Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Some heuristics for the hybrid flow shop scheduling problem with setup and assembly operations

Parviz Fattahi[a], Seyed Mohammad Hassan Hosseini[b*] and Fariborz Jolai[c]

[a]*Department of Industrial Engineering, Bu-Ali Sina University, Hamedan, Iran*
[b]*Department of Industrial Engineering, Payame Noor University, Tehran, Iran*
[c]*Department of Industrial Engineering, University of Tehran, Tehran, Iran*

C H R O N I C L E

Article history:
Received January 2 2013
Received in revised format
March 14 2013
Accepted March 14 2013
Available online
March 15 2013

Keywords:
Scheduling
Hybrid flow shop
Assembly operation
Setup
Heuristic

A B S T R A C T

This paper presents a two-stage hybrid flow shop scheduling problem with setup and assembly operations. The proposed study of this paper considers one kind of product with a quantity of demand where each product is made by assembling a set of different parts. At first, the parts are manufactured in a two-stage hybrid flow-shop and then the parts are assembled into products on assembly stage. Setup operations are needed when a machine starts processing the parts or it changes items. The considered objective is minimizing the completion time of all products. Since the problem is classified as NP-hard class, a combinatorial algorithm is proposed. The proposed algorithm is a three-step procedure where we use heuristic, genetic algorithm (GA), simulated annealing (SA), NEH and Johnson's algorithm. Three lower bounds are presented and improved to evaluate the proposed algorithms. An extensive computational experiment is conducted to compare the performances of the proposed algorithms.

1. Introduction

One of the most prominent tasks in service and manufacturing industries is to schedule arriving jobs such that some criteria hold (Hentsch et al., 2011). In a manufacturing system, scheduling procedure determines an exact production time and a machine assignment for each operation. In assembly scheduling problem, there is normally a preassembly stage followed by an assembly stage. When the preassembly stage is required prior to machining stage, the parts are processed independently and then, they are assembled into production during assembly stage and one important criterion for this kind of problem is to minimize the maximum job completion time (Koulamas et al., 2001). There are many industry applications where we face this kind of modeling formulation and many people are also interested in doing research in this area (Lee et al., 1993; Allahverdi et al., 2009; Naderi-Beni et al., 2012). Lee et al. (1993) described an application for a fire engine assembly plant while Potts et al. (2009) described an application in personal computer manufacturing industry. In particular,

* Corresponding author.
E-mail: sh.hosseini@phd.pnu.ac.ir (S. M. H. Hosseini)

doi: 10.5267/j.ijiec.2013.03.004

manufacturing of almost all items may be modeled as a two-stage assembly scheduling problem including machining operations and assembly operations (Allahverdi et al., 2009).

Lee et al. (1993) are believed to be first who contributed on 3-machine assembly-type flow shop scheduling problem significantly and since then this area of research has attracted many other researchers. However, generally speaking, scheduling for parts machining and planning for assembly operations have been independently studied (Yokoyama et al., 2005). The preassembly stage is considered as a two-stage hybrid flow shop in this study where we first complete a set of parts in hybrid flow shop and assemble them into a final product. Setup operation and setup time are needed when a machine start processing the parts or it changes items. The hybrid flow shop is a generalization of the classical flow shop in which there are parallel machines for some operations (Blazewicz et al., 2007; Pinedo 2008; Quadt et al., 2007; Ying et al., 2006). In this system, generally a set of *n* jobs must be processed in a series of *m* stages. Ruiz et al. (2010) reviewed the hybrid flow shop scheduling (HFS) problem and presented its characteristics completely.

The HFS is sometimes referred to as a flexible flow shop, compound flow shop, multi-processor flow shop, or flow shop with parallel machines (Pinedo 2008; Quadt et al., 2007; Ruiz et al., 2010). This generally comprises three sub-problems: Batching, loading, and sequencing. Batching occurs only if setup costs or times are not negligible and several jobs of the same product type have to be produced. A batching sub-procedure considers a number of units, which must be produced, consecutively, i.e. batch-size calculation. The majority of scheduling research considers setup times as negligible or part of processing time and only a few studies explicitly consider setup times among jobs (Quadt et al., 2007; Lin et al., 2012). The loading sub-problem refers to the allocation of operations to the parallel machines (Blazsik et al., 2008). After calculating batch-sizes, the loading of sub-problem is used to assign the batches to machines. Finally, the sequencing sub-problem is used to sequence the jobs assigned to a machine.

Garey and Johnson (1979) showed that the HFS problem with makespan objective is NP-complete, and for years, a large numbers of heuristics and approximation algorithms have been proposed for various HFS configurations (Ying et al., 2006; Ribas et al., 2010; Tseng et al., 2008; Jin et al., 2006). Gupta (1988) showed that HFS is limited to two processing stages, one stage contains at least two machines and the other one includes a single machine and the problem is NP-hard (Khalouli et al., 2010; Yang 2010). Therefore, it is obvious that the considered problem of this paper, with a more complex structure, is NP-hard.

There are two major types of studies in which both machining operations and assembly operations are treated: type I in which more complex models are built and priorities dispatching rules or heuristic methods are presented. Type II in which simpler models are considered and strict solution methods or theory associated with them is proposed (Yokoyama 2008). Lee et al. (1993) studied assembly-type flow-shop scheduling problem. They studied a two-stage assembly flow-shop scheduling problem with makespan objective function by assuming that each product is assembled in two types of parts. The first component of each product must be processed on the first machine and the second component is processed on the second machine. Finally, the third machine assembles the two parts into a product. They proved that the problem is strongly NP-complete and identified several special cases of the problem, which could be solved in polynomial time and suggested a branch and bound solution and also three heuristics (Lee et al., 1993).

Potts et al. (1995) extended the problem for the case of multiple fabrication machines in which there are *m* machines and one machine at the first and the second stages, respectively. They developed a heuristic algorithm with a worst-case ratio bound to minimize the makespan. Hariri and Potts (1997) also studied the same problem as Potts et al. (1995) proposed a branch and bound algorithm. Cheng and Wang (1999) considered minimization of makespan for a two-machine flow-shop scheduling with a special structure, developed several properties of an optimal solution, and obtained optimal schedules for some special cases. In their model, the first machine produces two types of parts, unique components and common components. The unique components are processed individually, the common components are processed in batches, and a setup is required to form each batch. The second machine assembles components into products.

Sung et al. (2008) studied a 2-stage assembly scheduling problem subject to component available time constraint where the objective function is minimization of makespan. In their problem, there are n jobs (products), each product consists of two components. One of the two components is outsourced subject to job-dependent lead-time and a single machine fabricates the other one. They showed that makespan minimization is equivalent to minimization of idle time of assembly (second) stage and the problem is NP-complete in the strong sense. Hence, some solution properties are characterized based on which three heuristic algorithms are derived.

Yokoyama (2001) studied a hybrid scheduling for the production system including parts machining and assembly operations. In his study, several different products are ordered to be produced, parts for the products are manufactured in a flow shop, and each product is produced by hierarchical assembly operations from the parts. The parts are assembled into the first sub-assembly, and several other parts and the first sub-assembly are assembled into the second sub-assembly. These assembly operations are continued until the last sub-assembly that is the final product is obtained. In his model, the objective is minimum weighted sum of completion time and decision variables are the sequence of products to be assembled and the sequence of parts to be processed. He introduced a branch and bound with two lower bounds, which can solve problems for up to 10 products with 15 parts.

Sun et al. (2003) studied 3-machine, assembly-type flow shop scheduling. They proposed some heuristics based on the basic idea of Johnson's algorithm and Gupta's idea. The heuristic algorithms can solve all of the worst cases, which cannot be solved by the existing heuristic. Yokoyama and Santos (2005) considered flow-shop scheduling with assembly operations. In their models, several different products are ordered. Each part for the products is processed on machine M_1 (the first stage) and then processed on machine M_2 (the second stage). Each product is processed (e.g., joined) with the parts by one assembly operation on assembly stage M_A (the third stage). The objective function is the same as Yokoyama (2001). They developed a solution procedure to obtain an ε_optimal solution based on a branch-and-bound method.

Allahverdi and Al-Anzi (2009) studied a two-stage assembly scheduling problem where there are m machines at the first stage and an assembly machine at the second stage. In their model, the setup times are treated as separate from processing times. They presented a dominance relationship and proposed three heuristics including a hybrid tabu search, a self-adaptive differential evolution (SDE), and a new self-adaptive differential evolution (NSDE). The results indicated that the NSDE performed the best heuristic for the problem even if setup times were ignored.

Al-Anzi and Allahverdi (2009) also considered the same problem as Allahverdi et al. (2009) where setup times were ignored. They proposed some heuristics based on tabu search (Tabu), particle swarm optimization (PSO), and self-adaptive differential evolution (SDE) along with the earliest due date (EDD) and Johnson (JNS) heuristics to solve the problem. Computational experiment revealed that both PSO and SDE performed better than tabu. Moreover, it was statistically shown that PSO performed better than SDE.

Fttahi et al. (2012) studied a HFS scheduling problem followed by an assembly stage. They studied this problem in condition that a number of products of different kinds are produced and for each kind just one product is needed and hence the setup time is ignored. They presented a mathematical model for this problem and a series of heuristic algorithms based on the basic idea of Johnson's algorithm. The result indicated that the proposed algorithms had good efficiency for solving the considered problems.

To the best of our knowledge, there has been no study on assembly scheduling problem with hybrid flow shop for machining and setup time for the parts. Setup includes work to prepare the machine, process, or bench for product parts or the cycle. This includes obtaining tools, positioning work in process material; return tooling, cleanup, setting the required jigs and fixtures, adjusting tools, and inspecting material. Allahverdi et al. (1999) reviewed scheduling literature involving setup considerations. For a separable setup, two problem types exist. In the first type, setup depends only on the job to be processed; hence it is called sequence-independent. In the second, setup depends on both the job to be processed and the immediately preceding job; hence it is called sequence-dependent. Furthermore, treating setup time separately from processing time allows operations to be performed simultaneously and hence improves performance. This concept is inherent in recent production management philosophies and techniques such as just-in-time (JIT), optimized production technology (OPT), group technology (GT), cellular manufacturing (CM), and time-based competition (Allahverdi et al., 1999).

