Real-Time Order Acceptance and Scheduling Problems in a Flow Shop Environment Using Hybrid GA-PSO Algorithm

Rahman, H. F.; Janardhanan, M. N.; Nielsen, I. E.

Published in:
IEEE Access

DOI (link to publication from Publisher):
10.1109/ACCESS.2019.2935375

Publication date:
2019

Document Version
Publisher's PDF, also known as Version of record

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 15, 2019, accepted August 4, 2019, date of publication August 14, 2019, date of current version August 27, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2935375

Real-Time Order Acceptance and Scheduling Problems in a Flow Shop Environment Using Hybrid GA-PSO Algorithm

HUMYUN FUAD RAHMAN[1], MUKUND NILAKANTAN JANARDHANAN[2], AND IZABELA EWA NIELSEN[3]

[1]School of Engineering and IT, University of New South Wales, Canberra, ACT 2620, Australia
[2]Mechanics of Materials Research Group, Department of Engineering, University of Leicester, Leicester LE1 7RH, U.K.
[3]Department of Materials and Production, Aalborg University, 9220 Aalborg, Denmark

Corresponding author: Mukund Nilakantan Janardhanan (mj251@le.ac.uk)

ABSTRACT With the emergence of new Industry 4.0 technologies, real-time order acceptance and scheduling is a key problem in a make-to-order (MTO) production system where customers place orders in real-time and the decision maker has to make acceptance or rejection decisions based on the available resources at that point in time. This paper focuses on simultaneously accepting orders and scheduling decisions in real-time, as is required for the operation of an MTO flow shop production system, a topic that has received little attention in academia due to the complexity of the problem. This paper presents a hybrid genetic algorithm and particle swarm optimization algorithm (GA-PSO) to solve the considered problem. A detailed computational study based on realistic problem instances has been conducted. In this study, the hybrid GA- and PSO-based approach performed better than other state-of-the-art approaches reported in the literature.

INDEX TERMS Order acceptance and scheduling, particle swarm optimization, real-time order arrival, genetic algorithm, flow shop scheduling.

I. INTRODUCTION

The permutation flow shop scheduling problem (PFSP) is one of the most challenging scheduling problems that arises in manufacturing industries such as pharmaceuticals, food processing, steel, automobiles, and semiconductors [1], [2]. In a traditional PFSP, a set of n jobs is scheduled in a set of m machines, where each job has to follow the same processing order in all machines. For solving PFSPs, makespan minimization is a common measure of performance. In flow shop manufacturing systems, the production system can be either make-to-order (MTO) or make-to-stock (MTS) production systems [1]. Over the last decade, a huge volume of research has been reported on PFSPs that considers static situations where an order (consisting of a set of jobs) is processed by the set of machines and scheduling is performed only once. In static single and multiple order PFSPs, the production managers receive a single order (for single order PFSPs) or a pool of orders (for multiple orders) at the beginning of production, along with a due date which is known in advance. For static problems, both single and multiple order PFSPs are expected to be solved only once. Johnson [3] introduced the idea of the static single order or traditional PFSP, proposing an optimal algorithm for two and special three machine single order PFSPs. Until then single order PFSPs have been widely studied by researchers. To solve single order PFSPs with more than two machines, many researchers proposed detailed algorithms, such as the branch and bound (B&B) [4], and integer programming [5] algorithms. However, as the problem is classified as NP Hard [6], these algorithms can only find optimal solutions for small-sized problems. To solve bigger-sized PFSPs, researchers have focused on developing heuristics [7] and meta-heuristic algorithms [8]–[17]. The next level of complexity of the order acceptance and scheduling problem is static multiple order PFSP, where the manufacturers receive a pool of customer orders at the beginning of production [18]. In this case, arrival time, due date and composition of each order are known in advance. Static order acceptance and scheduling problems have been studied for single and multiple machine environments [18]–[24].

The associate editor coordinating the review of this article and approving it for publication was Jagdish Chand Bansal.

Although static single and multiple order PFSPs have been widely studied in the literature, static scheduling problems have limited real-life applications since new orders arrive randomly in the system [25] and production capacity varies with time [26]. With the fourth generation industrial revolution (Industry 4.0) [27], multiple order PFSPs have become dynamic and customer oriented, whereby customers can place their orders with the desired order compositions (such as product specifications) and due dates into the system in real-time. In this case, the decision makers have to make two decisions sequentially. First, they must accept or reject the order while considering available machine capacity and the customer-specified due date. Second, the jobs involved in each new order must be scheduled with the existing accepted orders, which are either already being processed by the machines or are waiting in the processing queue. These decisions can be taken more effectively if the jobs are scheduled properly. Accordingly, the decision maker has to accept a set of orders that are feasible to complete when taking the production capacity available at that point in time into account, in addition to scheduling and assigning the jobs of the selected orders to the m machines on the shop floor. Therefore, for this class of problem, order acceptance decisions have to be made instantly, i.e. once the order is placed in the system. The considered problem is practical but more complex than the reported static PFSPs (static single order PFSPs and static multiple order PFSPs). To the best of the authors' knowledge, very few studies have been reported on real-time multiple order PFSPs, and the reported ones oversimplify the problem and are very challenging to implement in real-world MTO environments.

Due to these limited assumptions, there is a significant gap between theoretical flow shop scheduling research and practical manufacturing environments. This gap provides an opportunity to exploit the potential of advanced MTO flow shop environments arising due to the application of advanced manufacturing technology such as Industry 4.0 elements by developing novel scheduling algorithms for finding effective solutions for real-time multiple order PFSPs. In multiple order PFSPs, individual orders can be considered as a single order PFSP, but with no prior information about the order's arrival time, due date and composition of the order. Due to the high level of complexity, classical optimization techniques are not suitable for solving the problem under study. Therefore, it is essential to develop an efficient search technique to achieve a fast and feasible solution in near real-time. Metaheuristic algorithms have a good track record of solving complex combinatorial optimization problems similar to the scheduling problem under study. To solve this problem, this paper proposes a hybrid approach by integrating a genetic algorithm (GA) and particle swarm optimization (PSO), referred to hereafter as a GA-PSO-based real-time strategy for solving real-time multiple order PFSPs. The objective of the problem is to maximize the number of accepted orders. Researchers have been proposing hybrid algorithms for the last few years [28], and they are gaining popularity because of their ability to combine the strengths of different methods under a common scheme. By hybridizing two metaheuristics, the resulting algorithm reduces each individual metaheuristic algorithm's weaknesses and improves overall performance for several complex optimization problems [29], [30], similar to the problem under study. Despite of its real-industrial applications, these problems are hardly studied and reported in the literature. To the authors' best knowledge, real-time multiple order problem was first studied Rahman, *et al.* [31]. They proposed a GA based real-time approach for solving the problem. As real-time multiple-order PFSP is a complex scheduling problem, there is still room for improvement to find better schedules for a newly arrived order and therefore, increase the chance of its acceptance. Motivated by this fact, major contribution of this present study is to propose and test hybrid GA-PSO based approach for solving the real-time multiple order PFSPs. By hybridizing GA and PSO, better solutions are obtained than those obtained using individual algorithms (e.g. only GA or only PSO based approach) in short computational time which is critical in real time scheduling. For comparison, PSO based approach is also developed. Based on the experimental results it can be seen that the proposed hybrid GA-PSO-based real-time approach show superior performance when compared to the PSO-based real-time approach and the state-of-the-art approach reported in Rahman, *et al.* [31] based on the same set of problem instances generated in [31]. Finally, performance of the proposed approaches has been evaluated based on a real-world sanitary ware production system [31]. The proposed approach will be advantageous for real time decision making that can help production managers to ensure profitability and provide higher customer satisfaction.

The remainder of the paper is organized as follows: the relevant literature is discussed in Section II, and Section III formally defines the problem under study. The proposed algorithms for solving real-time multiple PFSPs are described in Section IV. An experimental analysis and an assessment of the effectiveness of the proposed approaches are presented in Section V. Finally, Section VI presents the conclusion of this study and possible future research directions.

II. LITERATURE REVIEW

In MTO systems, production managers have to manage order acceptance and scheduling decisions for a single order or stream of orders that dynamically enter either a single or multiple machine production system [18]. This section provides a brief literature review about dynamic order acceptance and scheduling problems in both single and multiple machine environments. Dynamic order acceptance and scheduling problems in single [32] and multi-machine [31], [33]–[37] environments produce practical and realistic solutions; however, it is significantly more complicated than static order acceptance and scheduling problems.

