IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 17, 2021, accepted June 3, 2021, date of publication June 29, 2021, date of current version July 9, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3093336

An Improved Evolution Strategy Hybridization With Simulated Annealing for Permutation Flow Shop Scheduling Problems

BILAL KHURSHID[1], SHAHID MAQSOOD[2], MUHAMMAD OMAIR[2], BISWAJIT SARKAR[3], IMRAN AHMAD[1], AND KHAN MUHAMMAD[4]

[1]Department of Industrial Engineering, University of Engineering and Technology, Peshawar 25000, Pakistan
[2]Department of Industrial Engineering, University of Engineering and Technology at Jalozai Campus, Peshawar 25000, Pakistan
[3]Department of Industrial Engineering, Yonsei University, Seoul 03722, South Korea
[4]Department of Mining Engineering, University of Engineering and Technology, Peshawar 25000, Pakistan

Corresponding author: Biswajit Sarkar (bsbiswajitsarkar@gmail.com)

ABSTRACT Flow Shop Scheduling Problem (FSSP) has significant application in the industry, and therefore it has been extensively addressed in the literature using different optimization techniques. Current research investigates Permutation Flow Shop Scheduling Problem (PFSSP) to minimize makespan using the Hybrid Evolution Strategy (HES_{SA}). Initially, a global search of the solution space is performed using an Improved Evolution Strategy (I.E.S.), then the solution is improved by utilizing local search abilities of Simulated Annealing (S.A.). I.E.S. thoroughly exploits the solution space using the reproduction operator, in which four offsprings are generated from one parent. A double swap mutation is used to guide the search to more promising areas in less computational time. The mutation rate is also varied for the fine-tuning of results. The best solution of the I.E.S. acts as a seed for S.A., which further improved the results by exploring better neighborhood solutions. In S.A., insertion mutation is used, and the cooling parameter and acceptance-rejection criteria induce randomness in the algorithm. The proposed HES_{SA} algorithm is tested on well-known NP-hard benchmark problems of Taillard (120 instances), and the performance of the proposed algorithm is compared with the famous techniques available in the literature. Experimental results indicate that the proposed HES_{SA} algorithm finds fifty-four upper bounds for Taillard instances, while thirty-eight results are further improved for the Taillard instances.

INDEX TERMS Permutation flow shop scheduling problems, improved evolution strategy, simulated annealing, Taillard problems, makespan.

I. INTRODUCTION

In a flow shop production environment, machines are arranged in series, and the product is moved from one machine to the next machine in a fixed sequence [1]. In FSSP, when the processing sequence for all the machines is the same, it is termed Permutation Flow Shop Scheduling (PFSSP). It has a wide range of applications in the industries, i.e., automobile, pharmaceutical, fertilizer, and food industry, and several researchers in literature have addressed it. The FSSP was first proposed by Johnson to minimize makespan. Since then, makespan is considered as most used objective in the literature to optimize PFSSP (Pinedo [1]).

The associate editor coordinating the review of this manuscript and approving it for publication was Noel Crespi.

Makespan is the total time required to complete all the jobs on all the machines [2]. For the current world's dynamic environment, the makespan criterion is considered the most relevant for PFSSP [3]. PFSSP is regarded as a complex problem (Yenisey and Yagmahan [4]), and it is NP-hard (Garey, Johnson [6]).

II. LITERATURE REVIEW

PFSSP is addressed in the literature using different optimization techniques, including Exact methods, Heuristics, and Meta-heuristics. Numerous researchers used exact methods to solve flow shop problems. Initially, Schrage [7] applied branch and bound (B&B) to minimize the 2-machines flow shop problem's mean completion time.

Moursli and Pochet [8] minimized the initial gap between the upper and lower bound to 50%, using the B&B method. Chung, Flynn [9] studied PFSSP to minimize total flow time using the B&B method. Ronconi [10] minimized the makespan of PFSSP using the B&B method. Ng, Wang [11] introduced numerous dominance properties to minimize total completion time in PFSSP. Moukrim, Rebaine [12] proposed the B&B method for 2-machine PFSSP. Nagano, Robazzi [13] suggested an improved B&B using a new machine-based lower bound that considers machine blocking and idleness. Isenberg and Scholz-Reiter [14], Rossit, Tohmé [15], and Meng, Zhang [16] used mathematical programming techniques to solve PFSSP.

However, most of the literature used exact methods to solve small instances of PFSSP's. Algorithms before Moursli were applicable to test problems with several jobs less than 15, while Moursli and Della's algorithms are suitable to test problems with several jobs and machines less than 20 and 45, respectively. Except for Ronconi [10], the most recent exact methods can solve problems with jobs up to 20 and machines up to 8. PFSSP is a combinatorial optimization problem [17]; hence, it becomes difficult to solve these complex problems using exact methods when problem size increases. Therefore, heuristics and Meta-heuristics have been used to solve complex PFSSP in literature.

Johnson developed the first heuristic technique in 1951. Since then, several researchers have proposed heuristic techniques to solve the PFSSP of various sizes in reasonable computational time to minimize makespan. Tseng and Lin [18] classified heuristics into constructive, improvement, and composite heuristics.

Initially, Palmer [19] proposed a constructive heuristic to minimized makespan in PFSSP. Nawaz, Enscore [20] developed a constructive heuristic to solve PFSSP to minimize makespan for n-jobs with m number of machines. This technique processes jobs with the highest processing time first by Nawaz, Encore, and Ham (N.E.H.). A heuristic is the best heuristic for optimizing PFSSP's. Ronconi [21] proposed three constructive heuristics for PFSSP, namely MinMax (MM), MinMax based on N.E.H. (M.M.E.), and Profile Fitting based on N.E.H. (P.F.E.); he compared the results with N.E.H., both the M.M.E. and P.F.E. outperformed N.E.H. Heuristic. Ribas, Companys [22] improved the M.M.E. algorithm using the Blocking flow shop problem's reversible property. Shao, Shao [23] proposed two constructive methods and two Iterated greedy (I.G.) algorithms for distributed blocking FSSP. To avoid local minima, he used acceptance criteria based on fuzzy characteristics. Other common constructive heuristics are developed by Liu and Reeves [3]; Kalczynski and Kamburowski [24]; Pan and Wang [25]; Benavides and Ritt [26]. However, constructive heuristics produce infeasible results and takes considerable computational time to find near optimum solutions [27].

Besides, there are improvement heuristics proposed in the literature to solve PFSSP (e.g., Suliman [28]; Chen, Tzeng [29]; Ye, Li [30]). These heuristics improved some of the already developed heuristics by considering specific knowledge of some problems. Moreover, PFSSP has also been addressed by composite heuristic (e.g., Benavides and Ritt [26]; Ribas, Companys [31]; Lin, Wang [32]), which combines different heuristics to solve PFSSP. Composite heuristics have yielded much better results as compared to constructive and improvement heuristics. However, most of the heuristics developed in literature are independent of a time limit, and they stop after a predefined number of steps. It can give a possibility to trap in the local optima.

Therefore, meta-heuristics have been developed in the literature to search near-optimal solutions considering the termination criteria (e.g., CPU time, number of iterations). Meta-heuristics can obtain better solutions than heuristics, but they require more computational time (Tseng and Lin [18]). In literature, Meta-heuristic algorithms have been developed to solve PFSSP by several researchers. Most significant meta-heuristics used in literature are Artificial Bee Colony Algorithm (ABC) (Deng, Xu [33]; Han, Gong [34]; Li and Pan [35], [36]), Differential Evolution Algorithm (DE)(Liu, Yin [37]), Evolutionary Algorithm (EA)(Qian, Wang [38], Yeh and Chiang [39]), Genetic Algorithms (GA) (Caraffa, Ianes [40]; Ruiz, Maroto [41]; Vallada and Ruiz [42]; Akhshabi, Haddadnia [43]; Andrade, Silva [44]), Hybrid Discrete Differential Evolution (HDDE) (Wang, Pan [45]), Hybrid Differential Evolution Algorithm (HDEA) (Liu, Yin [37]), Simulated Annealing (SA) (Laha and Chakraborty [46]; Lin and Ying [47]; Moslehi and Khorasanian [48]) Lin, Cheng [49], Tabu Search (TS) (Taillard [50]; Grabowski and Wodecki [51]; Grabowski and Pempera [52]; Arık [53]), TS and ABC (Li and Pan [35]), Hybrid Whale optimization algorithm(HWO) (Abdel-Basset, Manogaran [54]), Particle swarm optimization (PSO) (Zhao, Qin [55]) Evolution Strategy (ES) (de Siqueira, Souza [56]; Khurshid, Maqsood [57]), among others. These algorithms have found competitive results for different PFSSP's compared to heuristics; however, they require more computational time, as they initiate from a sequence constructed by heuristics and is iterated until termination criteria are achieved.

Over the past years, significant research has been carried out on combining various Meta-heuristics. So that valuable features of each Meta-heuristic are used to get the desired results. A good option is to combine a global search technique with a local search technique to fine-tuning results. In this research, I.E.S. is combined with S.A. to minimize the makespan of PFSSP. I.E.S. performs best for global search; however, sometimes it gets stuck around local minima. Hence I.E.S. is combined with S.A., as S.A. avoids local minima and finds the best solution available in its neighborhood. S.A. was first used by.Kirkpatrick, Gelatt [58] to solve the traveling salesman problem. S.A. is a stochastic local search method taken from nature. In annealing, metals are slowly cooled to form a uniform crystallization instead of fast cooling, leading to poor crystallization. Similarly, the search process for a global minimum in S.A. mimics the crystallization cooling method. S.A. starts from a random solution and then finds the best solution available in its neighborhood.

E.S. is a type of evolutionary algorithm that mimics natural evolution to solve optimization problems [59]. E.S. has been developed in Germany by Rechenberg in the late 1960s, which operates with a population of size (μ+, λ), where μ stands for individual parent and λ represents the offspring. Rechenberg [60] completed the first dissertation in the field of E.S. Rechenberg used rectangular corridor and hypersphere models for the approximate analysis of the (1 + 1)-E.S. with Gaussian mutation. E.S. is an iterative process that uses a population of individual solutions to search the solution space [61]. Each individual represents a possible solution to the optimization problem. E.S. has been developed for numerical optimization problems and is widely used for its efficiency and robustness.

The performance of E.S. is mostly dependent on the adjustment of its internal parameters, i.e., mutation strength [61]. In E.S., all parents can be chosen to produce offsprings, as there is no compulsion that parents involved should be different. In E.S., there are no mating selection criteria. In literature, different reproduction operators have been used in ES i.e. (1 + 1), (1 + 4), (1 + 9) and (1 + 16) [62]. One parent can produce 1, 4, 9, and 16 offsprings in these operators, respectively.

E.S. has been used in flow shop problems of limited size. For example, de Siqueira, Souza [56] applied E.S. on hybrid flow shop problems to minimize makespan considering 50 jobs and eight machines. They used a random N.E.H. heuristic and Iterated Greedy Search (I.G.S.) meta-heuristic to create the solutions' initial population. Khurshid, Maqsood [57] used Hybrid Evolution Strategy for Robust PFSSP to minimize the makespan. Khurshid, Maqsood [63] used a fast E.S. algorithm to solve Carlier and Reeves benchmark PFSSP and validated the algorithm's result to solve a battery manufacturing case from the industry. In addition to flow shop problems, E.S. is also used in the evolutionary design of digital circuits (Miller [64]), forecasting foreign currency exchange rates (Rehman, Khan [65]), and for feedforward and recurrent networks (Mahsal Khan, Masood Ahmad [66]). However, Limited researchers used E.S. to solve PFSSP of large sizes instances. Furthermore, E.S. is better in performance than the other meta-heuristics, including G.A. (Costa and Oliveira [67]), and is used in current research to solve the considered PFSSP. In Table 1, various techniques used for solving PFSSP are summarized.

