Proceedings of the International Conference on Industrial Engineering and Operations Management
Washington DC, USA, September 27-29, 2018

Scheduling Complex Products with Assembly Operations in Hybrid Flow Shop Environment

Vahid Kayvanfar
Department of Industrial Engineering
Amirkabir University of Technology
424 Hafez Ave., 15875-4413 Tehran, Iran
v.kayvanfar@aut.ac.ir

GH.M. Komaki
Department of Marketing and Business Analytics, College of Business
Texas A&M University-Commerce
2200 Campbell St., Commerce, Texas 75429, USA
mohamed.komaki@tamuc.edu

Shaya Sheikh
Department of Management Science Studies, School of Management
New York Institute of Technology
1855 Broadway, New York, NY 10023, USA
ssheik11@nyit.edu

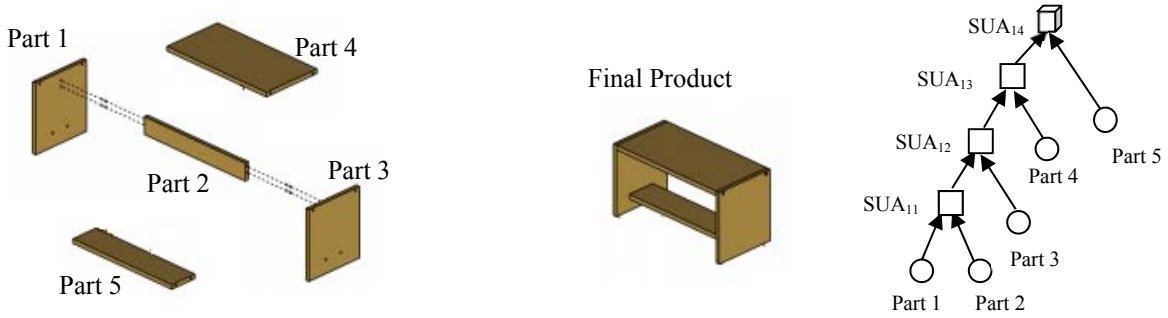
Ehsan Teymourian
Department of Management Science and Information Systems
Rutgers, the state university of New Jersey
Newark & New Brunswick, NJ, USA
ehsan.teymourian@rutgers.edu

Abstract

The hybrid flowshop scheduling problem with assembly operations is dealt with in this study in which the hybrid flowshop is followed by single assembly machine. Each final product has predefined structure and includes several parts with known bill of material (BOM). Parts are firstly processed in hybrid flowshop and thereafter, assembled based on structure of products. Each product has assembly operation(s) and the product is considered complete once its last assembly operation is processed. The goal is to find a schedule which minimizes the last product completion time. For the addressed problem, a mathematical model is developed and two heuristic algorithms as well as two meta-heuristic ones based on Ant Colony Optimization (ACO) are then proposed to solve the large-scale instances. The heuristic algorithms are developed versions of the existing well-known techniques which are employed to tackle the mentioned problem in two phases:1) finding the sequence of assembly operations/products, 2) finding the parts sequence. Due to precedence relation of parts and assembly operation, two versions of the ACO algorithm are proposed. The first version is basic ACO which has been adapted to the studied problem however the second one uses two pheromones where the first pheromone is employed to find the sequence of assembly operations and the second one is designed to find the parts sequence. Also, a lower bound based on the bottleneck stage is then proposed. In order to test efficiency of the methods, a comprehensive experiment is conducted.

Keywords
Scheduling; Hybrid flow shop; Assembly operation; Multi-Level Products; Meta-heuristics.

1. Introduction
Scheduling problems in conventional hybrid flow shop have been extensively studied. Most of these researches assume simple string type and independent jobs. However, in reality, final products are complex with several components, assembly and subassembly operations. In spite of this fact, few studies have addressed assembly operations. In this paper, Assembly Hybrid Flow Shop (AHFS), hybrid flow shop (HFS) followed by an assembly station, is investigated. Indeed, the HFS is the machining stage that components of the products should be processed and at the assembly stage there is single flexible machine (robot) or team of workers which joins the processed parts together. HFS has at least two stages where each stage has parallel machines and one stage has more than one machine. Preemption is not allowed; also, there is no machine breakdown. The assembly stage or team of workers joins parts of products based on the predefined assembly structure of products. In this problem, there are H products with similar assembly structure and each product may have components and/or assembly operation(s) and the goal is finding the sequence of components and products such that the last product completion time, i.e., C_{max}, is minimized. This manufacturing system is suitable for producing customized products in large volume and low cost in short period of time; therefore, it can provide competitive advantage in the market. A lot of real production systems such as automobile industry (Fattahi *et al*., 2013), fire engine factory (Lee *et al*., 1993), personal computers factory (Potts *et al.*, 1995), aircraft factory (Jiang and Wang, 2014) can be modelled as AHFS. As an application of AHFS, consider the following example which is inspired from wood industry. Consider the presented desk in Figure 1a as a final product which has five components and the precedence relation of parts is shown in Figure 1b in which parts, assembly operations and final product are shown with circle, square, and cubic square, respectively; and SUA_{hr} represents *r*th assembly (or subassembly) operation of product *h.* Timbers with different sizes and quality are row materials of the shop and they need to go under some processes such as cutting and furnishing to produce different parts of the desk. As can be seen from Figure 1b, this product has four assembly operations where in the first assembly operation, SUA_{11}, parts 1 and 2 are joined together and then in the assembly operation SUA_{12}, part 3 and SUA_{11} are assembled together. This process continues until the last assembly operation is completed. Obviously, the product completion time is the completion time of the product's last assembly operation (at the assembly stage). Suppose there is customer order to produce large number of this desk, then AHFS can be a prefect model to produce the order in reasonable time.



a: Components of the desk (Agrawala et al, 2003) b: Assembly structure of the desk

Figure1. An example of assembly operations in the wood industry

2. Literature review
Structure of products could be distinguished by "*Assembly Level*." The products based on the number of assembly operations at each assembly level could be grouped into *Simple* and *Complex* products. Simple products have only one assembly operations while complex products have more than one assembly operation. In Figure 2, Products 1, 2 and 3 are simple while Products 4 and 5 are complex.

