
w_{im}^t	1 if part i is assigned to machine m at stage t, $\forall i \epsilon \{1,..,N\}, \forall t \epsilon \{1,\ldots,S\}, \forall m \in \{1,..,M_t\}$ (HFS); 0 otherwise
l_{hgprk}	1 if rth assembly operation (layer) of product p exactly after gth assembly operation of product h is processed on kth machine at assembly stage (stage S+1), $\forall k \{1,..,M_{S+1}\}, \forall p,h \epsilon \{1,..,H\}, \forall g \epsilon \{0,..,I_h\}, \forall r \epsilon \{0,..,I_p\}$; 0 otherwise
C_{Max}	The maximum completion time of the AHFS system

Min C_{Max} (1)
Subject to:

$\sum_{j=1}^{N} x_{ijm}^t = \sum_{d=1}^{N} x_{dim}^t$ $\quad \forall i \epsilon \{1,..,N\}, \forall t \epsilon \{1,\ldots,S\}, \forall m \in \{1,..,M_t\}$ (2)
$\sum_{j=1}^{N} x_{ijm}^t = w_{im}^t$ $\quad \forall i \epsilon \{1,..,N\}, \forall t \epsilon \{1,\ldots,S\}, \forall m \in \{1,..,M_t\}$ (3)
$\sum_{m=1}^{m_t} w_{im}^t = 1$ $\quad \forall i \epsilon \{1,..,N\}, \forall t \epsilon \{1,\ldots,S\}$ (4)
$\sum_{i=1}^{N} x_{ijm}^t = 1$ $\quad \forall j = 0, \forall t \epsilon \{1,\ldots,S\}, \forall m \in \{1,..,M_t\}$ (5)
$\sum_{j=1}^{N} x_{ijm}^t = 1$ $\quad \forall i = 0, \forall t \epsilon \{1,\ldots,S\}, \forall m \in \{1,..,M_t\}$ (6)
$x_{ijm}^t \le y_{ijm}^t$ $\quad \forall i,j \epsilon \{1,..,N\}, \forall t \epsilon \{1,\ldots,S\}, \forall m \in \{1,..,M_t\}$ (7)
$y_{ijm}^t + y_{jim}^t = w_{im}^t \, . \, w_{jm}^t$ $\quad \forall i,j \epsilon \{1,..,N\}, \forall t \epsilon \{1,\ldots,S\}, \forall m \in \{1,..,M_t\}$ (8)
$y_{ijm}^t + y_{jdm}^t = y_{idm}^t + 1$ $\quad \forall i,j,d \epsilon \{1,..,N\}, \forall t \epsilon \{1,\ldots,S\}, \forall m \in \{1,..,M_t\}$ (9)
$C_i^t + (\sum_{m=1}^{m_t} y_{ijm}^t - 1).M \le C_j^t - P_j^t$ $\quad \forall i,j \epsilon \{1,..,N\}, \forall t \epsilon \{1,\ldots,S\}$ (10)
$\sum_{g=0}^{I_h} \sum_{m=1}^{M_{S+1}} l_{hgprk} = 1$ $\quad \forall p,h \epsilon \{1,..,H\}, \forall r \epsilon \{1,..,I_p\}$ (11)
$\sum_{r=0}^{I_p} l_{hgprk} \le 1$ $\quad \forall p,h \epsilon \{1,..,H\}, \forall g \epsilon \{1,..,I_h\}, \forall k \in \{1,..,M_{S+1}\}$ (12)
$\sum_{g=0}^{I_h} l_{hgfek} - \sum_{r=0}^{I_p} l_{feprk} = 0$ $\quad \forall p,h,f \epsilon \{1,..,H\}, \forall e \epsilon \{1,..,I_f\}, \forall k \in \{1,..,M_{S+1}\}$ (13)
$Co_{hg} + \left(\sum_{m=1}^{M_{S+1}} l_{hgprk} - 1\right).M \le Co_{pr} - q_{pr}$ $\quad \forall p,h \epsilon \{1,..,H\}, \forall g \epsilon \{1,..,I_h\}, \forall r \epsilon \{1,..,I_p\}$ (14)
$C_i^S + q_{pr} \le Co_{pr}$ $\quad \forall p \epsilon \{1,..,H\}, \forall r \epsilon \{1,..,I_p\}, \forall i \epsilon Q_{pr}$ (15)
$Co_{pe} + q_{pr} \le Co_{pr}$ $\quad \forall p \epsilon \{1,..,H\}, \forall r \epsilon \{1,..,I_p\}, \forall e \epsilon J_{pr}$ (16)
$Co_{pr} \le Com_p$ $\quad \forall p \epsilon \{1,..,H\}, \forall r \epsilon \{1,..,I_p\}$ (17)
$Com_p \le Cmax$ $\quad \forall p \epsilon \{1,..,H\}$ (18)
$x_{ijm}^t, y_{ijm}^t, w_{im}^t, l_{hgprk} \in \{0,1\}$ $\quad \forall i,j \epsilon \{1,..,N\}, \forall t \epsilon \{1,\ldots,S\}, \forall m \in$ (19)