In this paper, an I.E.S. algorithm is hybridized with S.A. to minimize makespan for PFSSP. I.E.S. is recommended for global search; however, it tends to get trapped in local minima after few iterations. Hence, to use a salient feature of the local search technique, it is hybridized with SA. S.A. avoids local minima by accepting new solutions in its neighborhood even if it is inferior to the previous solution. Combining both these algorithms gives improved results for PFSSP. The following section reports the problem statement, which provides assumptions used in PFSSP. Next, the methodology is presented, which explains the proposed improvement over E.S. and S.A. Computational experiments, and results are shown in section 4, and the final section reports the conclusions and recommendations.

III. PROBLEM STATEMENT

PFSSP can be formulated as follows. Flow-shop scheduling involves n number of processed on m number of machines in the sequence of machines arranged in the shop. The processing time of Job J_i on machine M_j is given as $P_{i,j}$. The machine executes only one job, and it is processed in the same order. The goal is to find an optimum sequence so that the makespan (C_{max}) is reduced. Processing times are known in advance, and they are non-negative with fixed values. The assumptions used in the current problem and the objective function and constraints are as follows.

- At any time, one and only one job is operated by a machine.
- Anticipation is not permissible, all jobs are independent, and any job can be started as first.
- Machine downtime is ignored, and machines are continuously available.
- The Job processing sequence is the same for each machine.
- The setup time is incorporated into the machine processing times.

For n jobs and m machines, the makespan can be calculated using Eq. 1- Eq. 4.

$$C_{max} = \max(C_{a,J_1}, C_{1,J_b}, \ldots, C_{a,J_b} \quad (1)$$

where,

$$C_{a,J_1} = \sum_{l=1}^{a} P_{l,J1} \quad a = 1, \ldots, m \quad (2)$$

$$C_{1,J_b} = \sum_{l=1}^{b} P_{1,J_l} \quad b = 1, \ldots, n \quad (3)$$

$$C_{a,J_b} = \max(C_{a-1,J_b}, C_{a,J_{b-1}}) + P_{a,J_b}$$
$$a = 2, \ldots, m \quad b = 2, \ldots, n \quad (4)$$

Minimization of makespan is the most common objective for PFSSP as it directly correlates to the maximum utilization of machines [2]. This research aims to reduce makespan for PFSSP using a Hybrid E.S. In this research 120, Taillard PFSSP's comprises 12 different problem sets, ranging from 20 jobs and five machines to 500 jobs 20 machines are solved using the proposed technique.

IV. METHODOLOGY

A. INTRODUCTION TO ES

Evolutionary strategy imitates the principle of natural evolution to solve parameter optimization problems. E.S. was

TABLE 1. Summary of literature on single objective PFSSP's.

Author	Objective Function					Technique
	Makespan	Tardiness	Mean Completion Time	Total Flow Time	Taillard Problems	
Suliman [28]	✓					Heuristic
Moursli and Pochet [8]	✓					B&B
Caraffa, Ianes [40]	✓					GA
Chung, Flynn [9]				✓		B&B
Della Croce, Ghirardi [68]	✓					Lagrangean Approach
Ronconi [10]	✓				✓	B&B
Ruiz, Maroto [41]	✓					GA
Grabowski and Pempera [52]	✓				✓	TS
Kalczynski and Kamburowski [24]	✓					Heuristic
Qian, Wang [38]	✓					HDE
Zobolas, Tarantilis [69]	✓					Hybrid Metaheuristic
Ng, Wang [11]				✓		B&B
Wang, Pan [45]	✓				✓	HDDE Algorithm
Vallada and Ruiz [42]		✓				GA
Deng, Xu [33]				✓		A.B.C. Algorithm
Akhshabi, Haddadnia [43]	✓					Parallel GA
de Siqueira, Souza [56]	✓				✓	ES
Liu, Yin [37]	✓					HDEA
Moslehi and Khorasanian [48]	✓					HVNSA
Li and Pan [35]			✓			Novel Hybrid Algorithm
Benavides and Ritt [26]	✓					Heuristic
Ribas, Companys [31]	✓					Heuristic
Ye, Li [30]	✓					Heuristic
Lin, Wang [32]	✓					Heuristic
Abdel-Basset, Manogaran [54]	✓				✓	H.W.O.
Yeh and Chiang [39]	✓	✓			✓	EA
Zhao, Qin [55]	✓				✓	PSO
Andrade, Silva [44]				✓	✓	G.A.
Khurshid, Maqsood [57]	✓					ES & TS
Khurshid, Maqsood [63]	✓					ES
Arık [53]	✓				✓	TS
Lin, Cheng [49]	✓					SA
Nagano, Robazzi [13]	✓					B&B
Shao, Shao [23]	✓					B&B & IG
Proposed Research	✓				✓	HES_{SA}

introduced by [60]. E.S. depends on the collective learning model gathered from natural evolution and principles of reproduction, recombination, mutation, and selection. During the optimum search, E.S. tries to adapt its strategy parameters by using a collective self-learning mechanism. In E.S., strong emphasis is done on the mutation to create offsprings. For faster results, mutation parameters are changed during the execution of the program. In the evolution strategy, floating-point representation is used, and mutation is the only recombination operator. Initially, experiments were performed having one descendant and one ancestor per generation, and mutation was done by subtracting two numbers drawn from a binomial distribution. The offspring replaced the ancestor if it was found better. After the arrival of computers, this two-membered or $(1+1)$-E.S. technique is complemented by the multi-membered version with recombination. Now within one cycle, parents create offsprings. Two or more parents may be involved in the recombination step, two extreme forms known as intermediate and discrete, respectively. In intermediate recombination, parental variable average values are shifted to the new offspring, while discrete recombination selects each component from one parent at random.

The basic steps of E.S. are as follows:

Step 1: Initialization
Step 2: Reproduction
Step 3: Recombination
Step 4: Mutation
Step 5: Selection
Step 6: Termination

B. SIQUEIRA E.S. FOR HYBRID FLEXIBLE FLOW LINE PROBLEMS

Siqueira (2013) used E.S. to minimize the makespan of Hybrid Flexible Flow Line Problems (HFFL). The pseudocode for the Siqueira E.S. Algorithm is shown in Figure 1.

Pseudo Code for HFFL Algorithm

Step 1: *Set Initial parameters*

Step 2: *Generate Initial Population (Half population generated by N.E.H. heuristic, and other half population generated by Iterated Greedy Search metaheuristic)*
Step 3: Reproduction (1+1, from 2 parents randomly two offsprings are generated)
Step 4: Step 5: Mutation (Block reallocation)
Step 5: Selection (Deterministic)
Step 6: Repeat until stopping criteria is met

FIGURE 1. **Pseudocode for HFFL algorithm.**

For reproduction Siqueira (2013) used (1 + 1)-E.S., from 1 parent, one offspring was generated, and the selection pools consist of 2 entities. Although less computational time is consumed for the (1+1)-E.S. reproduction operator, the selection pool is tiny to exploit the solution space thoroughly. Therefore ample iterations are required to find the optimum solution.

The mutation rate used by Siqueira (2013) is not changed constantly; however, to increase genetic variation in the population and improve results in fewer iterations, a variable mutation rate should be used. Siqueira(2013) applied E.S. on HFFL problems with jobs up to 50 and machines up to 8. E.S. should be tested on complex benchmark PFSSP's (i.e., Taillard, Vallada, Carlier, and Reeves Flow Shop problems) to validate its performance.

C. THE PROPOSED I.E.S. ALGORITHM

In this proposed I.E.S., the following improvements have been made as compared to Siqueira (2013).

- To thoroughly exploit solution space, (1 + 4)-E.S. has been used instead of (1 + 1)-E.S. Four offsprings are generated from one parent. The selection pool consists of 5 entities, one parent, and four offsprings.
- For maximum exploitation of solution space in minimum computational time, Double swap mutation is used.
- Initially, a high mutation rate is used; however, the mutation rate varies to avoid local minima and fine-tuning results. Variation in mutation rate is the crucial advantage of E.S.
- To test I.E.S. on a complex problem, it has been applied to Taillard Problems with the number of jobs ranging from 20 to 500 and the number of machines ranging from 5 to 20. (Taillard problems are the most complex benchmark flow shop problems available in the literature).

Pseudocode for the proposed I.E.S. is shown in Figure 2. Flowcart for HES_{SA} is shown in graphical form in Figure 5.

1) SELECTION OPERATOR (PARENT)

The parent population is randomly generated. For a population size of five, the randomly generated parent population is as follows.

Parent		2	1	4	3	5

2) REPRODUCTION OPERATOR

Siqueira (2013) used (1 + 1)-E.S. for reproduction, although (1 + 1)-E.S. is fast, but the solution space is not thoroughly exploited. To overcome this problem, (1 + 4)-E.S. is used in this paper as it explores more solution space and find better results from small to large sized problems. The reproduction operator selects the parents who take part in the generation of offsprings. From 1 parent, four offsprings are generated randomly, as shown in Figure. 3. Other reproduction operators, i.e. (1 + 5), (1 + 9), and (1 + 16), can be used; however, they will take ample computational time to solve complex scheduling problems.

3) RECOMBINATION OPERATOR

Recombination operator brings similarities between parents and their offsprings. Recombination itself has no benefit; however, it is useful when combined with enormous mutation strength and selection. A mutation is mandatory for evolutionary progress and new offsprings production; however, most offsprings are harmful. The selection operator must select suitable mutants. The recombination then extracts standard features, i.e., the similarity in these selected individuals and reduce uncorrelated part. Hence the chosen similarities are the most beneficial ones. Discrete recombination is used in this research. In discrete recombination, variable values of individuals are exchanged. Equal probability is used by the parent to share its variable with the offspring and is done randomly.

4) DOUBLE SWAP MUTATION OPERATOR

The mutation operator is the most important operator of the E.S. besides the selection and reproduction operator. It introduces genetic variation in the population. In E.S., mutation operators are problem-dependent. Their accurate design is essential in E.S. The double Swap mutation operator is used in

```
Pseudo Code for I.E.S. Algorithm
Step 1:   Set the input parameters
          Population size=5
          No of generations, n
          Mutation rate, m=40%
          Seed
Step 2:   Initialization: gen=1: n, gen is the present Generation
Step 3:   Evaluation: Calculate makespan for the parent solution
Step 4:   Generate a new population randomly
Step 5:   Reproduction (1+4), from one parent four offsprings are generated
Step 6:   Recombination (Discrete)
Step 7:   Mutation (Double Swap Mutation)
Step 8:   Calculate makespan for all newly generated and mutated offsprings
Step 9:   Selection of the fittest gene (Deterministic)
Step 10:  Term the fittest gene as the parent for the next iteration
Step 11:  If gen<n
            Then go to step 2 and Set gen=gen+1;
          else
          Record Makespan and schedule for the best solution;
```

FIGURE 2. Pseudo code for IES.

Parent	2	1	4	3	5
Offspring 1	2	3	4	1	5
Offspring 2	2	1	5	3	4
Offspring 3	2	1	4	5	3
Offspring 4	2	4	1	3	5

FIGURE 3. (1 + 4) reproduction strategy.