Hybrid flow shop scheduling problems with different criteria have been addressed extensively. According to Ruiz and Vázquez-Rodríguez (2010), the makespan is a well-studied criterion among other criteria. The proposed methods to tackle the HFS can be grouped into exact method (Branch and Bound (B&B), dynamic programming and so on), heuristic and metaheuristic algorithms. Dispatching rules as simple heuristic algorithms such as shortest processing time (SPT) are the simplest heuristic algorithms which find sequence of jobs in short computational time and their performance is fairly good. For example, Santoset al. (1996), Brah and Wheeler (1998) and Kurz and Askin (2003) investigated the performance of some dispatching rules. Due to NP-hard nature of HFS, majority of studies proposed

meta-heuristic algorithms to tackle HFS problem. For example, one can mention Genetic Algorithm (GA) (Ruiz and Maroto,2006; Kahraman et al., 2008), Tabu Search (TS) (Haouari and M'Hallah, 1997; Nowicki and Smutnicki,1998), Simulated Annealing (SA) (Haouari and M'Hallah, 1997; Jinet al., 2006; Mirsanei et al., 2011), Cuckoo Search (CS) Algorithm (Marichelvam et al., 2014), Particle Swarm Optimization (PSO) (Alaykyran et al., 2007; Naderi et al., 2014), Artificial Bee Colony (ABC) (Pan et al., 2014) as well as hybrid metaheuristic algorithms. Unfortunately, authors addressing HFS neither have used the same test problems to evaluate the performance of the proposed metaheuristic algorithms nor the same lower bound is used to compare them (Ribaset al., 2010); therefore, it makes difficult to identify the best performing algorithm among the proposed algorithms. For comprehensive reviews, readers are referred to Linn and Zhang (1999), Wang (2005), Ruiz and Vázquez-Rodríguez (2010), and Ribas et al. (2010).

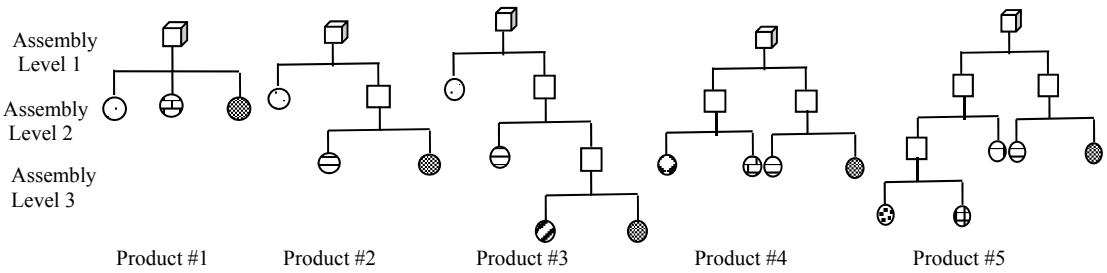


Figure 2. Illustrative example of products with different assembly levels

Simple version of AHFS is agile manufacturing systems where has two stages; production (fabrication) and assembly stage where the fabrication stage feeds the assembly stage. The fabrication and assembly stages may have either single machine or (identical) parallel machines. In agile manufacturing systems, products could be either simple or complex. Kusiak (1989) introduced two-stage agile manufacturing system each stage with single machine and showed that Maximum Level of Depth First (MLDF) provides the minimum makespan for simple products. Also, Kusiak (1989) developed an algorithm to find the sequence of complex products based on decomposition of assembly operations and MLDF. Kim et al. (1996) investigated minimizing total tardiness and earliness in two-stage flow shop with multi-level products and proposed SA as well as GA, and concluded that SA outperforms GA. He et al. (2001) considered the same problem where the first stage has multiple identical parallel machines and proposed a heuristic algorithm based on the MLDF and also developed a lower bound. He and Babayan (2002) investigated the two-stage agile manufacturing system that stage one has only one machine and the second stage has multiple parallel assembly machines and each product has multiple assembly operations. They showed that MLDF can provide the optimal solution for this system. Then, they proposed heuristic algorithms based on MLDF to minimize the makespan. Gaafar and Masoud (2005) addressed the same problem and proposed four hybrid meta-heuristic algorithms based on GA and SA. Later, Gaafar et al (2008) for the same problem proposed GA and a modified GA based on the PSO algorithm called MGA and showed that MGA outperforms the existing algorithms such as He and Babayan's (2002) algorithm. Liao and Liao (2008) investigated this problem and developed Ant Colony Optimization (ACO) algorithm and showed that ACO outperforms the proposed algorithms by He and Babayan (2002) as well as Gaafar and Masoud (2005). Mahdavi et al. (2011) investigated HFS with complex products, each of which has multiple assembly operations. Moreover, they presented some heuristic techniques. Fattahi *et al.* (2013) considered two-stage hybrid flowshop followed by assembly stage with simple products. They proposed simple heuristic algorithms based on the Johnson's algorithm as well as lower bound. Recently, Bhongade and Khodke (2014) addressed AHFS with complex products and proposed GA.

