Archive of SID

Mazandaran University of Science and Technology

Available online at www.iiec2017.com

13th International Conference on Industrial Engineering

(IIEC 2017)

Iran Institute of Industrial Engineering

A hybrid algorithm for flexible job shop scheduling problem with an assembly stage and sequence dependent setup time

Naeeme Bagheri Rad[a], Parviz Fattahi[b] and Fatemeh Daneshamooz[c]

[a] *Msc Student of Industrial Engineering, Faculty of Technical and Engineering, Bu-Ali Sina University, Iran*
E-mail: n.bagheri89@gmail.com

[b] *Associate Professor Department of Industrial Engineering, Faculty of Technical and Engineering , Alzahra University, Iran*
E-mail: Fattahi@basu.ac.ir

[c] *PhD Student of Industrial Engineering, Faculty of Technical and Engineering, Bu-Ali Sina University, Iran*
E-mail: f.daneshamooz@gmail.com

Abstract

A flexible job shop scheduling problem with assembly operations and sequence dependent setup time is studied in this paper. In this problem, at the first stage, the parts are processed in a flexible job shop system and then they are assembled in an assembly stage to produce products. Setup time is needed when a machine starts processing the parts or it changes items. At first, a linear model is introduced to express the problem. The objective is to minimize the completion time of all products. Then since the problem is NP-hard, a hybrid metaheuristic algorithm is presented. The proposed algorithm is called a Hybrid Particle Swarm Optimization with a Parallel Variable Neighborhood Search algorithm (HPSOPVNS). Finally, the numerical experiments are used to evaluate and validate the performance of the mathematical model and proposed algorithm. The computational results show that hybrid algorithm achieves better performance than Particle Swarm Optimization algorithm (PSO).

Keywords:
Flexible job shop scheduling, Assembly, Setup time, Particle swarm optimization, Parallel variable neighborhood search

1. Introduction

The two-stage assembly scheduling problem has many applications in industry and hence has received increasing attention from researchers recently [1,2]. Lee et al [1] described an application in a fire engine assembly plant while Potts et al [3] described an application in personal computer manufacturing. In particular, manufacturing of almost all items may be modeled as a two-stage assembly scheduling problem [2]. Hence considering production stages simultaneously in scheduling problems have received an increasing attention of researchers recently.

In this article, a flexible job shop scheduling problem (FJSP) with assembly operations is studied. In this problem, a set of parts are produced in independent production lines and finally in assembly stage are converted into desired product. This type of production systems can be a solution to respond market pressures to produce diverse products. FJSP problem is an extension of the classical job shop scheduling problem. The general JSP is strongly NP-hard [4]. This makes flexible job shop scheduling problem with assembly operations problem more complexity to solve due to the consideration of three sequence of operation, assignment of machines to operations and assembly stage. Therefore flexible job shop scheduling problem with assembly operations problem is NP-hard too.

The remainder of this paper is organized as follows. A literature review of the problem is presented in Section 2. In Section 3, the flexible job shop scheduling problem with assembly operations and sequence dependent setup times is described and the assumptions, parameters and decision variables and the mathematical model of the problem are presented. In section 4 is presented a numerical example. In section 5 the proposed hybrid algorithm and structure of proposed algorithm are explained. Experimental results are described in Section 6. Finally, the conclusion of the work and the future research are described in Section 7.

2. Literature review

In this section, the papers related to two-stage scheduling problems are reviewed. Lee et al in 1993 studied for the first time a two-stage production scheduling problem with the objective of makespan minimization. They considered a simple problem in which each product is assembled from two types of parts. The first component of each product must be processed on the first machine and the second

www.SID.ir

Archive of SID

component is processed on the second machine. Potts et al. [3] extend the problem to the case of multiple fabrication machines in which there are m machines and one machine at the first and second stages, respectively. Cheng and Wang [5] studied the problem of scheduling the fabrication and assembly of components in a two-machine flow shop with objective of minimizing the makespan. In their model, the first machine produces two types of parts, unique components and common components. The second machine assembles the components into the products. Tozkapan et al [6] studied a two-stage assembly scheduling problem with objective of minimizing the total weighted flow time. They considered m-1 machines in the first stage and a machine assembly in the second stage. They proposed a branch-and-bound algorithm and a lower bound and dominance criterion. Al-Anzi and Allahverdi [7] studied the two-stage assembly flow shop scheduling problem with objective minimize maximum lateness. They are considered the setup times the separate from processing times. They proposed a self-adaptive differential evolution heuristic to solve the scheduling problem for the first time. Also, they compared performance self-adaptive differential evolution heuristic with those of PSO, TS and EDD algorithms. Yokoyama [8] considered a scheduling model for flow-shop scheduling problem with setup and assembly operations with objective the mean completion time for all products. In the problem, operations are divided to several blocks. Each block consists of the operations of machining, operations of setup, and the assembly operation for one or more products .Also, they proposed a branch-and-bound method and the pseudo-dynamic programming. Allahverdi and Al-Anzi [3] are proposed the model for two-stage assembly scheduling problem with objective of minimizing the total completion time with setup times with assuming that setup times is separate from processing times. They presented a dominance relation and propose three heuristics: a hybrid tabu search, self-adaptive differential evolution (SDE) and a new self-adaptive differential evolution (NSDE). They showed NSDE algorithm the better performance than the other two.

Zhang [9] studied job shop scheduling with assembly operations with objectives of minimizing the total completion time and on-time delivery rate. They proposed the genetic algorithm with a rule based on the coding procedure. Shokrollahpour et al [10] studied two-stage assembly flow shop scheduling problem with objective of minimizing the weighted sum of makespan and mean completion time. They proposed an imperialist competitive algorithm (ICA) to solve the problem.