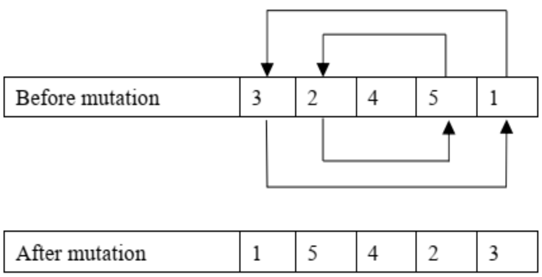


FIGURE 4. Double swap mutation.

this research; the procedure of double swap mutation operator (with 40% mutation rate) is illustrated in Figure. 4. The position of Gene 1 is interchanged with Gene 5, while the position of Gene 2 is interchanged with Gene 4 simultaneously. Double swap mutation takes less time and guides the solution to more promising areas.

The mutation rate varies after a specified time interval to reduce genetic variation with an increasing number of iterations and fine-tune the results. Variable mutation rate increases the chances of attaining the best results in minimum computational time and prevents the algorithm from trapping in the local minima. The mutation rate varies depending on the size of problems as for large-size problems low mutation rate is used; otherwise, the mutation operator becomes a random search operator. Taillard 120 benchmark problems can be divided into five categories depending on the number of jobs, i.e., 20, 50, 100, 200, and 500, respectively. For each category, a specific mutation rate is used depending on the computational time. The mutation rate against the number of jobs is mentioned in Table 2.

5) SELECTION OPERATOR (SURVIVOR)

From λ descendants, μ best individuals are deterministically chosen. In 1975, [70] introduced two new multi membered-ES survivor selection schemes, i.e. $(\mu + \lambda)$-E.S., (μ, λ)-E.S.

In $(\mu + \lambda)$-E.S., parents, and offsprings are considered in the selection pool, $(\mu + \lambda)$-the selection is recommended for the combinatorial optimization problem. While in (μ, λ)-E.S., only offsprings are considered in the selection pool, while parents die out of the selection pool, (μ, λ)-Selection is recommended for real-valued parameter optimization.

In this paper $(\mu + \lambda)$ selection scheme is used as it guides solutions to promising areas. Since four offsprings are generated from 1 parent, hence the selection pool consists of 5 entities. The parent can survive for many generations unless replaced by a better offspring.

6) TERMINATION

Termination criteria, i.e., the maximum number of iterations, computational time, and fitness value, are commonly used.

TABLE 2. Mutation rate against the number of jobs for taillard instances.

Number of Jobs	Time Interval	Mutation Rate
20	For < 200 ms	40%
	For> 200 and < 400 ms	30%
	For> 400 ms	20%
50	For < 20 s	40%
	For> 20 and < 40 s	30%
	For> 40 s	20%
100	For < 40 s	30%
	For> 40 and < 80 s	20%
	For> 80 s	10%
200	For < 5 min	20%
	For> 5 and < 10 min	10%
	For> 10 min	4%
500	For < 35 min	20%
	For> 35 and < 70 min	10%
	For> 70 min	4%

The stopping criteria used in HES_{SA} is the Maximum computational time, set at $n^2/2 \times 10$ ms for each instance. Hence the whole algorithm constituting of I.E.S. and S.A. is run for $n^2/2 \times 10$ ms.

D. SIMULATED ANNEALING

S.A. is a local search procedure originating from material science and was initially used as a simulation model in the solids' annealing process. S.A. does not guarantee an optimal solution; however, it will find a better neighborhood solution. At each iteration, S.A. searches within the neighborhood and evaluates the possible candidate solutions. Based on the acceptance-rejection criteria, the candidate solution is either accepted or rejected, and the correct selection of these criteria has a significant effect on the performance of the S.A. algorithm. The main criteria in designing the S.A. algorithm are i) Schedule representation ii) Neighborhood design, iii) Searching within the neighborhood, and iv) Acceptance-rejection criteria. In S.A., a probabilistic procedure is used for the acceptance-rejection criteria.

Several iterations are performed in S.A. At iteration k, the best-known schedule is termed as S_o, while the current schedule is termed as S_k. $G(S_o)$ and $G(S_k)$ are the corresponding values. $G(S_o)$ is also termed as aspiration criteria. S.A. algorithm moves from one schedule to another in search of an optimal schedule. At iteration k, a new schedule is searched in the neighborhood of Sk. A candidate schedule Sc is either randomly selected or through a genetic operator. A move is made If $G(S_c) < G(S_k)$ and set $S_{k+1} = S_c$. If $G(S_c) < G(S_0)$, then S_0 is equal to S_c. While a move is allowed with probability $P(S_k, S_c)$ if $G(S_c) \geq G(S_k)$,

$$P(S_k, S_c) = e^{\left\{\frac{G(S_k)-G(S_c)}{\beta_k}\right\}} \tag{5}$$

If $G(S_c) \geq G(S_k)$, then a random number (μ_k) between 0 and 1 and compared with probability if $\mu_k \leq P(S_k, S_c)$, then set $S_{k+1} = S_c$ else set $S_{k+1} = S_k$. β_k is the cooling parameter (analogous to the annealing process). Its initial value is between 0.9 to 0.95, which reduces with an increase in the number of iterations.

Unlike E.S., in S.A., the worst move is allowed, giving S.A. a chance to escape local minima and find a suitable solution in a later search. As β_k reduces with the number of iterations, the acceptance probability of a non-improving search is minimal as the number of iterations approaches its limit. If a neighborhood is worse, then acceptance probability ensures that a move is avoided. Several stopping criteria can be used in S.A., i.e., the number of iterations, the value of an objective function is met, or no improvement is observed for a specific interval. In this S.A., the computational time is used.

The best solution of I.E.S. is used as a seed for the S.A. algorithm. S.A. algorithm uses it as the initial schedule and then finds candidate schedules in its neighborhood. Mutation, cooling parameter, and acceptance-rejection criteria induce randomness in the solution search procedure. Insertion mutation is used in S.A. while cooling parameters vary between 0.95 and 0.6. The pseudocode for the S.A. algorithm is shown in Figure 6.

V. COMPUTATIONAL RESULTS

A. EXPERIMENTAL SETUP

The algorithm is coded in MATLAB and run on a Core™i5 with 2.6 GHz and 4 G.B. memory and tested on Taillard [71] benchmark PFSSP. Taillard PFSSP is the most challenging combinatorial optimization problem. A cushion for improvement is still available; for more than 100, the Upper bound schedule is still unknown for most instances. Taillard instances data is taken from OR Library. The benchmark set contains 120 different problems and divided into 12 groups, with each group containing ten instances with machines ranging from 5 to 20 and jobs ranging from 20 to 500. For each instance, Taillard has used a seed. Computational time was used as the termination criteria, and each instance was run for $n^2/2 * 10$ ms.

B. COMPARISON OF RESULTS

Results of the empirical tests for the suggested HES_{SA} are reported, and computational results from algorithms of Zobolas, Tarantilis [72], Chen, Huang [73], Marinakis, and Marinaki [74], and Abdel-Basset, Manogaran [54]. Zobolas, Tarantilis [72] suggested a Hybrid meta-heuristic ($NEGA_{VNS}$) by combining a Greedy randomized constructive heuristic, a Genetic algorithm, and a Variable

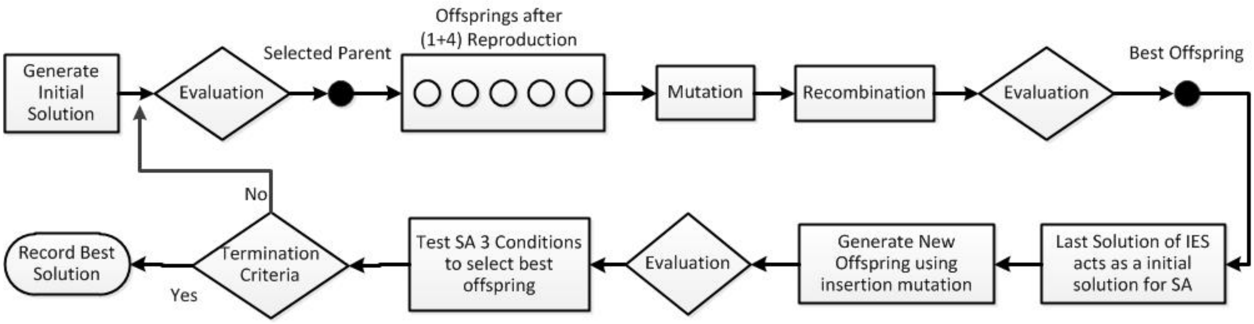


FIGURE 5. Flowchart for HES_{SA} algorithm.

Pseudo Code for S.A. Algorithm

Step 1: Initialize input parameters

The best-known schedule from I.E.S. is termed S_o

Value of best-known schedule, $G(S_o)$

Cooling parameter, β_k (Maximum value is set at 0.95 while minimum value is set at 0.6)

Step 2: For $k=1: n$

Step 3: $S_o=S_k$

Step 4: Hence So for $k=1$, $S_o=S_1$,

Step 5: Then $G(S_o)=G(S_1)$

Step 6: Perform Insertion Mutation in S_1 and generate new candidate schedule, S_c

Step 7: Compute Cmax for the new candidate schedule, $G(S_c)$

Step 8: Probability of moving from S_k to S_c schedule at K^{th} iteration = $P(S_k, S_c) = e^{\left(\frac{G(S_k)-G(S_c)}{\beta_k}\right)}$

Step 9: Now test the three conditions

If
$G(S_o) < G(S_c) < G(S_k)$,
Then $S_{k+1} = S_c$
Go to step (10)

Else If
$G(S_c) \leq G(S_o)$,
Then $S_o = S_{k+1} = S_c$
Go to step (10)

Else if
$G(S_c) > G(S_k)$,
THEN generate a random number $U_k \sim [0, 1]$
IF $U_k \leq P(S_k, S_c)$,
THEN $S_{k+1} = S_c$

Else $S_{k+1} = S_k$

Step 10: Set $k=k+1$
$B_{k+1} = (\beta_k - 0.01)$
If $k <= n$
Then Go to Step (2)

Step 11: Else Stop and record Makespan and Schedule of the best iteration

FIGURE 6. Pseudocode for S.A. algorithm.

neighborhood search (V.N.S.) method to minimize the makespan of PFSSP. For the first time, the algorithm combined G.A. with a V.N.S. for fine-tuning results and escaping local minima. The algorithm was coded in C++ and run a Pentium®IV with 2.6 GHz and 1 GB Ram and tested on Taillard [71] benchmark instances. The maximum running time for the algorithm was $n \times m/10$ s. Marinakis and Marinaki [74] proposed a Particle swarm optimization algorithm (PSOENT) with a new algorithmic nature-inspired technique to minimize the makespan of PFSSP. The PSOENT algorithm combines the PSO algorithm with expanding neighborhood topology, a variable neighborhood search technique, and a

TABLE 3. PRD comparison of HES_{SA} values with $NEGA_{VNS}$, PSOENT, RDPSO, and H.W.A. algorithms.