3. Problem description

There are H products with the same hierarchy assembly structure, Bill of Materials (BOM), where should be processed through S-stage of HFS followed by single flexible assembly machine. The HFS has at least two processing stages ($S \geq 2$) in which at least one of these stages has more than one machine. Also, all machines are identical at each stage. Moreover, only one part could be processed on any machine at a time and preemption is not allowed. Parts should first be processed in the HFS and thereafter in the assembly stage if all components of the specific assembly operation are available, it could be assembled. Each product may have several assembly operations. Components (children) of assembly operations (parents) could be "parts" and/or "assembly operations" which hereafter called sub-assembly (SUA). Assembly operations must be assigned to the assembly machine with respect to their hierarchical structure constraint. The completion time of each product equals to the completion time of its last assembly operation. The goal

is to find the parts and assembly operations sequence which minimizes the completion time of the last product, makespan. In this problem, there are H products which have the same structure. Parts should be first processed at HFS stages and subsequently at the assembly stage. The completed parts in the last stage of the HFS should be joined together based on the products BOM. Each product has some assembly operations in which components of the assembly operation in the maximum assembly level are only parts while components of the other assembly operations could be parts and/or assembly operations (see Figs. 1and 2). Each product completion time equals to the completion time of the last assembly operation in assembly level 1. Decision variables are defined so as to determine the parts sequence in HFS stages and assembly operations sequence in the assembly stage such that minimize C_{max}.

3.1 Assumption

There are *S*+1 stages in which the first *S* stages are stages of HFS where each stage consists of identical parallel machines. The last stage, stage S+1, is the assembly stage with only one assembly machine. All machines never break and are available all the times. All parts are available at time zero and all processing/assembly times are deterministic and known in advance. At any stage, each part/assembly operation could be processed only by one machine at a time and also each machine could process only one part at once. The BOM of each product is known and all products have the same structure. Processing of an assembly operation at the assembly stage could be started only when all its components are ready. Also, the product completion time equals to the completion time of its last assembly operation.

3.2 Notations

3.2.1. Subscripts

H	Number of products
N	Number of parts
S+1	Number of stages
p,h,f	Index for product (p,h,f= 1,2,…H)
i,j,l	Index for parts (i,j,l= 0,1,…N)
t	Index for stage (t = 1, 2,…S+1)
k	Index for machine (k =1, 2, …m_t)
I_p	Total number of assembly operation of product p
g,r,e	Index of assembly operation of product p (g,r,e = 0,1,…,I_p)

3.2.2. Input parameters

P_i^t	Processing time of part $i$at stage t, $t\in$ {1,2,…,S}
m_t	Number of machines at stage t; $t\in$ {1,2,…,S}
q_{pr}	Assembly time of SUA_{pr} at assembly stage (S+1)
Q_{pr}	The set of parts which are components of SUA_{pr}
J_{pr}	The set of assembly operations which are components of SUA_{pr}

3.2.3. Decision variables

C_i^t	Completion time of part i at stage t, $t\in$ {1,2,…,S}
Co_{pr}	Completion time of SUA_{pr}
Com_p	Completion time of product p
x_{ijm}^t	1 if part j is processed exactly after part $i$on machine m at stage t, $t\in$ {1,2,…,S}; 0 otherwise
y_{ijm}^t	1 if processing of part i precedes part j on machine m at stage t, $t\in$ {1,2,…,S}; 0 otherwise
w_{im}^t	1 if part i is assigned to machine m at stage t, $t\in$ {1,2,…,S}; 0 otherwise
xx_{hgpr}	1 if SUA_{pr} exactly after SUA_{hg} is processed at assembly stage; 0 otherwise
yy_{hgpr}	1 if processing SUA_{hg} precedes SUA_{pr} at assembly stage; 0 otherwise
C_{max}	maximum completion time of products
M	A big positive arbitrary number

3.3 Mathematical Model

Min C_{max} (1)

Subject to:

$$\sum_{i=0}^{N} x_{ijm}^{t} = \sum_{k=0}^{N} x_{jkm}^{t} \qquad \forall j\epsilon\{1,..,N\}, \forall t\epsilon\{1,\dots,S\}, \forall m \in \{1,..,m_t\} \qquad (2)$$

$$\sum_{j=0}^{N} x_{ijm}^{t} = w_{im}^{t} \qquad \forall i\epsilon\{1,..,N\}, \forall t\epsilon\{1,\dots,S\}, \forall m \in \{1,..,m_t\} \qquad (3)$$

$$\sum_{m=1}^{m_t} w_{im}^{t} = 1 \qquad \forall i\epsilon\{1,..,N\}, \forall t\epsilon\{1,\dots,S\} \qquad (4)$$

$$\sum_{i=1}^{N} x_{ijm}^{t} = 1 \qquad \forall j = 0, \forall t\epsilon\{1,\dots,S\}, \forall m \in \{1,..,m_t\} \qquad (5)$$

$$\sum_{j=1}^{N} x_{ijm}^{t} = 1 \qquad \forall i = 0, \forall t\epsilon\{1,\dots,S\}, \forall m \in \{1,..,m_t\} \qquad (6)$$

$$x_{ijm}^{t} \le y_{ijm}^{t} \qquad \forall i,j\epsilon\{1,..,N\}, \forall t\epsilon\{1,\dots,S\}, \forall m \in \{1,..,m_t\} \qquad (7)$$

$$y_{ijm}^{t} + y_{jim}^{t} = w_{im}^{t} \,.\, w_{jm}^{t} \qquad \forall i,j\epsilon\{1,..,N\}, \forall t\epsilon\{1,\dots,S\}, \forall m \in \{1,..,m_t\} \qquad (8)$$

$$y_{ijm}^{t} + y_{jdm}^{t} = y_{idm}^{t} + 1 \qquad \forall i,j,d\epsilon\{1,..,N\}, \forall t\epsilon\{1,\dots,S\}, \forall m \in \{1,..,m_t\} \qquad (9)$$