$$\{1,..,M_t\}, \forall k\{1,..,M_{S+1}\}, \forall p, h\epsilon\{1,..,H\}$$
$$, \forall g\epsilon\{1,..,I_h\}, \forall r\epsilon\{1,..,I_p\}$$

Constraint (1) establishes the objective function, minimization of the makespan for the AHFS system. In the above proposed mathematical model, constraints (2-10) are related to the HFS stages, whereas constraints (11-18) take care of sequence of assembly operations at the assembly stage.

Constraints (2-4) indicate the assignment of parts to corresponding machines, where constraints (2) and (3) ensure that each part has one prior and one next part in its queue on the machine which processes them. Also, constraint (4) guarantees that each part will be processed only on one machine in each stage. Constraints (5) and (6) like constraints (2) and (3) are the flow conservation constraints and confirm that virtual part (i=0) is visited exactly once in each machine queue. Constraints (7-9) seek to make the precedence relationship between each pair of parts i and j assigned to a machine m in each stage S. Inequalities (7) impose the fact that if the edge (i,j) is identified on the queue of machine m, then j follows i. Constraints (8) ensure that between each pair of parts i and j assigned to a same machine m in a considered stage S, only one of them is processed before the other one. Transitivity constraints are enforced in constraints set (9), which state that if part i is visited before j and d is visited after j on machine, so part d must be visited after i. Constraints (10) guarantee that processing start time of part j must be greater than or equal to the completion time of each part i which precedes part j (on machine m) on that stage.

Constraints (11), (12), and (13) ensure that each subassembly operation is processed precisely once at assembly stage. In particular, constraints (11) guarantee that for each subassembly J_{pr}, there is a unique machine such that J_{pr} is processed either first or after another operation (J_{hg}) on that machine. The inequalities (12) imply that at stage S+1, there is maximum one machine for each subassembly, where it has a successor or is processed at last. Ultimately, for each subassembly, there is one and only one machine satisfying both of the previous two conditions by (13).

Constraints (14-16) take care of the completion times of the subassemblies at stage S+1 (assembly stage). Constraints (14) guarantee that processing start time of subassembly p of product r must be grater or equal than completion time of each subassembly g of product h which precedes it. Constraints (15) and (16) enforce that each subassembly completion time must be greater than its pertaining components and subassemblies completion times in BOM structure plus its processing time. Completion time of each product is calculated in constraints (17) and is equal to the completion time of its last assembly operation. Constraints (18) likewise compute the completion time of whole system (AHFS) which is equal to the last product completion time. Finally constraints (19) identify the variables domain in the presented model as full binary.

3.3 Numerical Example

Suppose there are three products which their BOM which are similar to Product #2 in Fig. 1, see Fig. 2. Also, suppose there is two-stage HFS where each stage has two machines as well as the assembly stage, i.e., M_{11} and M_{12} are processing machines at the first stage, M_{21} and M_{22} are processing machines at the second stage, and ASM_1 and ASM_2 are assembly machines. The processing and assembly times are given in Table 1.