The rest of this paper is organized as follows: In section 2, the problem is described completely and it is clarified with a numerical example. The solving approach and procedure consist of the solution methodology, the heuristics algorithms based on GA, and SA will be expression in section 3. Three lower bounds are presented and improved in section 4. Design of problems, computational experiment, results, and the performance evaluation of the proposed algorithms are given in section 5. Finally, in section 6, concluding remarks and summary of the work are given and direction for the future research are offered.

2. Problem description

In this paper, an assembly production system including machining operations and assembly operations is studied. In this system, one kind of product with a quantity of demand is considered. Each product is made by assembling a set of several different parts. At first, the parts are manufactured in a hybrid flow-shop consisting of a single machines in stage 1 and m machines in stage 2. Setup operation and setup time are needed when a machine start processing the parts or it changes items. After manufacturing the parts, they are assembled into the products on an assembly stage. The assembly operation cannot be started for a product until the set of parts are completed in machining operations and the objective is to minimize the completion time of all products.

Machining, setup and assembly operations are performed based on blocking approach. Each block consists of ‘‘the machining operations and the setup operations for the parts of one or several products’’ and ‘‘the assembly operation(s) to assemble those parts into product(s)’’. Therefore, the concept of a block is quite different from a lot. The parts of the same item in a block are processed, successively. In this problem, there are three decision variables as follow:

- The number of blocks
- Block sizes that define the number of parts set in each block
- Sequencing the parts

Since there is only one machine in stage 1, there is no loading problem in this stage. In stage 2, every machine, which becomes idle sooner, will start new arrival part. One of the applications of this problem is body making of car manufacturing industrials. As shown in Fig. 1, a car making manufactory generally contains of the production engine, chassis and body. The body making includes press shop, assembly and painting. The press shop, which produces some parts such as doors and roofs has usually a flow shop or hybrid flow shop format.

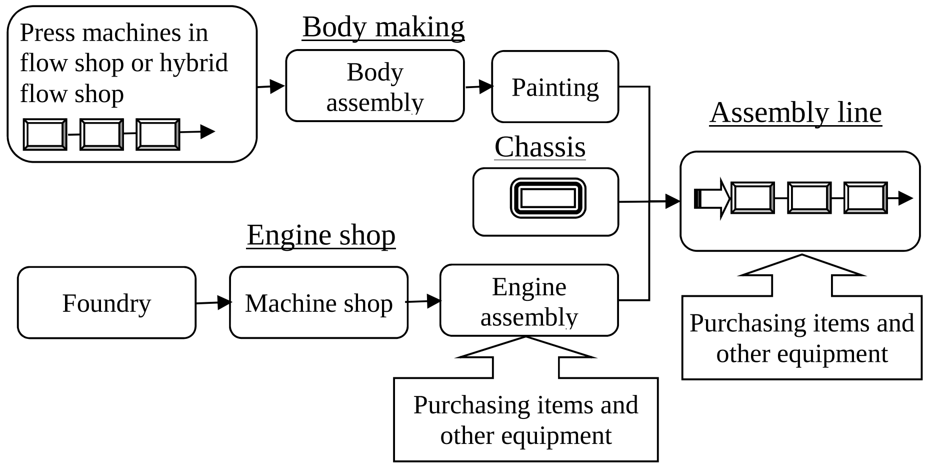

Press machines in flow shop or hybrid flow shop
Body making
Body assembly
Painting
Chassis
Assembly line
Engine shop
Foundry
Machine shop
Engine assembly
Purchasing items and other equipment
Purchasing items and other equipment

Fig. 1. A car making manufactory in generally

Fig. 2 shows a schematic view of an assembly production system including machining operations followed by an assembly section. The inputs contain row material, parts or unfinished products are processed on hybrid flow shop stage. When the set of parts for a product complete, they joined at assembly stage. Typically, buffers are located among stages to store intermediate products (Quadt et al., 2007) and it is assumed that there is no limited in buffer storages. The number of machines at hybrid flow shop is 1 and m machines at first and second stage respectively.

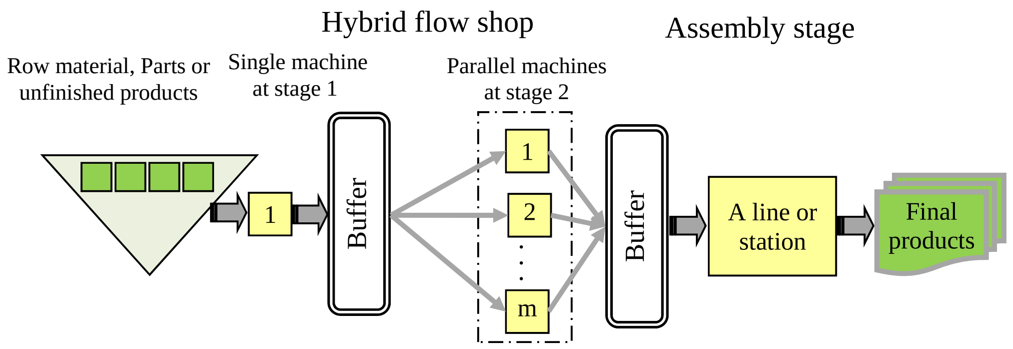

Hybrid flow shop
Assembly stage
Row material, Parts or unfinished products
Single machine at stage 1
Parallel machines at stage 2
1
Buffer
1
2
m
Buffer
A line or station
Final products

Fig. 2. A schematic view of the considered problem

2.1. Notations

The following notations are considered for this problem:

H	The number of products,
j	Item index of part (j = 1, 2, . . . ,/J/),
J	The set of item indices of parts,
l	Stage index (l =1,2),
S_{lj}	Setup time of a part of item j on stage l (l=1, 2),
P_{lj}	Processing time of a part of item j in stage l (l=1, 2),
m	Number of parallel machine in stage 2,
k	Machine index in stage 2,
A_h	Assembly time of a product,
C_h	Completion time of the hth product (h =1, 2, …, H).

Also decision variables of the problem are as follow:

B	The total number of blocks (parameter b will be item index of block),
ψ_b	Block size, that is, the number of products included in the bth block,

P_b	Sequence of item indices of parts in the b th block (we represent the sequence as a vector whose elements are item indices of parts)

The problem is to decide above three decision variables, and the objective function of problem is expressed as: $Z = \max(C_h)$

2.2. Assumptions

Basic assumptions of this paper are provided as follows,

- A number of a kind product is produced. A product is made by assembling a set of several different parts.
- All parts are available at time zero to process.
- The parts are manufactured in a two-stage hybrid flow-shop consisting of 1 machine in stage 1 and m uniform machines in stage 2.
- Setup operation and setup time are needed when a machine starts processing the parts or it changes items.
- Machining operations, setup operations and assembly operations are partitioned into several blocks. Each block consists of ‘‘the machining operations and the setup operations for the parts for one or several products’’ and ‘‘the assembly operation to assemble those parts into products’’. The parts of the same item in a block are processed successively.
- Sequencing of the parts is the same in all blocks.
- Each kind of part in each block is processed only by one machine in stage 2 of hybrid flow shop.
- Assembly operation for a product will not start until the set parts are completed.
- When assembly operation of a product is started, it doesn't stop until completed.
- There is no limited in buffer storages.

2.3. A numerical example

In order to clarify the problem, consider a simple numerical example as Table 1. Assume there is one machine in stage 1 and two machines in stage 2 in the hybrid flow shop. In addition, there is an assembly stage at the end of production system. Total number of products is $H = 4$ and the number of items of parts is /*J*/ = 3. At first, each part is processed in the hybrid flow shop. Then, the three parts are assembled into a product on assembly stage. The data for processing time of machining operation, assembly and setup time are given in Table 1.

Four solutions are shown in Fig. 3, Fig. 4, Fig. 5 and Fig. 6 for this problem. As shown in Fig. 3, the operations are partitioned into four blocks and each block consists of the machining operations, the setup operations and the assembly operations for one products. The completion time of products is 54 in this plan. In Fig. 4, the operations are partitioned into two blocks. Each block consists of the machining operations, the setup operations and the assembly operations for two products. The completion time of products is 50 in this plan. Fig. 5 shows another schedule of this problem. According to this schedule, the operations are partitioned into two blocks. The first block consists of the machining operations, the setup operations and the assembly operations for three products. In addition, the second block consists of the machining operations, the setup operations and the assembly operations for one product. The completion time of products is 49 in this plan. Fig. 6 shows the best schedule in which the blocks are the same as Fig. 5 but the scheduling of parts in blocks are different from Fig. 5. The completion time of products is 44 in this plan. The yellow color in Table 1 and Figs. 3-6 denote setup operations, the gray cells denote assembly operations and the other colors denote process operations of the parts.

Table 1
Processing time of machining and assembly and setup time

Operation and stages		*j*=1		*j*=2		*j*=3	
		Setup time	Process time	Setup time	Process time	Setup time	Process time
Hybrid flow shop	Stage 1	2	4	1	1	1	2
	Stage 2	1	2	2	3	2	4
Assembly		6					