$$C_i^t + P_j^t + (\sum_{m=1}^{m_t} y_{ijm}^{t} - 1).M \le C_j^t \qquad \forall i,j\epsilon\{1,..,N\}, \forall t\epsilon\{1,\dots,S\} \qquad (10)$$

$$\sum_{r=0}^{I_p} xx_{hgpr} = 1 \qquad \forall p,h\epsilon\{1,..,H\}, \forall g\epsilon\{0,..,I_h\}, \qquad (11)$$

$$\sum_{g=0}^{I_h} xx_{hgpr} = 1 \qquad \forall p,h\epsilon\{1,..,H\}, \forall r\epsilon\{0,..,I_p\} \qquad (12)$$

$$xx_{hgpr} \le yy_{hgpr} \qquad \forall p,h\epsilon\{1,..,H\}, \forall g\epsilon\{0,..,I_h\}, \forall r\epsilon\{0,..,I_p\} \qquad (13)$$

$$yy_{hgpr} + yy_{hgpr} = 1 \qquad \forall p,h\epsilon\{1,..,H\}, \forall g\epsilon\{0,..,I_h\}, \forall r\epsilon\{0,..,I_p\} \qquad (14)$$

$$yy_{hgfe} + yy_{fepr} = 1 + yy_{hgpr} \qquad \forall p,h,f\epsilon\{1,..,H\}, \forall g\epsilon\{0,..,I_h\}, \forall r\epsilon\{0,..,I_p\}, \forall e\epsilon\{0,..,I_f\} \qquad (15)$$

$$Co_{pr} - q_{pr} \ge C_i^S \qquad \forall p\epsilon\{1,..,H\}, \forall r\epsilon\{0,..,I_p\}, \forall i\epsilon Q_{pr} \qquad (16)$$

$$Co_{pr} - q_{pr} \ge Co_{pe} \qquad \forall p\epsilon\{1,..,H\}, \forall r\epsilon\{0,..,I_p\}, \forall e\epsilon J_{pr} \qquad (17)$$

$$Co_{hg} - q_{hg} \ge Co_{pr} - M.yy_{hgpr} \qquad \forall p,h\epsilon\{1,..,H\}, \forall g\epsilon\{0,..,I_h\}, \forall r\epsilon\{0,..,I_p\} \qquad (18)$$

$$Com_p \ge Co_{pr} \qquad \forall p\epsilon\{1,..,H\}, \forall r\epsilon\{0,..,I_p\} \qquad (19)$$

$$Cmax \ge Com_p \qquad \forall p\epsilon\{1,..,H\} \qquad (20)$$

$$x_{ijm}^{t}, y_{ijm}^{t}, w_{im}^{t}, yy_{hgpr}, xx_{hgpr} \in \{0,1\} \qquad \forall i,j\epsilon\{1,..,N\}, \forall t\epsilon\{1,\dots,S\}, \forall m \in \{1,..,m_t\}, \forall p,h\epsilon\{1,..,H\}, \forall g\epsilon\{0,..,I_h\}, \forall r\epsilon\{0,..,I_p\} \qquad (21)$$

Constraint (1) presents the objective function which minimizes the makespan of the AHFS. In the model, Constraints (2-10) are related to the HFS stages and Constraints (11-20) take care the sequence of assembly operations at the assembly stage. Constraints (2-4) are related to parts assignment onto machines. Actually, Constraints (2) and (3) ensure that each part has one prior and one subsequent part in its queue on the machine which is allocated to process it. Also, Constraint (4) guarantees that each part will be processed on only one machine at each stage. Constraints (5) and (6) similar to Constraints (2) and (3) are the flow conservation constraints and confirm that the 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 stage *S*. Inequalities (7) impose that if the edge (i,j) is used on machine *m*, then *i* precedes *j* in their queue. Constraints set (8) ensure that between each pair of parts *i* and *j* assigned to the 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 *k* and part *j* is visited after *k* on machine *m*'s queue, so part *j* must be visited after *i* in that queue. Constraints (10) guarantee that if part *i* precedes part *j*, then the processing start time of part *j* on machine *m* must be greater than or equal to the completion time of each part *i* on that stage. Constraints (11-15) similar to constraints (2-9) consider this fact that there is only one machine and all parts as well as subassemblies must be processed on such a machine. Constraints (16) and (17) enforce that each assembly completion time must be greater than completion times of its components plus its processing time. Constraints set (18) work as constraints set (10) in assembly stage, i.e., it guarantees that the processing start time of SUA_{hg} must be grater or equal than completion time of each SUA_{hr} which precedes it. Constraints set (19) calculate the completion time of each product which equals to the completion time of its last assembly operation. Constraints

set (20) compute the completion time of the whole system (AHFS) which equals to the last product completion time. Finally, constraint (21) shows that the variables used in our model are binary. It should be noticed that the right hand side of Eq. (8) is nonlinear. Accordingly, in order to make it linear an attempt is made using introducing the following variable (z) as well as Constraints (22) and (23) to overcome the nonlinearity of term $x.y$:

$$(z-1) \geq (x-1) + (y-1) \quad (22)$$

$$2(z-1) \leq (x-1) + (y-1) \quad (23)$$

where x and y are two binary variables and z is a new intermediary binary variable. We substitute z instead of two multiplied binary variables ($x.y$) to resolve Constraints (22) and (23).

4. Solution approach

Since the studied problem is NP-Hard, the computational time of exact methods is not affordable even for small size instances. Therefore, to solve the medium to large size instances in reasonable computational time, heuristic and meta-heuristic algorithms are developed to find the near optimum solution.