Algorithm	C	$NEGA_{VNS}$	PSOENT	RDPSO	HWA	HES_{SA}	$NEGA_{VNS}$	PSOENT	RDPSO	HWA	HES_{SA}
n x m		C_{max}	C_{max}	C_{max}	C_{max}	C_{max}	PRD	PRD	PRD	PRD	PRD
20 - 5	1278	1278	1278	1278	1278	1278	0.00	0.00	0.00	0.00	0.00
20 - 10	1582	1582	1582	1582	1582	1582	0.00	0.00	0.00	0.00	0.00
20 - 20	2297	2297	2298	2297	2297	2297	0.00	-0.04	0.00	0.00	0.00
50 - 5	2724	2724	2724	2724	2724	2724	0.00	0.00	0.00	0.00	0.00
50 - 10	2991	3021	3092	3051	3021	3024	-1.00	-3.38	-2.01	-1.00	-1.10
50 - 20	3850	3874	4004	3950	3876	3889	-0.62	-4.00	-2.60	-0.68	-1.01
100 - 5	5493	5493	5493	5493	5493	5493	0.00	0.00	0.00	0.00	0.00
100 - 10	5770	5770	5851	5790	5776	5776	0.00	-1.40	-0.35	-0.10	-0.10
100 - 20	6202	6303	6430	6414	6280	6257	-1.63	-3.68	-3.42	-1.26	-0.89
200 - 10	10862	10885	10953	10872	10885	10872	-0.21	-0.84	-0.09	-0.21	-0.09
200 - 20	11195	11339	11571	11535	11335	11287	-1.29	-3.36	-3.04	-1.25	-0.82
500 - 20	26040	26228	26737	26656	26388	26187	-0.72	-2.68	-2.37	-1.34	-0.56
Average							-0.46	-1.61	-1.16	-0.49	-0.38

***PRD Values calculated at 30 runs of each instance**

path relinking technique. Starting from a small-sized neighborhood, the neighborhood sizes increase with each iteration, and the neighborhood ends to a limit so that all swarms are included in it. The algorithm utilizes the global neighborhood technique's exploration ability and exploitation ability of the local neighborhood technique. The algorithm was coded in Fortran 90 and tested on Taillard [71] benchmark instances, and the termination criteria for instances with 20, 50, 100, 200, and 500 jobs was 60 sec, 120 sec, 180 sec, 300 sec, and 500 sec respectively.

Chen, Huang [73] proposed a revised discrete particle swarm optimization algorithm (RDPSO) for PFSSP to minimize makespan. A new particle swarm learning strategy is introduced in the RDPSO algorithm for guiding the search to find the personal and global best solutions. A new filtered local search is applied to avoid premature convergence, which guides the search to new solution areas and avoids already reviewed regions. The algorithm was tested on Taillard [71] benchmark instances with a P.C. having Intel Pentium IV at 2.6 GHz. The termination criteria were 1000 iterations for all the instances, and the population size is set at 60. Abdel-Basset, Manogaran [54] proposed a Hybrid algorithm (H.W.A.) that combined a Whale optimization algorithm as a Local search strategy to minimize makespan in PFSSP. To use the most considerable rank value, discrete search space is required in the algorithm. By using swap mutation, the candidate solution's diversity is improved, and to escape local optima. An insert-reversed block operation is incorporated in the algorithm. The performance of the initial solution was improved by using the N.E.H. heuristic [20]. The algorithm was coded in Java and run on a Core TMi5-3317U with 1.7 GHz and 4 GB Ram. The algorithm was tested on Taillard [71], Carlier [75], Reeves [76], and.Heller [77] benchmark instances.

For a fair comparison of HES_{SA} with $NEGA_{VNS}$, PSOENT, RDPSO, and H.W.A., the termination criteria for all these algorithms were computational time. The maximum computational time was set at $n^2/2 * 10$ ms. These algorithms were tested on the same processor, i.e., Core TMi5 with 2.6 GHz and 4 GB RAM.

The effectiveness of the suggested technique is analyzed in terms of solution quality. For each group, the quality of the algorithm was evaluated using Eq. 6.

$$PRD = \frac{100 * (C - C^m)}{C} \tag{6}$$

where

PRD- Percentage relative difference
C^m = *Makespan found from the algorithm of* HES_{SA}, $NEGA_{VNS}$, *PSOENT, RDPSO and HWA*
C = *Upper bound for Taillard Instances*

In the case of HES_{SA}, the values are averaged values over 30 runs of each instance. We summarized the results in Table 3. A positive value of PRD shows that the results are better than C, and the best values of PRD are highlighted in Bold. For the first three groups where the number of jobs is 20 respectively, PRD values for all these algorithms, i.e., HESSA, NEGAVNS, RDPSO, and H.W.A., are zero, which means all algorithms have found the same optimal makespan for these groups. PRD values for PSOENT are zero for Group 1 and 2 while it is -0.04 for group 3; hence, it cannot find the optimal schedule for Group 3 and performs inferior compared to other algorithms. For Group 4, the P.D.R. values for HES_{SA}, $NEGA_{VNS}$, PSOENT, RDPSO, and H.W.A. are also zero. So for this group, all algorithms have found the upper bound values. Moreover, their performance is at par with each other. For Group 5 (50- 10), HES_{SA} performs better than PSOENT, RDPSO. However, $NEGA_{VNS}$ and H.W.A. perform slightly better than HES_{SA}. For Group 6 (50- 20) and 8 (100- 10), $NEGA_{VNS}$ performs better than all other algorithms. Group 7 (100- 5) all algorithms have found the upper bound makespan and have performed equally well. For Group 9 (100- 20), 10 (200- 10), 11 (200- 20), and 12 (500- 20), HES_{SA} outperforms $NEGA_{VNS}$, H.W.A., RDPSO, and PSOENT as its PRD values are minimum for these four groups, and also these four groups contain the most challenging instances of Taillard Flow shop problems.

The average PRD values for HES_{SA}, $NEGA_{VNS}$, H.W.A., RDPSO, and PSOENT are −0.38, −0.46, −0.49, −1.16, and −1.61, respectively. In terms of overall performance, HES_{SA} outperformed all others algorithms as its overall PRD value is less than all other algorithms. As the Average PRD value of HES_{SA} is minimum than all other algorithms, it shows the HES_{SA} algorithm's robustness for all size problems, i.e., small, medium, and significant problems.

C. NEW UPPER BOUNDS AND IMPROVED SOLUTION FOR TAILLARD INSTANCES

Table 4 makespan values of HESSA compared with the makespan values of $NEGA_{VNS}$, PSOENT, RDPSO, and H.W.A. algorithms. The HES_{SA} algorithm has found 54 Upper bound makespan values for Taillard instances (Highlighted in bold). While makespan values of 38 solutions are improved (Highlighted in bold and underlined). For the remaining 28 Taillard instances, makespan values are very close to NEGAVNS and H.W.A. algorithms' makespan values. For the first 4 Groups (TA 01-30), all algorithms find the upper bound makespan for Taillard Instances except for TA- 07. So all algorithms can find the optimal schedules for small-size problems. For Group 5 (50- 5), all algorithms' performance is leveled as they find the same makespan values. Afterward, with the increase in the Several jobs and machines, PSOENT and RDPSO, lag behind other algorithms. In the case of Group 6 (50 -20), both PSOENT and RDPSO fall behind HES_{SA}, $NEGA_{VNS}$, and H.W.A. algorithms. In comparison, H.W.A. performs better than other algorithms in this group. In Group 8, HES_{SA}, improve the solution for seven instances and performs better than other instances.

The main feature of the HES_{SA} algorithm is its robustness and effectiveness in solving all sized problems, as it has found Upper bounds for small instances, i.e., 20 × 5 (TA- 01), and even for large instances, i.e., 200 × 10 (TA- 95). Moreover, it has improved solutions from medium-sized problems to large-sized problems (Highlighted in bold and underlined in Table 4).

Figure 7-10 provides a graphical view and comparison of makespan values of $NEGA_{VNS}$, PSOENT, RDPSO, and H.W.A. with HES_{SA}. For all the 120 Taillard instances for PFSSP. Each Figure covers 30 Taillard instances. Figure 7 compares all algorithms' makespan values for TA 01-30 instances, and it appears that almost all algorithms found an upper bound for small-sized Taillard instances. For instance, in TA-07 (20 × 5), the makespan values for $NEGA_{VNS}$, PSOENT, RDPSO, and H.W.A. with HES_{SA} is 1239, while for the upper bound is 1234. This is currently the only unresolved problem in the first 30 Taillard instances whose Upper bound is still pending. Figure 8 compares makespan values for the following 30 instances, i.e., TA 31-60. From Figure 8, it is apparent that HES_{SA} provides a lower makespan compared to $NEGA_{VNS}$, PSOENT, RDPSO, and H.W.A. Results of PSOENT and RDPSO are inferior to $NEGA_{VNS}$, H.W.A., and HES_{SA}. Figure 9 compares the makespan values for the following 30 instances, i.e., TA 61-90. From Figure 8, it is apparent that HES_{SA} provides the best results for all the instances compared to $NEGA_{VNS}$, PSOENT, RDPSO, and H.W.A. However, the results of PSOENT, RDPSO are much inferior to other algorithms. Figure 10 compares the makespan values for the last 30 instances, i.e., TA 91-120. From Figure 9, it is apparent that HES_{SA} performs best for all the Taillard instances. Results of PSOENT, RDPSO are inferior to other algorithms. While results of $NEGA_{VNS}$, H.W.A., and HES_{SA} are at a level with each other and is, for some instances, H.W.A. provides the best results.

From Figure 7-10, it is eminent that the makespan values of HES_{SA} are minimal compared to $NEGA_{VNS}$, PSOENT, RDPSO, and H.W.A. algorithms, and it is equally efficient to small, medium, and large-sized problems.

D. COMPARISON OF MAKESPAN WITH-SEED AND WITHOUT-SEED HES_{SA} ALGORITHM

In Table 5, a comparison is made for the makespan calculated with-seed and without-seed HES_{SA} algorithm for the twelve different groups of Taillard Problems. The value of % Diff Makespan for each instance is calculated using Eq. 7.

$$\%Diff\ C_{max} = \frac{100 * \left(C^{seed} - C^{w/seed}\right)}{C^{seed}} \tag{7}$$

where

C^{seed} = *Makespan calculated using Taillard seed*
$C^{w/seed}$ = *Makespan calculated without using Taillard seed*

It shows that the with-seed HES_{SA} algorithm exploits more solution space and finds better makespan values than the without-seed HES_{SA} algorithm. % Diff Cmax values depict the algorithm's performance; a negative value of the % Diff Cmax shows that the makespan with-seed algorithm has a better makespan value compared to a without-seed algorithm, as shown in Table 5. Makespan values are calculated at two termination criteria based on computational time, i.e., $n^2/2$ ms and $n^2/2 \times 10$ ms. for the 1st termination criteria ($n^2/2$ ms), all twelve %Diff Cmax values are negative. For the

TABLE 4. Makespan comparison of HES_{SA} with $NEGA_{VNS}$, PSOENT, RDPSO, and H.W.A. algorithms.