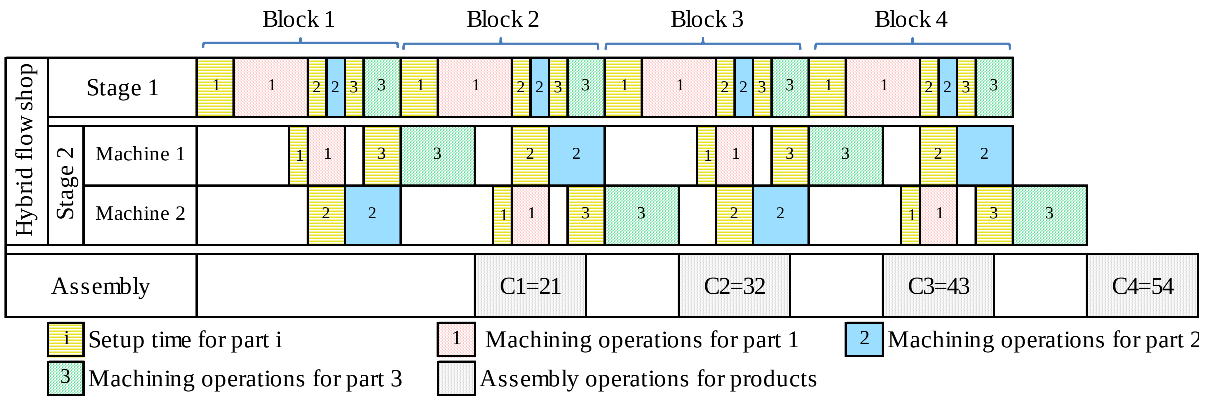


Fig. 3. A schedule for numerical example with C max = 54

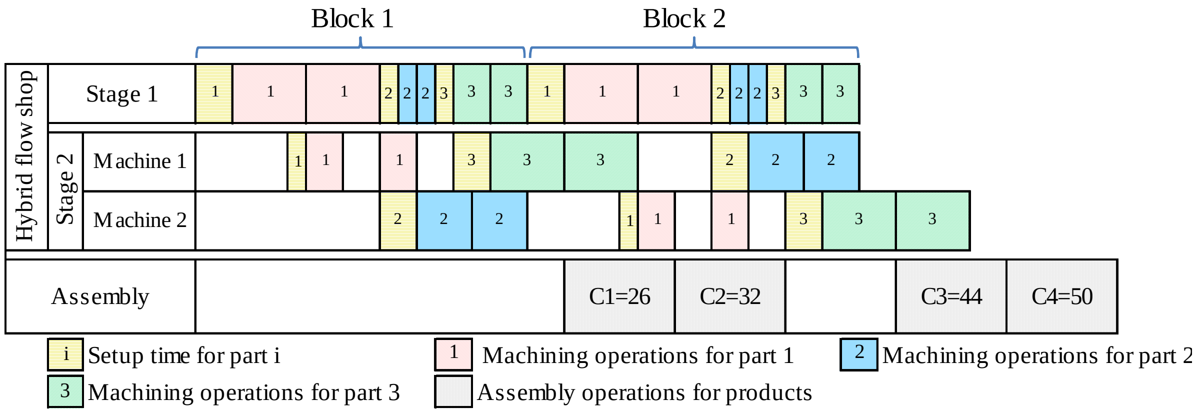


Fig. 4. A schedule of the numerical example with C max = 50

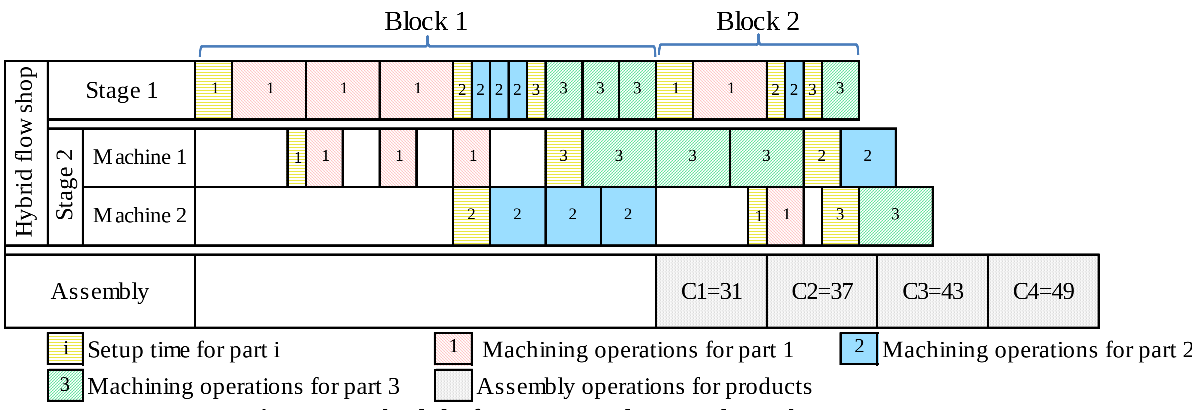


Fig. 5. A schedule for numerical example with C max = 49

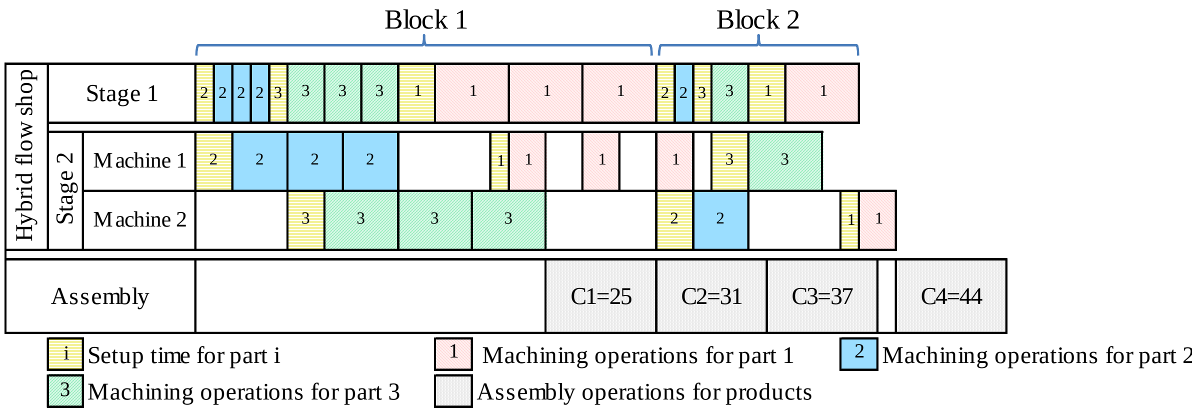


Fig. 6. The best schedule for numerical example with C max = 44

Table 2
Four schedules of the numerical example

Parameters	Figures									
	3				4		5		6	
Total number of blocks (B)	4				2		2		2	
Index of block (b)	1	2	3	4	1	2	1	2	1	2
Block size (ψ_b)	1	1	1	1	2	2	3	1	3	1
Sequence of parts (P_b)	1-2-3				1-2-3		1-2-3		2-3-1	
The amount of Z	54				50		49		44	

The results of Figs. 3 to 6 indicate that we can improve the objective function of this problem by changing three factors. 1. The number of blocks, 2.The size of blocks, and 3.The sequence of parts in every block. In Fig. 4 and Fig. 5, there are two blocks and so the objective function is better than Fig. 3 in which there are four blocks. Both Fig. 4 and Fig. 5 have two blocks, but the schedule of Fig. 5 is better because of the sizes of blocks. Finally, Fig. 6 with Z=44 is the best. The difference between Fig. 5 and Fig. 6 is in the sequence of the parts. These comparisons are performed under the condition that the sequence of the parts are the same in all blocks and also each block is processed only by one machine in stage 2 of hybrid flow shop.

The solutions show that in Fig. 6, the completion time has been improved more than 18.5% compared with Fig. 3. In addition, the idle time of machines in hybrid flow shop has been decreased more than 27.7% and the idle time of assembly stage has been decreased more than 33.3%.

3. The proposed solving approach and procedure

A taxonomy for flexible flow shop scheduling procedures is presented by Daniel et al. (2007), which is the first segmentation between optimal and heuristic procedures. The majority of optimal procedures for scheduling flexible flow lines are based on Branch & Bound algorithm. The heuristic algorithms are divided into holistic and decomposition approaches. In holistic approach, the complete scheduling problem is considered in an integrated and in decomposition approach; the overall scheduling problem is divided into segments, which are considered, consecutively. While the decomposition allows a simplification of the overall problem, it neglects the interdependencies among different segments. Therefore, decomposition approaches require strategies to effectively handle these interdependencies.

As mentioned before, the considered problem is strongly NP-hard and the computational time increases exponentially as the problem size increases, and hence, it is unlikely to find an optimal solution in reasonable amount of time. Therefore, some heuristics and metaheuristics are developed and justified in holistic approach for the considered problem. Metaheuristic search methods include local search or neighborhood search algorithms, which are iterative procedures and start with an initial solution and browse through the search space, picking up better solutions on the way, which could give good, or eventually optimal, solutions (Schneider 2011). Therefore, this study considered the application of a genetic algorithm (GA), a simulated annealing (SA) and a heuristic algorithm to solve the considered problem with minimization of the makespan. Both GA and SA have been successfully used in solving scheduling problems (Gaafar et al., 2005).

3.1. Solution methodology classification: N/S/Q

The main decision variables of the considered problem are the number of blocks, the size of each block and sequence of the parts considered fixed in all blocks. The assignment parts to machines are ignored because in stage 1, there is only one machine and in stage 2, it is assumed that the first idle machine will process the first part that arrives from stage 1. The notation $N/S/Q$ of Kumar et al. (2000) is modified for the considered problem. This notation is used throughout this paper to indicate the decision methodology used in each of three variables where:

N is method used for deciding the number of blocks,
S is method used for sizing each block,
Q is method used for sequencing the parts in each block.

For example, Fixed/SA/NEH indicates that there are fixed numbers of blocks, SA is used to obtain the size of blocks, and lastly, the sequence of the parts is determined by the NEH algorithm. As regards the size of blocks is dependent on the number of blocks, the size of blocks has to be determined after deciding the number of blocks. Therefore, a sweep method is used to obtain the best amount for number of blocks. Figure 7 shows an overview of solution methodology and its steps are explained in this paper.

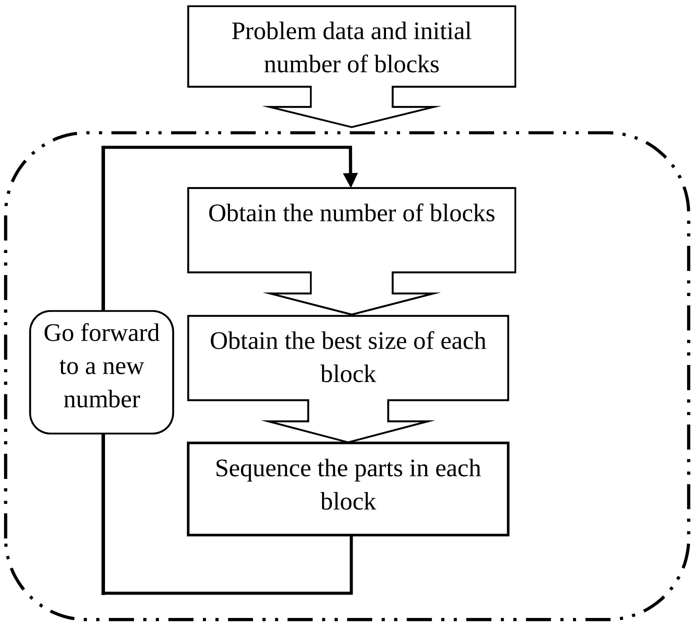

Problem data and initial number of blocks
Obtain the number of blocks
Go forward to a new number
Obtain the best size of each block
Sequence the parts in each block

Fig. 7. An overview of the proposed solution methodology

Hence the *SH* heuristic algorithm described below is used to obtain the initial value for the number of blocks and improved it.