4.1 Heuristic algorithms

Two sequencing algorithms are employed in this section, where the first one is based on Johnson's Algorithm and the second one is based on SPT rule. In both algorithms, first assembly operations are converted to an artificial job by computing their *processing times* (PT) which is the total average processing time of assembly operation parts with respect to the number of processors at each stage, the number of parts and total assembly times, see Eq. (24).

In the first algorithm, the problem is converted into an artificial two-stage flow shop problem where HFS is the first stage and the assembly stage is the second one. The processing time of an assembly operation is represented by PT as shown in Eq. (24) and its assembly operation at the assembly stage.

$$PT_{hr} = \sum_{t=1}^{S} \left\lceil \frac{\sum_{i \in Q_{hr}} p_i^t}{\min(|Q_{hr}|, m_t)} \right\rceil \quad (24)^1$$

In the second algorithm, total processing time of each assembly operation is computed. At each iteration of the second algorithm, only the schedulable assembly operations, the assembly operations that their preceding assembly operations have been scheduled earlier, are considered.

The above tow algorithms determine the sequence of assembly operations. Another algorithm to determine parts sequence of each assembly operation and assign parts to machines is developed. This algorithm uses NEH's algorithm (Nawaz et al., 1983) to find the parts sequence of each assembly operation. Also, we use the first available machine (FAM) to assign parts to the machines at each stage of HFS which results in the earliest completion time of part at that stage.

4.2 Metaheuristic algorithms

In this paper, we propose two metaheuristic algorithms based on ACO; ACO_A and ACO_B. In the first one, ACO_A, ACO is applied to find the sequence of parts without using structure of products while the second one is two-level ACO where first it finds the sequence of assembly operations, and then finds the sequence of parts of each assembly operation.

5. The lower bound

A standard approach to evaluate the algorithms' effectiveness is to compare the algorithms solutions with a lower bound. To do so, a LB based on bottleneck stage is established in this study. As mentioned earlier, the problem at the hand has two sections; HFS and the assembly stage, where both of these sections could be bottleneck stage. In the following we discussed the proposed LB.

5.1 When the assembly stage is bottleneck

Since there is only one machine at assembly stage, most likely it will work as bottleneck. Then, after starting the first assembly operation, the busy period of this stage will start. The first assembly operation should be in the maximum assembly level. So, we need to find an assembly operation with the earliest starting time. The proposed lower bound is shown by Eq. (25) where $\min(EST_{hr})$ is defined by Eq. (26).

[1] $\lceil x \rceil$ represents the smallest integer greater to real number x, and $|Q_{hr}|$ is the number of parts of SUA_{hr}.

$$LB_I = \min_{\substack{h\in\{1,2,..,H\}\\ r\in SAO}} (EST_{hr}) + \sum_{h=1}^{H} \sum_{r=1}^{I_h} q_{hr} \quad (25)$$

In this LB, the candidate assembly operation is chosen among SAOs which has the earliest completion time. This time, for each assembly operation, is defined as the earliest starting time of processing the assembly operation at the assembly stage plus its assembly time. In order to find the earliest starting time, we suggest using the following equation based on the presented LB by Santos *et al.* (1995).

$$EST_{hr} = \max\{\max_{i\in Q_{hr}} \sum_{t=1}^{S} p_i^t \, , \max_{t\in\{1,2,..,S\}} \{\min_{i\in Q_{hr}} \sum_{t=1}^{t-1} p_i^t + \frac{1}{m_t} \sum_{i\in Q_{hr}} p_i^t + \min_{i\in Q_{hr}} \sum_{t=t+1}^{S} p_i^t\}\} \quad (26)$$

where EST_{hr} is the earliest starting time of processing SUA_{hr} at the assembly stage and LSA and RSA are the same as in Santos *et al.* (1995) except that only parts of SUA_{hr} should be considered, i.e., $i \in Q_{hr}$. Simply, LSA(y,t) is the total parts processing time from the first stage to stage *t* in non-decreasing order and RSA(y,t) is the total parts processing time starting from stage *t* to the last processing stage, *S*, in non-decreasing order. For more information one could refer to Santos *et al.* (1995). Finally, the earliest completion time of each assembly operation could be found as:

5.2 When HFS is bottleneck

In this case, HFS section will be busy until processing the last part, and then the associated assembly operation related to that part should be processed at the assembly stage. We simply state this LB as completion time of the last processed part at HFS section and the assembly operation time with the minimum processing time as presented below.

$$LB_{II} = C_{max}^{San} + \min_{h\in H} q_{hI_h} \quad (27)$$

Where C_{max}^{San} is the proposed LB by where should be calculated for all parts are presented below.

$$C_{max}^{San} = \max_{t\in\{1,2,..,S\}} \{\min_{i\in N} \sum_{t=1}^{t-1} p_i^t + \frac{1}{m_t} \sum_{i\in N} p_i^t + \min_{i\in N} \sum_{t=t+1}^{S} p_i^t\} \quad (28)$$

And finally, $LB = \max\{LB_I, LB_{II}\}$

6. Experimental results

6.1 Test instances

To evaluate the performance of the proposed algorithms, several instances are randomly generated and test. The required data to generate instances are; number of products (*P*), BOM of products, number of HFS stages (*S*), number of machines at each stage (m_t) and finally parts and assembly operations processing times. In this study, all stages are supposed to have the same number of machines to simplify generating the instances. Also, the number of processing stages of HFS (*S*) are considered *S* = 2, 3 and 4. The number of machines in HFS stages (m_t) is either 2 3 or 4. Parts processing times and assembly times are distributed uniformly over range [10–50] and range [20–100], respectively. Four different sizes of products, i.e., P = 5, 10, 30, and 50 are considered in this research. In Figure 2, five BOM of products are presented, they are used to generate products where each assembly operation can have 2,3,4, and 5 parts. In total, 720 instances are randomly generated. We compere the proposed algorithms with proposed GA by Bhongade and Khodke (2014). Parameters of GA are considered as suggested by Bhongade and Khodke (2014), population size of GA is 60 for phase I and 30 for phase II and genetic operators are one-point crossover and shift mutation, mutation and crossover rates are 0.2 and 0.8, respectively. The 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.