Al-Anzi and Allahverdi [11] considered two-stage assembly scheduling problem to minimize total completion time. They proposed artificial immune system algorithm (AIS), and compared performance of AIS algorithm with the best known heuristic in the literature. They showed that performance of proposed algorithm is better than the best known heuristic in the literature. Fattahi et al [12] studied hybrid flow shop scheduling problem with assembly operations and setup time with the objective makespan minimization. They proposed a branch and bound algorithm hierarchy and three lower bounds and an upper bound in condition: the first stage and second stage of the hybrid flow shop and the assembly stage is bottleneck. In order to reduce the time of computing algorithm and improve the efficiency, they proposed an upper bound according to GRASP algorithm. Allahverdi et al [13] studied a two-stage assembly flow shop with objective of minimizing the number of tardy jobs. They proposed genetic algorithm, improved genetic algorithm, simulated annealing algorithm with three different neighborhood structures, Dhouib et al.'s simulated annealing algorithm and an improved cloud theory-based simulated annealing algorithm.

The flexible job shop scheduling problem with an assembly is strongly NP-hard. Therefore, to obtain the optimal solution is not possible by exact methods and at the real dimensions for problems with the medium and large size. So, the heuristic or metaheuristic algorithms must be used. In this article, proposed a hybrid algorithm based on particle swarm optimization to solve the problem studied. In this proposed hybrid algorithm, we used the PSO algorithm for global exploration at search space and Parallel Variable Neighborhood Search (PVNS) algorithm for local search at around solutions.

3. Problem description and mathematical formulation

Studied problem in this research is modelling and sequence of jobs in flexible job shop scheduling problem with an assembly stage. In this problem, each product is produced from assembling a set of several different parts. First stage is flexible job shop scheduling problem that is defined as scheduling J part $\{J_{1p}, J_{2p}, \dots, J_{np}\}$ of P product $\{P_1, P_2, \dots, P_p\}$ on set M of machines including m machines $\{M_1, M_2, \dots, M_m\}$. For each operation, there is a set of machines. So, for each operation, there is possibility to choose a machine i among several machines assigned. In this state, each part has h_j operation that must be done respectively. Index i denotes a machine, index j denote parts and index h denotes operation. So, operation h of part j of product p ($h=1,\dots,h_j$, $p=1,\dots,P$, $j=1,\dots,n$) is shown with notation O_{jph} , with the processing time p_{ijph}. Suppose that $O_{j'p'h'}$ is scheduled to follow immediately after O_{jph}. Then $O_{j'p'h'}$ requires time for a setup, $s_{ijphj'p'h'}$. In this model we assume that $s_{ijpj'p'}=s_{ijphj'p'h'}$. Second stage is assembly scheduling problem that is defined as scheduling P product with assembly time A_p. The assembly operations of each product can be started until the set of parts required for a product is processed in the first stage.

General scheme of the considered problem is shown in Figure 1. In the first stage of production, row material or parts are processed in flexible job shop production system and readied parts required for a product. When required parts of a product are complete, they together join in the assembly stage and connect in the assembly stage.

www.SID.ir

Archive of SID

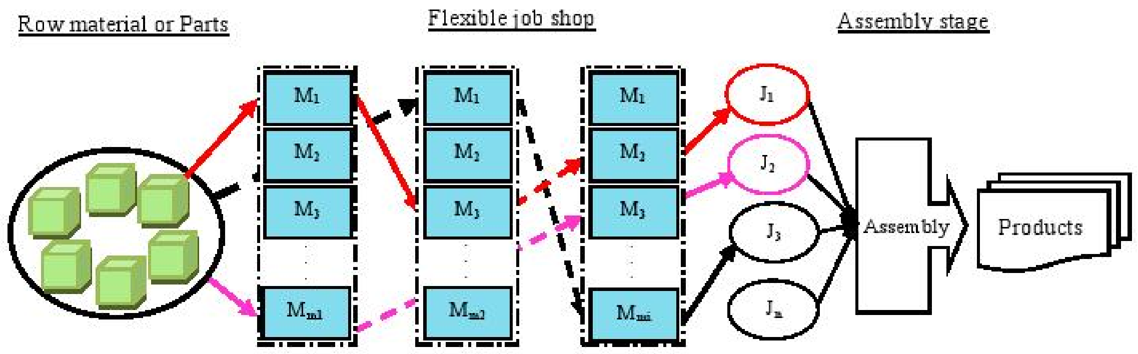


Figure 1- A schematic view of the considered problem

Assumptions:

1) All machines and parts are accessible at time zero.
2) Demand of final products is specified.
3) Each machine can process only one part at a time and each part can be processed only on one machine.
4) Time processing of parts and assembly of products is deterministic.
5) When processing a part is started, it doesn't stop until completed, (job interrupt is not allowed).
6) Assembly operation is started for a product when a set of parts are completed in the first stage.
7) Each product is composed of j part.
8) A dummy job 0 is used to signify the starting of a job on each machine. And, an operation that is scheduled to follow a dummy job is actually the first operation to process.