3.2. SH heuristic algorithm for determining the number of blocks

In order to determine the number of blocks, a new heuristic algorithm is proposed. This algorithm named *SH* heuristic algorithm, scan the scope of the possible number of blocks with search and test less than 15 percent of total possible number. This algorithm is described step by step as below:

```
Step1: consider $B_1 = 1$ and $B_2 = H$ are two amounts for number of blocks and calculate the makespan according them that is called $M_1$and $M_2$ respectively.
Step 2:     While (round $|B_1 - B_2|$) $> 1$ do the following.
    If $M_2 \geq M_1$
        $B_2 = (B_1 + 3 \times B_2)/4$
        Calculate the new amounts of $M_2$
    Else
        $B_1 = (3 \times B_1 + B_2)/4$
        Calculate the new amounts of $M_1$
    End
Step 3:     Return the best number for blocks $(Min(B_1,\ B_2))$.
```

The initial values for the number of blocks are 1 and *H*. It is clear that when we have only one block, that block will contain all parts, which must be processed one by one in two-stage of hybrid flow shop. On the other hand, when the number of blocks is *H*, each block contains a set of parts that is needed for one product. It is clear that the size of each block will be 1. Each time, *SH* algorithm is run and finds a new number for blocks, two algorithms based on SA and GA are used. The implementation is illustrated in section 3.3 to determine the size of each block and after that two algorithms based on *NEH* and Johnson is used as it is stated in section 3.4 for sequencing the parts for all blocks.

3.3 The proposed algorithms for sizing each block

In order to determine the size of each block, we propose two algorithms based on SA and GA.

3.3.1 The proposed SA algorithm

Simulated annealing (SA) is a neighborhood search technique and it has produced good results for combinatorial problems. A standard SA procedure begins by generating an initial solution, randomly. At each stage, the new solution taken from the neighborhood of the current solution is accepted as the new current solution if it has a lower or equal cost; if it has a higher cost, it is accepted with a probability that decreases as the difference in the costs increases and as the temperature of the method decreases. This temperature, which is simply a positive number, is periodically reduced by a temperature scheme, so that it moves gradually from a relatively high value to near zero as the method progresses. Thus, at the start of SA, most deteriorating moves are accepted, but at the end only ameliorating ones are likely to be accepted. The method converges to a local optimum as the temperature approaches zero, but because SA has performed many perturbations at higher temperatures, which have pushed the search path into new areas, a better local optimum solution should hopefully be reached. In this paper, the entire SA procedure is terminated when the temperature reaches a pre-specified value T_f. In this section, a simulated annealing algorithm is presented to find good solutions for the block sizing problem. Therefore, we use a matrix $S_{1\times B}$ to represent the size of blocks. The members of this matrix are $\psi_1, \psi_2, \psi_3, \dots, \psi_B$ with two below conditions:

$$\sum_{b=1}^{B} \psi_b = H$$

$$\psi_b \geq 1 \quad \text{for } b = 1, 2, 3, \dots, B$$

The initial block sizing is performed with a random operator that assigns a size to each block considering two above conditions.

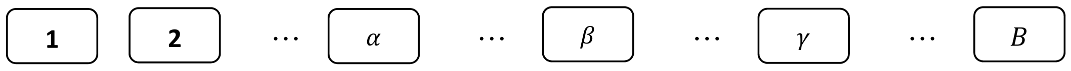
Assignment algorithm

```
Step 1: Initialization
    Step 1.1: Obtain an initial block sizing ($F^C$ and $F^*$).
    Step 1.2: Initiate the initial temperature ($T_0$, $T_C = T_0$), final temperature ($T_f$), cooling rate ($r$), and iteration number in each temperature ($L$).
Step 2: While not yet frozen ($T_C \succ T_f$), do the following.
    Step 2.1: Perform the following loop $L$ times.
        Step 2.1.1: Done *neighborhood search algorithm* and select a neighbor $F^{C'}$ of $F^C$.
        Step 2.1.2: Compute $\Delta = R_{F^{C'}} - R_{F^C}$ (done the computation of makespan)
        Step 2.1.3: If $\Delta \leq 0$ Then $F^C = F^{C'}$.
                        Compute $\Delta_b = R_{F^C} - R_{F^*}$.
                        If $\Delta_b < 0$ set $F^* = F^C$.
        Step 2.1.4: If $\Delta > 0$ select a random variable $P \backsim U(0,1)$.
                    If $e^{-\Delta/T} > P$ set $F^C = F^{C'}$
    Step 2.2: Set $T_C = r \times T_C$
Step 3: Return the best solution found for F*.
```

Neighborhood search algorithm

A neighborhood search is proposed in this section. This algorithm has two steps as follow:

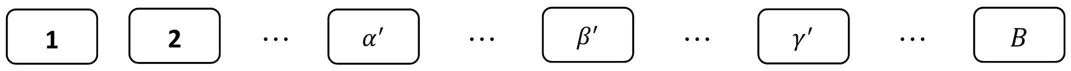
Step 1: select three blocks from B exist blocks. We call these three blocks as α , β , γ



Step 2: change the size of blocks α , β , γ and define the new size as below:

1. Select $\psi_{\alpha'}$	$1 \leq \psi_{\alpha'} \leq \psi_\alpha + \psi_\beta + \psi_\gamma - 2$
2. Select $\psi_{\beta'}$	$1 \leq \psi_{\beta'} \leq \psi_\alpha + \psi_\beta + \psi_\gamma - \psi_{\alpha'} - 1$
3. determine $\psi_{\gamma'}$	$\psi_{\gamma'} = \psi_\alpha + \psi_\beta + \psi_\gamma - \psi_{\alpha'} - \psi_{\beta'}$

Now we have new three new blocks that different with their old only in size.



In this neighborhood structure, all blocks have a chance to be selected and changed. Also all neighbors are placed in a feasible region. Through some experiments the best parameter for SA were obtained as follow:

- ✓ The initial temperature $T_0 = 1000$
- ✓ The final temperature $T_f = 10$
- ✓ The cooling rate $r = 0.95$
- ✓ The number of iteration in each temperature $L = 25$

3.3.2 The proposed GA algorithm

A GA (Holland, 1975) is a search technique that imitates the natural selection and biological evolutionary process. GA has been used in a wide variety of applications, particularly in combinatorial optimization problems and they were proved to be able to provide near optimal solutions in reasonable time (Anandaraman, 2011; Luo et al., 2011; Chakrabortty et al., 2013). A GA starts with a population of randomly generated candidate solutions (called chromosomes). A chromosome is represented by a string of numbers called genes. Each chromosome in the population is evaluated according to some fitness measure, which reflects the objective function value and the satisfaction of problem constraints. The better the fitness value, the more chances are given for the individual to be selected as a parent. New chromosomes are built of a reproduction operator that usually consists of crossover and mutation procedures. The crossover procedure produces the offspring from two parent individuals by combining and exchanging their elements. Certain pairs of chromosomes are selected on the basis of their fitness. Each of these pairs combines to produce new chromosomes (offspring) and some of the offspring are randomly modified. The mutation procedure adds small random changes to a chromosome (Maniezzo et al., 2009).

A new population is then formed replacing some of the original population by an identical number of offspring and the process is repeated until a stopping criterion is met.

In this paper, detail of the proposed GA with necessary illustrations is considered as follow. Each solution (chromosome) is represented as a vector $1 \times \mathrm{B}$ whose elements (genes) are item indices the size of *b*th block. An initial population of chromosomes is randomly generated. After some experiments, the best population size is obtained 20. The evaluation parameter *f(c)* is the value of objective function. This value is used to measure the fitness of a chromosome. For each partial (chromosome), block sizing is constructed and the makespan of jobs is calculated. The roulette wheel selection and tournament selection without replacement are used to select two chromosomes for crossover. Marimuthu et al. (2008) illustrated this selection method in a genetic algorithm for scheduling *m*-machine flow shop with lot streaming. In this paper, initially, the probability of choosing each chromosome $\mathrm{p(c)}$ is estimated as follows,

$$p(c) = \frac{f^*(c)}{\sum_{c=1}^{pop\_size} f^*(c)} \qquad \text{where } f^*(c) = 1/f(c)$$

Then, the cumulative probability $cp(c)$ for all chromosomes is found out.

$$\mathrm{c}p(c) = \sum_{c=1}^{c} p(c)$$

Then the random number 'r' between 0 and 1 is spun and a chromosome 'c' is selected, satisfying the following condition:

$$\mathrm{c}p(c-1) \le r < \mathrm{c}p(c)$$

This selection process is repeated as many times as the number of parents. The two-point crossover operator is applied to each pair of parent chromosomes with a probability of $\mathrm{p_c}$ (probability of crossover). From sensitivity analysis, probability of crossover is arrived as 0.85. According to the results of experiments, two kinds of mutations are used in this algorithm. According to the first one, the gene being considered is removed from its position and put into another randomly chosen position of the same chromosome with a probability ${}^{\mathrm{p_m}}/_{2}$, where $\mathrm{p_m}$ is mutation probability. According to the second one, two genes (blocks) α, β are chosen randomly from each chromosome with a probability ${}^{\mathrm{p_m}}/_{2}$ and their sizes are changed as below:

$$1 \le \psi_{\alpha'} \le \psi_\alpha + \psi_\beta - 1\ ,$$

$\psi_{\beta'} = \psi_\alpha + \psi_\beta - \psi_{\alpha'}$.

For the considered problem, this mutations scheme work better than other mutations methods. The probability of mutation p_m is arrived as 0.20 through trials. For each chromosome, a random number is generated ($0 < r < 1$). If the random number is less than 0.1 the chromosome is selected for mutation kind 1 and if the random number is between 0.1 and 0.2, the chromosome is selected for the second type of mutation. Seventy percent of chromosomes are selected from the present population and thirty percent of chromosomes are chosen from children (offspring) as new population and generation. These chromosomes are selected on the basis of their fitness (minimum evaluation makespan). Finally, the stopping criterion is based on the number of iteration without improvement that is considered up to 15 iterations. Fig. 8 shows a view of the proposed GA for determination the size of blocks.

