Vol. 13, No. 1, 2026, Pages 97-108

JIEMS
Journal of Industrial Engineering and Management Studies

journal homepage: www.jiems.icms.ac.ir

A hybrid algorithm to minimize total tardiness in cyclic flexible job shop scheduling with assembly stage and sequence-dependent setup time

Naeeme Bagheri Rad [1], Javad Behnamian [1*]

[1] Department of Industrial Engineering, Faculty of Engineering, Bu-Ali Sina University, Hamedan, Iran

Received: Oct 2025-15 / Revised: Mar 2026-10 / Accepted: Jul 2026-15

Abstract

This paper studies a cyclic flexible job shop scheduling problem with assembly operations and sequence-dependent setup time, with the objective of minimizing total tardiness. In this production system, in the first stage, parts are produced in a flexible job shop system, and in the second stage, they are assembled into finished products. In this problem, products must be delivered in predetermined batch sizes within a definite time window. To solve the problem, a mixed-integer linear model is developed. Since the problem is NP-hard, a hybrid genetic algorithm and a parallel variable neighborhood search algorithm are proposed to solve medium- and large-sized instances. The hybrid approach leverages the global search capabilities of the genetic algorithm and the robust local refinement of the parallel variable neighborhood search, ensuring better escape from local optima and improved convergence rates. After fine-tuning the hybrid algorithm using the Taguchi method, the obtained results are compared with those obtained from GAMS and a conventional genetic algorithm on various instances. The computational results show that the proposed hybrid algorithm not only reduces total tardiness more effectively but also offers significant computational efficiency improvements, thereby demonstrating its practical applicability in complex manufacturing scenarios characterized by stringent delivery requirements.

Keywords: Assembly sequence planning; Cyclic scheduling; Flexible job shop; Setup time; Hybrid algorithm; Flexible manufacturing

Paper Type: Original Research

1. Introduction

Due to strong competition and limited resources, scheduling is a very important decision-making process in both production and non-production systems. In scheduling and sequencing operations, one of the important issues is production planning. Flexible job shop scheduling is an important field in scheduling theory that has significant applications in industry. Setup time is an important factor in flexible job shop scheduling problems. In this paper, a flexible job shop scheduling problem with sequence-dependent setup time is discussed. In the real world, job scheduling and assembly operations are performed simultaneously. Thus, to more closely reflect real-world conditions, these two stages are considered together. In the paper, first, the parts are manufactured in a flexible job shop, and then they are assembled into products in the assembly stage. For each product, the time interval between delivery times is called the cycle time. Since the cycle time varies for different products, the smallest common multiple of the cycle times is considered the global cycle. The goal is to minimize total delay in a cycle. Today, every company must adapt its production planning to customer demand. This fact forms the basis of the philosophy of Just-in-Time (JIT) production. This is the main reason for the shift in industrial policy of many producers from one product to several types of products. The cyclic policy is used for this change. An important application of the cyclic policy is in industries trying to use common resources to produce different types of products. Therefore, a cyclic flexible job shop scheduling problem with assembly operations is studied. Some of the problem applications in the real world include the automobile industry, home appliance manufacturing, etc. The general scheme for the above problem is shown in Figure1.

*Corresponding Author: Behnamian@basu.ac.ir

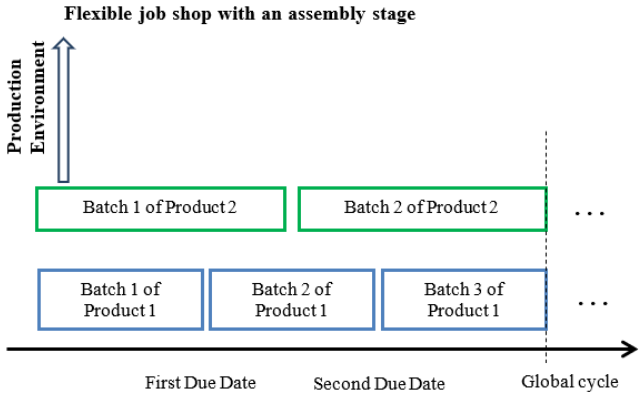

Figure1. Scheme general of the considered problem

The rest of this paper is organized as follows. Section 2 presents a review of the literature on the problem. Section 3 introduces the modeling of a cyclic flexible job shop scheduling problem with an assembly stage. Section 4 proposes the hybrid genetic algorithm and parallel variable neighborhood search algorithm and its structure. Section 5 presents the computational experiments. Finally, Section 6 provides the conclusion and future research.