The considered problem in this paper is similar to dynamic order acceptance and scheduling problems [32]. The relationships between four different order-acceptance strategies are

reported: 1) accepting orders based on workload estimation, 2) order selection based on aggregated load profile, 3) order acceptance based on the effect of tardiness of previous accepted orders, and 4) order selection based on the current state of the production schedule to maximize the utilization. Of the three order selection strategies, using available information on the current state of production is widely considered to be superior to the other two approaches. Duenyas and Hopp [33] considered stochastic order arrival and extended the problem for a job shop environment with customer tolerance limits based on order acceptance or rejection policies to maximize expected profit. Duenyas [34] extended this study by considering multiple customer classes and customer-quoted due dates. For multiple machine environments, Nandi and Rogers [35] proposed a simulation-based order acceptance and scheduling strategy. To solve this problem, they set profit maximization as the objective and considered two different types of customer order (regular and urgent). Later, they proposed another approach based on a simulation tool [36], which considered the resources required to process the order and its load on the busiest machine. They also proposed a priority rules (earliest due date, minimum slack per operation, and first come, first served) -based approach for scheduling the orders in the system. Moreira and Alves [37] used simulations to minimize lateness penalties and improve workload performance in a job shop environment. They also utilized priority rules to schedule the orders. Tang *et al.* [38] studied a dynamic order acceptance and scheduling problem, considering the minimization of average flow time, number of tardy jobs, and tardiness as their objectives. To solve the problem, they developed a neural network approach that integrates six priority rules. For real-time multiple order PFSPs the only study available in the literature is presented in [31]. They propose an order acceptance heuristic and GA-based real-time strategy to solve the problem. Since real-time multiple order PFSPs are classified as NP Hard, there is scope to improve order acceptance and scheduling decisions in real-time multiple order PFSPs. Another approach is proposed by .Eriksen and Nielsen [39], who investigated the number of incoming customer order requests that need to be aggregated in order to establish a stable inflow of orders.

From the above mentioned literature review, it can be summarized that order acceptance and scheduling problems have been studied with some restrictive assumptions, which are: (1) as with typical order acceptance and scheduling problems, most of the studies, when formulating their problems, considered static conditions, such as order arrival, due date, and composition, to be known. However, in practice, orders enter the systems in a dynamic fashion; (2) most of the studies considered order acceptance/rejection decisions and scheduling separately and sequentially; (3) most of the studies of dynamic order acceptance and scheduling considered small-sized problems (with a small number of machines); and (4) when scheduling complex orders, most of the studies considered only simple priority rule-based approaches. These assumptions isolates the above mentioned studies from real-life manufacturing scenarios. Hence, this paper aims to derive an approach for solving real-time PFSPs in an efficient manner, which is more realistic and relevant.

Over the last few years bio-inspired optimization techniques have become popular in solving wide range of solving real world complex decision making problems [40]. Like other optimization domains, there has been a growing literature in the field of bio-inspired techniques for solving different types of scheduling problems, e.g. job shop [41], flow shop [1], single machine [42], assembly line production [43], resource constrained project scheduling [44].

Among these scheduling problems, real-time or dynamic scheduling problem is more challenging than the static scheduling. Since real-time events may cause a change in scheduling, the initial feasible and good schedule may turn into infeasible after uncertain event occurs [45]. Hence, real-time scheduling problem is a complex decision-making process considering two factors: (1) difficulty in finding an optimal or near-optimal solution for the problems and (2) finding that solution within reasonable amount of time due to real-time decision process. Considering these factors, bio-inspired algorithms are excellent choice for solving such problems, as they can meet these criteria. Furthermore, these algorithms are easy to implement. A good number of research studies have been published on solving real-time scheduling problems by efficient bio-inspired optimization techniques. For example, Rossi and Dini [46] proposed a genetic algorithm based approach for solving job shop scheduling problem (JSSP) considering the following real-time events: dynamic batch arrival, unavailability of parts in production floor, and machine breakdowns. Souier, *et al.* [47] proposed six metaheuristic-based optimization techniques for solving the real-time alternative route selection problem in flexible manufacturing environment with the objective of reduction of the congestion in the system. Among them three meta-heuristic techniques are bio-inspired algorithms. The study was extended by [48] by considering uncertainty in the problem and the authors proposed a genetic algorithm based approach for solving the problem. Tang, *et al.* [49] proposed an improved differential evolution algorithm for solving dynamic or real-time scheduling in steelmaking-continuous casting production problems. Similarly, Lu, *et al.* [50] proposed a grey wolf optimizer for solving dynamic or real-time scheduling problems in a real-world welding industry. Sama, *et al.* [51] proposed an ant colony-based approach for solving real-time train scheduling and routing problems in a railway network. Based on our preliminary research and the literature review, it can be seen that researchers have focused on developing and utilizing bio-inspired optimization techniques for solving real-time complex scheduling problems from different real-life environments (e.g. manufacturing and transportation), which motivate us to propose efficient GA-PSO (bio-inspired optimization technique) based approach for solving real-time multiple order permutation flow shop scheduling problem.

III. PROBLEM DEFINITION AND ASSUMPTIONS

This section briefly describes the formulation of real-time multiple order PFSPs similar to the one reported in [31]. Due to dynamic or real-time behavior of the problem, this problem is the more complex than the single order or static multiple order PFSPs. The problem is defined as the 'real-time scheduling problem' since in the shop floor environment, customers place orders in the system in real-time and decision makers do not have prior information about the orders (e.g. arrival time, number of jobs per order, and due date). While scheduling and processing orders on the production floor, the decision makers requires to make order acceptance and scheduling decisions for newly arrived orders. The following measures of performance and constraints of a multiple order PFSP are described as follows.

$j-$ index of a machine from flow shop

$m-$ total number of machines on the production floor

$g-$ the index of an order

$i-$ the index of a job from the order g

$G-$ total number of customer orders arriving during a single production shift

$p_{ij}^{g}-$ processing time of i^{th} job from order g in j^{th} machine

S_j^g- start time of the first job from order g in j^{th} machine

F_j^g- completion time of the last job from order g in j^{th} machine

C_{max}^g- makespan of order g

C_{com}^g- completion time of order g

d_g- delivery time or due date of order g

The completion time for each order can be expressed as:

$$C_{com}^g = S_j^g + C_{max}^g \quad \text{for } g = 1, 2 \ldots . G \tag{1}$$

Maximum completion time or makespan, C_{max}^g (equivalent to makespan of a static single order PFSP), of an order g is the time difference between the start time of the first job at the first machine and the end time of the last job from the same order.

At the beginning of production, completion time of the first accepted order is equal to its makespan, since all machines are available to process the job:

$$C_{com}^{g=1} = C_{max}^{g=1} \tag{2}$$

The start time of the first job from the first accepted order in the first machine is 0.

$$\text{where,} \quad S_{j=1}^{g=1} = 0 \tag{3}$$

Tardiness (lateness) of an order is

$$T_g = C_{com}^g - d_g \quad \text{for } g = 1, 2, \ldots . G \quad \text{and } d_g \geq 0 \tag{4}$$

The objective function of the real-time multiple order PFSP is to maximize the total number of accepted orders (TAO) in the production shift. This function is presented in Equation 5.

$$TAO = \sum_{g=1}^{G} A_g \tag{5}$$

where,

$$A_g = \begin{cases} A_g, & T_g \leq 0 \; order \; is \; accepted ï£¡ \\ 0, & order \; is \; rejected \end{cases} \tag{6}$$

In formulating the real-time multiple order PFSPs the following assumptions are held.

- Order composition (e.g. number of jobs per order and processing time), arrival times, and due dates are not known in advance.
- The manufacturer rejects the order if it is tardy (late).
- Throughout production, the production process is uninterrupted.
- Setup and transportation times are included in the processing time.
- If an order is accepted, it cannot be rejected.

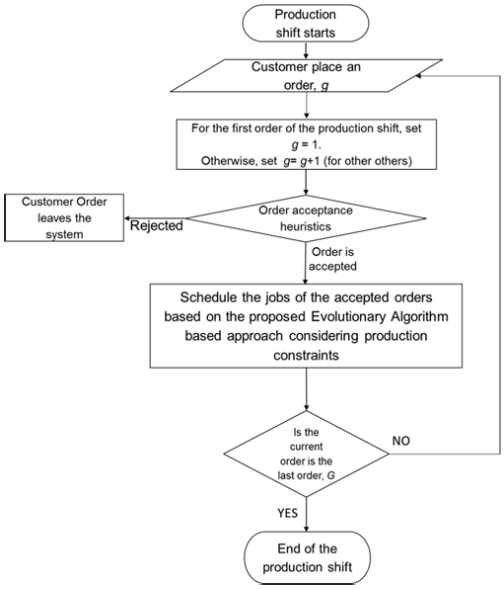

FIGURE 1. A framework for multiple order PFSPs.

IV. SOLUTION APPROACH

This section presents the proposed GA-PSO-based real-time strategy for solving multiple order PFSPs. A framework of the proposed real-time multiple order PFSP is presented in Figure 1. Based on Figure 1, first the order acceptance and rejection approach is presented. Following this, a description of the procedure of implementing GA and PSO algorithm for the hybrid GA-PSO algorithm for solving real-time multiple order PFSPs is described. The idea behind hybridizing GA-PSO is to combine the advantages of both of these algorithms and simultaneously overcome their disadvantages [52]. By combining the genetic algorithm operation and PSO operations, exploration and exploitation ability are further improved [29].

A. ORDER ACCEPTANCE AND REJECTION DECISIONS

The decision to accept or reject an order is made when the order enters the system. The decision depends on the

arrival time of the order, the availability of machines in the shops at that time, and the due date. It is to be noted that here machine availability means whether the machines are busy or unavailable for certain periods of time due to processing previously accepted orders. An order is accepted only if it can be completed before its due date. Therefore, to calculate the completion time of an order, a hypothetical schedule is generated based on machine availability or unavailability. In this study, order acceptance and rejection decisions are based on the order acceptance/rejection heuristic described in [31]. The schedule or completion time is calculated based on a metaheuristic algorithm which is presented in the next section.