3.4. NEH and Johnson's algorithm for sequencing the parts in each block

Two methods are used for sequencing the parts and it is supposed that the sequence of parts is the same in all blocks. One method is Johnson's algorithm and the other method is *NEH*. Based on Johnson's algorithm, the parts are sorted into two sets. Set U sorts the parts whose processing time on stage 1 is less than stage 2 in non-decreasing in P_{1j} and set *V* sorts the parts whose processing time on stage 1 is not less than stage 2 in non-increasing order in P_{2j}. The sequence of the parts is the set of *U* followed by *V* [8, 28]. According to algorithm *NEH* (Nawaz et al., 1983), the parts are sorted in non-increasing order of processing time. Table 3 summarizes these four heuristics,

Table 3
Four algorithms with their parameters

Algorithm H_i	Determination the number of blocks	Determination the size of blocks	Scheduling the parts
H_1 (SH/SA/Johnson)	SH	SA	Johnson
H_2 (SH/SA/NEH)	SH	SA	NEH
H_3 (SH/GA/ Johnson)	SH	GA	Johnson
H_4 (SH/GA/NEH)	SH	GA	NEH

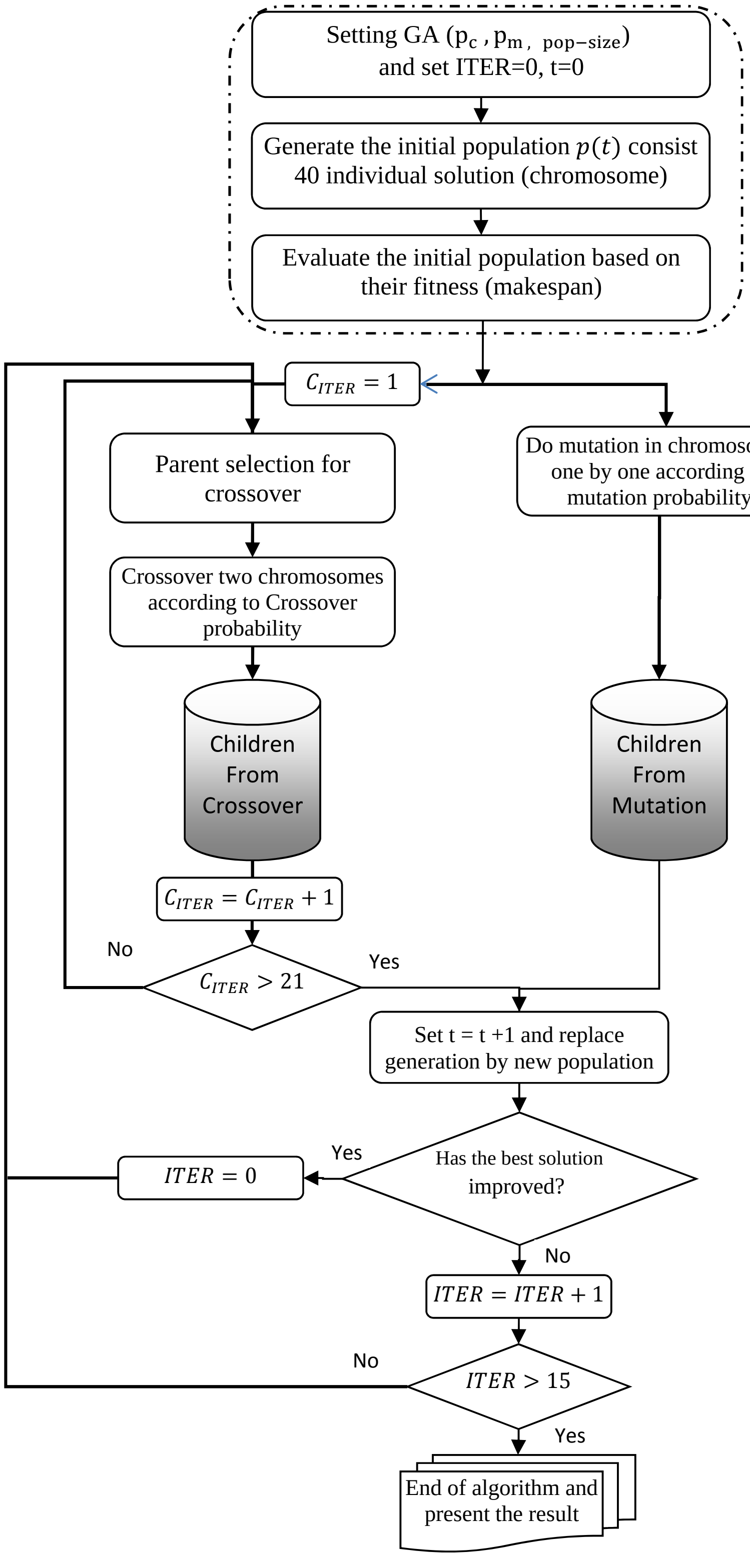
4. Three proposed lower bounds

In order to evaluate the heuristic solutions, three lower bounds of the makespan are introduced and improved for the considered problem. These lower bounds are improved in three conditions: a. when assembly stage is bottleneck, b. when the first stage of hybrid flow shop is bottleneck and c. when the second stage of hybrid flow shop is bottleneck. It is known that the SPT rule minimizes the total completion time in the case of parallel machines (Pinedo, 2002; Kadipasaoglu, 1997; Walter, 2011). Thus in order to present and improving the lower bounds, the SPT rule is used in this section.

4.1. When the first stage of hybrid flow shop is bottleneck

In case that the amount of processing time and/or set up time on the first stage is more than the second and assembly stage, the first stage will be bottleneck. In this condition, the second and assembly stage will always be waiting and the majority of makespan is depended on the first stage. Thus, the SPT that is computed as (1) is a good lower bound for problem in this case.

$$LB_1 = \sum_{j=1}^{/J/} S_{1j} + H * (\sum_{j=1}^{/J/} P_{1j}) + \min_j(P_{2j}) + A_h \tag{1}$$



Setting GA (p_c, p_m, pop−size) and set ITER=0, t=0
Generate the initial population $p(t)$ consist 40 individual solution (chromosome)
Evaluate the initial population based on their fitness (makespan)
$C_{ITER} = 1$
Parent selection for crossover
Do mutation in chromosomes one by one according to mutation probability
Crossover two chromosomes according to Crossover probability
Children From Crossover
Children From Mutation
$C_{ITER} = C_{ITER} + 1$
No
$C_{ITER} > 21$
Yes
Set t = t +1 and replace generation by new population
Yes
$ITER = 0$
Has the best solution improved?
No
$ITER = ITER + 1$
No
$ITER > 15$
Yes
End of algorithm and present the result

Fig. 8. GA model for determination the size of blocks

4.2. When the second stage of hybrid flow shop is bottleneck

Like before, when the second stage is bottleneck, the jobs will always have idle times to start at this stage (exception the first parts that is assign in primary positions) and assembly stage will always be waiting. In this condition, the majority of makespan is depended on the second stage of the hybrid flow shop. Thus, the SPT that is computed as Eq. (2) is a good lower bound for the considered problem in this case.

$$LB_2 = max\left\{\left(\frac{H*\left(\sum_{j=1}^{/J/} P_{2j}\right)+\sum_{j=1}^{/J/} S_{2j}}{m}\right), \left[\underset{j}{min}\left(S_{1j} + P_{1j}\right) + \frac{H*\sum_{j=1}^{/J/} P_{2j}}{m}\right]\right\} + A_h. \tag{2}$$

4.3. When the assembly stage is bottleneck

When the assembly stage is bottleneck, the total completion time at the assembly stage can be considered as a lower bound. In other words, the processing time of the pre-assembly stage is not considered, and the total completion time at the assembly stage is always no larger than the total completion time of any feasible schedule (Sung et al., 2008). This time is shown as Eq. (3).

$$\sum_{h=1}^{H} A_h = H * A_h \tag{3}$$

It is clear that there is always an idle time in the assembly stage to ready the first set of parts in preassembly stage. Therefore, in order to improve the lower bound by considering this idle time, a modified SPT rule is proposed. In other word, the minimum idle time for assembly stage is equal to minimum completion time of a set of parts for one product. So, this shortest process time can be computed as Eqs. (4-6) whichever is maximum dependent on condition:

$$SPT_1 = \sum_{j=1}^{/J/} S_{1j} + \sum_{j=1}^{/J/} P_{1j} + \underset{j}{min}(P_{2j}) \tag{4}$$

$$SPT_2 = \underset{j}{max}\left(max\left(\left(S_{1j} + P_{1j}\right), S_{2j}\right) + P_{2j}\right) \tag{5}$$

$$SPT_3 = max\left[\left(\underset{j}{min}\left(S_{1j} + P_{1j}\right) + \frac{\sum_{j=1}^{/J/} P_{2j}}{m}\right), \frac{\sum_{j=1}^{/J/} S_{2j}+\sum_{j=1}^{/J/} P_{2j}}{m}\right] \tag{6}$$

Therefore, the third lower bound will be obtained by adding the assembly time of all products to the maximum of SPT_i as below:

$$LB_3 = max\{SPT_1, SPT_2, SPT_3\} + H * A_h \tag{7}$$

If the assembly stage is bottleneck, then the parts always have some idle times to start at the assembly stage (exception the set of parts of the first product) and assembly stage will be always busy during flow time after completion all parts of the product on the first position. So, the total completion time of optimum solution can be close to the LB_3. Therefore, three lower bounds LB_1, LB_2, and LB_3 are used to cover all conditions and various process times and assembly times and the final lower bound (LB) will be defined as (8):

$$LB = max\{LB_1, LB_2, LB_3\} \tag{8}$$

5. Computational experiment and results

In this section, the computational experiment is carried out in order to evaluate the performance of the proposed heuristic algorithms. The tests have been performed on various condition of problem.

Therefore, first, the problems are designed and then comparison results of the proposed heuristic algorithms are presented. The present algorithms and lower bounds are coded in MATLAB 7/10/0/499 (R2010a). The experiments are executed on a Pc with a 2.0GHz Intel Core 2 Duo processor and 1GB of RAM memory.