2. Literature Review

In 1993, Lee et al. studied for the first time a two-stage production scheduling problem. In a two-stage production system, at the first stage, the parts are fabricated in a production system such as the job shop system or the flexible job shop system, and then assembled into products at the assembly stage. Sculli, Cheng, Zhang, and Dimyati studied job shop scheduling with assembly operations. A variant of the required setup time for an operation is the sequence-dependent setup time. In this type, the setup time depends on the immediately preceding operation. Rossi, Shen et al., Azzoz et al., Ramezani et al., and Mousakhani studied the flexible job shop scheduling problem and considered sequence-dependent setups.The number of studies on the cyclic flexible job shop scheduling problem is low, so the literature on cyclic scheduling, cyclic job shop scheduling, and cyclic flexible job shop scheduling is reviewed as follows. For the first time, Cuninghame-Green studied cyclic problems in the field of scheduling. He introduced the concept of cyclic operations to optimize the cycle period. Hanen and Munier introduced cyclic operations as operations that must be processed continuously over a period of time. They defined a linear constraint for each pair of cyclic operations.These constraints express the precedence relationships between operations, with the goal of optimizing cycle time. Brucker and Kampmeyer proposed a general framework for modeling and solving cyclic scheduling problems. The proposed framework includes various scheduling versions, such as cyclic job shop scheduling, cellular robotics, scheduling, and tool transfer between machines. They proposed a tabu search algorithm. In another paper, they studied the bottleneck in the cyclic job shop scheduling problem. Romanova and Servakh presented an algorithm to solve a particular type of cyclic job shop scheduling problem. In this problem, a fixed number of jobs can be processed simultaneously. These authors proposed a fully polynomial-time approximation scheme. Ouenniche and Bertrand studied the multi-product scheduling problem with fixed demand and limited time horizons. Their proposed approach consists of three main steps: sequencing, lot-sizing, and scheduling. Also, they proposed a cyclic scheduling framework, namely, the multiple cyclic method. This method first defines an initial cycle time and then, in the model, assumes that the cycle time for each product is an integer multiple of this value. Gnatowski et al. considered a cyclic job shop scheduling problem. They presented a mathematical model with the objective of minimizing the cycle time. These authors proposed a parallel tabu search algorithm. In this study, computational experiments were performed in a multi-processor environment. Jalilvand-Nejad and Fattahi in the cyclic flexible job shop scheduling problem, assumed that jobs are delivered in batches at definite time intervals. They proposed a framework to break down the infinite programming horizon into smaller periods. In order to solve the problem, these researchers proposed two algorithms based on genetic and simulated annealing (SA) algorithms. Quinton et al. studied a cyclic flexible job shop scheduling problem. They proposed an exact method and two heuristic algorithms. Elmi et al. presented a novel mixed-integer programming model for the cyclic job shops with blocking, where there are no intermediate buffers between the machines. This research is an extension of Fattahi et al.They proposed a particle swarm optimization-based algorithm to minimize the makespan in the assembly flexible job shop scheduling. Here, for the first time, under realistic assumptions, the

concept of cyclic scheduling and the assembly flexible job shop are integrated into the problem, and sequence-dependent setup time is considered.
The table below summarizes the key aspects of the most relevant studies, including their objective functions, assumptions, constraints, and solution approaches:

Table 1. Cyclic flexible job shop and job shop scheduling problem

Study	Objective Function	Key Assumptions	Key Constraints	Solution Approach
Lee et al.	– Introduced the two-stage scheduling concept (exact objective not explicitly specified)	Two-stage production system: fabrication followed by assembly	Sequential processing in two distinct stages	Theoretical/Exact methods
Sculli et al., Cheng, Zhang, Dimyati	Likely minimization of makespan or tardiness in job shop scheduling with assembly operations	Integration of assembly operations with job shop scheduling	Standard machine and assembly scheduling constraints	Heuristic and/or exact methods (details not fully specified)
Rossi et al., Shen et al., Azzoz et al., Ramezani et al., Mousakhani	Likely minimization of makespan/tardiness under sequence-dependent setups	Flexible job shop scheduling with sequence-dependent setup times	Machine assignment, processing times, and sequence-dependent setup times	Various (heuristics, approximation schemes, etc.)
Cuninghame-Green , Hanen and Munier, Brucker and Kampmeyer	Cycle period or cycle time optimization	Cyclic operation with repeated processes	Precedence relationships among operations in cycles	Linear programming formulations; tabu search (in later work)
Romanova and Servakh ,Ouenniche and Bertrand ,Gnatowski et al.	Minimization/optimization of cycle time	Cyclic scheduling with fixed job processing constraints	Cycle time defined as an integer multiple and simultaneous processing	Fully polynomial-time approximation scheme; exact methods; parallel tabu search
Jalilvand-Nejad and Fattahi	Framework aimed at effective scheduling in batch-based cyclic environments	Jobs are delivered in batches at definite time intervals; cyclic production assumptions	Decomposition of an infinite planning horizon into smaller periods	Genetic and simulated annealing (SA) algorithms
Quinton et al.	Addressing cyclic flexible job shop scheduling (objective details not fully specified)	Cyclic production environment	As defined by the specific problem formulation	Exact methods and two heuristic algorithms
Elmi et al.	– Typically aiming at cycle time minimization under blocking conditions	Cyclic job shops with blocking (no intermediate buffers)	Blocking constraints between machines	Mixed-integer programming (MIP) model
Fattahi et al.	Minimization of makespan in assembly flexible job shop scheduling	Integration of assembly process with job shop scheduling	Standard scheduling constraints for assembly flexible job shop	Particle swarm optimization (PSO) based algorithm
Our Study	Minimization of total tardiness	Integration of cyclic scheduling, assembly operations, and sequence-dependent setup times under realistic assumptions	Two-stage production; cyclic scheduling constraints; machine assignment; sequence-dependent setup; batching and assembly constraints	Hybrid metaheuristic combining Genetic Algorithm (GA) and Parallel Variable Neighborhood Search (PVNS)

Despite the diverse approaches in the literature, several research gaps remain:

1. Limited Focus on Cyclic Flexible Job Shop Scheduling with Assembly: Only a few studies have addressed scheduling problems that combine cyclic production with assembly operations. Moreover, existing works often consider simplified assumptions or focus on single aspects (e.g., makespan minimization) rather than comprehensive performance metrics such as total tardiness.
2. Insufficient Integration of Sequence-Dependent Setup Times: Although sequence-dependent setups have been incorporated in flexible job shop scheduling their integration within a cyclic and assembly environment remains underexplored.
3. Scalability and Computational Efficiency: Many exact methods and even some heuristics fail to effectively solve medium- and large-sized instances of such complex scheduling problems, emphasizing the need for more robust metaheuristic approaches.
4. Diverse Solution Approaches: Previous solution approaches have ranged from exact algorithms and mixed-integer programming models to heuristics such as tabu search, simulated annealing, and particle swarm optimization. However, few studies have successfully combined multiple metaheuristic strategies to balance global exploration and local exploitation.

   Our Study directly addresses these gaps by proposing a hybrid metaheuristic algorithm (GA-PVNS) that:

   - Integrates the concepts of cyclic scheduling, assembly operations, and sequence-dependent setup times within a unified mathematical and computational framework.
   - Focuses on minimizing total tardiness, a performance metric that is particularly relevant in modern production environments with strict delivery requirements.