B. SCHEDULING DECISIONS

Once an order is placed by a customer, a hypothetical schedule of that order is generated, considering machine availability/unavailability constraints at that point in time and the orders currently under process. On the basis of that hypothetical schedule, an order acceptance or rejection decision is made, if jobs of that order can be completed within its due date. Otherwise, the order is rejected. Therefore, generating an effective schedule within a reasonable time is critical in order acceptance/rejection decisions.

When an order is registered in the production system, a hypothetical schedule of that order is generated, since there is a likelihood that it cannot be processed immediately due to machine availability constraints. However, if the first machine is free when the job arrives, the job can be immediately processed into the first machine without violating the machine availability constraints, and without interrupting the other accepted orders which are already being processed by other machines. If the first machine or the following machines are not immediately free, then the orders need to wait in a queue for processing until the next machine is free. Scheduling the jobs of each order while considering the current state of availability at that time may reduce the completion time of each order and therefore increase the possibility of its acceptance. Therefore, on basis of that hypothetical schedule, an order acceptance or rejection decision is made on basis of local view of whether all jobs of that order can be completed within its due date or not. This approach is called real-time strategy for multiple order PFSPs. A simple example of 3 machine flow shop and 2 orders (e.g. order X and order Y), with Gantt charts in Figure 2 illustrates the overall process. Each of the order contains 5 jobs. In Figure 2, the top Gantt chart represents the order X. This order has the static single order makespan (and completion time) is 30 and the sequence: 2-5-3-4-1. The order X is accepted since it has the due date of 48 units of time ($>$ completion time, 30). Assume that order Y arrives in the production floor after 15 units of time from the start of processing the order X. Hence, the order Y arrives in the system while order X is in process. The due date for order Y is 57. The second Gantt chart of Figure 2 shows that the static single order makespan for order Y is also 30 and sequence: 2-5-3-4-1.

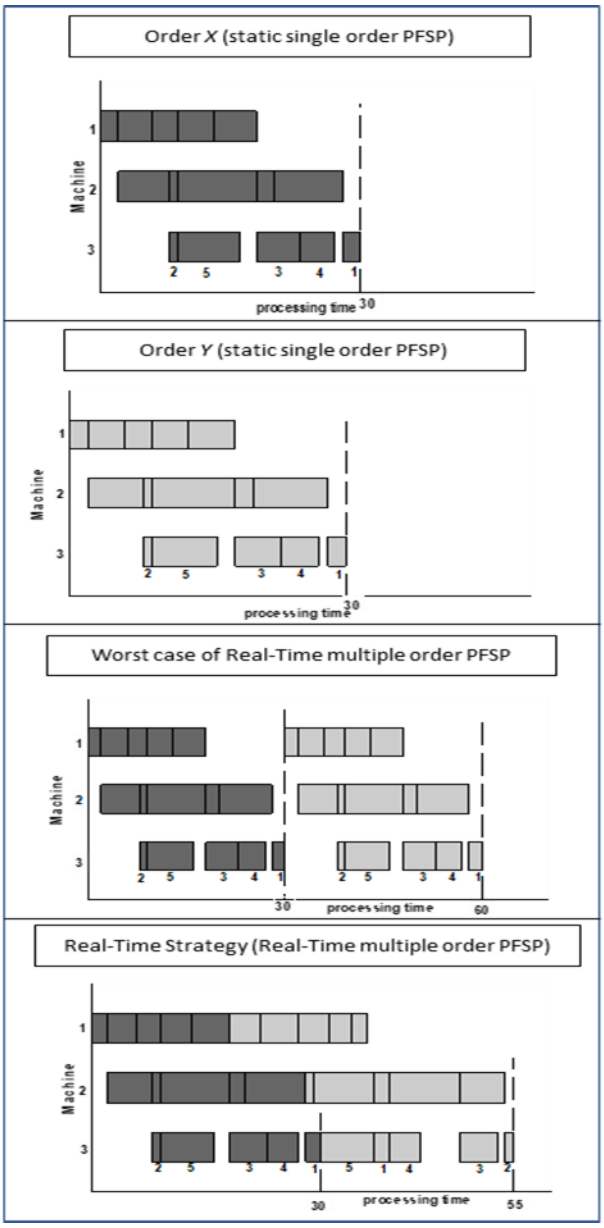

FIGURE 2. Real-time strategy for real-time multiple order PFSP.

In the worst-case scenario, order Y starts to process its first job just after finishing the last job of order X. It gives the completion time of order Y is 60 ($= 30+30$). The third Gantt chart shows the worst-case scenario. This scenario refers to the upper bound of the problem and it will be referred later. Now if the order Y starts to process as soon as the first machine completes order X and schedule the jobs of order Y while taking into consideration of availability of some machines for only certain time windows. For convenience of understanding, this approach is referred as Real-time approach for solving real-time multiple order PFSP. It is important to note that static single order schedule is generated assuming all machines are available at any time. Hence, the new schedule generated by real-time strategy varies from the static single order schedule. As shown in the last Gantt

chart of Figure 2, after applying the real-time strategy, the job sequence for order Y is 5-1-4-3-2, which is different from its static schedule. The completion time is 55 and it can also be accepted (<due date 57).

In this study a GA-PSO based real-time approach is proposed for solving multiple order PFSPs. The motivation for selecting GA and PSO algorithms for making critical scheduling decisions is to use the strength of both algorithms, since they have been developed with the aim of carrying out a global search with two main purposes: solving large problems in a reasonable time and obtaining robust algorithms. However, both the algorithms have strengths and weaknesses. In GA, if an individual is not selected from the information contained, that individual is lost, but PSOs keep this information in memory. On the other hand, PSOs waste computation time on poor individuals since they do not have a selection operator. By hybridizing GA and PSO one can combine the advantages of these algorithms, capable of reaching a global region (for GA) and group interaction boost in the search for optimum solutions (for PSO). In the following section, the proposed GA and PSO algorithms for the hybridized GA-PSO algorithm for solving multiple order PFSPs are presented.

C. GENETIC ALGORITHM

GA is a popular bio-inspired metaheuristic algorithm for finding optimal or near optimal solutions for complex combinatorial optimization problems [41], [53], [54]. The structure of GAs for solving multiple order PFSPs is described in this section. For solving different types of PFSPs, permutation representation of job sequences is widely accepted; a non-random initialization is proposed where the first member/individual of the initial generation starts with an NEH heuristic [7]. A certain percentage of members in the population is generated either by the random swapping of two jobs or by randomly inserting a job into a randomly selected position. Finally, the remaining members are generated randomly. A tournament selection technique [53] is applied to select parents to go through the crossover process to generate new offspring.

To maintain both quality and diversity in each generation, a new generational scheme is proposed whereby each population contains two sets of population – one set is the 80% of the population with high-quality solutions and remaining 20% is the diversified solutions. No duplication is allowed. Here, duplication refers to individuals with the same job sequence. Next, each of the o^{th} offspring will replace the existing member of the population in the same o^{th} position only if it is more diversified than the $(o\text{ -}1)^{th}$ member in the population. Exact position-based diversity measures [55] are applied for obtaining diversity. To avoid this situation a restart mechanism [11], [14], [15] is applied, where if the algorithm is unable to find better solutions after a certain number of generations, all individuals are sorted into three segments: the top 5% (the best individuals), the next best 45% (of medium quality), and the remaining 50% (the worst individuals). We apply an elitism strategy to preserve the best individuals from the current generation and transfer them to the next generation. In this algorithm, block order crossover (SBOX) [14] and shift mutations [11], [14], [56] have been selected as the reproduction operators as they generate promising candidate solutions. In SBOX, both parents' job sequence positions are checked one by one. Finally, to enhance the performance of the proposed algorithm, an insertion neighborhood-based [10], [14], [57] local search approach has been incorporated into the algorithm to enhance its performance.

D. PARTICLE SWARM OPTIMIZATION ALGORITHM

Particle Swarm Optimization (PSO) is a well-known metaheuristic algorithm which is based on the social behavior of flocks of birds or schools of fish [58]–[60]. PSO consists of a number of particles (a swarm) which represents a job sequence in the considered problem, where these particles move around in the search space and evolve, continuously trying to find a good feasible solution. Each particle is assigned with a velocity and moves around in the search space based on this velocity. Each particle keeps track of the best fitness they encounter, referred to as local best, and this information is passed between particles in order to determine the global best in the swarm. PSO is initiated with a set of solutions, referred to as a swarm, and each of the solutions is called a particle. The population of a PSO evolves with velocity and position updates, as shown in Equations 7 and 8. The algorithm terminates once the stopping conditions are met. The structure of a PSO for solving multiple order PFSPs is described below:

$$P_i^{t+1} = P_i^t + v_i^{t+1} \tag{7}$$

$$v_i^{t+1} = c_1 v_i^t + c_2 U_1 \left({}^e P_i^t - P_i^t\right) + c_3 U_2 \left(G^t - P_i^t\right) \tag{8}$$

where U_1 and U_2 are the velocity coefficients (random numbers between 0 and 1), v_i^t is the initial velocity, ${}^e P_i^t$ is the local best, G^t is the global best, P_i^t is the current particle position, and c_1, c_2 and c_3 are the learning coefficients. ${}^e P_i^t$ refers to the best position of the i^{th} particle at generation 't'. G^t refers to the best position of all particles at generation 't'.