5.1. Design of problems

In order to evaluate the proposed algorithms, 36 problems are designed in various conditions. For this, the following parameters are considered to generate randomly the test problems:

Numbers of jobs (H): 50, 100, 200, 500
Setup time of stage 1 (S_{1j}): generated from the discrete uniform distribution with range [50, 150].
Setup time of stage 2 (S_{2j}): generated from the discrete uniform distribution with range [100, 200].
Processing times of parts on stage 1 (P_{1j}): generated from the discrete uniform distribution with range [0, 100].
Processing times of parts on stage 2 (P_{2j}): generated from the discrete uniform distribution with range [200, 400].
Assembly times of a product (A_h): generated from the discrete uniform distribution with range [500, 700].
Number of a set parts (*/J/*): generated from the discrete uniform distribution with three ranges [3, 7], [7, 13], [10, 20].
Number of machine in second stage of HFS (*m*): 3, 5, 7

The distributions of parameters provide a good scope for problems and heuristic algorithms can be tested and evaluated under various conditions. According to the above parameters, 36 types of problems are generated and these problems are grouped into three classes. Class *A* contains the problems in which fabrication stage (hybrid flow shop) is bottleneck. Class *B* presents problems in which assembly stage is a bottleneck workstation. In problems of class *C,* there is no bottleneck and the setup and process time in stages 1 and 2 of hybrid flow shop are approximately equal to assembly times. These three classes of problems and other information about generated problems are considered and they demonstrated in Table 4.

5.2. Comparisons of results

This section presents the results of the heuristic methods described in Section 3. Each problem of Table 4 has been run ten times by each algorithm. Then computational experiments are carried out to evaluate the effectiveness of the proposed heuristic algorithms. In order to evaluate the performance of heuristic algorithms, we use the lower bounds of $\mathrm{C}_{\max}$ derived in Section 4. Based on the lower bounds of C_{max}, the percentage of deviation is defined as below:

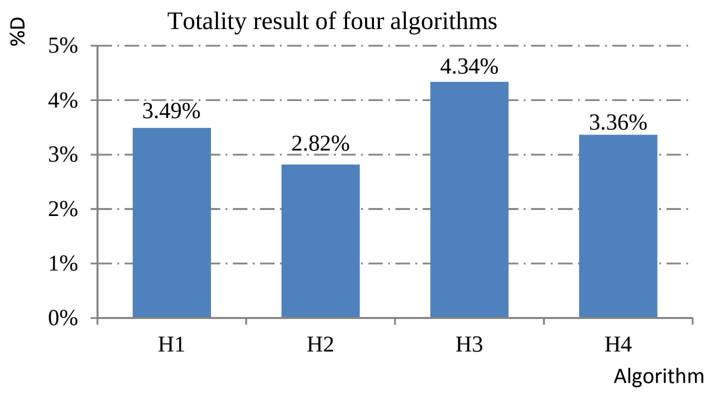
$$\%D = \{[C_{max}(H_i) - LB]/LB\} * \%100,$$

where $C_{max}(H_i)$ denotes the makespan of the schedule generated by heuristic algorithm H_i, for i= 1, 2, 3, 4 as described in section 3. Also LB presents the maximum of LB_i in each problem as described in section 4. The average results obtained of ten runs for each problem is presented in Table 6 of each problem and algorithm. The performances of each algorithm are presented in Fig. 9 and Table 5. As we can observe in panel I of Fig. 9, algorithm H_2 has the best performance with 2.82 deviation percentage from lower bound. That is GA is better than SA in sizing the blocks and the Johnson's algorithm is better than *NEH* in sequencing the parts for two stages of hybrid flow shop. Panel II shows that this rule is current for all three classes of problems, unless in class C. In class C of problem where there is a balance condition in production system, algorithm H_1 that uses GA for sizing the blocks and *NEH* for sequencing the parts has a better performance than algorithm H_4 that uses SA for sizing the blocks and Johnson's algorithm for sequencing the parts. Performances of four algorithms are presented in Table

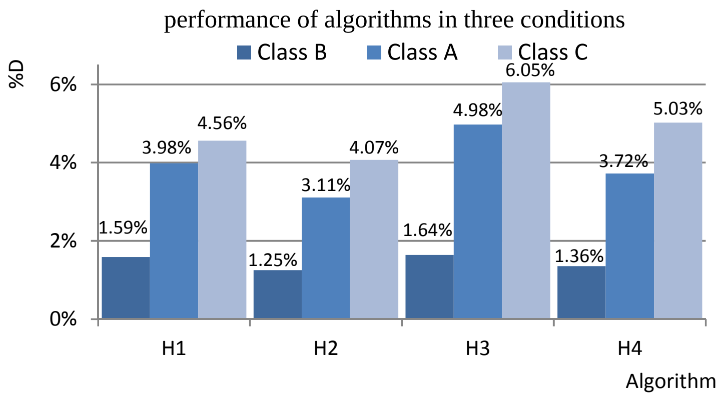
5. Table 5 shows that the performance of algorithms is kept during changing the size of problems. As seen from the Fig. 10, LB_3 depends on assembly time and it is constant approximately because the assembly time is supposed to be fixed in uniform distribution of range [500 , 700]. Two lower bounds LB_1 and LB_2 depend on process time of the first and the second stage of hybrid flow shop, respectively. In other word, the main amount of LB_1 depends on process time of parts in the first stage and the main amount of LB_2 depends on process time of parts in the second stage. It is clear that when the hybrid flow shop (fabrication stage) is bottleneck, LB_1 or LB_2 will be effective lower bound (class A). When assembly stage is bottleneck, the LB_3 is effective lower bound (class B). Finally, in balance condition, three lower bounds are equal approximately.

Table 4
Four generated classes of test problems

Problem	H	/J/	m	S_{1j}	P_{1j}	S_{2j}	P_{2j}	A_h	Class
H-A1	50	[3 , 7]	3	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	B
H-A2	50	[3 , 7]	5	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	B
H-A3	50	[3 , 7]	7	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	B
H-A4	50	[7 , 13]	3	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	A
H-A5	50	[7 , 13]	5	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	C
H-A6	50	[7 , 13]	7	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	C
H-A7	50	[10 , 20]	3	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	A
H-A8	50	[10 , 20]	5	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	A
H-A9	50	[10 , 20]	7	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	A
H-A10	100	[3 , 7]	3	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	B
H-A11	100	[3 , 7]	5	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	B
H-A12	100	[3 , 7]	7	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	B
H-A13	100	[7 , 13]	3	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	A
H-A14	100	[7 , 13]	5	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	C
H-A15	100	[7 , 13]	7	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	C
H-A16	100	[10 , 20]	3	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	A
H-A17	100	[10 , 20]	5	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	A
H-A18	100	[10 , 20]	7	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	A
H-A19	200	[3 , 7]	3	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	B
H-A20	200	[3 , 7]	5	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	B
H-A21	200	[3 , 7]	7	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	B
H-A22	200	[7 , 13]	3	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	A
H-A23	200	[7 , 13]	5	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	C
H-A24	200	[7 , 13]	7	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	C
H-A25	200	[10 , 20]	3	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	A
H-A26	200	[10 , 20]	5	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	A
H-A27	200	[10 , 20]	7	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	A
H-A28	500	[3 , 7]	3	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	B
H-A29	500	[3 , 7]	5	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	B
H-A30	500	[3 , 7]	7	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	B
H-A31	500	[7 , 13]	3	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	A
H-A32	500	[7 , 13]	5	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	C
H-A33	500	[7 , 13]	7	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	C
H-A34	500	[10 , 20]	3	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	A
H-A35	500	[10 , 20]	5	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	A
H-A36	500	[10 , 20]	7	[50, 150]	[0, 100]	[100, 200]	[200, 400]	[500, 700]	A



Panel I. Performance of algorithms in totality



Panel II. Performance of algorithms in three classes of problems

Fig. 9. Totally performance of four algorithms in scheduling the parts

Table 5
Comparison of results (%D) in problems with various sizes

Number of jobs (*H*)	Algorithm			
	H_1	H_2	H_3	H_4
50	7.4%	6.4%	8.5%	6.9%
100	5.4%	4.4%	6.6%	5.3%
200	3.2%	2.6%	4.5%	3.6%
500	2.8%	2.2%	3.4%	2.5%

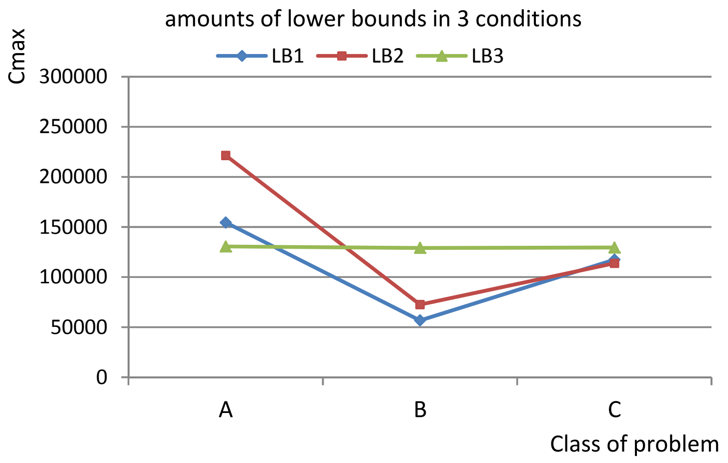


Fig. 10. Variations of lower bounds in three condition of bottleneck

Fig. 10 shows that when fabrication stage (hybrid flow shop) is bottleneck, the LB_2 is more active than LB_1. Table 7 presents the amount of lower bounds and the best solution with details. In this table, the red cells show the maximum lower bound (active lower bound). As seen from this table, in 79% of problem with class A the LB_2 presented better amount than LB_1 and only in 21% problem LB_1 is better. Orange cells show the worst result of the best algorithm in four sizes of problem. This table also shows that the worst performance of algorithms is obtained in condition that LB_1 is active lower bound. These results state that LB_1 was a weak lower bound and so it will be a good study to improve this lower bound.