- Improves scalability and solution quality by combining the global search ability of Genetic Algorithms with the local refinement capabilities of Parallel Variable Neighborhood Search, making it well-suited for medium- and large-sized problem instances.
- Demonstrates superior performance compared with traditional methods (such as standalone GA), as validated by extensive computational experiments and statistical analysis.

This comprehensive review and analysis not only underline the novelty of our work but also pave the way for future research in multi-objective, uncertain, and more complex scheduling environments.

3. Problem Description and Formulation

In this paper, the cyclic flexible job shop scheduling problem with an assembly stage is considered. In this problem, each product is produced by assembling a set of parts. First, the parts are processed in a flexible job shop system (first stage). In this system, there are 'j' parts of 'p' products, and 'm' different machines. For each operation, there is the possibility of choosing machine i from among several assigned machines. The h-th operation of batch k of part j of product p is shown with the notation Oj,p,k,h. Each operation Oj,p,k,h is performed by machine i with the processing time ψi,j,p,k,h. Setup time is needed when a machine starts processing the parts or when it changes items. In the proposed model, it is assumed that si,j,p,k,j',p',k' = si,j,p,k,h,j',p',k',h'. The setup time is sequence-dependent and represented by si,j,p,k,h,j',p',k',h'. Then, in the second stage, the parts are assembled with an assembly time Ap. The assembly operation for a product cannot commence until its corresponding set of parts has been completed during the machining operations. It is also assumed that each product has a due date. The due date for batch k of product p is shown with the notation dp,k; in other words, delivering batches of product p after their due dates is denoted by dtp,k. The assumptions used in this problem are:

- All machines and parts are available at the planning horizon.
- The demands for the final products are specified.
- Each machine can process only one part at a time, and each part can be processed on only one machine.
- The processing time of operations and the assembly time of products is deterministic.
- Each operation cannot be interrupted.
- Dummy job 0 is used to indicate the start of operations on each machine.
- When a set of parts is completed at the first stage, the assembly operation is started for a product.

3.1. Parameters and decision variables

The common notations used throughout this paper are provided below.

Indices:

p Product index(p=1,…,P);
j Sub-parts index(j=1,…, np);
h Operation index(h=1,…, hj);
i Machine index(i=1,…, m);
k Batch index (k=1,…,kp);

Parameters:

P,P' Number of products;
np,np' Number of sub-parts of product p,p';
hj,h'j' Number of operations for part j,j';
kp,kp' Number of batches in the short-term horizon for product p,p';
m Number of machines at the flexible job shop scheduling stage ;
kk'A Number of assigned products to assembly machine;
Oj,p,k,h Operation h of the batch k of the part j in product p;
Psi,j,p,h Processing time of operation Oj,p,k,h on machine i;
Ap Assembly time of product p;
ai,j,p,h 1, if operation Oj,p,k,h can be performed on machine i; 0 otherwise;
si,j,p,k,j',p',k' Requires time for a setup, if Oj',p',k',h' is scheduled to follow immediately after Oj,p,k,h Operation h of batch k of the part j of product p;
dp,k Due date for kth batch of product j;
L A large number;

Variables:

dtp,k Delay time of delivering the kth batch of product p;
tj,p,k,h Start time of the operation Oj,p,k,h;

fj,p,k,h Finish time of the operation Oj,p,k,h;
Stp,k Start time of the assembling the kth batch of product p;
Smkk' Start of working time for assembly machine in priority kk' in the second stage;
Ep,k Completion time of the kth batch of product p in the first stage;
yi,j,p,k,h 1, if machine i is selected for operation Oj,p,k,h; 0 otherwise;
xi,j,p,k,h,j',p',k',h 1, if Oj',p',k',h' precedes Oj,p,k,h immediately; 0 otherwise;
Zp,k,kk' 1, if product p of the kth batch is assembled on the assembly machine in priority kk'; 0 otherwise;

The model proposed in this research is the mathematical model initially proposed by Jalilvand-Nejad and Fattahi et al. With respect to the problem defined above and the mentioned assumptions, the mathematical model can be formulated as follows:

$$\text{Minimize } Z=\sum_{p}\sum_{k} dt(p,k) \tag{1}$$

Subject to:

$$t_{j,p,k,h}+y_{i,j,p,k,h}.p_{i,j,p,k,h} \le f_{j,p,k,h} \quad i=1,2,\ldots,m,\ j=1,2,\ldots,n_p,\ p=1,2,\ldots,P,\ h=1,2,\ldots,h_j,\ k=1,\ldots,K \tag{2}$$

$$f_{j,p,k,h} \le t_{j,p,k,h+1} \quad j=1,2,\ldots,n_p,\ p=1,2,\ldots,P,\ h=1,2,\ldots,h_j-1,\ k=1,\ldots,K \tag{3}$$

$$f_{j,p,k,h} \le E_{p,k} \quad j=1,2,\ldots,n_p,\ p=1,2,\ldots,P,\ h=1,2,\ldots,h_j,\ k=1,\ldots,K \tag{4}$$

$$y_{i,j,p,k,h} \le a_{i,j,p,h} \quad i=1,2,\ldots,m,\ j=0,1,\ldots,n_p,\ p=0,1,\ldots,P,\ h=1,2,\ldots,h_j,\ k=0,1,\ldots,K \tag{5}$$

$$\sum_{i=1}^{m} y_{i,j,p,k,h}=1 \quad j=0,1,2,\ldots,n_p,\ p=0,1,\ldots,P,\ h=1,2,\ldots,h_j,\ k=0,1,\ldots,K_p \tag{6}$$

$$t_{j,p,k,h}+p_{i,j,p,h}+s_{i,j,p,k,j',p',k'} \le t_{j',p',k',h'}+\left(1-x_{i,j,p,k,h,j',p',k',h'}\right).L \quad i=1,2,\ldots,m,\ j=0,1,\ldots,n_p,\ p=0,1,\ldots,P,\ h=1,2,\ldots,h_j,\ j'=1,\ldots,n_{p'},\ p'=1,\ldots,P',\ h'=1,2,\ldots,h'_{j'},\ k=0,1,\ldots,k_p,\ k'=0,1,\ldots,k_{p'} \tag{7}$$