Parameters and decision variables:

The following parameters and decision variables are used to formulate the problem:

Parameters:

P,P': Total number of products(p=1,2,…,P ,p'=1,2,…,P');
$n_p, n_{p'}$: Number of sub-parts product P,P' (j=1,2,…, n_p j'=1,2,…, $n_{p'}$);
$h_j, h_{j'}$: Number of operations part j,j' (h=1,2,…, h_j ,h'=1,2,…, $h'_{j'}$);
m: Number of machines at first stage (i=1,…, m);
k'_A : The number of assigned products to assembly machine (k'=1,…,k'_A);
$O_{j,p,h}$: Operation h of the part j in product p;
$p_{i,j,p,h}$: Processing time of operation $O_{j,p,h}$ on machine i;
$s_{i,j,p,j',p'}$: Requires time for a setup operation $O_{j,p,h}$, (operation $O_{j',p',h'}$ precedes operation $O_{j,p,h}$ immediately);
A_p : Assembly time of product p;
$a_{i,j,p,h}$: 1,if machine i is selected for operation $O_{j,p,h}$; 0 otherwise;
L : A large number;

Decision variables:

C_{max}: Makespan or maximal completion time of products;
$t_{j,p,h}$: Start time of the processing of operation $O_{j,p,h}$;
$f_{j,p,h}$: finish time of the processing of operation $O_{j,p,h}$;
St_p : Start time of the assembling of product P;
$Sm_{k'}$: Start of working time for assembly machine in priority *k'* in the second stage;
E_p: Completion time of products p in the first stage;
C_p: Completion time of products p;
$x_{i,j,p,h,k}$: 1, operation $O_{j',p',h'}$ precedes operation $O_{j,p,h}$ immediately; 0 otherwise;
$y_{i,j,p,h}$: 1, if operation $O_{j,p,h}$ can be performed on machine i; 0 otherwise;
$Z_{p,k'}$: 1, if product P is assembled on assembly machine in priority k'; 0 otherwise;
The mathematical model of problem is presented as follows:

$$\text{Min } Z=\left(C_{max}\right) \qquad (1)$$

Subject to:

$$C_{max} \geq C_p \quad ,\forall p \qquad (2)$$

$$t_{j,p,h} + y_{i,j,p,h} . p_{i,j,p,h} \leq f_{j,p,h}, i=1,2,\ldots,m, \\ j=1,2,\ldots,n_p, p=1,2,\ldots,P \ ,h=1,2,\ldots,h_j \qquad (3)$$

$$f_{j,p,h} \leq t_{j,p,h+1}, \quad j=1,2,\ldots, n_p, \\ p=1,2,\ldots,P, \ h=1,2,\ldots,h_j-1 \qquad (4)$$

$$f_{j,p,h} \leq E_p, j=1,2,\ldots, n_p \ ,p=1,2,\ldots,P, \\ h=1,2,\ldots,h_j \qquad (5)$$

$$y_{i,j,p,h} \leq a_{i,j,p,h}, i=1,2,\ldots,m, \ j=0,1,\ldots,n_p, \\ p=0,1,\ldots,P, \ h=1,2,\ldots,h_j \qquad (6)$$

$$\sum_{i=1}^{m} y_{i,j,p,h}=1, j=0,1,2,\ldots, n_p \ , \\ p=0,1,\ldots,P, \ h=1,2,\ldots,h_j \qquad (7)$$

www.SID.ir

Archive of SID

$$t_{j,p,h}+p_{i,j,p,h}+s_{i,j,p,j',p'} \le t_{j',p',h'}+\left(1-x_{i,j,p,h,j',p',h'}\right).L,$$
$$i=1,2,\ldots,m,\ j=0,1,\ldots,n_p, p=0,1,\ldots,P,$$
$$h=1,2,\ldots,h_j, j'=1,\ldots,n_{p'}, p'=1,\ldots,P',$$
$$h'=1,2,\ldots,h'_{j'} \qquad (8)$$

$$f_{j,p,h}+s_{i,j,p,j',p'} \le t_{j,p,h+1}+\left(1-x_{i,j',p',h',j,p,h+1}\right).L,$$
$$i=1,2,\ldots,m,\ j=1,\ldots,n_p, p=1,\ldots,P,$$
$$h=1,2,\ldots,h_j-1, j'=0,1,\ldots,n_{p'},$$
$$p'=0,1,\ldots,P',\ h'=1,2,\ldots,h'_{j'} \qquad (9)$$

$$\sum_{p=0}^{P}\sum_{j=0}^{n_p}\sum_{h=1}^{h_j} x_{i,j,p,h,j',p',h'}=y_{i,j',p',h'}, i=1,2,\ldots,m,$$
$$j'=1,\ldots,n_{p'}\ , p'=1,\ldots,P', h'=1,\ldots,h'_{j'} \qquad (10)$$

$$\sum_{p'=1}^{P'}\sum_{j'=1}^{n_{p'}}\sum_{h'=1}^{h'_j} x_{i,j,p,h,j',p',h'}=y_{i,j,p,h}, i=1,\ldots,m\ ,$$
$$j=0,1,\ldots,n_p\ , p=0,1,\ldots,P,\ h=1,\ldots,h_j \qquad (11)$$

$$E_p \le St_p, \forall p \qquad (12)$$

$$A_p+St_p \le C_p, \forall p \qquad (13)$$

$$Sm_{k'}+A_p.Z_{p,k'} \le Sm_{k'+1}, \forall p,$$
$$k'=1,2,\ldots,k'_A-1 \qquad (14)$$

$$Sm_{k'} \le St_p+(1-Z_{p,k'}).L, \forall p,k' \qquad (15)$$

$$Sm_{k'}+(1-Z_{p,k'}).L \ge St_p, \forall p,k' \qquad (16)$$

$$\sum_{k'=1}^{k'_A} Z_{p,k'}=1, p=1,2,\ldots,P \qquad (17)$$

$$x_{i,j,p,h,j,p,h}=0, i=1,2,\ldots,m\ , j=0,1,\ldots,n_p$$
$$, p=0,1,\ldots,P\ , h=1,2,\ldots,h_j \qquad (18)$$

$$x_{i,j,p,h,j',p',h'} \in \{0,1\}, i=1,2,\ldots,m,$$
$$j=0,1,\ldots,n_p, p=0,1,\ldots,P,$$
$$h=1,2,\ldots,h_j, j'=1,\ldots,n_{p'}$$
$$, p'=1,\ldots,P',\ h'=1,2,\ldots,h'_{j'} \qquad (19)$$