In Fig. 3, the optimal solution of this numerical example using CPELX Version 11.1 is obtained. As can be seen, the completion time of SUA_{21} and SUA_{22} are 26 and 56, respectively; therefore, completion time of Product 2 is 56. Also, the completion time of SUA_{11} and SUA_{12} are 41 and 56, and completion time of Product 1 is 56. Makespan (C_{max}) of the optimal schedule is 56.

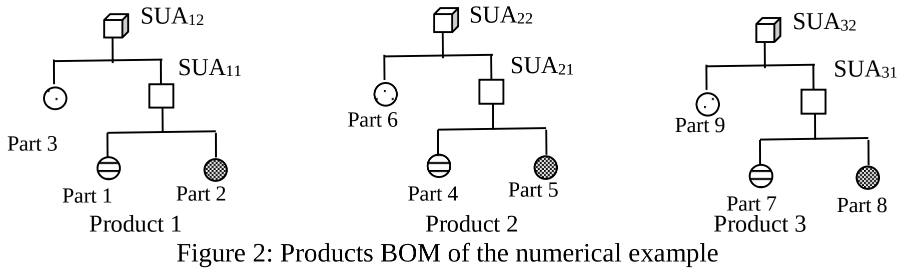

Figure 2: Products BOM of the numerical example

As can be seen from Fig. 3, parts in the higher assembly level of each product are processed earlier than other parts of the product, for example, parts 4 and 5 which belong to SUA_{21} are processed before 6.

Table 1: Processing time of parts and assembly operations

Part	1	2	3	4	5	6	7	8	9
Stage 1	5	7	10	8	6	9	10	5	9

Stage 2	8	6	7	5	6	8	5	7	6
Assembly operation	SUA_{11}	SUA_{12}	SUA_{21}	SUA_{22}	SUA_{31}	SUA_{32}			
Assembly time	12	15	13	11	14	10			

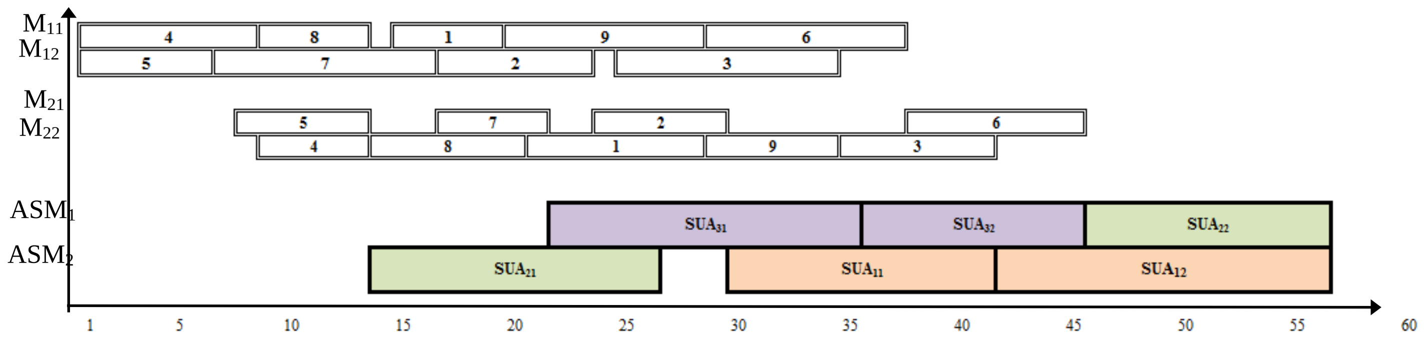

Figure 3: Gantt-chart of the optimal solution of the numerical example

4. Cuckoo Search Algorithm

Cuckoo Search (CS) Algorithm, developed by Yang and Deb (2009), is inspired from behaviour of cuckoo birds. Some species of cuckoos do not grow up and feed their chicks; instead they leave their eggs in nest of other host birds. Some birds will identify these eggs and they may through it away, while others may abandon their nests, and some other eggs will not be identified by the hosts. The growth rate of the cuckoos is extremely high, and they eat all of the provided food by the host such that the host’s own checks will die due to starving. After hatching, growing up and becoming mature bird, cuckoos leave the host’s nest and lay their own egg in other nests. They put their egg in host nests which has access to the higher level of food source. This process repeats until the stopping criterion is met.