Taillard Problem (TA)	Size	Seed	UB	$NEGA_{VNS}$	PSOENT	RDPSO	HWA	HES_{SA}
01	20 x 5	873654221	1278	1278	1278	1278	1278	**1278**
02		379008056	1359	1359	1359	1359	1359	**1359**
03		1866992158	1081	1081	1081	1081	1081	**1081**
04		216771124	1293	1293	1293	1293	1293	**1293**
05		495070989	1235	1235	1235	1235	1235	**1235**
06		402959317	1195	1195	1195	1195	1195	**1195**
07		1369363414	1234	1239	1239	1239	1239	**1239**
08		2021925980	1206	1206	1206	1206	1206	**1206**
09		573109518	1230	1230	1230	1230	1230	**1230**
10		88325120	1108	1108	1108	1108	1108	**1108**
11	20 x 10	587595253	1582	1582	1582	1582	1582	**1582**
12		1401007982	1659	1659	1659	1659	1659	**1659**
13		873136276	1496	1496	1500	1496	1496	**1496**
14		268827376	1377	1377	1377	1377	1377	**1377**
15		1634173168	1419	1419	1419	1419	1419	**1419**
16		691823909	1397	1397	1397	1397	1397	**1397**
17		73807235	1484	1484	1484	1484	1484	**1484**
18		1273398721	1538	1538	1544	1543	1538	**1538**
19		2065119309	1593	1593	1593	1593	1593	**1593**
20		1672900551	1591	1591	1591	1598	1591	**1591**
21	20 x 20	479340445	2297	2297	2298	2297	2297	**2297**
22		268827376	2099	2099	2101	2100	2099	**2099**
23		1958948863	2326	2326	2328	2326	2326	**2326**
24		918272953	2223	2223	2225	2223	2223	**2223**
25		555010963	2291	2291	2294	2294	2291	**2291**
26		2010851491	2226	2226	2229	2228	2226	**2226**
27		1519833303	2273	2273	2273	2273	2273	**2273**
28		1748670931	2200	2200	2202	2200	2200	**2200**
29		1923497586	2237	2237	2240	2237	2237	**2237**
30		1829909967	2178	2178	2178	2178	2178	**2178**
31	50 x 5	1328042058	2724	2724	2724	2724	2724	**2724**
32		200382020	2834	2834	2838	2836	2834	2836
33		496319842	2621	2621	2621	2621	2621	**2621**
34		1203030903	2751	2751	2751	2751	2751	**2751**
35		1730708564	2863	2863	2863	2863	2863	**2863**
36		450926852	2829	2829	2829	2829	2829	**2829**
37		1303135678	2725	2725	2725	2725	2725	**2725**
38		1273398721	2683	2683	2683	2683	2683	**2683**
39		587288402	2552	2552	2554	2555	2552	**2552**
40		248421594	2782	2782	2782	2782	2782	**2782**
41	50 x 10	1958948863	2991	3021	3092	3051	3021	3024
42		575633267	2867	2902	2942	2915	2891	**2882**
43		655816003	2839	2871	2926	2889	2869	**2852**
44		1977864101	3063	3070	3083	3071	3063	**3063**
45		93805469	2976	2998	3049	3024	3001	**2982**
46		1803345551	3006	3024	3056	3036	3006	**3006**
47		49612559	3093	3122	3144	3133	3126	**3122**
48		1899802599	3037	3063	3072	3049	3046	**3042**
49		2013025619	2897	2914	2952	2923	2897	2911
50		578962478	3065	3076	3143	3131	3078	3077

TABLE 4. *(Continued.)* Makespan comparison of HES_{SA} with $NEGA_{VNS}$, PSOENT, RDPSO, and H.W.A. algorithms.

51	50 x 20	1539989115	3850	3874	4004	3950	3876	3889
52		691823909	3704	3734	3838	3761	3715	**3714**
53		655816003	3640	3688	3788	3741	3653	3667
54		1315102446	3723	3759	3857	3806	3755	**3754**
55		1949668355	3611	3644	3732	3688	3649	**3644**
56		1923497586	3681	3717	3821	3758	3703	3708
57		1805594913	3704	3728	3855	3763	3723	3754
58		1861070898	3691	3730	3825	3788	3704	3711
59		715643788	3743	3779	3903	3831	3763	3772
60		464843328	3756	3801	3896	3830	3767	3778
61	100 x 5	896678084	5493	5493	5493	5493	5493	**5493**
62		896678084	5268	5268	5274	5268	5268	**5268**
63		1122278347	5175	5175	5179	5175	5175	**5175**
64		416756875	5014	5014	5023	5014	5018	**5014**
65		267829958	5250	5250	5255	5250	5250	**5250**
66		1835213917	5135	5135	5135	5135	5135	**5135**
67		1328833962	5246	5246	5251	5246	5246	**5246**
68		1418570761	5094	5094	5094	5094	5094	**5094**
69		161033112	5448	5448	5454	5448	5448	**5448**
70		304212574	5322	5322	5332	5322	5324	**5322**
71	100 x 10	1539989115	5770	5770	5851	5790	5776	5776
72		655816003	5349	5358	5407	5377	5362	5360
73		960914243	5676	5676	5691	5679	5691	5677
74		1915696806	5781	5792	5902	5849	5825	**5792**
75		2013025619	5467	5467	5588	5514	5491	**5467**
76		1168140026	5303	5311	5334	5308	5308	5311
77		1923497586	5595	5605	5658	5602	5608	**5596**
78		167698528	5617	5617	5695	5664	5630	5625
79		1528387973	5871	5877	5958	5907	5891	5891
80		993794175	5845	5845	5903	5857	5848	**5845**
81	100 x 20	450926852	6202	6303	6430	6414	6280	**6257**
82		1462772409	6183	6266	6489	6383	6278	**6223**
83		1021685265	6271	6351	6526	6437	6368	**6342**
84		83696007	6269	6360	6440	6407	6350	**6303**
85		508154254	6314	6408	6612	6509	6377	6380
86		1861070898	6364	6453	6633	6551	6430	**6427**
87		26482542	6268	6332	6605	6476	6354	**6306**
88		444956424	6401	6482	6724	6640	6515	**6472**
89		2115448041	6275	6343	6576	6462	6396	6380
90		118254244	6434	6506	6699	6593	6527	**6485**
91	200 x 10	471503978	10862	10885	10953	10872	10885	**10872**
92		1215892992	10480	10495	10610	10556	10512	**10487**
93		135346136	10922	10941	11040	10950	10965	**10941**
94		1602504050	10889	10889	10939	10893	10889	**10889**
95		160037322	10524	10524	10646	10537	10524	**10524**
96		551454346	10329	10346	10452	10378	10375	**10346**
97		519485142	10854	10866	10977	10882	10868	10868
98		383947510	10730	10741	10864	10777	10751	**10741**
99		1968171878	10438	10451	10498	10450	10465	**10451**
100		540872513	10675	10684	10810	10727	10727	**10680**

TABLE 4. *(Continued.)* **Makespan comparison of HES_{SA} with $NEGA_{VNS}$, PSOENT, RDPSO, and H.W.A. algorithms.**

101	200 x 20	2013025619	11195	11339	11571	11535	11335	**11287**
102		475051709	11203	11344	11729	11596	11517	**11277**
103		914834335	11281	11445	11757	11676	11481	**11418**
104		810642687	11275	11434	11713	11665	11405	**11376**
105		1019331795	11259	11369	11712	11548	11374	**11365**
106		2056065863	11176	11292	11699	11546	11335	11330
107		1342855162	11360	11481	11874	11702	11438	**11398**
108		1325809384	11334	11442	11813	11675	11530	**11433**
109		1988803007	11192	11313	11725	11554	11439	11356
110		765656702	11284	11424	11780	11683	11499	**11446**
111	500 x 20	1368624604	26040	26228	26737	26656	26388	**26187**
112		450181436	26520	26688	27497	27153	26714	26799
113		1927888393	26371	26522	27277	26923	26648	**26496**
114		1759567256	26456	26586	27080	26894	25656	26612
115		606425239	26334	26541	26915	26768	26579	**26514**
116		19268348	26477	26582	27203	26965	26666	26661
117		1298201670	26389	26660	27057	26799	26594	**26529**
118		2041736264	26560	26711	27270	27066	26711	26750
119		379756761	26005	26148	26622	26488	26228	26223
120		28837162	26457	26611	27164	26923	26695	26619

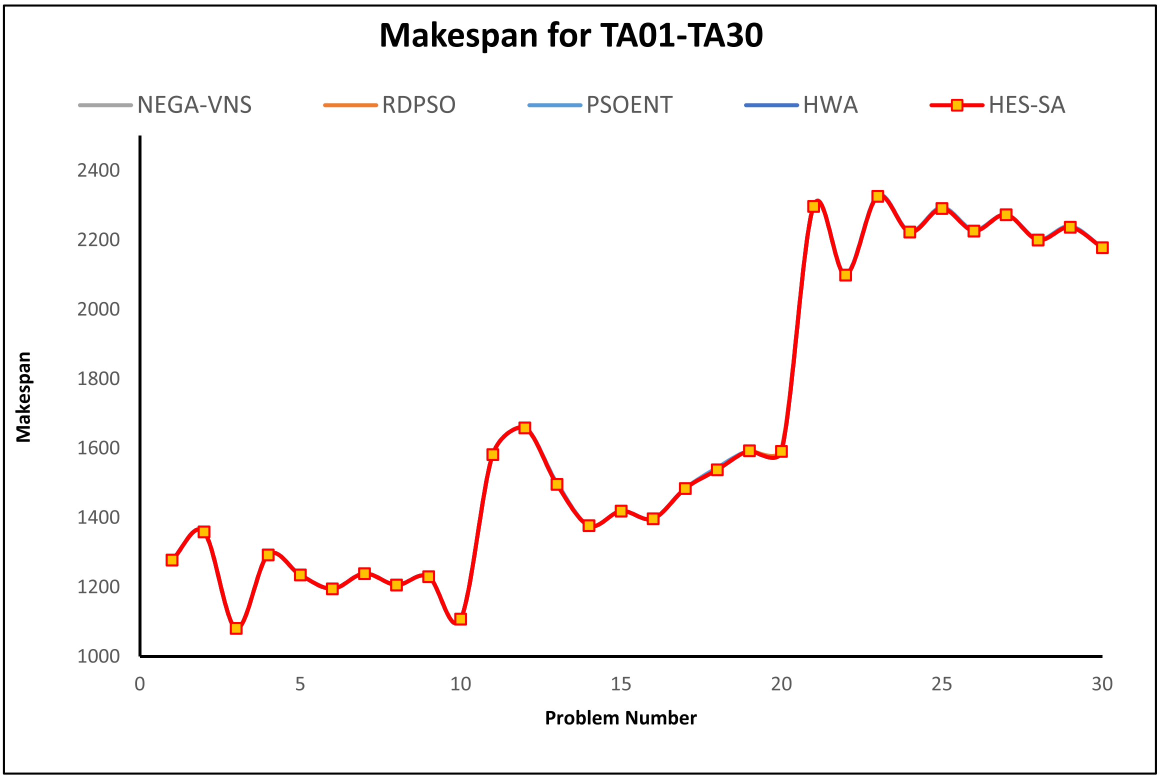


FIGURE 7. **Makespan for TA01-TA30.**

2nd termination criteria ($n^2/2 \times 10$ ms), all the twelve % Diff Cmax values are also negative. Hence by increasing the computational time, the % Diff Cmax values of the with-seed HES_{SA} algorithm are still better than the without-seed HES_{SA} algorithm as it explores more solution space. Hence, a with-seed HESSA algorithm should be used to start from a fixed starting point and then improve the solution, which helps the algorithm yield better results.