1) POPULATION INITIALIZATION

To enhance the performance of PSO, it is essential to have a good set of initial particles, all of which are generated randomly. However in this study, we adopt a procedure similar to the one utilized in the population generation of GA. This is done to improve the quality of the solutions obtained during the search process. In this study, the number of particles is equal to the population in GA. In the initial population of the PSO, the solutions generated by the GA are copied directly to the PSO.

2) VELOCITY UPDATE

By using the velocity vector the position of each particle is updated towards the good solution. Local best (personal best) and global best (best solution among the population)

are used to find better solutions using Equation 8. In the first iteration, each particle is randomly assigned with a velocity pair based on the size of the problem considered, where the velocity pair represents the position of the tasks in the sequence or particles and it is referred to as transposition. In this study, velocity pairs are randomly generated based on the utilized concept [61].

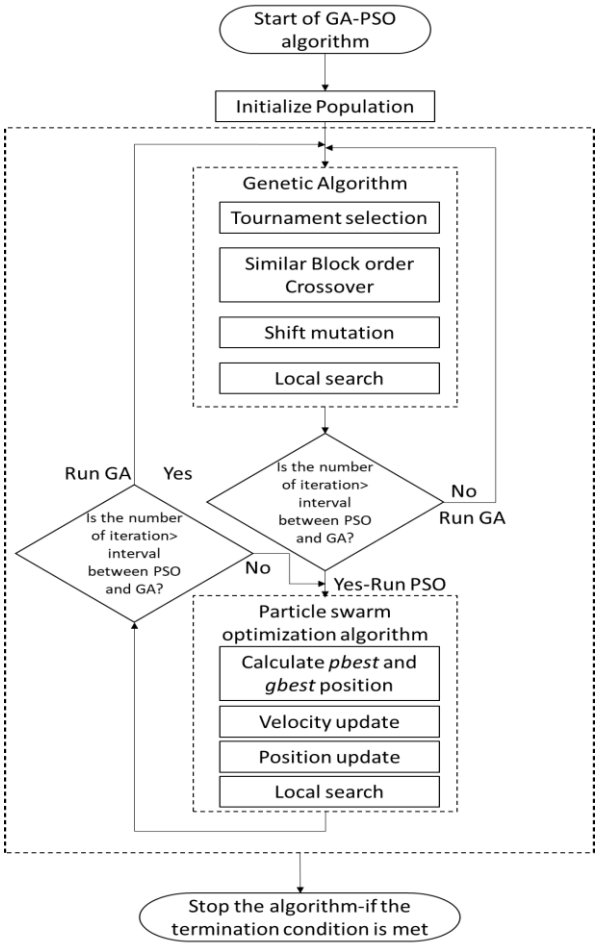

FIGURE 3. Flowchart of the proposed GA-PSO algorithm.

E. HYBRID GA-PSO ALGORITHM

In this section, the developed hybridized GA-PSO algorithm is described and a flowchart is presented in Figure 3. The balance between exploration and exploitation is further improved by combining GA with PSO algorithms. The idea of combining these two algorithms is to combine the social thinking ability of PSO with the global search ability of GA. The procedure starts by generating the initial population followed by the execution of genetic algorithm operations such as selection, crossover and mutation. A local search is also embedded to enhance the performance of the GA algorithm. The procedure is repeated for a certain set number of iterations, which is considered as one of the parameters during the experimentation process. The population is obtained after performing the GA operations after a set of iterations. The PSO algorithm starts with the generated population and randomly generates velocity for these populations, referred to as swarms. Using the velocity and position to update Equations 7 and 8, a new set of particles (swarm) is generated with a new velocity and a local search procedure is also embedded to improve the performance. This procedure keeps on repeating for a certain number of iterations and the best solution is obtained based on its fitness value in relation to the considered problem. The procedures for GA and PSO mentioned in Sections 4.3 and 4.4 are used for the proposed hybrid GA-PSO algorithm.

V. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, experimental results conducted for a multiple order PFSP with the proposed approaches are presented and analyzed. All the algorithms were implemented in C++ using a personal computer with an Intel core i7 processor (2.70 GHz) and 16 GB RAM, and Windows 7 OS. In the following section, a sensitivity analysis and obtained computational results are presented.

TABLE 1. Combinations of parameter values for experimental study.

Algorithm	Parameter	Factor level
GA	Tournament size	3, 4, 5
	Crossover rate	0.4, 0.5, 0.6
	Mutation rate	0.15, 0.20, 0.25
PSO	$c2$	1, 2, 3
	$c3$	1, 2, 3
Common	Interval between GA and PSO	20, 40, 60
	Restart interval	5, 10, 15
	Probability of local search	7%, 9%, 11%
	Population size	30, 40, 50

A. SENSITIVITY ANALYSIS

The performance of the proposed approaches is sensitive to their parameter settings. In this paper, parameters are calibrated based on Taguchi's method of experiment design [62] to achieve good solutions within reasonable computational times. Following the preliminary experiments, the parameters listed in Table 1 were considered for experimental study. Since there are 9 parameters and each of the parameters has 3 levels, we employ the orthogonal array $L_{27}(9^3)$. The total number of treatments was set to 27 and for each combination algorithms are executed 5 times independently and the stopping criterion for each run is set as a function of the number of machines, m, and the number of jobs per order, n, as: $m \times n \times$ 100 milliseconds. To test the parameters, all instances $\{20 \times 10, 50 \times 10, 100 \times 10, 200 \times 10\}$ from Taillard's benchmark [57] were chosen. The reason for performing this parameter testing in the benchmark for single order PFSP [57] is that the schedule generated by GA-PSO helps to reduce the makespan (or completion time) of an order. As a result, the shop floor may require less time to

TABLE 2. Response value and significance rank for PSO.

Level	Factor				
	$c2$	$c3$	Restart interval	Probability of local search	Population size
1	0.7425	0.7491	0.7553	0.7674	0.7546
2	0.7390	0.7564	0.7596	0.7422	0.7515
3	0.7779	0.7538	0.7445	0.7498	0.7534
Delta	0.0390	0.0073	0.0151	0.0251	0.0031
Rank	1	4	3	2	5

complete an order. If this increases, there is the possibility to accept more customer orders in real-time multiple-order PFSPs. For the PSO algorithm, 5 parameters are considered with 3 levels for each, and an L_{27} orthogonal array was used to tune the parameters of the PSO algorithm. Each treatment was run 5 times. To evaluate the performance of the GA-PSO and PSO algorithms in each run, an average relative performance of deviation (ARPD) is used and is expressed as follows:

$$ARPD = \frac{1}{5}\sum_{r=1}^{5}\left(\frac{f_i^r - f_{best}}{f_{best}} \times 100\right) \quad (9)$$

where f_i^r denotes the makespan obtained by an algorithm i in its r^{th} run and f_{best} indicates the best makespan value available in the literature. ARPD values for each treatment are omitted due to space constraints. However, ARPD values are available on request and will be uploaded to ResearchGate for the reader's benefit. Based on ARPD values, the response value and significance rank of each parameter for each level are presented in Tables 2 and 3. These tables also show delta values of each parameter, which is the difference between the highest and lowest average response values for each factor. For example, in Table 2, $c2$ has the delta value of 0.0390 (=0.7779 − 0.7390). It is important to note that a parameter is more significant that the other if it has higher delta value. For example, based on the delta value, from Table 3, it can be concluded that the tournament size parameter is the most significant parameter while the restart interval is the least significant parameter for the GA-PSO algorithm. For the PSO algorithm, $c2$ (learning coefficient for local best) is the most significant parameter and the number of particles is the least significant parameter. Finally, each parameter is set to the level which gives the minimum response value. For example, from Table 3, it is clear that the population size for GA-PSO algorithm is set to 40 (or level 2) since it has the minimum response values among all response values for the population size. Based on Taguchi analysis the following parameters for GA-PSO were selected for solving the considered problem: Tournament size = 5, Crossover rate = 0.6, Mutation rate = 0.15, $c2 = 3$, $c3 = 3$, Interval between GA and PSO = 60, Restart interval = 15, Probability of local search = 11%, and Population size (number of individuals or particles) = 40. For the PSO algorithm the following parameters were selected: $c2 = 2$, $c3 = 3$, Restart interval = 15, Probability of local search = 9%, and Number of particles = 40.