$$y_{i,j,p,h} \in \{0,1\}, i=1,2,\ldots,m, j=0,1,\ldots,n_p$$
$$,\ p=0,1,\ldots,P,\ h=1,2,\ldots,h_j \qquad (20)$$

$$Z_{p,k'} \in \{0,1\}, P=1,2,\ldots,P, k'=1,2,..,k'_A \qquad (21)$$

$$C_p, St_p, E_p \ge 0, P=1,2,\ldots,P \qquad (22)$$

$$t_{j,p,h}, f_{j,p,h} \ge 0, j=0,1,\ldots,n_p,$$
$$p=0,1,\ldots,P,\ h=1,2,\ldots,h_j \qquad (23)$$

$$Sm_{k'} \ge 0, k'=1,2,..,k'_A \qquad (24)$$

The objective function (1) indicates minimization of the completion time for all products (makespan). Constraint (2) determines the objective function. Constraints (3) and (4) enforce each job to follow a specified operation sequence. Constraint (5) determines the largest processing time the parts of a product. Constraint (6) enforces that the machine for each O_{jph} must be selected from the machine's alternatives of O_{jph}. Constraint (7) forces exactly one machine alternative to be selected for each O_{jph}. Constraints (8) and (9) force each machine to process one operation at a time and consider the setup times. Constraints (10) and (11) define circular permutations of operations on each machine. They eliminate alternative operations that are excluded in a final schedule. Constraint (10) selects one operation O_{jph} that immediately precedes a scheduled alternative operation Oj',p',h' and constraint (11) selects one operation Oj',p',h' that immediately follows a scheduled alternative operation O_{jph}. Constraint (12) determines the smallest start time of assembly stage. Constraint (13) represents completion time of products. Constraint (14) assembly machine can process only one product at a time. Constraints (15) and (16) force each product in the second stage can be start after its assigned machine is idle and previous product is completed. Constraint (17) assigns the each product to a priority at second stage.

4. Numerical example

In order to clarify the problem, consider a simple numerical example. The data for processing time of machining operation, assembly and setup time are given in Table 1 and Table 2. In this example, assume that there are three machines at first stage and one machine at assembly stage. Total number of products is P = 2. That is each product contains of two parts of different kind. At first, each part is processed on first stage (flexible job shop). Then, parts are assembled into a product on assembly stage. The amount of the objective function is 49 in this example (Figure 2).

Table 1-Processing time of machining and assembly

Products				1				2	
Parts		1		2			3	4	
Operations		1	2	1	2	3	1	1	2
Machine	M_1	5	7	4	3	-	-	10	5
	M_2	4	2	-	2	5	1	-	6
	M_3	-	5	7	-	7	4	7	-
Assembly time				6				5	

www.SID.ir

Archive of SID

Table 2-Sequence-dependent setup time

	M_1				M_2				M_3			
	(1,1)	(1,2)	(2,3)	(2,4)	(1,1)	(1,2)	(2,3)	(2,4)	(1,1)	(1,2)	(2,3)	(2,4)
Dummy	5	8	6	10	5	1	3	7	2	4	6	9
*(1,1)	0	7	5	9	0	2	6	9	0	7	2	10
(1,2)	4	0	1	4	8	0	4	3	1	0	8	6
(2,3)	9	3	0	8	5	3	0	7	9	3	0	6
(2,4)	2	6	7	0	10	6	2	0	5	4	4	0

(product, part) *

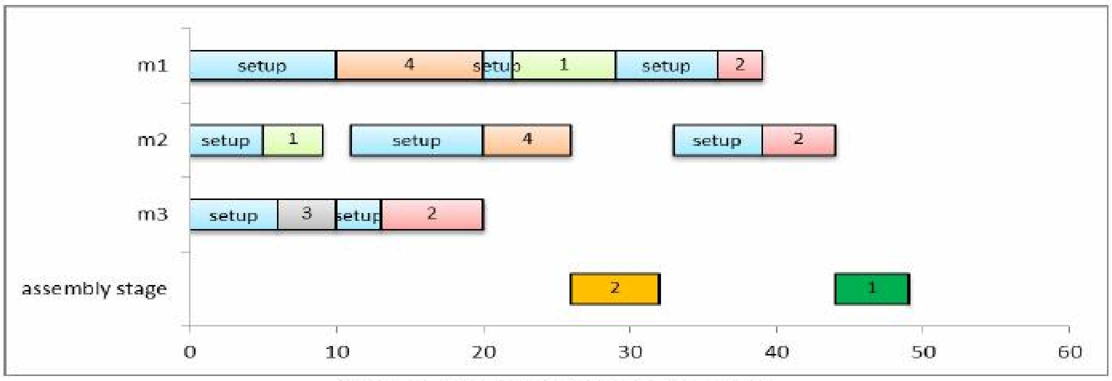


Figure 2- Gantt chart numerical example

5. Proposed algorithm

Flexible job shop scheduling problem with an assembly stage and sequence dependent setup times is strongly NP-hard. Therefore, to obtain the optimal solution is not possible for problem with the medium and large size in a reasonable time and needs using the heuristic algorithms or metaheuristic. So, to solve problem a hybrid metaheuristic method based on two methods of particle swarm optimization (PSO) and parallel variable neighborhood search (PVNS) is used.

5.1. Hybrid Particle Swarm Optimization and Parallel Variable Neighborhood Search (HPSOPVNS)

Particle swarm optimization is one of the most popular swarm intelligence. It was originally proposed by Kennedy and Eberhart as a simulation of the social behavior of social organisms such as bird flocking and fish schooling [14]. In the PSO algorithm, each solution of problem shows position of the particle in the search space that at first is initialized randomly .Changing the location of each particle in the search space influences best answer which so far has been obtained for each particle (Personal-Best) and the best answer has been obtained by all particles (Global-Best). The corresponding equations to position and velocity of the particle are defined as follow:

$$V_i(t)=W\times V_i(t-1)+c_1\times r_1\times(\text{P-Best}_i-X_i(t-1))+ \\ +c_2\times r_2\times(\text{G-Best}-X_i(t-1)) \quad (26)$$

$$X_i(t)=X_i(t-1)+V_i(t) \quad (27)$$

Where $V_i(t)$ and $X_i(t)$ are, respectively, velocity and position of particle i ,w is inertia weight, $r_1$and r_2 are two random number with uniform distribution in the interval [0,1] and c_1,c_2 are ,respectively, individual learning factor and social learning factor.