Table 1. Suggested value of parameters

Parameter	ACO_A	ACO_B
Ant_{Size} : number of ants	100	80
It_{MaxACO}: number of iterations of ACO	500	400
β: importance of heuristic desirability of parts	2	1.4

δ: importance of heuristic desirability (of assembly operations)	-	2.2
ρ: evaporation rate of parts pheromone	0.3	0.5
ρ_2: evaporation rate of assembly operations pheromone	-	0.1
q_0: probability of exploiting the best edge (between parts)	0.8	0.9
l_0: probability of exploiting the best edge (between assembly operations)	-	0.7
It_{MaxVNS}: : number of iterations of VNS	5	4

Appropriate values of the parameters of ACO_A and ACO_B are determined by initial experiments on quite large number of instances and based on these experiments, the parameters values are considered as presented in Table 1.

6.2 Analysis of Experiments

The Relative Percentage Deviation (RPD) is employed to compare the efficiency of the proposed algorithms as follows:

$$RPD = ((Alg_{sol}\text{-}LB)/LB)\times 100 \quad (29)$$

Where Alg_{sol} represents the objective value of the algorithms and LB represents the lower bound. The lower values of RPD are obviously preferred. Table 2 represents the RPD of algorithms for different number of products. It can be observed that H1 clearly outperforms H2 in all cases, moreover, performance of ACO_B and ACO_A are better than GA while ACO_B outperforms ACO_A. Generally, as the number of products increases, the performance of algorithms decreases.

Table 2. Ave. RPD of algorithm for all instances based on number of products

H	H2	H1	GA			ACO_B			ACO_A		
			best	ave	worst	best	ave	worst	best	ave	worst
5	12.50	9.27	2.37	4.06	6.08	0.50	0.95	1.49	1.31	3.05	5.44
10	12.90	10.01	2.49	4.27	6.42	0.60	0.90	1.24	1.45	3.23	5.69
30	13.83	11.03	2.86	4.69	6.79	0.64	1.13	1.70	1.63	3.22	5.28
50	14.58	11.52	3.24	5.01	6.94	0.66	1.20	1.81	1.84	3.39	5.27

Table 3 presents RPD of the algorithms based on assembly structure of products. Ranking of the algorithms in this table also is similar to the above mentioned conclusion. Also, as can be expected, as structure of products gets more complex, RPD of the algorithms increases. Interestingly, the worst performance of algorithms is in BOM type 2 and 3.

Table 3. Ave. RPD of algorithm for all instances based on type of products (BOM)

BOM	H2	H1	GA			ACO_B			ACO_A		
			best	ave	worst	best	ave	worst	best	ave	worst
1	12.87	9.74	2.46	4.10	5.93	0.52	0.92	1.49	1.51	3.09	5.10
2	13.83	11.00	2.88	4.82	7.05	0.64	1.14	1.67	1.82	3.47	5.71
3	13.48	10.48	2.63	4.57	6.92	0.63	1.11	1.68	1.48	3.36	5.87
4	13.94	10.55	2.75	4.39	6.27	0.53	0.97	1.48	1.49	3.00	4.94
5	13.13	10.52	2.97	4.67	6.62	0.70	1.07	1.48	1.48	3.19	5.49

Tables 4-6 present the RPD of algorithms based on number of stages (S), machines at each stage and number of parts of each assembly operation. All of the tables confirm the above conclusion about the ranking of the algorithms.

Table 4. Ave. RPD of algorithm for all instances based on number of processing stages

S	H2	H1	GA			ACO_B			ACO_A		
			best	ave	worst	best	ave	worst	best	ave	worst
2	13.07	10.29	2.72	4.57	6.75	0.62	1.02	1.49	1.84	3.50	5.75
3	13.32	10.45	2.68	4.34	6.34	0.57	1.03	1.51	1.40	3.05	5.14
4	13.97	10.64	2.82	4.62	6.57	0.62	1.08	1.68	1.43	3.12	5.36

In Figure 3, the computational time of metaheuristic algorithms is presented. As can be seen from the figure, the computational time of ACO_A is slightly higher than GA and computational time of GA is higher than ACO_B, i.e., in average ACO_B is faster than ACO_A and GA.

Table 5. Ave. RPD of algorithm for all instances based on number of machines

m_t	H2	H1	GA	ACO_B	ACO_A

			best	ave	worst	best	ave	worst	best	ave	worst
2	13.12	10.26	2.84	4.62	6.62	0.57	0.99	1.49	1.49	3.08	5.20
3	13.05	10.11	2.63	4.36	6.49	0.60	1.02	1.51	1.58	3.20	5.36
4	14.18	11.01	2.74	4.55	6.56	0.64	1.12	1.68	1.61	3.38	5.71

Table 6. Ave. RPD of algorithm for all instances based on number of parts of each assembly operation

Part	H2	H1	GA			ACO_B			ACO_A		
			best	ave	worst	best	ave	worst	best	ave	worst
2	13.09	10.11	2.71	4.41	6.36	0.64	1.08	1.62	1.58	3.18	5.27
3	13.43	10.78	2.58	4.38	6.54	0.52	0.87	1.31	1.45	3.02	5.22
4	13.60	10.37	2.93	4.75	6.85	0.61	1.10	1.63	1.48	3.28	5.73