B. COMPUTATIONAL RESULTS AND STATISTICAL ANALYSIS

Based on the size of an order (i.e. number of jobs) an algorithm (PSO and GA-PSO) was allowed to run for same amount of time. In order to ensure fair comparison, the following requirements need to be met: (1) implementation in the same programming language, (2) same termination criteria, and (3) same computational power. Thus, all algorithms were coded in C++. To address points 2 and 3, the GA-PSO-based RT and PSO-based RT algorithms were allowed to run on converted CPU times, while considering the same CPU architecture presented in [31]. The CPU time was converted using Equation 10 [63]. It is to be noted that as mentioned in the introduction and literature review sections, real-time multiple is a complex scheduling problem with may need large CPU time to achieve good schedules. On the other hand, decision makers need to make a decision in real-time. To address this challenge, the CPU times are scaled based on the size of the problem, i.e. number of jobs per order and it is negligible with respect to time window between an order arrival and its delivery time. Recall that our proposed PSO based real-time strategy and GA-PSO based real-time strategy were implemented in Intel Core i7, 2.70 GHz processor. Since the state-of-the-art method reported in [31] were the implemented in Intel Core i7, 2.80 GHz processor, the given CPU time is the 2.80GHz. The converted CPU times for each order is available in Table 9 (Appendix).

$$\text{Converted CPU time (sec)} = \frac{2.7\ \text{GHz}}{\text{given CPU speed (GHz)}}\ x\ \text{given CPU time (sec)} \quad (10)$$

In the experimental study, scenarios I, II, and III from [31] were considered. To compare the proposed techniques against the state-of-the-art methods [31], 46 test instances reported in [31] (40 problem instances for scenario I, 4 instances for scenario II and 2 instances for scenario III) and the same upper bound (UB) (the worst case) and lower bound (LB) (the best case) for the accepted orders as derived in [31] were used. Simulation results for scenario I (identical orders arrive), where each order contains 50 jobs (as in problem instance *Ta*056), are presented in Table 4. From Table 4 it can been seen that a total of 10 orders arrived in the production shift at different points in production, and they all had customer-specified due dates. The first column shows the order number, the following columns show the arrival time and due date of each order, which were generated randomly [31]. The fourth column shows the completion time of each order i.e. time taken to complete all jobs in an order in all machines. The column headed 'tardiness' shows the tardiness or earliness of each order. If the tardiness value of an order is positive, then the order can be completed before it is accepted for processing. Otherwise, the order is tardy and is rejected. Considering both the completion time of each order and the number of accepted orders, the proposed PSO based RT strategy performs better than the GA-based RS strategy [31], as the PSO based strategy allows 9 orders to be accepted. However, it was outperformed by the GA based [31] RT. On the other hand,

TABLE 3. Response value and significance rank for GA-PSO.

Level	Factors								
	Tournament size	Crossover rate	Mutation rate	$c2$	$c3$	Interval between GA and PSO	Restart interval	Probability of local search	Population size
1	0.3967	0.3864	0.3797	0.3848	0.3905	0.3873	0.3810	0.3816	0.3841
2	0.3825	0.3818	0.3812	0.3842	0.3780	0.3823	0.3819	03821	0.3788
3	0.3628	0.3749	0.3823	0.3742	0.3747	0.3736	0.3803	0.3795	0.3803
Delta	0.0328	0.0115	0.0026	0.0106	0.0158	0.0137	0.0017	0.0026	0.0053
Rank	1	4	8	5	2	3	9	7	6

TABLE 4. Comparison between algorithms for Scenario I.

Order Number	Arrival Time of an order	Due Date, d	Right Shifting (RS) strategy [31]			Real-Time multiple-order (RT) strategy [31]			Real-Time multiple-order (RT) strategy (PSO based)			Real-Time multiple-order (RT) strategy (GA-PSO based)		
			Completion Time, $C_{comp}(RS)$	Tardiness= $C_{comp}(RS)$-d	Status	Completion Time, $C(RT)$	Tardiness = $C_{comp}(RT)$-d	Status	Completion Time, $C(RT)$	Tardiness = $C_{comp}(RT)$-d	Status	Completion Time, $C(RT)$	Tardiness = $C_{comp}(RT)$-d	Status
1	0	6297	**3006**	-3291	A	**3006**	-3291	A	3048	-3249	A	**3006**	-3291	A
2	360	5984	5740	-244	A	5718	-266	A	5766	-218	A	**5716**	-266	A
3	4846	8573	8436	-137	A	8421	-152	A	8446	-127	A	**8410**	-161	A
4	8638	11983	**11644**	-339	A	**11644**	-339	A	11678	-305	A	**11644**	-339	A
5	13882	19126	**16888**	-2238	A	**16888**	-2238	A	16936	-2190	A	**16888**	-2238	A
6	15991	19606	19624	18	R	**19594**	-12	A	19638	32	R	**19594**	-12	A
7	19112	22294	22307	13	R	22288	-6	A	**22148**	-146	A	22290	-4	A
8	20764	24987	24997	10	R	24982	-5	A	**24881**	-106	A	24978	-9	A
9	23051	30116	27680	-2436	A	27662	-2454	A	**27575**	-2541	A	27662	-2454	A
10	25101	30468	33093	2625	R	30356	-112	A	**30269**	-199	A	30347	-121	A

like the RT strategy [31], the proposed GA-PSO based RT strategy allowed all 10 orders that arrived on the production floor to be accepted, and enhanced the completion time of 9 orders. Computational results for scenario II and scenario III are presented in Table 10 and Table 11 (Appendix).

In order to evaluate the overall performance of PSO-based RT and GA-PSO-based RT against the state-of-the-art method reported in [31], the average deviations from the UB and LB were calculated for each method. The effectiveness of each method (RS or RT) can be represented by Equations 11 and 12:

Average deviation from the LB,

$$D_L = [\sum_{i=1}^{Ns} (\frac{ri - LBi}{LBi})]/n_S \tag{11}$$

Average deviation from the UB,

$$D_U = [\sum_{i=1}^{Ns} (\frac{UBi - ri}{ri})]/n_S \tag{12}$$

where ri is the makespan of the i^{th} accepted order achieved by a particular technique. As the number of accepted orders may vary for both strategies, n_S is the total number of orders accepted by a technique.

A summary of the comparison between RS, RT obtained in [31] and the proposed PSO and GA-PSO based RT is presented in Table 5, where the same type of order arrives into the production system randomly. The first column in the Table shows the number of jobs for each order. The following column represents the average improvement from the worst-case scenario (i.e. deviation from the UB) for the 10 different problem instances [31] after applying each technique. The greater the value of D_U, the better the solution. The next column, D_L, presents the deviation from the LB, i.e. whether the orders overlapped and how effectively they could be scheduled into the available machines. From the results of D_U and D_L, it can be seen that PSO-based RT performs better than RS [31] only. However, from the results

TABLE 5. Same order.

Problem Size	D_U				D_L				Number of Accepted Orders			
	RS [31]	RT [31]	RT (PSO)	RT (GA-PSO)	RS [31]	RT [31]	RT (PSO)	RT (GA-PSO)	RS [31]	RT [31]	RT (PSO)	RT (GA-PSO)
20jobs	1.566026	2.045218	1.764676	**2.048284**	0.796381	0.501011	0.597619	**0.492937**	64	70	65	**72**
50jobs	0.3713685	0.4072689	0.40657	**0.471369**	0.24099229	0.218244499	0.230428	**0.211325**	73	**80**	73	**80**
100jobs	0.16546857	**0.18658584**	0.184625	0.18353	0.17880491	0.17248293	0.174575	**0.16298**	59	**62**	60	**62**
200jobs	0.11492327	0.11501577	0.115015	**0.195477**	0.06381179	0.062574632	0.06357	**0.062298**	59	65	59	**67**

TABLE 6. Mixed order with identical order sizes.

Problem Size	D_U				D_L				Number of Accepted Orders			
	RS [31]	RT [31]	RT (PSO)	RT (GA-PSO)	RS [31]	RT [31]	RT (PSO)	RT (GA-PSO)	RS [31]	RT [31]	RT (PSO)	RT (GA-PSO)
20jobs	0.145812	0.185154	0.150975	**0.187866**	0.054259	0.038151	0.053257	**0.034233**	14	17	14	**17**
50jobs	0.070463	0.0774516	0.076932	**0.132584**	**0.0435364**	0.0444811	0.435001	0.044791	10	**12**	11	11
100jobs	0.0204339	0.0215277	0.021005	**0.044983**	0.00742706	0.00630862	0.007271	**0.002612**	**7**	**7**	6	**7**
200jobs	0.0274414	0.0310825	0.029795	**0.115003**	**0.0049502**	0.00606717	0.005183	0.005375	13	12	12	**14**

TABLE 7. Mixed order with different order sizes.

Case	D_U				D_L				Number of Accepted Orders			
	RS [31]	RT [31]	RT (PSO)	RT (GA-PSO)	RS [31]	RT [31]	RT (PSO)	RT (GA-PSO)	RS[31]	RT[31]	RT (PSO)	RT (GA-PSO)
1	0.292583	0.144235	0.28101	**0.30481**	0.0285443	0.0231372	0.025504	**0.017071**	19	21	19	**22**
2	0.335286	0.335857	0.336014	**0.34125**	0.00712116	0.00665912	0.0067514	**0.006315**	**25**	**25**	**25**	**25**

TABLE 8. Comparisons between RS and RTs for practical production problems.