All the above results confirm that the proposed HES_{SA} has outperformed the algorithms of $NEGA_{VNS}$, PSOENT,

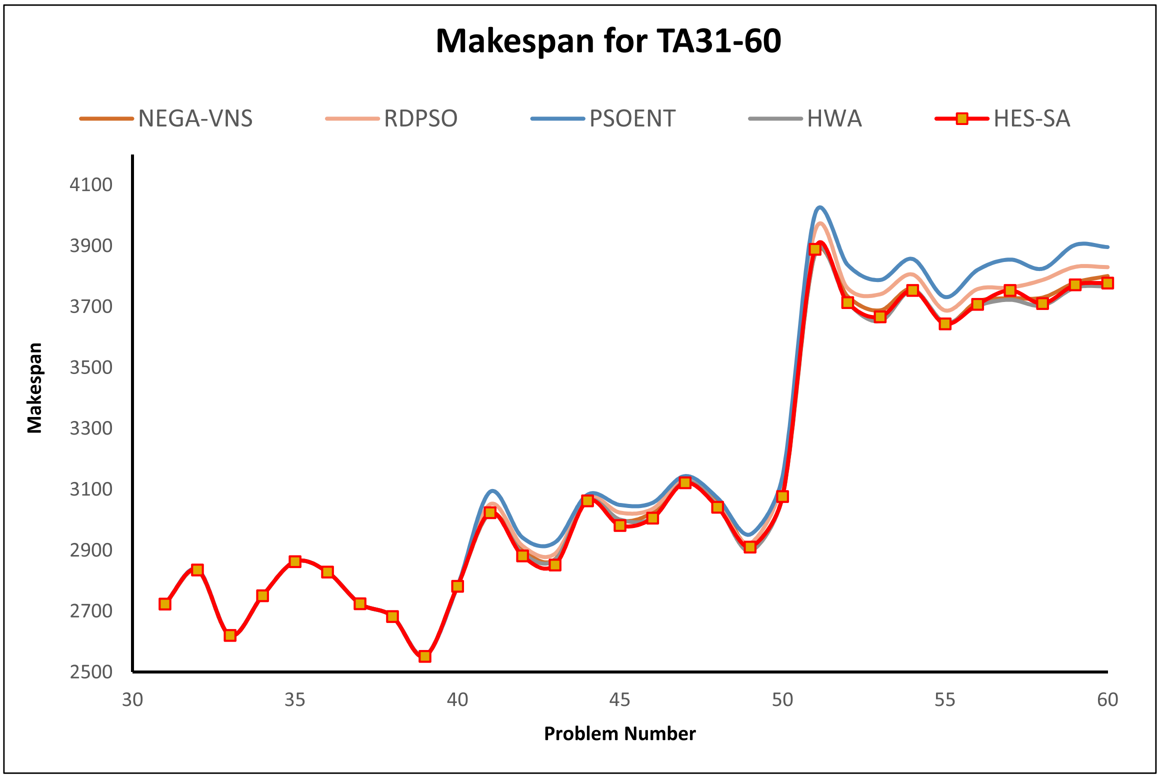


FIGURE 8. Makespan for TA31-TA60.

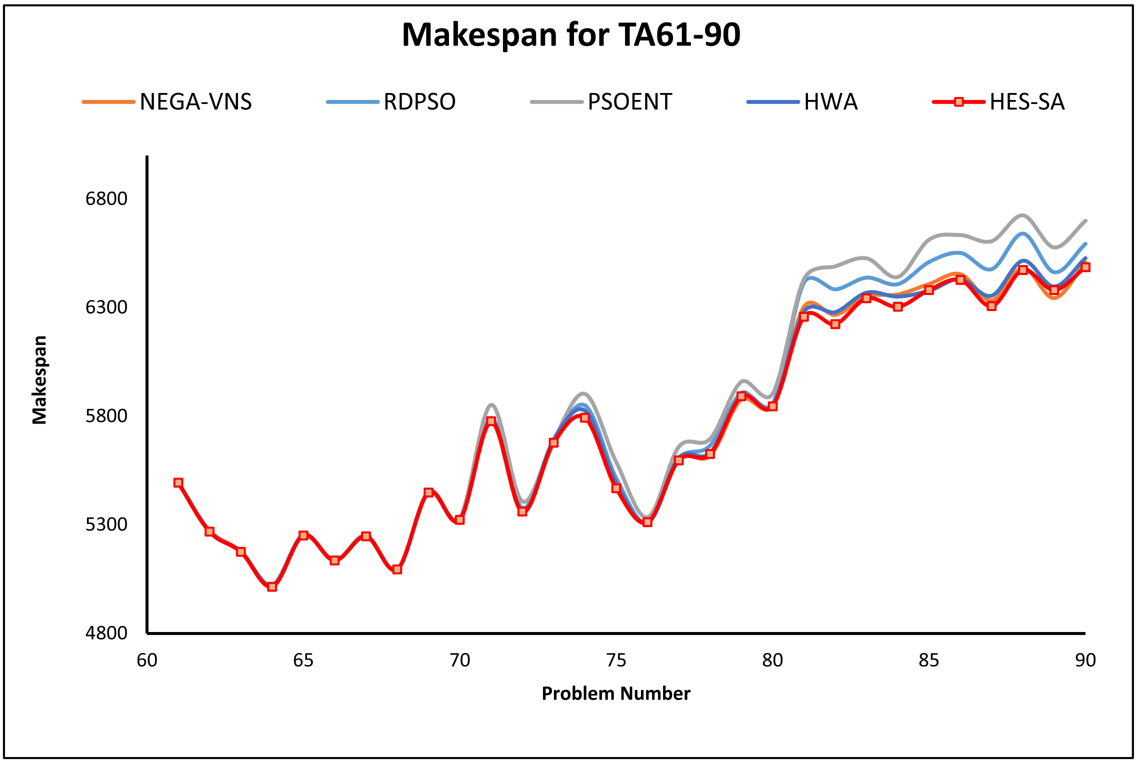


FIGURE 9. Makespan for TA61-TA90.

RDPSO, and H.W.A. in terms of makespan values. Also, with-seed HES_{SA} performs better than without-seed HES_{SA}; hence it is recommended to solve complex problems. HES_{SA} has been a robust technique as it has solved small, medium, and significant size problems, respectively. Since HES_{SA} has also proven its robustness and efficiency, it should be applied to real-life problems from industry to its effectiveness.

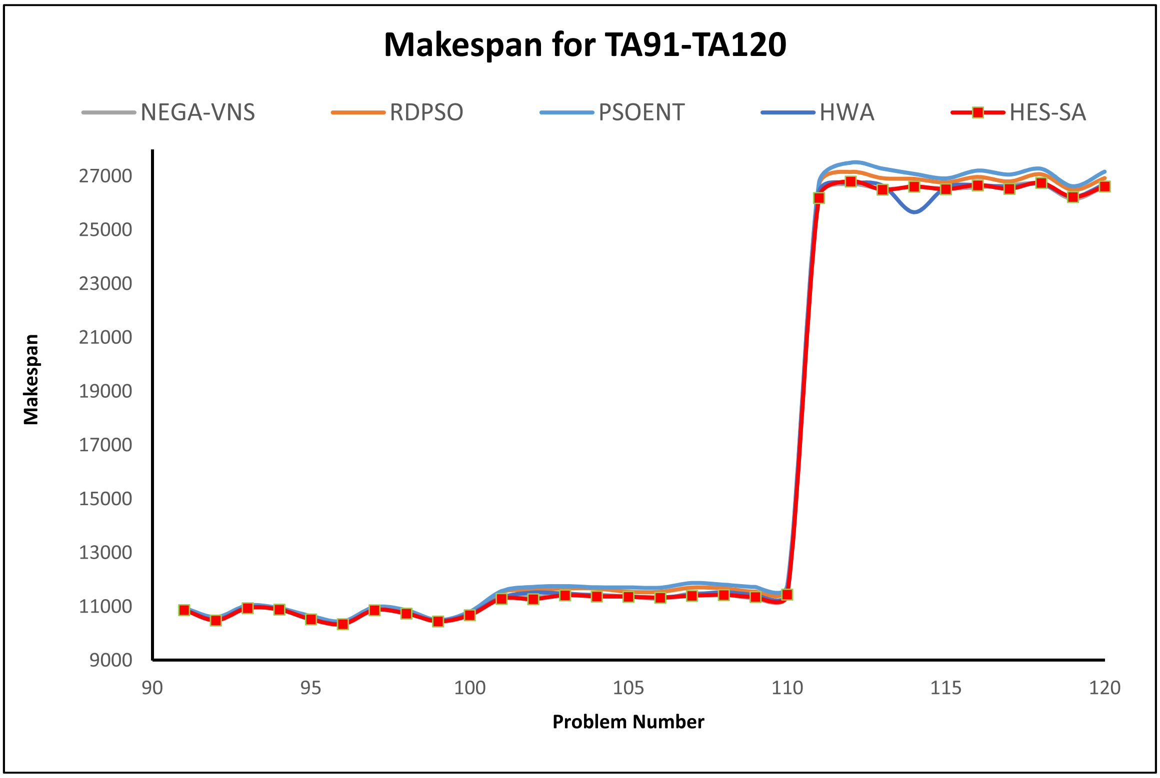


FIGURE 10. Makespan for TA91-TA120.

TABLE 5. Makespan and CPU time for different iterations using seed and without seed.

Algorithm	HES_{SA}			HES_{SA}		
Termination criteria	$n^2/2$ ms			$n^2/2 \times 10$ ms		
n x m	With-seed	Without-seed		With-seed	Without-seed	
	C^{seed}	$C^{w/seed}$	% Diff C_{max}	C^{seed}	$C^{w/seed}$	% Diff C_{max}
20 - 5	1278	1297	-1.49	1278	1297	-1.49
20 - 10	1628	1636	-0.49	1582	1602	-1.26
20 - 20	2341	2346	-0.21	2297	2321	-1.04
50 - 5	2752	2752	0.00	2724	2729	-0.18
50 - 10	3265	3286	-0.64	3024	3085	-2.02
50 - 20	4258	4260	-0.05	3889	4017	-3.29
100 - 5	5495	5529	-0.62	5493	5493	0.00
100 - 10	5925	5976	-0.86	5776	5824	-0.83
100 - 20	6589	6650	-0.93	6257	6538	-4.49
200 - 10	11022	11094	-0.65	10872	11056	-1.69
200 - 20	12113	12312	-1.64	11287	11807	-4.61
500 - 20	28457	28784	-1.15	26187	27092	-3.46

VI. CONCLUSION AND RECOMMENDATIONS

In this paper, Hybrid Evolution Strategy (HES_{SA}) is proposed to minimize makespan for PFSSP, and the results are validated on Taillard benchmark PFSSP. In HES_{SA}, an Improved Evolution Strategy is combined with simulated annealing to find optimal schedules for PFSSP, and the program was coded in MATLAB. To avoid trapping of I.E.S. local minima and fine-tune the results, it is hybridized with S.A. to improve the results further. The hybridization ensures that exploitation of solution space and exploration of neighbors can be carried out simultaneously. In I.E.S., double swap mutation is used to save computational time, and also, the mutation rate is varied to find better schedules. While in S.A., an insertion mutation is used, and the cooling parameter is gradually reduced for fine-tuning results. The results obtained from the proposed approach in terms of PRD and makespan values are compared with $NEGA_{VNS}$, PSOENT, RDPSO, and H.W.A. algorithms. Results suggest that the HES_{SA} algorithm is a robust technique and is equally applicable to small, medium, and significant size problems, as new upper bound are found in 54 instances, while improved makespan values are found for 38 instances.

Since HES_{SA} has been applied to the Scheduling problem for the first time, the following work can be performed in the future. For significant size problems (i.e., jobs ranging from 200 to 500 and machines up to 20), ample computational time is required to solve them; hence a quad swap mutation operator should reduce the computational time. Makespan minimization has been the performance measure in this research. Different performance measures can be implemented using HESSA, i.e., Tardiness, maximum utilization of the machine in future studies. By developing multi-objective HES_{SA}, this technique can be applied to multi-objective PFSSP's. Additionally, this technique should be applied to a real-life case from any industry to validate its practical implementation.