The algorithm PSO has high convergence speed. But at near the optimal point the search process strongly slows and does not explore the remaining space. In this article, in order to attain the quality solutions at improvement process, parallelization technique of variable neighborhood search algorithm is used. Mladenovic and Hansen [15] introduced variable neighborhood search as one of the well-known local search methods. The parallelization technique of VNS algorithm is used to improve the PSO algorithm. The pseudo-code of the PVNS algorithm is illustrated in Figure 3. In this structure n_{max}, n, pr_{max} and pr are, respectively, the number of neighborhood searches in the local search method, counter of neighborhood searches in the local search method, number of processor and counter of processor.

In this proposed approach, PSO algorithm is used for global exploration at search space and PVNS algorithm for local search at around solutions obtained in the each iteration. In this hybrid method at first, the best global solution is obtained by PSO algorithm. Then, the parallel variable neighborhood search is applied on the best global solution found in the each iteration. This step is repeated until the termination criteria are established.

```
Initialization:
Define of the neighborhood structures.
Choose stopping condition algorithm (maximum number of iterations).
Set G-best→x
    Set 1→k
      Repeat
        for each processor($p_{pr}$) do in parallel pr=1,…,$pr_{max}$
            Shaking:
```

www.SID.ir

Archive of SID

```
                Produce random solution ($x'_{pr}$) using the
                neighborhood structure 3.
            Local search:
               Consider solutions $x'_{pr}$;
               Set $1 \rightarrow n$, $1 \rightarrow h$;
               for $n=1:n_{max}$
                   if $h=1$ then use neighborhood
                   structure 1; (xl)
                   else if $h=2$ then use neighborhood
                   structure 2; (xl)
                   else if $h=3$ then use neighborhood
                   structure 4; (xl)
                   else if $h=4$ then use neighborhood
                   structure 5 ; (xl)
                   end if.
                   if $f(xl)<f(x'_{pr})$ then $xl \rightarrow x'_{pr}$ and
                   $1 \rightarrow h$ ;
                   else $h+1 \rightarrow h$;
                   end if
                   if $h=h_{max}+1$ then $1 \rightarrow h$;
                   end if
                   $n+1 \rightarrow n$;
             end for
          end for
         Updating:
         The best solution is selected among the solutions
         obtained and set equal to $x''$;
         if $x''$ is better $x$ then $x'' \rightarrow x$ , $1 \rightarrow k$;
        else $k+1 \rightarrow k$;
        end if
    Until $k>k_{max}$;
Until the stopping condition the PVNS algorithm is
attained;
```

Figure 3- Pseudo-code of PVNS algorithm

5.1.1. Structure of proposed algorithm

1- Solution representation

Each member of the population has two parts. First part: the sequence of operation and machine is assigned to operations at flexible job shop problem. Second part: sequence of products on assembly machine. In the first part, the position of each particle is shown by two vectors. 1) Machine assignment vector. 2) Operation sequence vector. Operation sequence vector is consists of two vectors. In the first vector, part, is used from the representation of Gen et al [16]. Each element (component) of the vector sequence shows an operation of job j. In this structure of representation, all operations of a job are displayed with the same symbol. Sequence of elements in this vector, respectively, displays the operations of different jobs. The second vector represents the number of product parts. In the machine assignment vector, each number represents selected machine for the corresponding operations in the operation sequence vector. Length of the operation sequence vector and machine assignment vector is equal to the total number of operations. In the second part, sequence of products is displayed by a vector. Each number represents the number of product. This structure of representation performs automatically feasibility of solutions updated. Figure 4 shows a feasible solution for flexible job shop with assembly problem with 2 products, 3 parts and 3 machines.

operation sequence	part	1	2	3	2	3	1	2
	product	2	1	1	1	2	1	2
machine assignment		2	1	3	1	1	2	3

a

product sequence	1	2

b

Figure 4- Schematic representation of a feasible solution

2-Initialization

Production of an initial population with quality is an important step for algorithm. The process of initializing includes phase of assignment of machine, operation sequence and sequence of products. In the problem studied, production of the initial population in the first phase has two parts: 1) Assignment of machines to operation. 2) Operation sequence. For generating initial assignments, two approaches presented by Pezzella et al [17] are utilized: AssignmentRule1 (search for the global minimum in the processing time table) and AssignmentRule2 (randomly permute jobs and machines in the processing time table). The combination of these two rules generates the initial set of assignments. 20% of initial assignments can be generated by rule 1 and 80% by rule 2. The sequencing of the initial assignments and sequence of products are obtained by random rule. In this rule, randomly, the symbol numbers (number of operations each job/number of products) are respectively sorted.

3- Neighborhood structure

Since the PVNS algorithm improves a solution using different neighborhood structures, in the following, five types of neighborhood structures used in PVNS algorithm are introduced:

1. Neighborhood structure 1
Neighborhood structure 1 changes the sequence of operations and sequence of products in the candidate solution string. In this structure an element is randomly chosen. Then operation/product of the selected cell with operation/product of the previous cell is replaced. Otherwise, operation of the selected with operation of the next one cell is replaced.