Other basic assumption of CS is as following; each cuckoo can lay one egg at a time, the total number of host nest is fixed, the best nests with high quality will be used in the next generation (or equivalently iteration), and P_α% of eggs will be identified by hosts. In Fig. 4, the steps of the basic CS are presented.

```
Initial Population of n host nests (i=1,2,…,$N_{max}$)
While Stopping criterion is not met do
  Generate a cuckoo i randomly by Levy flight (i=1,2,…,$N_{max}$)
  Choose a nest j randomly among $N_{max}$ nests
  If quality of generate cuckoo i is better that nest j
    Replace nest j by cuckoo i
  End if
  Kill $P_\alpha$ of worse nests
  Generate new nests by levy flight
  Keep the best solutions
End while
```

Figure 4: Pseudo-code of the basic CS

In the following, the steps of the proposed CS are discussed in detail.

4.1 Initial Population

In order to apply CS to the problem at hand, some preliminary explanations are needed. First of all, each egg in the nest and new egg (or chicks) represent a solution. Therefore, nest and egg both represent the solution of the problem. In this study, due to precedence constraint between parts and assembly operations, i.e., precedence among Q_{pr}, J_{pr} and rth assembly operation, we develop a novel solution representation scheme which is called block representation scheme. In this representation, parts of each assembly operation, Q_{pr}, are defined as a block. For instance, the optimal solution of the numerical example shown in Fig. 3 can be presented as following figure. As can be seen in Fig. 5, each block is shown with different colours and in total there are 6 blocks which is equals to the number of assembly operations in Fig.2. Also, in this representation, the parts which belong to the higher assembly levels have been processed before parts in the lower assembly level, for instance, parts 7 and 8 are processed before part 9.

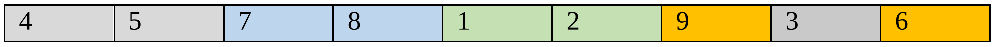

Figure 5: An example of solution representation scheme

In most of population based metaheuristic algorithms the initial solutions are generated randomly but as indicated by many researchers, the quality of initial population has undeniable effect on the quality of final solution as well as faster convergence of the algorithm. In this study, we have tested several rules and it turns out that in order to generate initial population, the following procedure yields better performance.

1-Consider only assembly operations in the highest assembly level, assign a random number between 0 and 1 to those assembly operations and sort them in non-decreasing order. Consider this sequence of assembly operations as partial sequence of assembly operations.
2-To find order of part of each assembly operation, assign a random number to each part of the each assembly operation and sort them in non-decreasing order, and put the parts in the partial sequence of assembly operations obtained in the Step1.
3-Repeat Steps 1 and 2 until sequence of all assembly operations and parts are determined.

In the above procedure, the solution is generated completely randomly. One solution can be obtained by modifying Shortest Processing Time (SPT) for complex products (Mahdavi et al., 2011), i.e., the order of parts and assembly operations are determined based on shortest processing and assembly times and the order of blocks are determined based on the total processing times of its part. Also, the well-known NEH algorithm can be applied to obtain a better solution. Then η% of N_{max} can be obtained by perturbing the second solution. In this study, the perturbation is defined inner-block and outer-block movement which are explained in the next subsection. The rest of initial populations, N_{max}-η%-2, are generated randomly as by the previously explained procedure.
Fitness of each individual is defined as $-C_{max}$ where C_{max} is makespan of the individual. Suppose N_{max} be the number of initial sequences, therefore, the above procedure needs to be repeated N_{max} times. Also, we set It_{Max}, maximum number of iteration of the algorithm, as stopping criterion.