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

$$\sum_{p=0}^{P}\sum_{j=0}^{n_p}\sum_{k=0}^{K_p}\sum_{h=1}^{h_j} x_{i,j,p,k,h,j',p',k',h'}=y_{i,j',p',k',h'} \quad i=1,2,\ldots,m,\ j'=1,\ldots,n_{p'},\ k'=1,\ldots,K_{p'},\ p'=1,\ldots,P',\ h'=1,\ldots,h'_{j'} \tag{9}$$

$$\sum_{p'=1}^{P'}\sum_{j'=1}^{n_{p'}}\sum_{k'=1}^{K_{p'}}\sum_{h'=1}^{h'_{j'}} x_{i,j,p,k,h,j',p',k',h'} = y_{i,j,p,k,h} \qquad i=1,\ldots,m\ ,\ j=0,1,\ldots,n_p\ , \quad p=0,1,\ldots,P,\ h=1,\ldots,h_j, k=0,1,...,K_p \tag{10}$$

$$E_{p,k} \le St_{p,k} \qquad \forall p,k \tag{11}$$

$$Sm_{kk'} + A_p.Z_{p,k,kk'} \le Sm_{kk'+1} \qquad \forall p,k,\quad k'=1,2,...,k'_A-1 \tag{12}$$

$$Sm_{kk'} \le St_{p,k} + (1-Z_{p,k,kk'}).L \qquad \forall p,k,kk' \tag{13}$$

$$Sm_{kk'} + (1-Z_{p,k,kk'}).L \ge St_{p,k} \qquad \forall p,k,kk' \tag{14}$$

$$\sum_{kk'=1}^{kk'_A} Z_{p,k,kk'} = 1 \qquad \forall p,k \tag{15}$$

$$dt_{p,k} \ge A_p + St_{p,k} - d_{p,k} \qquad \forall p,k \tag{16}$$

$$x_{i,j,p,k,h,j,p,k,h} = 0 \qquad i=1,2,\ldots,m\ ,\ j=0,1,\ldots,n_p, k=0,1,...,K_p \quad p=0,1,\ldots,P\ ,h=1,2,\ldots,h_j \tag{17}$$

$$x_{i,j,p,k,h,j',p',k',h'} \in \{0,1\} \qquad i=1,2,\ldots,m,\ j=0,1,\ldots,n_p, k=0,1,...,K_p \quad p=0,1,\ldots,P, h=1,2,\ldots,h_j, j'=1,\ldots,n_{p'} \quad p'=1,\ldots,P',\ h'=1,2,\ldots,h'_{j'}\ k'=1,...,K_{p'} \tag{18}$$

$$y_{i,j,p,k,h} \in \{0,1\} \qquad i=1,2,\ldots,m, j=0,1,\ldots,n_p, k=0,1,...,K_p \quad p=0,1,\ldots,P,\ h=1,2,\ldots,h_j \tag{19}$$

$$Z_{p,k,kk'} \in \{0,1\} \qquad p=1,2,\ldots,P,\ k=1,...,K_p, k'=1,2,..,k'_A \tag{20}$$

$$C_{p,k}, St_{p,k}, E_{p,k}, dt_{p,k} \ge 0 \qquad \forall p,k \tag{21}$$

$$t_{j,p,k,h}, f_{j,p,k,h} \ge 0 \qquad j=0,1,\ldots,n_p, k=0,1,...,K_p \quad p=0,1,\ldots,P,\ h=1,2,\ldots,h_j \tag{22}$$

$$Sm_{kk'} \ge 0 \qquad \forall kk' \tag{23}$$

The objective function (1) minimizes total tardiness. Constraints (2) and (3) define the operation sequence of each job. Constraint (4) determines the maximum processing time for the parts of a product. Constraint (5) selects the machine required for each operation from the machines assigned to that operation. Constraint (6) assigns a machine to each operation. Constraints (7) and (8) ensure that each machine processes only one operation at a time (with setup time considered). Constraints (9) and (10) ensure that each operation is performed on only one machine and assigned only one priority. Constraint (11) defines the earliest start time of the assembly stage. Constraint (12) enforces that a machine in the second stage first assembles the parts of the product at priority kk′ and then starts assembling the parts of the product at priority kk′+1. Constraints (13) and (14) ensure that each product in the second stage is processed only after its assigned machine is idle and the previous product is completed. Constraint (15) assigns each product to a priority in the second stage. Constraint (16) defines the delivery of the kth batch of product p.

4. Hybrid Genetic Algorithm and Parallel Variable Neighborhood Search

The cyclic flexible job shop scheduling problem with an assembly stage is an NP-hard problem. Therefore, the introduced mathematical model cannot be solved for medium- and large-sized instances. Thus, it is necessary to propose a metaheuristic algorithm. A hybrid metaheuristic algorithm based on a genetic algorithm and a parallel variable neighborhood search (PVNS) is proposed. A genetic algorithm, inspired by Darwin's evolutionary theory,