Order Number	Arrival Time of an order	Due Date, d	Right Shifting (RS) strategy [31]				Real-Time multiple-order (RT) strategy [31]				Real-Time multiple-order (RT) strategy (PSO based)				Real-Time multiple-order (RT) strategy (GA-PSO based)			
			Completion Time, $C_{comp}(RS)$	Tardiness= $C_{comp}(RS)-d$	Profit, $	Status	Completion Time, $C(RT)$	Tardiness = $C_{comp}(RT)-d$	Profit, $	Status	Completion Time, $C(RT)$	Tardiness = $C_{comp}(RT)-d$	Profit, $	Status	Completion Time, $C(RT)$	Tardiness = $C_{comp}(RT)-d$	Profit, $	Status
1	0	5189	**2889**	-2300	4915	A	**2889**	-2300	4915	A	2923	-2266	4845.64	A	**2889**	-2300	4915	A
2	141	5933	5492	-441	1157.74	A	5434	-499	1281.86	A	5482	-451	1179.14	A	**5426**	-507	1298.98	A
3	248	5874	8069	2195	-27	R	**7990**	2116	-27	R	8028	2154	-27	R	**7990**	2116	-27	R
4	755	6962	8044	1082	-88	R	**7985**	1023	-88	R	8028	1066	-88	R	8031	1069	-88	R
5	4498	8000	8038	38	-83	R	**7985**	-15	258.8	A	8028	28	-83	R	**7985**	-15	258.8	A
6	6638	7559	**9516**	1957	-106	R	10537	2978	-106	R	9554	1995	-106	R	9554	1995	-106	R
7	8790	10188	11686	1498	-22	R	11664	1476	-22	R	11711	1523	-22	R	**11659**	1471	-22	R
8	8798	13019	11688	-1331	1790.58	A	11688	-1331	1790.58	A	11716	-1303	1757.54	A	**11681**	-1338	1798.84	A
9	10414	12517	14297	1780	-163	R	**14245**	1728	-163	R	14288	1771	-171	R	14280	1763	-171	R
10	12500	18216	15390	-2826	5169.5	A	15377	-2839	5193.25	A	15415	-2801	5125.75	A	**15375**	-2841	5195.75	A
Total profit per production shift, $			12543.82				13033.39				12411.07				**13053.37**			

*A- Accepted, R-Reject

it is clear that the GA-PSO-based RT approach performs better than the others (smaller values for D_L and bigger values for D_U). The last four columns present the total number of orders accepted by the RS and RT techniques. For each of the problem instances, 10 orders were considered, where orders have different inter-arrival times and due dates. The GA-PSO-based RT strategy succeeded in accepting the most orders of any strategy. A comparison between the RS and RT techniques for mixed orders with the same-sized orders (scenario II) and different sized orders (scenario III) is presented in Tables 6 and 7 respectively. For scenarios II and III, 25 orders with different due dates arrived

TABLE 9. Time requirements.

Problem Size (number of jobs per order)	Average Computational time (second) in 2.80GHz processor	Average Computational time (second) in 2.70GHz processor allowed for GA-PSO and PSO based Real-time strategy
20	12	11.57
50	40	38.57
100	70	67.50
200	190	183.21

TABLE 10. Comparison between algorithms in Scenario II.

Order Number	Order Type	Arrival Time of an order	Due Date, d	Right Shifting (RS) strategy [31]			Real-time (RT) strategy [31]			Real-time (RT) strategy (PSO based)			Real-time (RT) strategy (GA-PSO based)		
				Completion Time, $C_{comp}(RS)$	Tardiness = $C_{comp}(RS)$-d	Status	Completion Time, C_{comp} (RT)	Tardiness = $C_{comp}(RT)$-d	Status	Completion Time, C_{comp} (RT)	Tardiness = $C_{comp}(RT)$-d	Status	Completion Time, C_{comp} (RT)	Tardiness = $C_{comp}(RT)$-d	Status
1	*Ta16*	0	1591	**1397**	-194	A	**1397**	-194	A	**1397**	-194	A	**1397**	-194	A
2	*Ta16*	7	3846	2501	-1345	A	**2496**	-1350	A	**2496**	-1350	A	**2496**	-1350	A
3	*Ta12*	1234	5626	3837	-1789	A	3750	-1876	A	**3749**	-1877	A	**3749**	-1877	A
4	*Ta11*	2920	4878	5049	171	R	**4898**	20	R	4913	35	R	4913	35	R
5	*Ta19*	3185	5934	5965	31	R	5892	-42	**A**	**4988**	-946	A	**4988**	-946	A
6	*Ta17*	5227	7365	7256	-109	A	7161	-204	A	**6711**	-654	A	**6711**	-654	A
7	*Ta13*	6407	7907	8467	560	R	8360	453	R	**7903**	-4	A	**7903**	-4	**A**
8	*Ta18*	8101	10752	9597	-1155	A	9597	-1155	A	9639	-1113	A	**9539**	-1213	A
9	*Ta18*	9382	11213	**10920**	-293	A	**10920**	-293	A	10930	-290	A	10926	-287	A
10	*Ta16*	10870	12641	**12267**	-374	A	**12267**	-374	A	12278	-385	A	**12267**	-374	A
11	*Ta18*	11872	13827	13489	-338	A	**13461**	-366	A	13471	-356	A	**13461**	-366	A
12	*Ta15*	12301	15487	14725	-762	A	**14630**	-857	A	**14630**	-857	A	**14630**	-857	A
13	*Ta15*	14462	15929	**15881**	-48	A	**15881**	-48	A	**15881**	-48	A	**15881**	-48	A
14	*Ta15*	14584	17565	17378	-187	A	17158	-407	A	**17027**	-538	A	**17027**	-538	A
15	*Ta14*	15310	18010	18285	275	R	18142	132	R	18027	17	R	**18011**	1	R
16	*Ta11*	17118	19766	19270	-496	A	19068	-698	A	**18700**	-1066	A	**18700**	-1066	A
17	*Ta12*	18598	21330	20483	-847	A	20265	-1065	A	**20257**	-1073	A	**20257**	-1073	A
18	*Ta19*	19006	22789	22435	-354	A	22950	161	R	21499	-1290	A	**21497**	-1292	A
19	*Ta15*	20929	22934	23505	571	R	**22441**	-493	A	22628	-306	A	22623	-311	A
20	*Ta19*	22284	24599	24963	364	R	**23877**	-722	A	23890	-709	A	23884	-715	**A**
21	*Ta15*	22877	25350	26137	787	R	25209	-141	A	25019	-331	A	**25013**	-337	A
22	*Ta11*	24099	25789	27065	1276	R	26359	570	R	26169	380	R	**26163**	374	R
23	*Ta11*	24736	27276	28349	1073	R	27689	413	R	**26318**	-958	A	**26318**	-958	A
24	*Ta16*	25962	27757	29163	1406	R	28471	714	R	27390	-370	A	**27387**	-370	A
25	*Ta17*	27044	29363	30436	1073	R	29758	395	R	28600	-763	A	**28599**	-764	**A**

randomly during a production shift. Table 7 compares the proposed and existing methods with regard to two independent problem instances, namely case 1 and case 2. From Tables 6 and 7, it can be seen that the proposed GA-PSO-based RT strategy outperforms the other methods.

C. REAL-WORLD CASE STUDY

This section presents a comparison of the proposed approaches based on practical data available from a sanitary ware-manufacturing industry located in Bangladesh. Relevant production data on inter-arrival times, due dates and costs is available in [31] and the computational results are presented in Table 8. The company can earn an income by accepting and completing an order on or before its due date. The company can also earn bonuses by producing the order before its due date, i.e. an earliness bonus. However, if the order is rejected then a fixed opportunity cost is imposed. From the production scenario, it can be seen that 10 orders entered the production system within a short span of time.

VI. CONCLUSION

Real-time order acceptance and scheduling problems are an important problem in a modern manufacturing system scenario. This study aims to develop new meta-heuristic-based strategies to optimize the number of orders accepted that arrive in real-time in flow shop environments. Hence, this paper proposes a hybrid GA-PSO algorithm-based RT strategy for solving real-time multiple order PFSPs. The parameters of each algorithm are calibrated using Taguchi methods. The proposed approaches were compared with

TABLE 11. Comparison between algorithms in Scenario III.