4.2 Levy Flight

In the basic CS, levy flight of cuckoo i is defined as following.

$$x_i^{(t+1)} = x_i^{(t)} + \alpha \oplus \text{Levy}(\lambda) \qquad (22)$$

where $x_i^{(t)}$ and $x_i^{(t+1)}$ represent the position of the cuckoo before and after applying levy flights, respectively, also $\alpha>0$ is step size (and should be determined based on the characteristics of the problem at the hand), and $\oplus$is entry-wise multiplication. Since the above representation is based on continues search space, the levy flight for the studied problem needs some adjustments. Before presenting the levy flight some definitions and explanations need to be introduced.

1. Neighbourhood Structure:

Consider Fig. 5 which represents a schematic solution representation of the problem. In scheduling problems such as permutation flow shop scheduling, there are two main neighbourhood search strategies, reinserting and swapping. In reinserting, some jobs are randomly selected and reinserted in randomly chosen positions while in swapping the position of selected jobs are switched. It is accepted that reinsertion movements yield higher performance than swap movements. Therefore, in this problem we apply reinsertion strategy to generate a neighbour of the current solution. The reinsertion can be defined in two levels, inner-block and outer-block movements; inner-block reinsertion refers to selecting parts from one block and changing their positions by applying reinsertion of it inside of the block while outer-block movement is applying reinsertion on the blocks, i.e., some blocks are randomly selected and reinserted in random positions. Note that in this case, the precedence relations need to be considered. For instance, the following figure represents inner and outer-block movements. In Fig. 6.a, the position of parts 7 and 8 has changed and in Fig. 6.b the block representing SUA_{21} has been reinserted after SUA_{32} which this movement is valid one since the precedence structure of the products are not violated.

2. The step of movement

Usually the step of a movement is defined as distance between two solutions. In this study, we also follow this definition of step. As an example, consider Fig. 6.a and Fig. 5 where the Fig. 6.a is obtained by reinserting Part 7 after Part 8, therefore, in this case the distance of these two solution is 1, since by one movement can be achieved one form other one. Now, consider Fig. 5 and Fig. 6.b, in this case it is needed to perform two movements to get one solution from other one, therefor, their distances from each other is two.

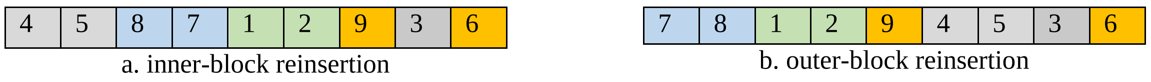

Figure 6: An example of outer-block and inner-block reinsertion

Levy flight in CS is resemble to intensification procedure which extensively searches neighbour of the current solution and sometimes have big steps. In order to apply levy flight in this study, we suggest the following procedure. *Generate a random number x, if x<0.5, perform outer-block movement, otherwise perform inner-block movement.*

4.3 Kill P_a% of new eggs

As mention earlier, at each iteration of the algorithm P_α% of new eggs will be killed which most of them have low quality or low fitness. So, after applying levy flights and generating the new eggs eliminate (kill) those solutions (eggs and nests) which have low fitness.

4.4 Keep the best eggs and nest

Sort the population based on their fitness ($-C_{max}$), and select the best N_{max} of them. If the stopping criterion is met stop, otherwise continue.

5. Computational Experiments

5.1 Test problems

The performance of the proposed LB and CS algorithm are evaluated through randomly generated instances. In order to generate instances the following information is needed; number of products (H), number of stages of HFS(S), number of machines at assembly stage (M_{S+1}), processing and assembly times, and assembly structure of products. We consider H={10,30,50},S={2,3,4}, the number of machines are {2,3,4}, processing and assembly times have uniform distributions as presented in table 2, and finally, the assembly structure of products are the same as presented in the Fig. 1. In total, 1215 instances are generated.