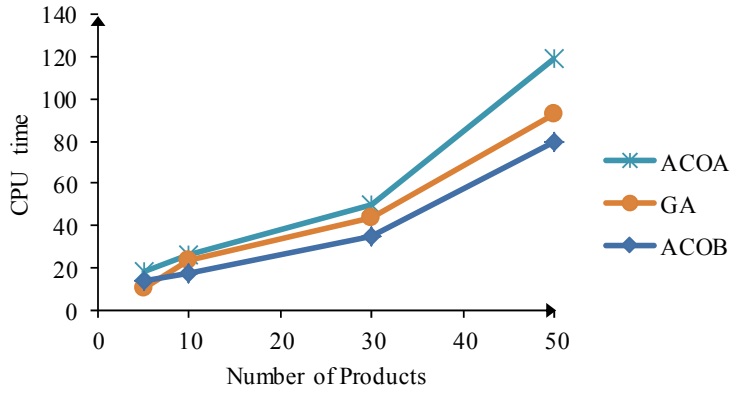


Figure 3. Computational time of metaheurisittc algorithms vs. number of products

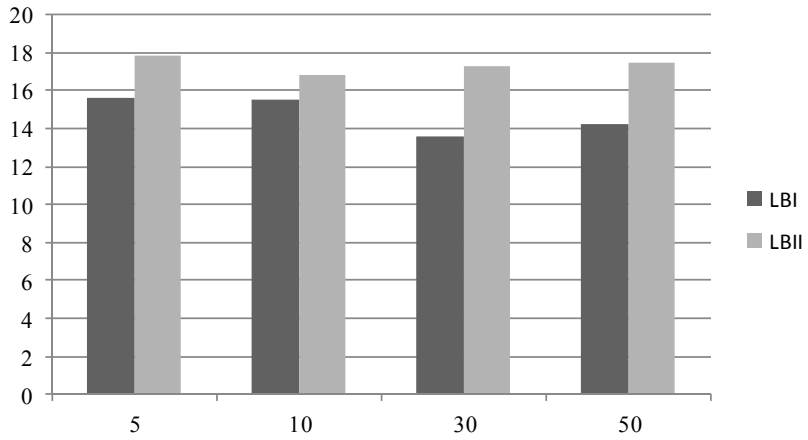


Figure 4. Average DVL of LB_I and LB_{II} from best known solutions

Performance of the proposed LB is measured as below. In Figure 4, performance of LB_I and LB_{II} based on number of products are presented. As can be seen, LB_I has better performance than LB_{II}. In average, DVL of LB_I is 14% which is fairly good.

$$DVL=100(Alg_{Sol} - LB)/Alg_{Sol} \quad (30)$$

7. Conclusion and future studies

This paper dealt with the hybrid flowshop scheduling problem with assembly operations where products are complex and have several components. The hybrid flow shop is machining section and the assembly stage which is a group of worker or a flexible machine assembles the parts based on the assembly structure of products. The main contribution of this paper is considering the machining and assembling the multi-level products which have been less investigated

in the literature. The goal of this research is to obtain a schedule having the minimum makespan. In this context, a mathematical model is firstly proposed for the considered problem. Thereafter, besides developing some heuristic techniques, two meta-heuristic algorithms based on Ant Colony Optimization (ACO) are proposed to solve the medium-to-large size instances. One of the proposed ACOs uses one pheromone to find the sequence of parts while the other one uses two types of pheromones which one of them is designed to find the sequence of assembly operations and the other one is to find the sequence of parts. Moreover, a lower bound based on bottleneck stage is developed. The computational results showed that the proposed lower bound is fairly good. Also, our experimental results revealed that the proposed ACOs outperform the existing GA proposed for this problem. Moreover, ACO with two pheromones has better performance than the basic ACO in term of computational time and relative error.

As a direction for future research, it would be interesting to consider limited buffer space between stages and setup times which are more realistic in real-world situations. In this study, it is assumed that the processing times are known and fixed, but in the real-world production environments this assumption is not valid. The processing time can increase or decrease due to unpredicted situations, for example see Kayvanfar et al. (2013) and Zarandi and Kayvanfar (2015). Extending this problem into the controllable processing time can be interesting research area.

References

Agrawala, M., Phan, D., Heiser, J., Haymaker, J., Klingner, J., Hanrahan, P., &Tversky, B., Designing effective step-by-step assembly instructions. In *ACM Transactions on Graphics (TOG)*, Vol. 22, No. 3, pp. 828-837, 2003.

Alaykýran, K., Engin, O., & Döyen, A., Using ant colony optimization to solve hybrid flow shop scheduling problems. *The international journal of advanced manufacturing technology*, *35*(5-6), 541-550, 2007.

Bhongade, A. S., &Khodke, P. M., A Genetic Algorithm for Flow Shop Scheduling with Assembly Operations to Minimize Makespan. *Journal of The Institution of Engineers (India): Series C*, *95*(2), 89-96, 2014.

Brah, S. A., & Wheeler, G. E., Comparison of scheduling rules in a flow shop with multiple processors: a simulation. *Simulation*, *71*(5), 302-311, 1998.

Fattahi, P., Hosseini, S.M.H., & Jolai, F., A mathematical model and extension algorithm for assembly flexible flow shop scheduling problem. *The International Journal of Advanced Manufacturing Technology*, 65, 787-802, 2013.

Gaafar, L. K., & Masoud, S. A., Genetic algorithms and simulated annealing for scheduling in agile manufacturing. *International Journal of Production Research*, 43(14), 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), 707-720, 2008.

Haouari, M., & M'Hallah, R., Heuristic algorithms for the two-stage hybrid flowshop problem. *Operations Research Letters*, *21*(1), 43-53, 1997.