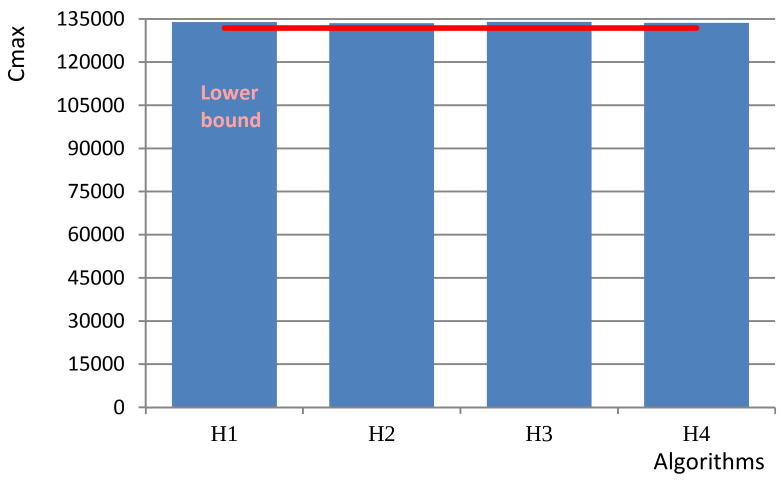
Table 6
Deviation percentage (%D) of solutions from lower bound in 36 problems

Problem name	H	/J/	*m*	Class	H_1	H_2	H_3	H_4
H-A1	50	[3 , 7]	3	B	3.6%	3.2%	4.3%	3.5%
H-A2	50	[3 , 7]	5	B	1.2%	1.0%	1.6%	1.2%
H-A3	50	[3 , 7]	7	B	1.2%	1.2%	1.4%	1.4%
H-A4	50	[7 , 13]	3	A	7.6%	6.1%	8.6%	6.4%
H-A5	50	[7 , 13]	5	C	10.8%	9.5%	12.5%	10.8%
H-A6	50	[7 , 13]	7	C	10.1%	9.2%	11.7%	10.5%
H-A7	50	[10 , 20]	3	A	5.8%	4.6%	6.2%	4.5%
H-A8	50	[10 , 20]	5	A	12.0%	9.7%	13.8%	10.8%
H-A9	50	[10 , 20]	7	A	12.9%	11.7%	14.7%	12.9%
H-A10	100	[3 , 7]	3	B	1.8%	1.6%	2.3%	2.3%
H-A11	100	[3 , 7]	5	B	0.6%	0.6%	0.9%	0.7%
H-A12	100	[3 , 7]	7	B	0.5%	0.4%	0.5%	0.5%
H-A13	100	[7 , 13]	3	A	5.5%	4.6%	6.6%	5.4%
H-A14	100	[7 , 13]	5	C	7.7%	6.2%	9.9%	8.4%
H-A15	100	[7 , 13]	7	C	8.1%	7.0%	10.2%	8.8%
H-A16	100	[10 , 20]	3	A	3.5%	2.7%	4.0%	2.8%
H-A17	100	[10 , 20]	5	A	7.5%	4.9%	9.3%	5.6%
H-A18	100	[10 , 20]	7	A	12.8%	11.4%	14.9%	13.5%
H-A19	200	[3 , 7]	3	B	2.2%	2.4%	3.8%	3.4%
H-A20	200	[3 , 7]	5	B	0.7%	0.4%	0.8%	0.7%
H-A21	200	[3 , 7]	7	B	0.3%	0.2%	0.4%	0.4%
H-A22	200	[7 , 13]	3	A	3.6%	2.7%	4.6%	3.3%
H-A23	200	[7 , 13]	5	C	4.2%	3.6%	6.4%	5.5%
H-A24	200	[7 , 13]	7	C	3.9%	3.6%	6.2%	5.9%
H-A25	200	[10 , 20]	3	A	2.2%	1.6%	2.7%	1.9%
H-A26	200	[10 , 20]	5	A	4.8%	3.5%	6.8%	4.5%
H-A27	200	[10 , 20]	7	A	6.2%	5.4%	9.0%	7.7%
H-A28	500	[3 , 7]	3	B	3.6%	3.0%	4.0%	3.2%
H-A29	500	[3 , 7]	5	B	1.3%	0.8%	0.5%	0.4%
H-A30	500	[3 , 7]	7	B	0.7%	0.3%	0.3%	0.2%
H-A31	500	[7 , 13]	3	A	3.1%	2.2%	3.5%	2.5%
H-A32	500	[7 , 13]	5	C	4.0%	3.4%	5.5%	4.0%
H-A33	500	[7 , 13]	7	C	2.9%	3.0%	3.5%	2.9%
H-A34	500	[10 , 20]	3	A	1.5%	0.8%	1.6%	0.8%
H-A35	500	[10 , 20]	5	A	3.6%	2.6%	5.0%	3.3%
H-A36	500	[10 , 20]	7	A	5.7%	5.4%	7.8%	6.8%

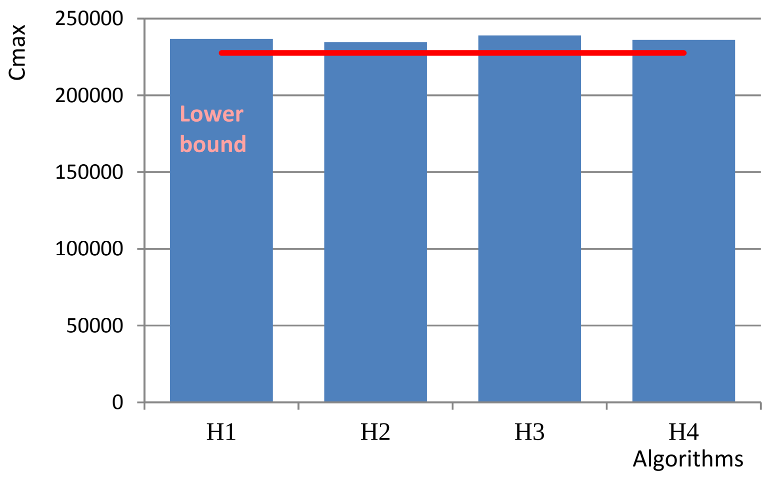
Table 7
The performance of lower bounds and the best solution in 36 problems

Problem name	H	/J/	m	Class	LB_1	LB_2	LB_3	The best solution	%D of the best algorithm
H-A1	50	[3 , 7]	3	B	15565	27504	30604	31583	3.2%
H-A2	50	[3 , 7]	5	B	11383	12630	31252	31564	1.0%
H-A3	50	[3 , 7]	7	B	13928	11167	30056	30416	1.2%
H-A4	50	[7 , 13]	3	A	26140	53559	30514	56826	6.1%
H-A5	50	[7 , 13]	5	C	27584	27293	35303	38657	9.5%
H-A6	50	[7 , 13]	7	C	27882	22968	32868	35892	9.2%
H-A7	50	[10 , 20]	3	A	41449	74062	32460	77395	4.5%
H-A8	50	[10 , 20]	5	A	42230	43081	32749	47259	9.7%
H-A9	50	[10 , 20]	7	A	39965	32180	32747	44641	11.7%
H-A10	100	[3 , 7]	3	B	23861	49141	59291	60240	1.6%
H-A11	100	[3 , 7]	5	B	33459	33638	62934	63311	0.6%
H-A12	100	[3 , 7]	7	B	26479	21222	60854	61097	0.4%
H-A13	100	[7 , 13]	3	A	56268	94093	62347	98421	4.6%
H-A14	100	[7 , 13]	5	C	49409	67520	59072	71706	6.2%
H-A15	100	[7 , 13]	7	C	43239	37846	63277	67706	7.0%
H-A16	100	[10 , 20]	3	A	77542	156961	62837	161199	2.7%
H-A17	100	[10 , 20]	5	A	81847	99420	63888	104292	4.9%
H-A18	100	[10 , 20]	7	A	78537	69334	64749	87491	11.4%
H-A19	200	[3 , 7]	3	B	50530	100015	126206	128983	2.2%
H-A20	200	[3 , 7]	5	B	63304	66319	122221	122710	0.4%
H-A21	200	[3 , 7]	7	B	53378	41842	121036	121278	0.2%
H-A22	200	[7 , 13]	3	A	99606	186509	124095	191545	2.7%
H-A23	200	[7 , 13]	5	C	122771	135087	116323	139950	3.6%
H-A24	200	[7 , 13]	7	C	105122	91184	130578	135279	3.6%
H-A25	200	[10 , 20]	3	A	151690	314375	122893	319405	1.6%
H-A26	200	[10 , 20]	5	A	176524	195048	121541	201875	3.5%
H-A27	200	[10 , 20]	7	A	161244	138384	123115	169952	5.4%
H-A28	500	[3 , 7]	3	B	103811	220072	317557	327084	3.0%
H-A29	500	[3 , 7]	5	B	159324	186445	312845	314097	0.4%
H-A30	500	[3 , 7]	7	B	125871	99991	307107	307722	0.2%
H-A31	500	[7 , 13]	3	A	272655	467929	310330	478224	2.2%
H-A32	500	[7 , 13]	5	C	304253	342018	311689	353646	3.4%
H-A33	500	[7 , 13]	7	C	258529	221554	318939	328188	2.9%
H-A34	500	[10 , 20]	3	A	396255	839458	291265	846173	0.8%
H-A35	500	[10 , 20]	5	A	411886	460671	307346	472648	2.6%
H-A36	500	[10 , 20]	7	A	377336	316431	306353	397713	5.4%

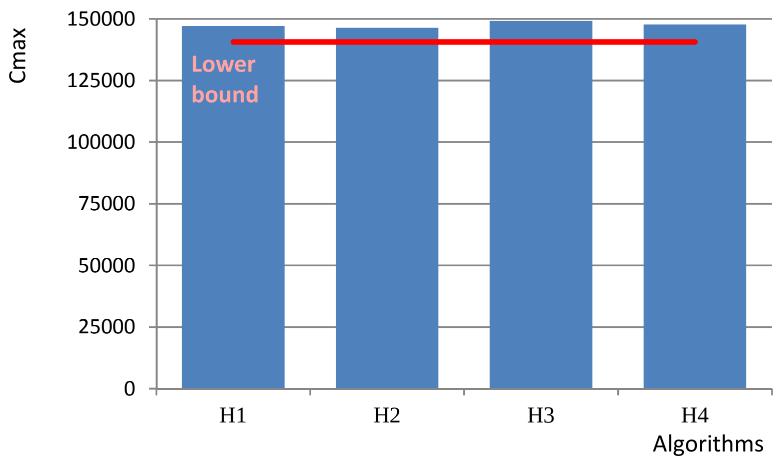
Fig. 11 with 5 panels shows the amount of four algorithms in terms of lower bound under different conditions. The algorithms have kept their good performances in all conditions. Panel *I* shows the result in condition that the assembly stage is bottleneck. In this case, all algorithms show the best performance. Panel *II* and *III* show the result of algorithms and lower bound situation when the hybrid flow shop is bottleneck and balance condition, respectively. Panel *II* is decomposed in two case. First, when the first stage of hybrid flow shop (single machine) is bottleneck and when the second stage (parallel machines) is bottleneck. These two cases are presented in panel *IV* and V, respectively.