was invented in 1975 by Holland et al. Today, this algorithm is one of the most widely used methods in combinatorial optimization problems. The genetic algorithm is a powerful search technique for solving combinatorial optimization problems. Although it has a good convergence speed, its convergence slows near the optimum point of the search process. In this case, the algorithm loses its ability to produce diverse chromosomes. Therefore, to avoid falling into a local optimum and premature convergence, a parallel variable neighborhood search algorithm is proposed. The rationale for employing this hybrid approach is multi-fold. First, while the GA exhibits strong global search capabilities by exploring vast areas of the solution space and quickly generating promising candidates, it often suffers from premature convergence—especially as the population loses diversity near optimal regions. This makes it difficult for the GA alone to finely explore the local neighborhood of high-quality solutions. Second, the variable neighborhood search (VNS) algorithm, proposed by Mladenovic and Hansen, is highly effective in local exploitation. It systematically explores different neighborhood structures, which facilitates an intensified search around locally optimal solutions. By parallelizing the VNS (PVNS), we can execute multiple independent neighborhood searches simultaneously, thereby significantly reducing computational time and increasing the thoroughness of the local search. Integrating these two methods leverages their individual strengths: the GA provides robust global exploration, while PVNS ensures efficient local refinement. This synergy leads to a balanced exploration–exploitation trade-off, which is essential for overcoming the complex landscape of NP-hard scheduling problems. Moreover, the hybridization allows our algorithm to better escape local optima and improve solution quality, especially in medium- and large-sized problem instances. Ultimately, the combination of GA and PVNS results in a more robust and computationally efficient approach to tackle our problem. The variable neighborhood search (VNS) algorithm was proposed by Mladenovic and Hansen. The VNS algorithm systematically changes the neighborhood. In order to reduce the computational time or increase the exploration of the search space, a parallelization strategy for the VNS is applied. In this algorithm, parallelization is based on increasing the number of independent searches within the search environment. In the PVNS algorithm, first, neighborhood structures are defined for searching the solution space. The main part of the algorithm consists of internal and external loops. The internal loop is responsible for searching the solution space, while the external loop controls the stopping condition of the algorithm. In this algorithm, the internal loop utilizes multiple processors. In each iteration of the internal loop, each processor, based on the input solutions, performs one iteration of the VNS algorithm, which includes the shake and local search methods. The steps of the PVNS algorithm are illustrated in Algorithm 1.

Algorithm 1: PVNS pseudo-code method

```
Repeat (External loop)
  Set k ← 1and initial solution ỹ;
    Repeat (inner loop)
      For each processor pr(h), h = 1,…, n_pr; % Start of generation
        Perturbation
          Generate a random solution y'(h) using the neighborhood structure 3;
        Local Search
          Solution Consider y'(h);
          Set n ← 1 and l ← 1;
          Do this for n = 1 to nmax
            If l = 1, then
                Apply neighborhood structure 1. (yp)
                Otherwise, if l = 2, use the neighboring structure of 2. (yp)
                Otherwise, if l = 3, use the neighborhood structure of 4. (yp)
                 If f (yp) ≤ f (y'(h)) then
                      y'(h) ← yp and l ← 1 otherwise l ← l + 1.
                 End if
            End if
            If l = lmax + 1 then;
                l ← 1 (lmax = 4)
            End if;
            n ← n + 1;
          End of the loop
          spr(h) ← y'(h) ;
      End of the loop % End of generation
    Update Answers:
    The best solution is among the solutions set y'';
    If y'' better than ỹ set
        ỹ ← y'' and k ← 1,
```

Otherwise; k ← k + 1;
End of if
Continue inner loop Until k>k_{max} (k_{max} = 4)
Continue until the stop conditions are met.

In the proposed algorithm, first the best global solution is obtained by the GA, and then the parallel variable neighborhood search algorithm is applied to the solution found at each iteration. This step is repeated until the termination criteria are met. As shown in this pseudo-code, the necessary neighbourhood structures for exploring the solution space are identified initially. The stopping condition of the algorithm is set as the maximum number of iterations determined at the initialization step. In this algorithm, its core consists of two loops: an internal loop that searches the solution space and an external loop that controls the stopping condition. The notation k represents the counter for the inner loop iterations, and kmax represents the maximum number of inner loop iterations, respectively. A number of processors are employed in the search process within the inner loop. In each iteration of the inner loop, each processor, based on the input solutions, performs an iteration of the successive VNS algorithm, which includes the perturbation and local search phases. After each iteration of the PVNS algorithm, the stopping condition is checked by the external loop. If the stopping condition is not met, the next iteration of the algorithm begins. The PVNS algorithm continues until the maximum number of iterations is reached. When the PVNS algorithm terminates, the final current solution is reported as y*, which represents the best solution. The flowchart of the proposed algorithm can be seen in Figure 2.

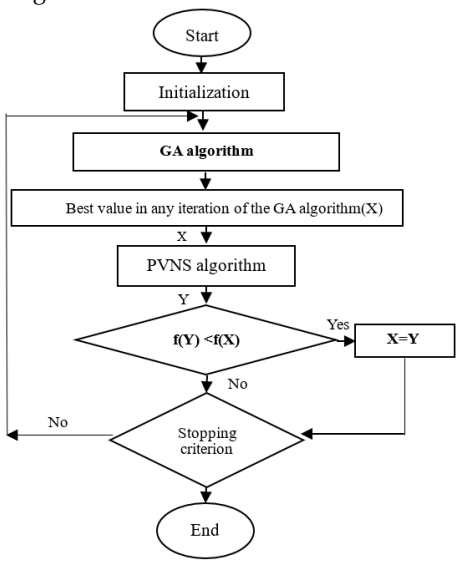

Figure 2. Flowchart of the proposed hybrid algorithm

4.1. Solution representation

The representation of each member of the population has two phases. The first phase represents the sequence of operations and the assignment of machines to operations in the cyclic flexible job shop scheduling problem, while the second phase represents the sequence of products in the assembly stage. In Phase 1, each element (component) of the part vector represents an operation of job j. In this solution representation structure, all operations of a job are represented with the same symbol. The length of the first phase vector is equal to the total number of operations. In the last row of both phases, a batch sequence vector is displayed. Each number in this vector represents the batch number to which the operation and product belong. Fig. 3 illustrates this solution representation.