ACKNOWLEDGMENT

The authors would like to thank Eric Taillard for the flow shop benchmark problems and their solution files.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] M. Pinedo, *Scheduling*. Cham, Switzerland: Springer, 2015, doi: 10.1007/978-3-319-26580-3.
[2] I. M. Alharkan, "Algorithms for sequencing and scheduling," Dept. Ind. Eng., King Saud Univ., Riyadh, Saudi Arabia, Tech. Rep., 2005.
[3] J. Liu and C. R. Reeves, "Constructive and composite heuristic solutions to the $P//\Sigma C_i$ scheduling problem," *Eur. J. Oper. Res.*, vol. 132, no. 2, pp. 439–452, 2001, doi: 10.1016/S0377-2217(00)00137-5.
[4] M. M. Yenisey and B. Yagmahan, "Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends," *Omega*, vol. 45, pp. 119–135, Jun. 2014, doi: 10.1016/j.omega.2013.07.004.
[5] A. Allahverdi, "The third comprehensive survey on scheduling problems with setup times/costs," *Eur. J. Oper. Res.*, vol. 246, no. 2, pp. 345–378, Oct. 2015, doi: 10.1016/j.ejor.2015.04.004.
[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, doi: 10.1287/moor.1.2.117.
[7] 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, doi: 10.1287/opre.13.3.400.
[8] O. Moursli and Y. Pochet, "A branch-and-bound algorithm for the hybrid flowshop," *Int. J. Prod. Econ.*, vol. 64, no. 1, pp. 113–125, 2000, doi: 10.1016/S0925-5273(99)00051-1.
[9] C.-S. Chung and J. O. F. Kirca, "A branch and bound algorithm to minimize the total flow time for m-machine permutation flowshop problems," *Int. J. Prod. Econ.*, vol. 79, no. 3, pp. 185–196, 2002, doi: 10.1016/S0925-5273(02)00234-7.
[10] D. P. Ronconi, "A branch-and-bound algorithm to minimize the makespan in a flowshop with blocking," *Ann. Oper. Res.*, vol. 138, no. 1, pp. 53–65, Sep. 2005, doi: 10.1007/s10479-005-2444-3.
[11] C. T. Ng, J.-B. Wang, T. C. E. Cheng, and L. L. Liu, "A branch-and-bound algorithm for solving a two-machine flow shop problem with deteriorating jobs," *Comput. Oper. Res.*, vol. 37, no. 1, pp. 83–90, Jan. 2010, doi: 10.1016/j.cor.2009.03.019.
[12] A. Moukrim, D. Rebaine, and M. Serairi, "A branch and bound algorithm for the two-machine flowshop problem with unit-time operations and time delays," *RAIRO-Oper. Res.*, vol. 48, no. 2, pp. 235–254, Apr. 2014, doi: 10.1051/ro/2014004.
[13] M. S. Nagano, J. V. S. Robazzi, and C. P. Tomazella, "An improved lower bound for the blocking permutation flow shop with Total completion time criterion," *Comput. Ind. Eng.*, vol. 146, Aug. 2020, Art. no. 106511.
[14] M.-A. Isenberg and B. Scholz-Reiter, "The multiple batch processing machine problem with stage specific incompatible job families," in *Dynamics in Logistics*. Berlin, Germany: Springer, 2013, pp. 113–124, doi: 10.1007/978-3-642-35966-8_9.
[15] D. Rossit, F. Tohmé, M. Frutos, J. Bard, and D. Broz, "A non-permutation flowshop scheduling problem with lot streaming: A mathematical model," *Int. J. Ind. Eng. Comput.*, vol. 7, no. 3, pp. 507–516, 2016, doi: 10.5267/j.ijiec.2015.11.004.
[16] L. Meng, C. Zhang, X. Shao, B. Zhang, Y. Ren, and W. Lin, "More MILP models for hybrid flow shop scheduling problem and its extended problems," *Int. J. Prod. Res.*, vol. 58, no. 13, pp. 3905–3930, Jul. 2020.
[17] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*, vol. 24. Berlin, Germany: Springer, 2003, doi: 10.1016/j.dam.2004.09.004.
[18] L.-Y. Tseng and Y.-T. Lin, "A genetic local search algorithm for minimizing total flowtime in the permutation flowshop scheduling problem," *Int. J. Prod. Econ.*, vol. 127, no. 1, pp. 121–128, Sep. 2010, doi: 10.1016/j.ijpe.2010.05.003.
[19] D. S. Palmer, "Sequencing jobs through a multi-stage process in the minimum total time—A quick method of obtaining a near optimum," *J. Oper. Res. Soc.*, vol. 16, no. 1, pp. 101–107, Mar. 1965, doi: 10.1057/jors.1965.8.
[20] M. Nawaz, Jr., E. E. Enscore, Jr., 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.
[21] D. P. Ronconi, "A note on constructive heuristics for the flowshop problem with blocking," *Int. J. Prod. Econ.*, vol. 87, no. 1, pp. 39–48, 2004, doi: 10.1016/S0925-5273(03)00065-3.
[22] I. Ribas, R. Companys, and X. Tort-Martorell, "An iterated greedy algorithm for the flowshop scheduling problem with blocking," *Omega*, vol. 39, no. 3, pp. 293–301, Jun. 2011, doi: 10.1016/j.omega.2010.07.007.
[23] Z. Shao, W. Shao, and D. Pi, "Effective heuristics and metaheuristics for the distributed fuzzy blocking flow-shop scheduling problem," *Swarm Evol. Comput.*, vol. 59, Dec. 2020, Art. no. 100747.
[24] P. Kalczynski and J. Kamburowski, "On the NEH heuristic for minimizing the makespan in permutation flow shops," *Omega*, vol. 35, no. 1, pp. 53–60, Feb. 2007, doi: 10.1016/j.omega.2005.03.003.
[25] Q.-K. Pan and L. Wang, "Effective heuristics for the blocking flowshop scheduling problem with makespan minimization," *Omega*, vol. 40, no. 2, pp. 218–229, Apr. 2012, doi: 10.1016/j.omega.2011.06.002.
[26] A. J. Benavides and M. Ritt, "Two simple and effective heuristics for minimizing the makespan in non-permutation flow shops," *Comput. Oper. Res.*, vol. 66, pp. 160–169, Feb. 2016, doi: 10.1016/j.cor.2015.08.001.
[27] W. Shao and D. Pi, "A self-guided differential evolution with neighborhood search for permutation flow shop scheduling," *Expert Syst. Appl.*, vol. 51, pp. 161–176, Jun. 2016, doi: 10.1016/j.eswa.2015.12.001.
[28] S. Suliman, "A two-phase heuristic approach to the permutation flow-shop scheduling problem," *Int. J. Prod. Econ.*, vol. 64, nos. 1–3, pp. 143–152, 2000, doi: 10.1016/S0925-5273(99)00053-5.
[29] C.-L. Chen, Y.-R. Tzeng, and C.-L. Chen, "A new heuristic based on local best solution for permutation flow shop scheduling," *Appl. Soft Comput.*, vol. 29, pp. 75–81, Apr. 2015, doi: 10.1016/j.asoc.2014.12.011.
[30] H. Ye, W. Li, and A. Abedini, "An improved heuristic for no-wait flow shop to minimize makespan," *J. Manuf. Syst.*, vol. 44, pp. 273–279, Jul. 2017, doi: 10.1016/j.jmsy.2017.04.007.
[31] I. Ribas, R. Companys, and X. Tort-Martorell, "Efficient heuristics for the parallel blocking flow shop scheduling problem," *Expert Syst. Appl.*, vol. 74, pp. 41–54, May 2017, doi: 10.1016/j.eswa.2017.01.006.
[32] J. Lin, Z.-J. Wang, and X. Li, "A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem," *Swarm Evol. Comput.*, vol. 36, pp. 124–135, Oct. 2017, doi: 10.1016/j.swevo.2017.04.007.
[33] G. Deng, Z. Xu, and X. Gu, "A discrete artificial bee colony algorithm for minimizing the total flow time in the blocking flow shop scheduling," *Chin. J. Chem. Eng.*, vol. 20, no. 6, pp. 1067–1073, 2012, doi: 10.1016/S1004-9541(12)60588-6.

[34] Y.-Y. Han, D. Gong, and X. Sun, "A discrete artificial bee colony algorithm incorporating differential evolution for the flow-shop scheduling problem with blocking," *Eng. Optim.*, vol. 47, no. 7, pp. 927–946, Jul. 2015, doi: 10.1080/0305215X.2014.928817.

[35] J.-Q. Li and Q.-K. Pan, "Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm," *Inf. Sci.*, vol. 316, pp. 487–502, Sep. 2015, doi: 10.1016/j.ins.2014.10.009.

[36] H. Xuan, H. Zhang, and B. Li, "An improved discrete artificial bee colony algorithm for flexible flowshop scheduling with step deteriorating jobs and sequence-dependent setup times," *Math. Problems Eng.*, vol. 2019, pp. 1–13, Dec. 2019.

[37] Y. Liu, M. Yin, and W. Gu, "An effective differential evolution algorithm for permutation flow shop scheduling problem," *Appl. Math. Comput.*, vol. 248, pp. 143–159, Dec. 2014, doi: 10.1016/j.amc.2014.09.010.

[38] B. Qian, L. Wang, R. Hu, W.-L. Wang, D.-X. Huang, and X. Wang, "A hybrid differential evolution method for permutation flow-shop scheduling," *Int. J. Adv. Manuf. Technol.*, vol. 38, nos. 7–8, pp. 757–777, Sep. 2008, doi: 10.1007/s00170-007-1115-8.

[39] T.-S. Yeh and T.-C. Chiang, "An improved multiobjective evolutionary algorithm for solving the no-wait flow shop scheduling problem," in *Proc. IEEE Int. Conf. Ind. Eng. Eng. Manage. (IEEM)*, Dec. 2018, pp. 142–147, doi: 10.1109/IEEM.2018.8607486.

[40] V. Caraffa, "Minimizing makespan in a blocking flowshop using genetic algorithms," *Int. J. Prod. Econ.*, vol. 70, no. 2, pp. 101–115, 2001, doi: 10.1016/S0925-5273(99)00104-8.

[41] 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, Oct. 2006, doi: 10.1016/j.omega.2004.12.006.

[42] E. Vallada and R. Ruiz, "Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem," *Omega*, vol. 38, nos. 1–2, pp. 57–67, Feb. 2010, doi: 10.1016/j.omega.2009.04.002.

[43] M. Akhshabi, J. Haddadnia, and M. Akhshabi, "Solving flow shop scheduling problem using a parallel genetic algorithm," *Procedia Technol.*, vol. 1, pp. 351–355, Jan. 2012, doi: 10.1016/j.protcy.2012.02.073.

[44] C. E. Andrade, T. Silva, and L. S. Pessoa, "Minimizing flowtime in a flowshop scheduling problem with a biased random-key genetic algorithm," *Expert Syst. Appl.*, vol. 128, pp. 67–80, Aug. 2019, doi: 10.1016/j.eswa.2019.03.007.

[45] L. Wang, Q.-K. Pan, P. N. Suganthan, W.-H. Wang, and Y.-M. Wang, "A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems," *Comput. Oper. Res.*, vol. 37, no. 3, pp. 509–520, Mar. 2010, doi: 10.1016/j.cor.2008.12.004.

[46] D. Laha and U. K. Chakraborty, "Minimising total flow time in permutation flow shop scheduling using a simulated annealing-based approach," *Int. J. Automat. Control*, vol. 4, no. 4, pp. 359–379, 2010, doi: 10.1504/IJAAC.2010.035525.

[47] S.-W. Lin and K.-C. Ying, "Minimizing makespan and total flowtime in permutation flowshops by a bi-objective multi-start simulated-annealing algorithm," *Comput. Oper. Res.*, vol. 40, no. 6, pp. 1625–1647, Jun. 2013, doi: 10.1016/j.cor.2011.08.009.

[48] G. Moslehi and D. Khorasanian, "A hybrid variable neighborhood search algorithm for solving the limited-buffer permutation flow shop scheduling problem with the makespan criterion," *Comput. Oper. Res.*, vol. 52, pp. 260–268, Dec. 2014, doi: 10.1016/j.cor.2013.09.014.