2. Neighborhood structure 2
In this structure an operation is randomly chosen out of operations of candidate solution string which are capable of being processed on more than one machine. Then, among the available machines for the selected operation, a new

www.SID.ir

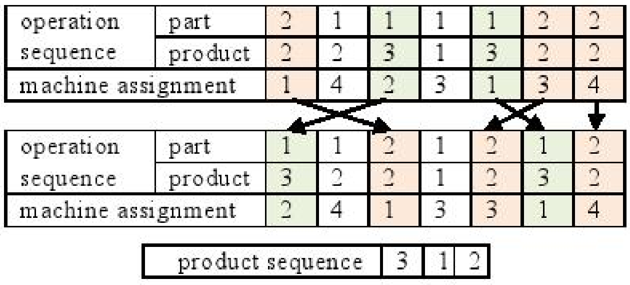
Archive of SID

machine is chosen at random and this operation will be assigned to it.

3. Neighborhood structure 3
Neighborhood structure 3 is a combination of neighborhood structure 1 and neighborhood structure 2. Neighborhood structure 1 and neighborhood structure 2 are simultaneously implemented on the candidate solution.

4. Neighborhood structure 4
In this neighborhood structure, at first, machines with most and lowest workload from candidate solution string are selected. Then, an operation is randomly selected from the machines with the most workload and is assigned to machine with lowest workload. Workload of machine is equal to the sum of processing times of the operations is assigned to that machine.

5. Neighborhood structure 5
Neighborhood structure 5 changes the sequence of operations in the candidate solution string. In this neighborhood structure at first, two parts of two or a product are randomly chosen. If the number of operations is equal in two selected parts, places of operations of two parts in candidate solution string, one to one, due to the number of operations are swapped. On the contrary, if the number of operations is not equal in two selected parts, operations related to part with lesser number of operations, with regard to sequence constraints are transferred to the location of the part with more numbers of operations. And the operations related to part with more number of operations, occupy their remaining cells and all cells part with lesser number of operations with regard to sequence constraints. An example of how to run the neighborhood structure 5 is presented in Figure 5. In this example, part 1 of product 3 and part 2 of product 2 were selected for replacement.



operation sequence	part	2	1	1	1	1	2	2
	product	2	2	3	1	3	2	2
machine assignment		1	4	2	3	1	3	4

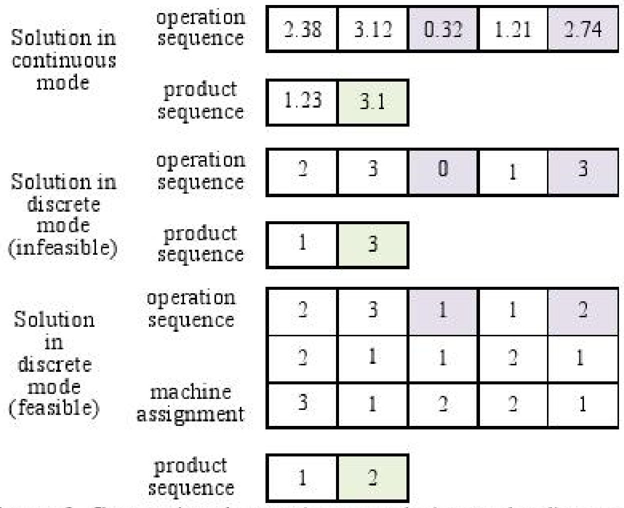
operation sequence	part	1	1	2	1	2	1	2
	product	3	2	2	1	2	3	2
machine assignment		2	4	1	3	3	1	4

product sequence	3	1	2

Figure 5- Example of the neighborhood structure 5

4- Compliance the PSO algorithm with discrete space

Solution of the flexible job shop problem with assembly is sequence of discrete numbers while PSO algorithm is designed to solve problem with continuous space. In the proposed algorithm after each time the displacement particles, each of them is converts to an acceptable discrete solution and then objective function is calculated for them. In this method, position of the particles is converted to the nearest integer. Vector obtained in this step show infeasible solution of problem. Therefore to produce a feasible solution, each of vector elements are reviewed and elements value of which is not equal with number of parts or number of operations of parts, specified and is converted to the nearest integer required. Figure 6 shows the updated position of the particle (continuous solution) and also how its conversion into a discrete solution. In this figure, nearest integer to 0.32 is 0, because do not have part 0, number 0.32 to 1 is converted. Also, nearest integer to 2.74 is 3, because the number 3 has been created twice and also part 3 has only one operation, so the number 2.74 is converted to the possible nearest integer, number 2. And in the product sequence, nearest integer to 3.1 is 3, but because the product 2 not has been created, number 3.1 is converted to the possible nearest integer, number 2. At the end, the second vector of sequence (product) and machine assignment vector are updates.



Solution in continuous mode	operation sequence	2.38	3.12	0.32	1.21	2.74
	product sequence	1.23	3.1			
Solution in discrete mode (infeasible)	operation sequence	2	3	0	1	3
	product sequence	1	3			
Solution in discrete mode (feasible)	operation sequence	2	3	1	1	2
		2	1	1	2	1
	machine assignment	3	1	2	2	1
	product sequence	1	2			

Figure 6- Conversion the continuous solution to the discrete

6. Experimental result

In this section, the proposed mathematical model is coded in GAMS software and also the proposed metaheuristic algorithm is coded in MATLAB (R2011a). For obtain the better and more sustainable results of the metaheuristic algorithms, from Minitab software is used for design the experiments.The results of Taguchi method are shown for each level of factors in small size, medium size and large size in Table 3, respectively. As regards information related to problems solved does not exist at similar articles for the problem studied. In this article, from random samples were used for validation of the problem. How to produce them is shown in Table 4.