Part	1	1	2	1	1	2	1	2	2	2	2	1
Product	1	2	2	1	2	1	1	2	1	2	2	1
Machine	3	3	1	2	1	3	1	3	2	3	2	2
Batch	1	2	1	2	1	2	1	1	1	2	2	2

Phase 1

Product	1	2	2	1
Batch	1	2	1	2

Phase 2

Figure 3. Schematic representation of a solution.

4.2. Initialization

The sequence of operations (parts and products) in Phase 1 and the sequence of products in Phase 2 are obtained by a random rule. To generate a machine vector in the first phase, based on Pezzella et al. two rules are used as follows:

- Assignment Rule 1: Search for the global minimum in the processing timetable.
- Assignment Rule 2: Randomly permute jobs and machines in the table.

In the proposed algorithm, 20% of the initial assignments are generated using Rule 1 and 80% using Rule 2.

4.3. Selection Operator

Selection is a method that randomly selects chromosomes from the population for reproduction. In this study, the Roulette Wheel Selection operator was used. In this method, the wheel is divided into segments equal in number to the population members, and the area of each segment is proportional to the fitness of the corresponding chromosome. Parents are selected randomly, and if a new individual's fitness is better than or equal to that of its predecessor, it will be accepted.

4.4. Crossover Operator

In the current study, a single-point crossover is used. In this operator, a random crossover point is selected, and the segments beyond this point are exchanged, resulting in the production of two new chromosomes. Considering that the solution representation consists of two parts, the single-point crossover is applied to both sections.

4.5. Mutation Operator

The mutation operator introduces random alterations in the genes with a certain probability, thereby maintaining diversity in the search. The mutation operator used in this algorithm is the substitution mutation. In this method, two random positions in the string are chosen, and the values at those positions are interchanged. In this process, a number in the interval [0, 1] is generated. If this value is less than the mutation probability, the replacement is performed. The introduced mutation is applied to both parts of the solution.

4.6. Neighborhood Structure

The process of transitioning from one solution to its neighboring solution is performed using a key factor called the neighborhood structure. In the proposed algorithm, several neighborhood structures are used. The following four neighborhood structures are introduced:

- Neighborhood Structure 1: In this structure, a cell is randomly selected, and the operation/product in the selected cell is replaced with that from its preceding cell.
- Neighborhood Structure 2: In this neighborhood structure, a random operation that can be processed on more than one machine is selected. Then, a new machine is chosen at random and assigned to this operation.
- Neighborhood Structure 3: Neighborhood Structures 1 and 2 are simultaneously run on the candidate solution.
- Neighborhood Structure 4: In this structure, first, the machine with the highest workload and the machine with the lowest workload from the candidate solution are selected. Then, one operation is randomly selected from the machine with the highest workload and assigned to the machine with the lowest workload.

5. Computational Results

In this section, the proposed mathematical model and hybrid algorithm are validated. For this purpose, the mathematical model is solved using GAMS software, and the results are compared with those obtained by the hybrid algorithm in small-sized instances. In this regard, a genetic algorithm is used as a benchmark to measure the performance of the hybrid algorithm. This algorithm was implemented in accordance with the algorithm presented by Jalilvand-Nejad and Fattahi. Both the proposed algorithm and the GA were coded using MATLAB software. To evaluate the performance of the proposed algorithm, the relative percentage deviation (RPD) (Eq. (24)) factor is used.

$$\text{RPD}= \frac{\text{Algorithm}_{\text{solution}} - \text{Minimum}_{\text{solution}}}{\text{Minimum}_{\text{solution}}} \times 100 \qquad (24)$$

In this study, the Taguchi method is used for parameter tuning of the proposed algorithm. In this method, for eight parameters of the hybrid algorithm, three levels are considered. The results of the Taguchi method for each level of factors are shown in Figure 3 and Table 2. In the diagram of SN-ratios, the level of factors with the highest SN-ratios value is selected as the best factor level [31]. In order to evaluate the performance of the proposed algorithm and the quality of results, 18 instances are randomly generated. The information of generated instances is shown in Table 3. Also, for each instance, the algorithm is run ten times and instances are categorized into three groups (small, medium and large). In this section, the performance and performance of the hybrid algorithm are compared with the GA algorithm in solving different problems. The obtained results in solving small problems are shown in Table 3. The stopping condition of the algorithm is the maximum number of iterations that is specified in the initialization step.

Table 2. Factors and suitable values of the GA-PVNS algorithm parameters

Metaheuristic	Parameters	Factors level	Suitable level
GA	Initial population: nPop	30-40-50	40
	Probability of crossover: pc	0.5-0.8-1	0.5
	Probability of mutation: pm	0.5-0.8-1	0.5
	Maximum number of iterations of the proposed algorithm: Max iTT	80-120-150	80
PVNS	Number of processors: n-pr	3-4-5	3
	Max it	60-70-80	80
	nmax	30-40-50	30
	kmax	2-3-4	4

Table 3. Size of random data

Parameters	Values
Number of machines	{2,3,...,7,8}
Number of products	{2,3,...,9,10 }
Number of parts	{1,2,3,...,5}
Processing time of operation	Uniform distribution (1,5)

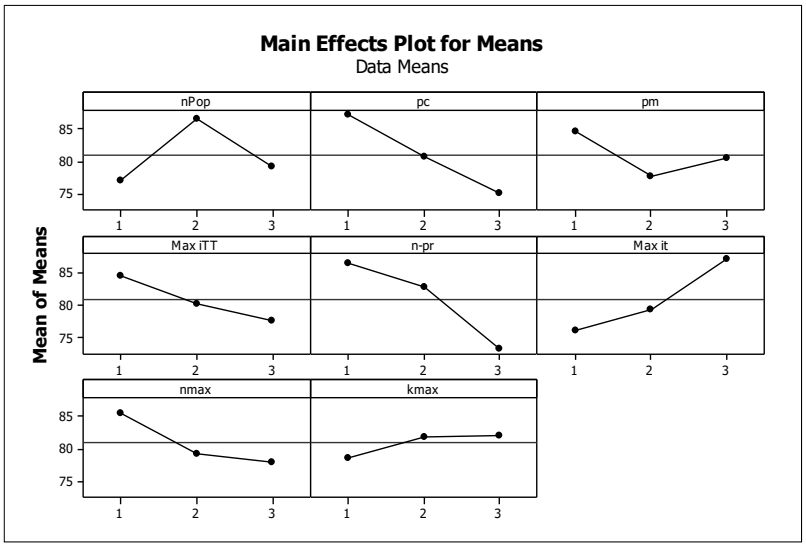

Figure 3. Diagram of average values SN ratios for each level of factors in the GA-PVNS algorithm.