Order Number	Order Type	Arrival Time of an order	Due Date, d	Right Shifting (RS) strategy [31]			Real-time (RT) strategy [31]			Real-time (RT) strategy (PSO based)			Real-time (RT) strategy (GA-PSO based)		
				Completion Time, $C_{comp}(RS)$	Tardiness = $C_{comp}(RS)$-d	Status	Completion Time, C_{comp} (RT)	Tardiness = C_{comp} (RT)-d	Status	Completion Time, C_{comp} (RT)	Tardiness = C_{comp} (RT)-d	Status	Completion Time, C_{comp} (RT)	Tardiness = C_{comp} (RT)-d	Status
1	*Ta47*	0	4027	3115	-912	A	3115	-912	A	3124	-903	A	**3107**	-920	A
2	*Ta14*	2254	3887	3643	-244	A	**3638**	-249	A	4165	278	R	3643	-244	A
3	*Ta16*	4027	6787	**5424**	-1363	A	**5424**	-1363	A	**5424**	-1363	A	**5424**	-1363	A
4	*Ta49*	4679	9732	9746	14	R	9670	-62	A	8048	-1684	**A**	**8032**	-1700	A
5	*Ta17*	14282	16644	**15766**	-878	A	**15766**	-878	A	**15766**	-878	**A**	**15766**	-878	**A**
6	*Ta92*	14427	25409	24743	-666	A	**24739**	-670	A	25949	540	R	**24739**	-670	A
7	*Ta18*	19287	23441	22574	-867	A	22499	-942	A	**20825**	-2616	A	**20825**	-2616	A
8	*Ta71*	22461	31839	31246	-593	A	31246	-593	A	28252	-3587	A	**28232**	-3607	A
9	*Ta78*	26769	38278	36759	-1519	A	36614	-1664	A	33615	-4663	A	**33598**	-4680	A
10	*Ta12*	31753	34762	33679	-1083	A	**33637**	-1125	A	34878	116	R	**33637**	-1125	A
11	*Ta47*	39733	44745	**42848**	-1897	A	**42848**	-1897	A	42871	-1874	A	**42848**	-1897	A
12	*Ta94*	42341	60741	60751	10	R	60726	-15	**A**	53520	-7221	A	**53502**	-7239	A
13	*Ta47*	63319	67976	**66434**	-1542	A	**66434**	-1542	A	66452	-1524	A	**66434**	-1542	A
14	*Ta17*	63765	67958	65853	-2105	A	**65831**	-2127	A	67653	-305	A	67640	-318	A
15	*Ta49*	71866	75361	**74768**	-593	A	**74768**	-593	A	74798	-563	A	**74768**	-593	A
16	*Ta74*	75647	84354	83258	-1096	A	83236	-1118	A	81498	-2856	A	**81473**	-2881	A
17	*Ta44*	78321	82979	83491	512	R	**83419**	440	R	84291	1312	R	84308	1329	R
18	*Ta41*	79959	85336	86593	1257	R	**85478**	142	R	85770	434	R	**85478**	142	R
19	*Ta95*	86101	103716	108689	4973	R	105001	1285	R	96646	-7070	A	**96638**	-7078	**A**
20	*Ta73*	93996	104935	104115	-820	A	104087	-848	A	102124	-2811	A	**102112**	-2823	A
21	*Ta72*	107889	114760	**113251**	-1509	A	**113251**	-1509	A	113253	-1507	A	**113251**	-1509	A
22	*Ta18*	116783	118809	**118321**	-488	A	**118321**	-488	A	118327	-482	A	**118321**	-488	A
23	*Ta71*	121105	128063	127844	-219	A	127843	-220	A	**126905**	-1158	A	127843	-220	A
24	*Ta99*	122912	143164	143382	218	R	**143365**	201	R	143373	201	R	**143365**	201	R
25	*Ta47*	138444	142737	142099	-638	A	141537	-1200	A	141588	-1149	A	**141530**	-1200	A

*Status: A-accepted order, R-rejected order

current state-of-the-art approaches. The computational results based on realistic problem instances show that the hybridization of GA with PSO is advantageous, since GA-PSO-based RT shows superiority in terms of solution quality and computational time. Two key advantages of this strategy over the existing methodologies are the better customer satisfaction gained by accepting a greater number of orders and better production capacity utilization thanks to an effective approach that is suitable for practical decision-making on a real-time basis. This study opens up several future research avenues for studying and investigating different hybrid algorithms for solving different realistic make-to-order production problems. Other realistic settings could consider process interruptions, energy consumption, sequence-dependent setups, buffer size constraints, and shortages in material supplies. The proposed algorithms can be used to solve these problems in the future.

APPENDIX

See Tables 9–11.

REFERENCES

[1] H. F. Rahman, R. Sarker, and D. Essam, "A genetic algorithm for permutation flow shop scheduling under make to stock production system," *Comput. Ind. Eng.*, vol. 90, pp. 12–24, Dec. 2015.

[2] H. F. Rahman, R. Sarker, and D. Essam, "A genetic algorithm for permutation flowshop scheduling under practical make-to-order production system," *AI EDAM*, vol. 31, no. 1, pp. 87–103, Feb. 2017.

[3] S. M. Johnson, "Optimal two-and three-stage production schedules with setup times included," *Naval Res. Logistics*, vol. 1, no. 1, pp. 61–68, Mar. 1954.

[4] E. Ignall and L. Schrage, "Application of the branch and bound technique to some flow-shop scheduling problems," *Oper. Res.*, vol. 13, no. 3, pp. 400–412, Jun. 1965.

[5] W. J. Selen and D. D. Hott, "A mixed-integer goal-programming formulation of the standard flow-shop scheduling problem," *J. Oper. Res. Soc.*, vol. 37, no. 12, pp. 1121–1128, Dec. 1986.

[6] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Math. Oper. Res.*, vol. 1, no. 2, pp. 117–129, May 1976.

[7] M. Nawaz, E. E. Enscore, and I. Ham, "A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, 1983. doi: 10.1016/0305-0483(83)90088-9.

[8] J. Grabowski and M. Wodecki, "A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion," *Comput. Oper. Res.*, vol. 31, no. 11, pp. 1891–1909, Sep. 2004. doi: 10.1016/S0305-0548(03)00145-X.

[9] M. F. Tasgetiren, Y. C. Liang, M. Sevkli, and G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem," *Eur. J. Oper. Res.*, vol. 177, no. 3, pp. 1930–1947, 2007.

[10] I. H. Osman and C. N. Potts, "Simulated annealing for permutation flow-shop scheduling," *Omega*, vol. 17, no. 6, pp. 551–557, 1989. doi: 10.1016/0305-0483(89)90059-5.