Table 2: Processing time and assembly operations

Problem Type	Processing time of parts	Assembly times
PT1	U(0,100)	U(0,50)
PT2	U(0,20)	U(0,50)
PT3	U(0,100)	U(0,20)

We compare CS with SA (Mahdavi et al., 2011), and due to random nature of meta-heuristic algorithm we run each algorithm five times. All algorithms are coded in MATLAB R2013a and run on PC Intel (R) Core ™2Duo 2.66 GHz and 4GB RAM.
As mentioned in Section 4, CS has few parameters to set, and based on initial experiments, we set the parameters as following; It_{Max}=400, P_a=0.2, N_{max}=H, and η=0.5.

5.2 Experimental Result

In order to measure the performance of the algorithms we use Relative Deviation Percentage (RDP) which is calculated as below.

$$RDP=100(Alg_{Sol}- Alg_{Sol*})/ Alg_{Sol*} \qquad (23)$$

where Alg_{Sol} and Alg_{Sol*} represent the solution of the algorithms and the best solution found, respectively.
In the following tables, the summary of results is presented where Table 3.a represents the RDP of algorithms with respect to types of products, as can be seen as structure of the products become more complex, the RDP increases where in all cases RDP of CS is lower than SA which indicates that CS has better performance than SA. In Table 3.b RDP of algorithms based on number of products are presented, as a general conclusion, as number of products increases, the RDP of both algorithms increase while in all cases CS outperforms SA. Table 3.c and Table 3.d represent the RDP of algorithms based on number of stage and machines, respectively. As can be seen in Table 3.c, as number of stages increase, the RDP of algorithms increases, and for S=2 RDP of CS is 3.47% which represent the excellent performance of CS. Table 3.d presents the RDP of algorithms based on number of machines which the behaviour of algorithms is almost the same as discussed pattern based on number of stages. In Table 3.e, RDP based on problem types is presented, surprising performance of CS for PT2 is better than other problem types while SA has better performance in PT1, and the worst performance of both algorithms is for PT3, however, the performance

of CS is a lot better than SA. In total RDP of CS and SA are 6.6% and 12.2% respectively, therefore, CS outperforms SA.

Table 3: RDP of algorithms based on different characteristics

a.RDP based on BOM of Products

BOM	#1	#2	#3	#4	#5
SA	11.73	12.08	11.79	12.21	13.14
CS	4.27	6.35	6.41	7.31	8.69

b.RDP based on number of products

H	10	30	50
SA	10.32	12.39	13.88
CS	5.83	6.32	7.67

c. RDP based on number of stages

S	2	3	4
SA	9.45	12.29	14.76
CS	3.47	6.93	9.42

d. RDP based on number of machines

m	2	3	4
SA	10.31	12.05	14.23
CS	5.43	5.64	8.69

e. RDP based on problem type

PT	PT1	PT2	PT3
SA	11.21	12.09	13.13
CS	6.42	5.84	7.69

6. Conclusion

This paper dealt with Hybrid flow shop (HFS) with assembly operations where products have complex assembly structure. In this study, there is S-stage HFS where each stage has identical parallel machines and machines never breakdown. Also, assembly stage has identical parallel machines. Components of assembly operations first need to be processed at S stages of HFS and then based on products assembly structure, the processed components can be assembled together at the assembly stage by one of the parallel machine with respect to their hierarchy structure. The completion time of each product is completion time of the last assembly operation of each product. Goal is finding sequence of products, assembly operations, and parts such that makespan is minimized.

For this problem, a mathematical model, and Cuckoo Search algorithm, bio-inspired meta-heuristic algorithm, are proposed. In the proposed CS, the presentation of the solution is novel which each assembly operation is presented as block and then the precedence relation of parts and assembly operations are considered. The computational experiments showed that CS clearly dominates SA in all cases, it term of both computational time and quality.

References

Al-Anzi, F., & Allahverdi, A, A hybrid tabu search heuristic for the two-stage assembly scheduling problem. International Journal of Operations Research, 3, pp109-119,2006.

Fister Jr, I., Yang, X. S., Fister, D., & Fister, I. Cuckoo search: A brief literature review. In *Cuckoo search and firefly algorithm* (pp. 49-62). Springer International Publishing, 2014..