www.SID.ir

Archive of SID

Table 3-Parameter designs to Taguchi method

Metaheuristic	Parameters	Categories problems		
		Large	Medium	Small
	Swarm size	45	45	45
PSO	C1	1.5	0.5	1.5
	C2	1.5	0.5	0.5
	w	1	0.8	1
	Max iter	100	100	100
	pr_{max}	3	4	3
PVNS	Max it	70	60	70
	n_{max}	40	50	40
	k_{max}	4	4	4

Table 4-Amounts of random data

Values	Symbols	Parameters
{2,3,…,50}	P	Number of products
{2,3,…,12}	n	Number of parts
{2,3,…,17}	m	Number of machine at first stage
uniform distribution	Ps	Processing time
uniform distribution	Ap	Assembly time
uniform distribution	s	Setup time

To evaluate the method of proposed solution, instances are categorized into three groups with small size, medium size and large size. The effectiveness of algorithm is compared using a well-known criterion, Relative Percentage Deviation (RPD). RPD factor is computed as Equation 28.

$$RPD = \frac{Algorithm_{solution} - Minimum_{solution}}{Minimum_{solution}} * 100 \qquad (28)$$

For the problems in the small size the optimal solution is obtained by GAMS. The obtained results are specified in Table 5. With regard to the values of RPD, HPSOPVNS and PSO algorithms have achieved the optimal solution, similar to optimal solution obtained by GAMS. Problems with medium and large size are solved only by the HPSOPVNS and PSO algorithm. Ten instances with medium size are generated and shown in Table 6. Diagram of average RPD for problems with medium size is shown in Figure 7. According to the RPD values in the figure, HPSOPVNS algorithm has better performance than PSO algorithm. Ten instances with large size are generated in Table 7 .On the basis of the RPD values presented in this table and diagram of average RPD (Figure 8), HPSOPVNS algorithm has better performance.

Table 5- Computational results the small problems

Problem no.	Size*	Gams			HPSOPVNS			PSO		
		CPU time(s)	C_{max}	RPD	CPU time(s)	C_{max}	RPD	CPU time(s)	C_{max}	RPD
S_1	2,3,22	2	40	0	30	40	0	15	40	0
S_2	2,4,4,3	5	41	0	99	41	0	29	41	0
S_3	3,6,33	9	50	0	128	50	0	62	50	0
S_4	3,3,6,5	45	45	0	127	45	0	49	45	0
S_5	4,2,4,4	25	61	0	170	61	0	72	61	0
S_6	4,3,4,5	85	39	0	151	39	0	64	39	0
S_7	5,2,4,5	69	55	0	160	55	0	67	55	0
S_8	5,4,4,6	132	67	0	147	67	0	67	67	0
S_9	8,2,4,4	317	61	0	182	61	0	79	61	0
S_{10}	6,3,4,6	452	63	0	368	63	0	148	63	0

*Number of (product, maximum number of parts, maximum number of operations, maximum number of machines)

Table 6- Computational results the medium problems

Problem no.	Size	HPSOPVNS			PSO		
		CPU time(s)	C_{max}	RPD	CPU time(s)	C_{max}	RPD
M_1	8,2,6,6	487	92	0	144	113	22.83
M_2	8,2,6,8	506	109	0	129	132	21.1
M_3	9,2,6,9	579	104	0	211	127	22.11
M_4	10,5,6,6	864	152	0	439	181	19.08
M_5	9,3,6,11	907	88	0	324	118	34.09
M_6	6,5,7,12	638	76	0	271	103	35.53
M_7	8,4,6,12	842	96	0	324	117	21.87
M_8	13,2,6,9	989	137	0	325	180	31.39
M_9	13,3,5,5	815	202	0	251	231	14.36
M_{10}	12,2,6,10	768	128	0	376	161	25.78

www.SID.ir

Archive of SID

Table 7- Computational results the large problems

Problem no.	Size	HPSOPVNS			PSO		
		CPU time(s)	C_{max}	RPD	CPU time(s)	C_{max}	RPD
L_1	9,4,6,14	853	142	0	698	162	14.08
L_2	13,3,6,17	1014	167	0	736	194	16.17
L_3	11,12,3,6	1347	187	0	751	217	16.04
L_4	17,4,9,9	1309	204	0	829	235	15.2
L_5	25,2,10,14	1159	145	0	736	174	20
L_6	30,6,4,8	1638	217	0	912	245	12.9
L_7	33,4,7,6	1735	230	0	948	261	13.48
L_8	38,3,9,7	2014	314	0	1237	346	10.19
L_9	42,2,6,7	1968	339	0	1103	371	9.44
L_{10}	50,3,6,9	2264	317	0	1276	360	13.56