For these problems, the solution is computed using GAMS software. Due to the complexity of the problem, medium- and large-sized instances are solved using the GA-PVNS and GA algorithms. The results are shown in Table 3. Considering the RPD values, the average RPD for medium- and large-sized instances is plotted in Figure 4 and Figure 5, respectively. Based on the RPD values and the average RPD diagrams for medium- and large-sized instances, the GA-PVNS algorithm exhibits better performance than the other algorithms. Also, in this paper, to compare the capabilities of the algorithms in finding good solutions in medium-sized and large-sized instances, they were statistically examined. In this regard, first, the normality of the data was examined. The Shapiro-Wilk test was used to check the normality of the RPD values. The test results regarding the normality of the algorithms are shown in Table 4. The output of this test shows that the P-value for the RPD for at least one of the two algorithms is less than 0.05. It can be concluded that the RPD values of the algorithms do not follow a normal distribution. Therefore, the non-parametric Kruskal-Wallis test was used for statistical analysis.

Table 3. Computational results sample problems.

		GAMS		GA-PVNS		GA	
	Problem	Total Tardiness	RPD	Total Tardiness	RPD	Total Tardiness	RPD
Small	P1	0	0	0	0	0	0
	P2	0	0	0	0	0	0
	P3	0	0	0	0	0	0
	P4	1	0	1	0	1	0
	P5	3	0	3	0	3	0
Medium	P6	-	-	63	0	75	19.04
	P7	-	-	63	1.61	62	0
	P8	-	-	305	0	364	19.34
	P9	-	-	160	0	187	16.87
	P10	-	-	192	0	221	15.1
	P11	-	-	204	0	218	6.86
	P12	-	-	682	0	761	11.58
	P13	-	-	311	0	365	17.36
Large	P14	-	-	1893	0	2197	16.05
	P15	-	-	1523	0	1847	23.04
	P16	-	-	3527	0	3704	5.01
	P17	-	-	3441	0	3906	13.51
	P18	-	-	1824	0	2203	20.78

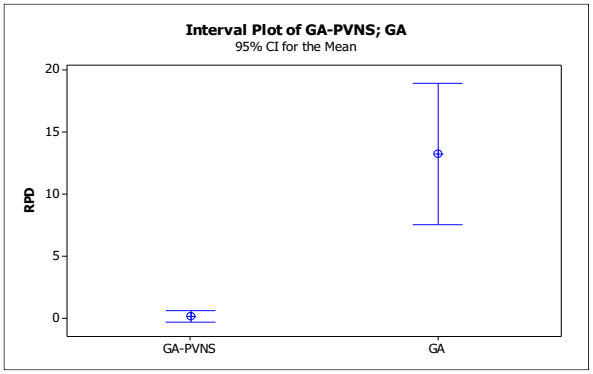

Figure 4. Diagram of average RPD for medium problems

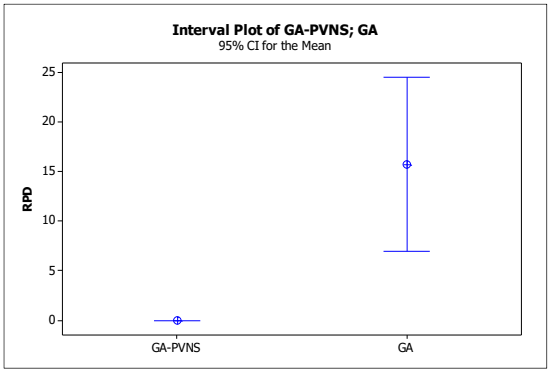

Figure 5. Diagram of average RPD for large problems

The results of the statistical analysis for the RPD values are shown in Table 5. As shown in this table, the P-value for the RPD is less than 0.05, leading to the rejection of the null hypothesis. Therefore, we conclude that there is a significant difference between the GA-PVNS and GA algorithms. According to the statistical test results, it can be concluded that the GA-PVNS algorithm performs better than the GA.

Table 4. Shapiro-Wilk test results for the RPD values of algorithms

	w	DF	P-value
GA-PVNS	0.311	13	0.000
GA	0.928	13	0.325

Table 5. Kruskal-Wallis test results for the RPD values of algorithms

KW chi-squared	18.04
DF	1
p-value	0.000

6. Conclusion and further research

In this paper, a hybrid algorithm is proposed for the cyclic flexible job shop scheduling problem with assembly operations and setup time. The objective function is to minimize total tardiness. Initially, a mathematical model was developed. Due to the complexity of the considered problem, a hybrid metaheuristic algorithm based on a Genetic Algorithm (GA) and Parallel Variable Neighborhood Search (PVNS) is proposed to solve medium- and large-sized instances. To validate the hybrid algorithm, it was compared with the GA algorithm. Computational results showed that the GA-PVNS algorithm performs better than the GA algorithm. There were two major limitations to this study. The first is the lack of consideration of uncertainty in the problem. The failure to consider solution methods based on exact algorithms also constitutes a limitation of the current study. Based on the introduced limitations and shortcomings of the current research, suggestions for future research include: (i) the development of a multi-objective model, (ii) the incorporation of uncertainty into the problem parameters, (iii) the development of exact algorithms, and (iv) the investigation of other effective metaheuristic algorithms.