Gaafar, L. K., & Masoud, S. A., Genetic algorithms and simulated annealing for scheduling in agile manufacturing. International Journal of Production Research, 43(14),pp 3069-3085,2005.

Gaafar, L. K., Masoud, S. A., & Nassef, A. O., A particle swarm-based genetic algorithm for scheduling in an agile environment. Computers & Industrial Engineering, 55(3), pp 707-720, 2008.

Guo, P., Cheng, W., & Wang, Y. Parallel machine scheduling with step deteriorating jobs and setup times by a hybrid discrete cuckoo search algorithm. arXiv preprint arXiv:1309.1453,2013.

He, D., & Babayan, A., Scheduling manufacturing systems for delayed product differentiation in agile manufacturing. International Journal of Production Research, 40(11), pp 2461-2481,2002.

He, D., Babayan, A., & Kusiak, A., Scheduling manufacturing systems in an agile environment. Robotics and Computer-Integrated Manufacturing, 17(1),pp 87-97,2001.

Kusiak, A., Aggregate scheduling of a flexible machining and assembly system. IEEE Transactions on Robotics and Automation, 5, 451 – 459,1989.

Layeb, A., A novel quantum inspired cuckoo search for knapsack problems. *International Journal of Bio-Inspired Computation*, *3*(5),pp 297-305,2011.

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

Liao, C. J., & Liao, C. C., An ant colony optimisation algorithm for scheduling in agile manufacturing. International Journal of Production Research, 46(7), pp 1813-1824, 2008.

Mahdavi, I., Komaki, G. M., Kayvanfar, V. 2011, September. Aggregate hybrid flowshop scheduling with assembly operations. Processing of IEEE 18Th International Conference on the Industrial Engineering and Engineering Management (IE&EM), Changchun, China, Sept.3-5 , pp. 663-667.

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

Ouaarab, A., Ahiod, B., & Yang, X. S., Discrete cuckoo search algorithm for the travelling salesman problem. *Neural Computing and Applications*, *24*(7-8), pp 1659-1669,2014.

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

Yang, X. S., & Deb, S. (2009, December). Cuckoo search via Lévy flights. In Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on (pp. 210-214). IEEE.

Biography

Mohamed Komaki is accomplished B.Sc. in Industrial Engineering at Sharif University of Technology, Tehran, Iran (2001–2006) and M.Sc. in Industrial Engineering at Mazandaram University of Science & Technology, Babol, Iran (2007–2010). He is currently studying System Engineering at Case Western Reserve University, Cleveland, USA. His research interests include scheduling, optimization and multi-criteria decision making.

Shaya Sheikh obtained his Ph.D. from Case Western Reserve University in 2013. He has worked as Scheduling and Optimization Scientist at Lancaster Laboratories. He is now assistant professor of management science at New York Institute of Technology. His research interests include scheduling optimization, multi-criteria decision making, and energy supply chain management.

Behnam Malakooti: Professor Malakooti obtained his PhD in 1982 from Purdue University. He has consulted for numerous industries and corporations, including General Electric, Parker Hannifin, and B.F. Goodrich. He has published over 100 papers in technical journals. In his work, systems architectures, space networks, optimization, multiple criteria & intelligent decision making, trait analysis of biological systems, adaptive artificial neural networks, and artificial intelligence theories and techniques are developed and applied to solve a variety of problems. His current research is on design and protocols analysis for NASA space-based networks. Recently Professor Malakooti developed a four-dimensional approach for decision-making process typology and risk analysis. Decision-making typology accurately identifies the four types of decision makers' behavior: Information processing, creativity, risk, and decisiveness approach. It also provides a basis for developing the next generation of intelligent robots. See http://car.cwru.edu/decision/ for computerized survey. He has made contributions to manufacturing systems developing computer aided approaches for manufacturing/production design, planning, operations, facility layout, assembly systems, scheduling, MEMS, and machine set-up, tool design, and machinability.