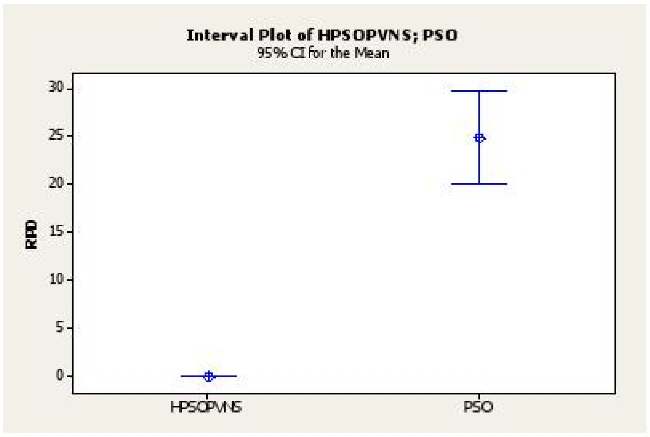


Figure 7- Diagram of average RPD with confidence interval 95% in medium problems

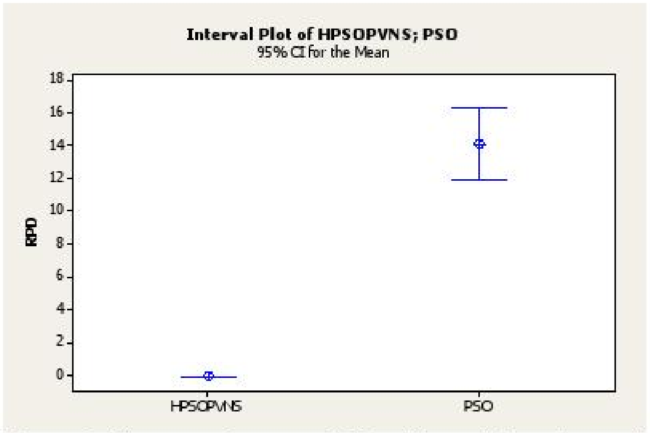


Figure 8- Diagram of average RPD with confidence interval 95% in large problems

7. Conclusion and future work

In this paper, a flexible job shop scheduling problem with assembly operations and sequence dependent setups was studied. This production system contains a flexible job shop followed by an assembly stage. In this production system, parts are produced in the flexible job shop and products are assembled after the parts in the assembly stage are ready. Due to high complexity of the flexible job shop scheduling problem with assembly operations, a hybrid metaheuristic algorithm the based on the PSO and PVNS optimization methods is proposed to solve the problem. In this hybrid approach, particle swarm optimization algorithm to explore the search space and the parallel variable neighborhood algorithm to exploit the best found solution in the each iteration are deployed. In order to evaluate the performance of proposed algorithm, we have compared its performance on numerical experiments with PSO algorithm. The results show that HPSOPVNS algorithm achieves better performance than PSO algorithm. As a further study, consideration of the buffer limitation is suggested. Also, the study of other objective functions such as sum of tardiness, sum of earliness, mean completion time, and so on will be useful in this problem.

References

[1] Lee, C.Y., Cheng, T. C. E. and Lin, B. M. T, "Minimizing the makespan in the 3-machine assembly-type flow shop scheduling problem," *Management Science*, Vol.39, No. 3, 1993, pp. 616-625.

[2] Allahverdi, A. and Al-Anzi, F. S, "The two-stage assembly scheduling problem to minimize total completion time with setup times," *Computers & Operations Research*, Vol.36, No.10, 2009, pp. 2740-2747.

[3] Potts, C.N., Sevast'Janov, S.V., Strusevich, V.A., Van Wassenhove, L.N. and Zwaneveld, C.M, " The two-stage assembly scheduling problem: complexity and approximation, " *Operations Research*, Vol.43, 1995, pp.346–355.

[4] Garey, M. R., Johnson, D. S. and Sethi, R, "The complexity of flow shop and job shop scheduling," *Mathematics of Operations Research*, Vol.1, No.2, 1976, pp.117-129.

[5] Cheng, T. C. E., and Wang, G, "Scheduling the fabrication and assembly of components in a two-machine flow shop," *IIE Transactions*, Vol.31, No.2, 1999, pp.135-143.

[6] Tozkapan, A., Kirca, O. and Chung, C-S, "A branch and bound algorithm to minimize the total weighted flow time for the two-stage assembly scheduling problem," *Computers & Operations Research*, Vol.30, No.2, 2003,

www.SID.ir

Archive of SID

pp.309-320.

[7] Al-Anzi, F. S. and Allahverdi, A, "A self-adaptive differential evolution heuristic for two stage Assembly scheduling problem to minimize maximum lateness with setup times," *European Journal of Operational Research*, Vol.182, No. 1, 2007, pp.80-94.

[8] Yokoyama, M, "Flow-shop scheduling with setup and assembly operations," *European Journal of Operational Research*, Vol.187, No. 3, 2008, pp.1184–1195.

[9] Zhang, R, "Simulation-based Genetic Algorithm for Job Shop Scheduling with Assembly Operations," *International Journal of Advancements in Computing Technology*, Vol.3, No.10, 2011, pp.132.

[10] Shokrollahpour, E., Zandieh, M. and Dorri, B, "A novel imperialist competitive algorithm for bi-criteria scheduling of the assembly flow shop problem," *International Journal of Production Research*, Vol.49, No.11, 2011, pp.3087-3103.

[11] Al-Anzi, F. S., and Allahverdi, A, "An artificial immune system heuristic for two-stage multi-machine assembly scheduling problem to minimize total completion time," *Journal of Manufacturing Systems*, Vol.32, No.4, 2013, pp. 825– 830.

[12] Fattahi, P., Hosseini, S. M. H., Jolai, F. and Tavakkoli-Moghaddam, R, "A branch and bound algorithm for hybrid flow shop scheduling problem with setup time and assembly operations," *Applied Mathematical Modelling*, Vol.38, No.1, 2014, pp.119-134.

[13] Allahverdi, A., Aydilek, A. and Aydilek, H, "Minimizing the number of tardy jobs on a two-stage assembly flow shop," *Journal of Industrial and Production Engineering*, Vol.33, No.6, 2016, pp.391-403.

[14] Eberhart, R., Kennedy, J, "A new optimizer using particle swarm theory," *In Conference: Proceedings of the sixth international symposium on micro machine and human science*, Nagoya, Japan, 1995, pp. 39- 43.

[15] Mladenovic, N., and Hansen, P, "Variable neighborhood search," *Computers and Operations Research*, Vol.24, No.11, 1997, pp. 1097-1100.

[16] Gen, M., Tsujimura, Y. and Kubota, E, "Solving job-shop scheduling problem using genetic algorithms," *In: Proceedings of the 16th international conference on computer and industrial engineering*, Ashikaga, Japan, 1994, pp. 576-579.

[17] Pezzella, F., Morganti, G. and Ciaschetti, G, "A genetic algorithm for the flexible job-shop scheduling problem," *Computers & Operations Research*, Vol.35, No.10, 2008, pp. 3202–3212.

www.SID.ir