[11] G. I. Zobolas, C. D. Tarantilis, and G. Ioannou, "Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm," *(in English), Comput Oper Res*, vol. 36, no. 4, pp. 1249–1267, Apr. 2009. doi: 10.1016/j.cor.2008.01.007.
[12] C. Rajendran and H. Ziegler, "Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs," *Eur. J. Oper. Res.*, vol. 155, no. 2, pp. 426–438, Jun. 2004. doi: 10.1016/S0377-2217(02)00908-6.
[13] R. Ruiz and C. Maroto, "A comprehensive review and evaluation of permutation flowshop heuristics," *Eur. J. Oper. Res.*, vol. 165, no. 2, pp. 479–494, Sep. 2005.
[14] R. Ruiz, C. Maroto, and J. Alcaraz, "Two new robust genetic algorithms for the flowshop scheduling problem," *Omega*, vol. 34, no. 5, pp. 461–476, 2006.
[15] H. F. Rahman, R. A. Sarker, and D. L. Essam, "A memetic algorithm for permutation flow shop problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2013, pp. 1618–1625.
[16] P. Dasgupta and S. Das, "A discrete inter-species cuckoo search for flowshop scheduling problems," *Comput Oper Res*, vol. 60, pp. 111–120, Aug. 2015. doi: 10.1016/j.cor.2015.01.005.
[17] M. Abdel-Basset, G. Manogaran, D. El-Shahat, and S. Mirjalili, "A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem," *Future Gener. Comput. Syst.*, vol. 85, pp. 129–145, Aug. 2018. doi: 10.1016/j.future.2018.03.020.
[18] S. A. Slotnick, "Order acceptance and scheduling: A taxonomy and review," *Eur. J. Oper. Res.*, vol. 212, no. 1, pp. 1–11, Jul. 2011.
[19] S. A. Slotnick and T. E. Morton, "Selecting jobs for a heavily loaded shop with lateness penalties," *Comput Oper Res*, vol. 23, no. 2, pp. 131–140, Feb. 1996. doi: 10.1016/0305-0548(95)00015-E.
[20] H. F. Lewis and S. A. Slotnick, "Multi-period job selection: Planning work loads to maximize profit," *Comput. Oper. Res.*, vol. 29, no. 8, pp. 1081–1098, Jul. 2002. doi: 10.1016/s0305-0548(00)00105-2.
[21] S. A. Slotnick and T. E. Morton, "Order acceptance with weighted tardiness," *Comput. Oper. Res.*, vol. 34, no. 10, pp. 3029–3042, Oct. 2007. doi: 10.1016/j.cor.2005.11.012.
[22] W. O. Rom and S. A. Slotnick, "Order acceptance using genetic algorithms," *Comput. Oper. Res.*, vol. 36, no. 6, pp. 1758–1767, 2009.
[23] Y.-Y. Xiao, R.-Q. Zhang, Q.-H. Zhao, and I. Kaku, "Permutation flow shop scheduling with order acceptance and weighted tardiness," *Appl. Math. Comput.*, vol. 218, no. 15, pp. 7911–7926, Apr. 2012.
[24] S.-W. Lin and K.-C. Ying, "Order acceptance and scheduling to maximize total net revenue in permutation flowshops with weighted tardiness," *Appl. Soft Comput.*, vol. 30, pp. 462–474, May 2015.
[25] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 4th ed. New York, NY, USA: Springer, 2012.
[26] G. Schmidt, "Scheduling with limited machine availability," *Eur. J. Oper. Res.*, vol. 121, pp. 1–15, Feb. 2000.
[27] F. Qian, W. Zhong, and W. Du, "Fundamental theories and key technologies for smart and optimal manufacturing in the process industry," *Engineering*, vol. 3, no. 2, pp. 154–160, Apr. 2017.
[28] R. C. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *Proc. Int. Conf. Evol. Program.*, 1998, pp. 611–616.
[29] H. Garg, "A hybrid PSO-GA algorithm for constrained optimization problems," *Appl. Math. Comput.*, vol. 274, pp. 292–305, Feb. 2016.
[30] A. Gálvez and A. Iglesias, "A new iterative mutually coupled hybrid GA–PSO approach for curve fitting in manufacturing," *Appl. Soft Comput.*, vol. 13, no. 3, pp. 1491–1504, Mar. 2013.
[31] H. F. Rahman, R. Sarker, and D. Essam, "A real-time order acceptance and scheduling approach for permutation flow shop problems," *Eur. J. Oper. Res.*, vol. 247, no. 2, pp. 488–503, Dec. 2015. doi: 10.1016/j.ejor.2015.06.018.
[32] F. Wester, J. Wijngaard, and W. R. M. Zijm, "Order acceptance strategies in a production-to-order environment with setup times and due-dates," *Int. J. Prod. Res.*, vol. 30, no. 6, pp. 1313–1326, Jun. 1992.
[33] I. Duenyas and W. J. Hopp, "Quoting customer lead times," *Manage. Sci.*, vol. 41, no. 1, pp. 43–57, Jan. 1995. doi: 10.1287/Mnsc.41.1.43.
[34] I. Duenyas, "Single facility due date setting with multiple customer classes," *Manage. Sci.*, vol. 41, no. 4, pp. 608–619, Apr. 1995. doi: 10.1287/mnsc.41.4.608.
[35] A. Nandi and P. Rogers, "Using simulation to make order acceptance/rejection decisions," *Simulation*, vol. 80, no. 3, pp. 131–142, Mar. 2004.
[36] P. Rogers and A. Nandi, "Judicious order acceptance and order release in make-to-order manufacturing systems," *Prod Plan Control*, vol. 18, no. 7, pp. 610–625, Oct. 2007. doi: 10.1080/09537280701582422.
[37] M. R. Moreira and R. A. Alves, "A methodology for planning and controlling workload in a job-shop: a four-way decision-making problem," *Int. J. Prod. Res.*, vol. 47, no. 10, pp. 2805–2821, May 2009. doi: 10.1080/00207540701725083.
[38] L. Tang, W. Liu, and J. Liu, "A neural network model and algorithm for the hybrid flow shop scheduling problem in a dynamic environment," *J. Intell. Manuf.*, vol. 16, no. 3, pp. 361–370, Jun. 2005.
[39] P. S. Eriksen and P. Nielsen, "Order quantity distributions: Estimating an adequate aggregation horizon," *Manage. Prod. Eng. Rev.*, vol. 7, no. 3, pp. 39–48, Sep. 2016.
[40] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*. London, U.K.: Luniver Press, 2010.
[41] S. K. Hasan, R. Sarker, and D. Essam, "Genetic algorithm for job-shop scheduling with machine unavailability and breakdowns," *Int. J. Prod. Res.*, vol. 49, no. 16, pp. 4999–5015, Aug. 2011.
[42] J. Liu and L. Tang, "A modified genetic algorithm for single machine scheduling," *Comput. Ind. Eng.*, vol. 37, nos. 1–2, pp. 43–46, Oct. 1999.
[43] J. M. Nilakantan and S. Ponnambalam, "Robotic U-shaped assembly line balancing using particle swarm optimization," *Eng. Optim.*, vol. 48, no. 2, pp. 231–252, Feb. 2016.
[44] D. Debels and M. Vanhoucke, "A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem," *Oper. Res.*, vol. 55, no. 3, pp. 457–469, 2007.
[45] D. Ouelhadj and S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *J. Scheduling*, vol. 12, no. 4, p. 417, 2009.
[46] A. Rossi and G. Dini, "Dynamic scheduling of FMS using a real-time genetic algorithm," *Int. J. Prod. Res.*, vol. 38, no. 1, pp. 1–20, Jan. 2000.
[47] M. Souier, Z. Sari, and A. Hassam, "Real-time rescheduling metaheuristic algorithms applied to FMS with routing flexibility," *Int. J. Adv. Manuf. Technol.*, vol. 64, nos. 1–4, pp. 145–164, 2013.
[48] M. Souier, M. Dahane, and F. Maliki, "An NSGA-II-based multiobjective approach for real-time routing selection in a flexible manufacturing system under uncertainty and reliability constraints," *Int. J. Adv. Manuf. Technol.*, vol. 100, nos. 9–12, pp. 2813–2829, Feb. 2019.
[49] L. Tang, Y. Zhao, and J. Liu, "An improved differential evolution algorithm for practical dynamic scheduling in steelmaking-continuous casting production," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 209–225, Apr. 2014.
[50] C. Lu, L. Gao, X. Li, and S. Xiao, "A hybrid multi-objective grey wolf optimizer for dynamic scheduling in a real-world welding industry," *Eng. Appl. Artif. Intell.*, vol. 57, pp. 61–79, Jan. 2017.
[51] M. Sama, P. Pellegrini, A. D'Ariano, J. Rodriguez, and D. Pacciarelli, "Ant colony optimization for the real-time train routing selection problem," *Transp. Res. B, Methodol.*, vol. 85, pp. 89–108, Mar. 2016.
[52] A. F. Ali and M. A. Tawhid, "A hybrid particle swarm optimization and genetic algorithm with population partitioning for large scale optimization problems," *Ain Shams Eng. J.*, vol. 8, no. 2, pp. 191–206, Jun. 2017.
[53] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Mach. Learn.*, vol. 3, nos. 2–3, pp. 95–99, 1988.
[54] J. H. Holland, "Adaptation in natural and artificial systems," in *An Introductory Analysis with APPLICATION to Biology, Control, and Artificial Intelligence*. Ann Arbor, MI, USA: Univ. Michigan Press. 1975.
[55] M. Sevaux and K. Sörensen, "Permutation distance measures for memetic algorithms with population management," in *Proc. 6th Metaheuristics Int. Conf.*, 2005, pp. 1–8.
[56] T. Murata, H. Ishibuchi, and H. Tanaka, "Genetic algorithms for flowshop scheduling problems," *Comput. Ind. Eng.*, vol. 30, no. 4, pp. 1061–1071, Sep. 1996. doi: 10.1016/0360-8352(96)00053-8.
[57] E. Taillard, "Some efficient heuristic methods for the flow-shop sequencing problem," *Eur. J. Oper. Res.*, vol. 47, no. 1, pp. 65–74, Jul. 1990. doi: 10.1016/0377-2217(90)90090-X.
[58] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Micro Mach. Human Sci.*, Apr. 1995, pp. 39–43.
[59] Y. Shi, "Particle swarm optimization: developments, applications and resources," in *Proc. Congr. Evol. Comput.*, Sep. 2001, pp. 81–86.
[60] G. Onwubolu and M. Clerc, "Optimal path for automated drilling operations by a new heuristic approach using particle swarm optimization," *Int. J. Prod. Res.*, vol. 42, no. 3, pp. 473–491, Feb. 2004.
[61] J. M. Nilakantan, G. Q. Huang, and S. Ponnambalam, "An investigation on minimizing cycle time and total energy consumption in robotic assembly line systems," *J. Cleaner Prod.*, vol. 90, pp. 311–325, Jun. 2015.

[62] V. N. Nair, B. Abraham, J. MacKay, G. Box, R. N. Kacker, T. J. Lorenzen, J. M. Lucas, R. H. Myers, G. G. Vining, J. A. Nelder, and M. S. Phadke, "Taguchi's parameter design: A panel discussion," *Technometrics*, vol. 34, no. 2, pp. 127–161, 1992.
[63] X. Yuan, L. Wang, Y. Yuan, Y. Zhang, B. Cao, and B. Yang, "A modified differential evolution approach for dynamic economic dispatch with valve-point effects," *Energy Convers. Manage.*, vol. 49, no. 12, pp. 3447–3453, 2008.

HUMYUN FUAD RAHMAN received the Ph.D. degree in computer science from the School of Engineering and IT, University of New South Wales, Canberra, Australia, where he is currently a Research Associate. His research lies in the area of manufacturing system optimization considering uncertainty and disruptions, Industry 4.0, project and supply chain management. His research interests include evolutionary computation, manufacturing system optimization, and supply chain management.

MUKUND NILAKANTAN JANARDHANAN received the Ph.D. degree in manufacturing Engineering from Monash University, in 2015. From 2015 to 2018, he was a Postdoctoral Fellow with Aalborg University, Denmark. Since March 2018, he has been a Lecturer in engineering management with the University of Leicester, Leicester, U.K. He has published over 30 articles in reputed journals and conferences. His research interests include manufacturing, production planning, and control.

IZABELA EWA NIELSEN received the master's degree in engineering from the Opole University of Technology and the Ph.D. degree (Hons.) from the Faculty of Production Engineering, Warsaw University of Technology, in 2005, with the focus on application of constraint logic programming techniques in production flow planning. She is currently a Professor with the Department of Materials and Production, Aalborg University, Denmark. She has published over 140 articles in journals, books, and conferences. Her research interests include planning, scheduling, and optimization problems. She has a special emphasis on automated manufacturing, transportation, and production systems. She is an Associate Editor for two international journals and serving as an Editorial Board Member for several reputed journals.