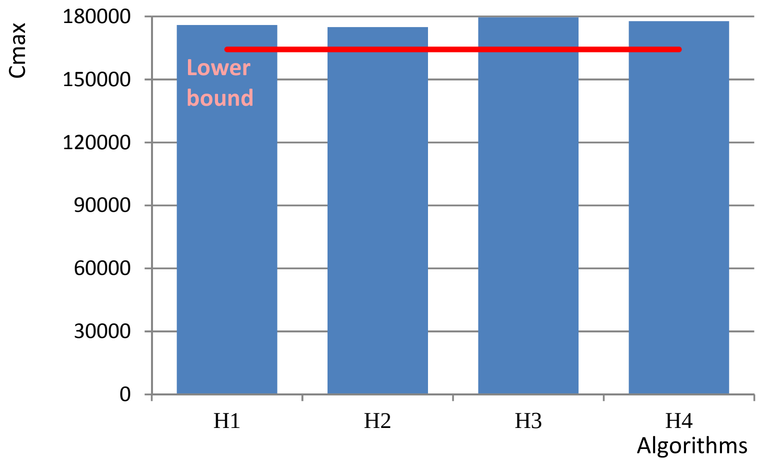
Panel *I*. The results when the assembly stage is bottleneck



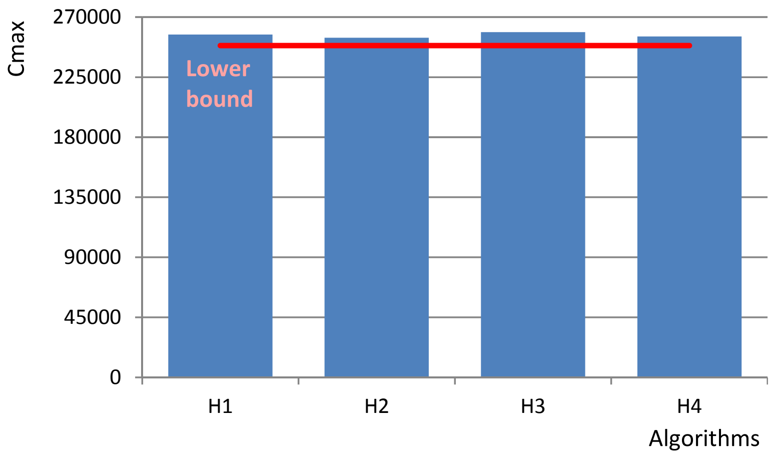
Panel *II*. The results when the hybrid flow shop is bottleneck



Panel *III*. The results in balance condition



Panel *IV*. The results when the first stage of hybrid flow shop is bottleneck



Panel *V*. The results when the second stage of hybrid flow shop is bottleneck

Fig. 11. The performance of algorithms in comparison of lower bound in 5 conditions of bottleneck

6. Concluding remarks and directions for future research

In this paper, an assembly production system including machining operations and assembly operations has been considered. The machining stage is a hybrid flow shop with two stages. All parts have to be processed on stage 1 and 2, respectively and when a set of parts is complete, the assembly of product is started. A number of products of the same kind have to be produced. Because of the setup times, it is better to process the parts in batches instead of individual. Each set of parts is called a block. Three decision variables of the problem are: the number of blocks, the size of each block, and sequencing the parts. So a three steps approach was considered to solve this problem. A heuristic is used for deciding the number of blocks, Johnson's and NEH algorithm for sequencing the parts and finally GA and SA for sizing the blocks. Four algorithms are presented by combination of this method. Three lower bounds presented and improved to evaluate the performance of algorithms.

Various problems in different condition design have been considered and four algorithms have applied to each problem, several times. The results have shown that SH/GA/John′s performed the best performance in all circumstances. Deviation of this algorithm from lower bound is 2.82% in average. After that SH/SA/John′s with 3.36% deviation came in the second position.

The worst result happened when the first stage of hybrid flow shop was bottleneck and it maintained low performance in terms of the lower bounds. Therefore, building some lower bounds for this problem under the condition that the first stage is bottleneck could be considered as a good future study. It is a good idea, in each block, that the first part be the same as the last part in previous block to eliminate its setup. Therefore, a new modeling formulation by considering this issue could be considered as a new study.

References

Al-Anzi, F.S., & Allahverdi, A. (2009). Heuristics for a two-stage assembly flow shop with bicriteria of maximum lateness and makespan. *Computers & Operations Research*, 36, 2682-2689.

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

Allahverdi, A., Gupta, J.N.D., & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega, International Journal of Management Science*, 27, 219-239.

Anandaraman, C. (2011). An improved sheep flock heredity algorithm for job shop scheduling and flow shop scheduling problems. *International Journal of Industrial Engineering Computations*, 2, 749–764.

Blazewicz, J., Ecker, K., Pesch, E., Schmidt. G., & Weglarz, J. (2007). Handbook on Scheduling From Theory to Application. *Springer, USA.*

Blazsik, Z., Imreh, C., & Kovacs, Z. (2008). Heuristic algorithms for a complex parallel machine scheduling problem. *Central European Journal of Operations Research*, 16, 379–390.

Chakrabortty, R.K., & Hasin, Md. A. A. (2013). Solving an aggregate production planning problem by using multi-objective genetic algorithm (MOGA) approach. *International Journal of Industrial Engineering Computations,* 4, 1–12.

Cheng, T.C.E., & Wang, G. (1999). Scheduling the fabrication and assembly of components in a two-machine flow shop. *IIE Transactions*, 31, 135-143.

Fattahi, P., Hosseini, S.M.H., & Jolai, F. (2012). A mathematical model and extension algorithm for assembly flexible flow shop scheduling problem. *International Journal of Advance Manufacture Technology,* DOI 10.1007/s00170-012-4217-x.

Gupta, J.N.D. (1988). Two stage hybrid flowshop scheduling problem. *Journal of Operational Research, Society* 39 (4), 359–364.

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

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, 547-556.

Hentsch, K., & Kochel, P. (2011). Job scheduling with forbidden setups and two objectives using genetic algorithms and penalties. *Central European Journal of Operations Research,* 19, 285-298.

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

Kadipasaoglu, S.N. (1997). A note on scheduling hybrid flow systems. *International Journal of Production Research,* 35(5), 1491-1494.

Khalouli, S., Ghedjati, F., & Hamzaoui, A. (2010). A meta-heuristic approach to solve a JIT scheduling problem in hybrid flow shop. *Engineering Applications of Artificial Intelligence,* 23, 765–771.

Koulamas, Ch., & Kyparisis, G. (2001). The three-stage assembly flow shop scheduling problem. *Computers & Operations Research*, 28, 689-704.

Kumar, S., Bagchi, T.P., & Sriskandarajah, C. (2000). Lot streaming and scheduling heuristics for m-machine no-wait flowshops. *Computers & Industrial Engineering,* 38, 149-172.

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.

Luo, H., Huang, G.Q., Zhang, Y.F., & Dai, Q.Y. (2011). Hybrid flowshop scheduling with batch-discrete processors and machine maintenance in time windows. *International Journal of Production Research*, 49(6), 1575-1603.

Maniezzo, V., Stutzle, T., & Vob, S. (2009). Hybridizing Metaheuristics and Mathematical Programming. *Springer, London*.

Marimuthu, S., Ponnambalam, S.G., & Jawahar, N. (2008). Evolutionary algorithms for scheduling m-machine flow shop with lot streaming. *Robotics and Computer-Integrated Manufacturing,* 24, 125–139.

Naderi-Beni, M., Tavakkoli-Moghaddam, R., Naderi, B., Ghobadian, E., & Pourrousta, A. (2012). A two-phase fuzzy programming model for a complex bi-objective no-wait flow shop scheduling. *International Journal of Industrial Engineering Computations*, 3, 617–626.

Pinedo, M.L. (2008). Scheduling. Theory, Algorithms, and Systems. Third Edition. *Springer, USA*.

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.

Quadt, D., & Kuhn, H. (2007). A taxonomy of flexible flow line scheduling procedures. *European Journal of Operational Research*, 178, 686-698.

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

Ruiz, R., & Vazquez-Rodriguez, J.A. (2010). Invited Review The hybrid flow shop scheduling problem. *European Journal of Operational Research,* 205, 1-18.

Schneider, U. (2011). A tabu search tutorial based on a real-world scheduling problem. *Central European Journal of Operations Research*, 19, 467–493.

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

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

Tseng, C.T., & Liao, C.J. (2008). A particle swarm optimization algorithm for hybrid flow-shop scheduling with multiprocessor tasks. *International Journal of Production Research,* 46(17), 4655-4670.

Walter, R. (2011). Comparing the minimum completion times of two longest-first scheduling-heuristics. *Central European journal of operations research,* doi:10.1007/s10100-011-0217-4.

Yang, J. (2010). A new complexity proof for the two-stage hybrid flow shop scheduling problem with dedicated machines. *International Journal of Production Research*, 48(5), 1531-1538.

Ying, K.Ch., & Lin, Sh.W. (2006). Multiprocessor task scheduling in multistage hybrid flow-shops: an ant colony system approach. *International Journal of Production Research*, 44(16), 3161-3177.

Yokoyama, M. (2001). Hybrid flow-shop scheduling with assembly operations. *International Journal of Production Economics,* 73, 103-116.

Yokoyama, M. (2008). Flow-shop scheduling with setup and assembly operations. *European Journal of Operational Research*, 187, 1184–1195.

Yokoyama, M., & Santos, D.L. (2005). Three-stage flow-shop scheduling with assembly operations to minimize the weighted sum of product completion times. *European Journal of Operational Research*, 161, 754-770.