References

T. Borreguero-Sanchidrián, R. Pulido, Á. García-Sánchez and M. Ortega-Mier, 2018. Flexible job shop scheduling with operators in aeronautical manufacturing: A case study. IEEE Access, 6: 224-233.

Lee, C.Y., Cheng, T.C.E., Lin, B.M.T. 1993. Minimizing the makespan in the 3- machine assembly-type flow shop scheduling problem. Management Science, 39(5): 616-625.

Sculli, D. 1979. Priority Dispatching Rules in Job Shops with Assembly Operations and Random Delays. OMEG, 8(2): 227-234.

Cheng, T.C.E. 1990. Analysis of material flow in a job shop with assembly operations. International Journal of Production Research, 28(7): 1369-1383.

Zhang, R.A. 2011. Simulation-based Genetic Algorithm for Job Shop Scheduling with Assembly Operations. International Journal of Advancements in Computing Technology, 3(10):132

Dimyati, T. 2007. Minimizing production flow time in a process and assembly job shop. Proceedings of the International Seminar on Industrial engineering and Management, Jakarta, 68- 73.

Rossi, A. 2014. Flexible job shop scheduling with sequence-dependent setup and transportation times by ant colony with reinforced pheromone relationships. International Journal of Production Economics, 153: 253-267.

Shen, L., Dauzère-Pérès, S., Neufeld, J.S. 2018. Solving the flexible job shop scheduling problem with sequence-dependent setup times. European Journal of Operational Research, 256(2):503-516.

Azzouz, A., Ennigrou, M., Said, L.B. 2017. A self-adaptive evolutionary algorithm for solving flexible job-shop problem with sequence-dependent setup time and learning effects. 2017 IEEE Congress on Evolutionary Computation (CEC).

RamezaniEmai, P., Rabiee, M., Jolai, F. 2015. No-wait flexible flowshop with uniform parallel machines and sequence-dependent setup time: a hybrid meta-heuristic approach. Journal of Intelligent Manufacturing, 26(4): 731-744.

Mousakhani, M. 2013. Sequence-dependent setup time flexible job shop scheduling problem to minimise total tardiness. International Journal of Production Research, 51(12): 3476-3487.

Cuninghame-Green, RA. 1960. Process synchronisation in a steelworks- a problem of feasibility, Proceedings of the 2nd international conference on operational research, 323-328.

Cuninghame-Green, RA. 1962. Describing industrial processes with interference and approximating their steady-state behavior, Operational Research Society, 13(1): 95-100.

Hanen, C., & Munier, A. 1997. Cyclic scheduling on parallel processors: An overview. In P. Chretienne, E. G. Coffman, J. K. Lenstra, Z. Liu (Eds.), Scheduling theory and its applications. New York: Wiley.

Brucker, P., & Kampmeyer, T. 2008a. A general model for cyclic machine scheduling problems. Discrete Applied Mathematics, 156(13): 2561–2572.

Brucker, P., & Kampmeyer, T. 2008b. Cyclic job shop scheduling problem with blocking. Journal of Intelligent Manufacturing, 159(1): 161–181.

Romanova, A. A., & Servakh, V. V. 2009. Optimization of processing identical jobs by means of cyclic schedules. Journal of Intelligent Manufacturing, 3(4), 496–504.

Ouenniche, J., & Bertrand, J. W. M. 2001. The finite horizon economic lot sizing problem in job shops: The multiple cycle approach. International Journal of Production Economics, 74, 1: 49–61.

Bożejko, W., Gnatowski, A., Pempera, J., Wodecki, M. 2017. Parallel tabu search for the cyclic job shop scheduling problem, Computers & Industrial Engineering,113:512-524.

Jalilvand-Nejad.A., Fattahi, P. 2015. A mathematical model and genetic algorithm to cyclic flexible job shop scheduling problem. Journal of Intelligent Manufacturing, 26(6): 1085-1098.

Quinton, F., Hamaz, I., Houssin, L 2018. Algorithms for the flexible cyclic job shop problem, 14th International Conference on Automation Science and Engineering (CASE), Munich, Germany.

Elmi, A, Thiruvady, DR, Ernst, AT. Blocking Cyclic Job-Shop Scheduling Problems. Algorithms. 2022; 15(10):375.

Fattahi, P., Bagheri Rad, N., Daneshamooz, F. and Ahmadi, S. 2020. A new hybrid particle swarm optimization and parallel variable neighborhood search algorithm for flexible job shop scheduling with assembly process", Assembly Automation.

Garey M.R, Johnson D. S, Sethi R. The complexity of flow shop and job shop scheduling. Mathematics of Operations Research 1976; 1(2): 117–29.

Mladenovic, N. and Hansen, P. 1997.Variable neighborhood search", Computers and Operations Research, 24(11), pp. 1097-1100.

Yazdani, M., Amiri, M., & Zandieh, M. 2010. Flexible job shop scheduling with parallel variable neighborhood search algorithm. Expert system with applications, 37(1): 678-687.

Pezzella, F., Morganti, G., Ciaschetti, G. 2008. A genetic algorithm for the Flexible Job-shop Scheduling Problem. Computers & Operations Research, 35(10):3202-3212.

Cui, Z., Chang, Y., Zhang, J., Cai, X., & Zhang, W. 2019. Improved NSGA-III with selection-and-elimination operator, Swarm and Evolutionary Computation, 49: 23-33.

Koohestani, B. 2020. A crossover operator for improving the efficiency of permutation-based genetic algorithms, Expert Systems with Applications, 151: 113381.

Cazacu, R. 2017. Comparative Study between the Improved Implementation of 3 Classic Mutation Operators for Genetic Algorithms, Procedia Engineering, 181: 634-640.

Komijan, R.A., Tavakkoli-Moghaddam, R., Dalil, S. 2020. A mathematical model for an integrated airline fleet assignment and crew scheduling problem solved by vibration damping optimization. Scientia Iranica.