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

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

Jiang, T. H., & Wang, H. X., Study on the self-evolution problem of an aircraft-engine assembly workshop with uncertain number of assembly times. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, 0954405414535593, 2014.

Jin, Z., Yang, Z., & Ito, T., Metaheuristic algorithms for the multistage hybrid flowshop scheduling problem. *International Journal of Production Economics*, *100*(2), 322-334, 2006.

Kahraman, C., Engin, O., Kaya, I., & Kerim Yilmaz, M., An application of effective genetic algorithms for solving hybrid flow shop scheduling problems.*International Journal of Computational Intelligence Systems*, *1*(2), 134-147, 2008.

Kayvanfar, V., Mahdavi, I., & Komaki, G. M., A drastic hybrid heuristic algorithm to approach to JIT policy considering controllable processing times. *The International Journal of Advanced Manufacturing Technology*, 69(1-4), 257-267, 2013.

Kim, J. U., & Kim, Y. D. Simulated annealing and genetic algorithms for scheduling products with multi-level product structure. *Computers & Operations Research*, *23*(9), 857-868, 1996.

Kurz, M. E., & Askin, R. G., Comparing scheduling rules for flexible flow lines. *International Journal of Production Economics*, 85(3), 371-388, 2003.

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

Lee, C.Y., Cheng, T.C.E., & Lin, B.M.T., Minimizing the makespan in the 3-machine assembly-type flowshop scheduling problem. *Management Science*, 39, 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), 1813-1824, 2008.

Linn, R., & Zhang, W., Hybrid flow shop scheduling: a survey. *Computers & industrial engineering*, 37(1), 57-61, 1999.

Mahdavi, I., Komaki, G.M., & Kayvanfar, V., Aggregate hybrid flow shop scheduling with assembly operations, in: *Industrial Engineering and Engineering Management (IE&EM), 2011 IEEE 18Th International Conference*, pp. 663–667, 2011.

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

Mirsanei, H. S., Zandieh, M., Moayed, M. J., & Khabbazi, M. R., A simulated annealing algorithm approach to hybrid flow shop scheduling with sequence-dependent setup times. *Journal of Intelligent Manufacturing*, *22*(6), 965-978, 2011.

Naderi, B., Gohari, S., & Yazdani, M., Hybrid flexible flowshop problems: Models and solution methods. *Applied Mathematical Modelling*. DOI: 10.1016/j.apm.2014.04.012, 2014.

Nawaz, M., Enscore, E. E., & Ham, I., A heuristic algorithm for the *m*-machine, *n* job flow-shop sequencing problem. *Omega*, 11, 91-95, 1983.

Nowicki, E., & Smutnicki, C., The flow shop with parallel machines: A tabu search approach. *European Journal of Operational Research*, *106*(2), 226-253, 1998.

Pan, Q. K., Wang, L., Li, J. Q., & Duan, J. H., A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespanminimisation. *Omega*, *45*, 42-56, 2014.

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

Ribas, I., Leisten, R., & Framiñan, J. M., Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research*, *37*(8), 1439-1454, 2010.

Ruiz, R., & Maroto, C., A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operational Research*, *169*(3), 781-800, 2006.

Ruiz, R., & Vázquez-Rodríguez, J.A., The hybrid flow shop scheduling problem. *European Journal of Operational Research*, 205, 1-18, 2010.

Santos, D. L., Hunsucker, J. L., & Deal, D. E., An evaluation of sequencing heuristics in flow shops with multiple processors. *Computers & industrial engineering*, *30*(4), 681-691, 1996.

Santos, D.L., Hunsucker, J.L., & Deal, D.E., Global lower bounds for flow shops with multiple processors. *European Journal of Operational Research*, 80, 112-120, 1995.

Wang, H., Flexible flow shop scheduling: optimum, heuristics and artificial intelligence solutions. *Expert Systems*, *22*(2), 78-85, 2005.

Zarandi MHF, & Kayvanfar, V., A bi-objective identical parallel machine scheduling problem with controllable processing times: a just-in-time approach. *International Journal of Advanced Manufacturing Technology*, 77(1–4):545–563, 2015.

Biographies

Vahid Kayvanfar is a lecturer at Amirkabir University of Technology (Tehran Polytechnic). He got his PhD in Industrial Engineering from Amirkabir University of Technology in 2017. He is reviewer of more than 30 ISI journals. His research interests mostly include supply chain management soft computing, applied operations research, production planning and scheduling in different areas. Now, he has a HOT paper in Journal of Computational Science and published several journal and conference papers in the above-mentioned areas. He also has written "Advanced Operations Research" book in Farsi. Moreover, he has performed some practical projects in different organizations and taught several courses such as operations research, probability & engineering statistics, etc. in different universities during recent 7 years.

GH.M. Komaki is an Assistant Professor of business analytics at Texas A&M University-Commerce. He has 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), Ph.D. in Systems Engineering at Case Western Reserve University, Cleveland, USA (2012-2018). His research interests include scheduling, optimization and multi-criteria decision making.

Shaya Sheikh is currently serving as assistant professor of operations management at New York Institute of Technology school of management. He obtained his Ph.D. from Case Western Reserve University in 2013. He His research interests include supply chain, smart homes, and scheduling.

Ehsan Teymourian is a PhD candidate in Management Science-Operations Research at Rutgers Business School Newark & New Brunswick, NJ, USA. Mr. Teymourian holds both Bachelor and Master of Science degrees in Industrial Engineering from Mazandaram University of Science & Technology in Iran. He has published several research papers in Operations Management, Scheduling, Optimization and Algorithms, and Logistics. He is already working on Stochastic Optimizations and Machine Learning Optimization algorithms.