[49] S.-W. Lin, C.-Y. Cheng, P. Pourhejazy, and K.-C. Ying, "Multi-temperature simulated annealing for optimizing mixed-blocking permutation flowshop scheduling problems," *Expert Syst. Appl.*, vol. 165, Mar. 2021, Art. no. 113837.

[50] 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.

[51] 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.

[52] J. Grabowski and J. Pempera, "The permutation flow shop problem with blocking. A tabu search approach," *Omega*, vol. 35, no. 3, pp. 302–311, Jun. 2007, doi: 10.1016/j.omega.2005.07.004.

[53] O. A. Arık, "Population-based tabu search with evolutionary strategies for permutation flow shop scheduling problems under effects of position-dependent learning and linear deterioration," *Soft Comput.*, vol. 25, pp. 1–18, Aug. 2020, doi: 10.1007/s00500-020-05234-7.

[54] 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.

[55] F. Zhao, S. Qin, G. Yang, W. Ma, C. Zhang, and H. Song, "A factorial based particle swarm optimization with a population adaptation mechanism for the no-wait flow shop scheduling problem with the makespan objective," *Expert Syst. Appl.*, vol. 126, pp. 41–53, Jul. 2019, doi: 10.1016/j.eswa.2019.01.084.

[56] E. C. de Siqueira, M. J. F. Souza, S. R. de Souza, M. F. de França Filho, and C. G. Marcelino, "An algorithm based on evolution strategies for makespan minimization in hybrid flexible flowshop scheduling problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2013, pp. 989–996, doi: 10.1109/CEC.2013.6557675.

[57] B. Khurshid, S. Maqsood, M. Omair, R. Nawaz, and R. Akhtar, "Hybrid evolution strategy approach for robust permutation flowshop scheduling," *Adv. Prod. Eng. Manage.*, vol. 15, no. 2, pp. 204–216, Jun. 2020, doi: 10.14743/apem2020.2.359.

[58] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[59] Z. Michalewicz, "Evolution strategies and other methods," in *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Germany: Springer, 1996, pp. 159–177, doi: 10.1007/978-3-662-03315-9.

[60] I. Rechenberg. (1970). *Optimierung Technischer Systeme Nach Prinzipien der Biologischen Evolution*. [Online]. Available: https://www.jstor.org/stable/23679080

[61] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies—A comprehensive introduction," *Natural Comput.*, vol. 1, no. 1, pp. 3–52, 2002, doi: 10.1023/A:1015059928466.

[62] P. C. D. Paris, E. C. Pedrino, and M. C. Nicoletti, "Automatic learning of image filters using Cartesian genetic programming," *Integr. Comput.-Aided Eng.*, vol. 22, no. 2, pp. 135–151, Feb. 2015, doi: 10.3233/ICA-150482.

[63] B. Khurshid, S. Maqsood, M. Omair, B. Sarkar, M. Saad, and U. Asad, "Fast evolutionary algorithm for flow shop scheduling problems," *IEEE Access*, vol. 9, pp. 44825–44839, 2021, doi: 10.1109/ACCESS.2021.3066446.

[64] J. F. Miller, P. Thomson, and T. C. Fogarty, "Designing electronic circuits using evolutionary algorithms. Arithmetic circuits: A case study," in *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*. Hoboken, NJ, USA: Wiley, 1997.

[65] M. Rehman, G. M. Khan, and S. A. Mahmud, "Foreign currency exchange rates prediction using CGP and recurrent neural network," *IERI Procedia*, vol. 10, pp. 239–244, Jan. 2014, doi: 10.1016/j.ieri.2014.09.083.

[66] M. M. Khan, A. M. Ahmad, G. M. Khan, and J. F. Miller, "Fast learning neural networks using Cartesian genetic programming," *Neurocomputing*, vol. 121, pp. 274–289, Dec. 2013, doi: 10.1016/j.neucom.2013.04.005.

[67] L. Costa and P. Oliveira, "Evolutionary algorithms approach to the solution of mixed integer non-linear programming problems," *Comput. Chem. Eng.*, vol. 25, nos. 2–3, pp. 257–266, 2001, doi: 10.1016/S0098-1354(00)00653-0.

[68] F. D. Croce, M. Ghirardi, and R. Tadei, "An improved branch-and-bound algorithm for the two machine total completion time flow shop problem," *Eur. J. Oper. Res.*, vol. 139, no. 2, pp. 293–301, 2002, doi: 10.1016/S0377-2217(01)00374-5.

[69] G. I. Zobolas, C. D. Tarantilis, and G. Ioannou, "Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm," *Comput. Oper. Res.*, vol. 36, no. 4, pp. 1249–1267, Apr. 2009, doi: 10.1016/j.cor.2008.01.007.

[70] H.-P. Schwefel, "Evolutionsstrategie und numerische optimierung," Technische Univ. Berlin, Berlin, Germany, Tech. Rep., 1975.

[71] E. Taillard, "Benchmarks for basic scheduling problems," *Eur. J. Oper. Res.*, vol. 64, no. 2, pp. 278–285, 1993, doi: 10.1016/0377-2217(93)90182-M.

[72] G. I. Zobolas, C. D. Tarantilis, and G. Ioannou, "Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm," *Comput. Oper. Res.*, vol. 36, no. 4, pp. 1249–1267, Apr. 2009.

[73] C.-L. Chen, S.-Y. Huang, Y.-R. Tzeng, and C.-L. Chen, "A revised discrete particle swarm optimization algorithm for permutation flow-shop scheduling problem," *Soft Comput.*, vol. 18, no. 11, pp. 2271–2282, Nov. 2014.

[74] Y. Marinakis and M. Marinaki, "Particle swarm optimization with expanding neighborhood topology for the permutation flowshop scheduling problem," *Soft Comput.*, vol. 17, no. 7, pp. 1159–1173, Jul. 2013, doi: 10.1007/s00500-013-0992-z.

[75] J. Carlier, "Ordonnancements a contraintes disjonctives," *RAIRO-Oper. Res.*, vol. 12, no. 4, pp. 333–350, 1978. [Online]. Available: http://www.numdam.org/item?id=RO_1978__12_4_333_0

[76] C. R. Reeves, "A genetic algorithm for flowshop sequencing," *Comput. Oper. Res.*, vol. 22, no. 1, pp. 5–13, Jan. 1995, doi: 10.1016/0305-0548(93)E0014-K.

[77] J. Heller, "Some numerical experiments for an M×J flow shop and its decision-theoretical aspects," *Oper. Res.*, vol. 8, no. 2, pp. 178–184, Apr. 1960.

BILAL KHURSHID was born in Peshawar, Pakistan, in 1983. He received the B.Sc. and M.Sc. degrees in mechanical engineering from the University of Engineering and Technology, Peshawar, in 2007 and 2012, respectively, where he is currently pursuing the Ph.D. degree with the Department of Industrial Engineering.

He has published several research articles in peer reviewed journals, such as *Advances in Production Engineering & Management* and IEEE Access. His current research interests include artificial intelligence, engineering optimization, evolutionary computation, and scheduling.

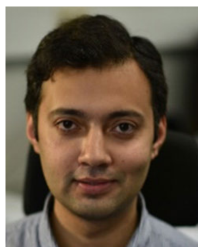
SHAHID MAQSOOD received the B.Sc. degree in mechanical engineering from the University of Engineering and Technology, Peshawar, in 1999, the M.S. degree in mechanical engineering from the Ghulam Ishaq Khan Institute of Science and Technology, Swabi, in 2008, and the Ph.D. degree in mechanical engineering from the University of Bradford, U.K., in 2012.

He has been a Professor with the Department of Industrial Engineering, University of Engineering and Technology, Jalozai Campus, since 2019. He has authored more than 50 research articles, some of which are published in international journals, such as *Advances in Production Engineering & Management*, *International Journal of Intelligent Systems Technologies and Applications*, *Mathematics*, *The International Journal of Advanced Manufacturing Technology*, *Advances in Mechanical Engineering*, *Journal of Ergonomics*, IEEE Access, and *International Journal of Progressive Sciences and Technologies*. His main research interests are manufacturing systems, scheduling, artificial intelligence, finite element analysis, and supply chain management.

MUHAMMAD OMAIR received the B.S. and M.S. degrees in industrial engineering from the University of Engineering and Technology, Peshawar, Pakistan, in 2010 and 2013, respectively, and the Ph.D. degree in industrial engineering from Hanyang University, South Korea, in 2019.

He has been an Assistant Professor with the Department of Industrial Engineering, University of Engineering and Technology, Jalozai Campus, since 2019. He has authored more than 20 research articles, some of which are published in international journals, such as *Applied Soft Computing Journal* of cleaner production, *Advances in Production Engineering & Management*, *RAIRO-Operations Research*, *Mathematics*, IEEE Access, *The International Journal of Advanced Manufacturing Technology*, and *International Journal of Environmental Research and Public Health*. His main research interests are supply chain management, inventory management, sustainability, manufacturing systems, scheduling, and mathematical modeling.

BISWAJIT SARKAR received the bachelor's and master's degrees in applied mathematics from Jadavpur University, Kolkata, India, in 2002 and 2004, respectively, the Master of Philosophy degree in the application of boolean polynomials from Annamalai University, Chidambaram, India, in 2008, and the Doctor of Philosophy degree in operations research from Jadavpur University, in 2010. He held a postdoctoral position with the Pusan National University, South Korea, from 2012 to 2013. He has dedicated his teaching and research abilities to various universities, including Hanyang University, South Korea (2014–2019), Vidyasagar University, India (2010–2014), and Darjeeling Government College, India (2009–2010). Under his supervision, 19 students have been awarded their Ph.D. degrees, and three students are awarded their master's degrees. He is currently an Associate Professor in industrial engineering with Yonsei University, South Korea. Since 2010, he has been published 218 journal articles in reputed journals of *Applied Mathematics and Industrial Engineering* and he has published three books. He is the Editorial Board Member of some reputed international journals of *Applied Mathematics* and *Industrial Engineering*. He is a member of several learned societies. In 2014, his article was selected as the best research paper at an international conference in South Korea. He has presented several research papers in international conferences as an invited speaker and chaired several sessions in several international conferences. He has received a Bronze Medal for his capstone achievement from Hanyang University, in 2016. He was a recipient of the Bharat Vikash Award as a Young Scientist from India, in 2016. He was also a recipient of the Hanyang University Academic Award as one of the most productive researchers, in 2017 and 2018. He has received an International Award from the Korean Institute of Industrial Engineers at KAIST, Daejon, South Korea, in 2017. He is the Topic Editor of the SCIE indexed journal *Energies*. He has served as the Guest Editor for five special issues of some SCIE indexed journals. He has been the best active author in the topic cluster of *Supply Chain Management and Industry* as SciVal (Scopus), since 2017. His SCI/SCIE/SSCI article publication average for the last three years is 31 articles per year.



IMRAN AHMAD received the B.S. and M.S. degrees in industrial engineering from the University of Engineering and Technology, Peshawar, Pakistan, in 2008 and 2012, respectively, and the Ph.D. degree in industrial management engineering from Hanyang University, South Korea, in 2019.

He has been working as an Assistant Professor with the Department of Industrial Engineering, University of Engineering and Technology, since 2019. His research publications have been published in some of the international journals, such as *International Journal of Environmental Research and Public Health*, IEEE Access, *Engineering Science and Technology*, *International Journal*, *Metals*, and *Congress of International Ergonomics Association*. His main research interests are manufacturing systems simulation, and human ergonomics.

KHAN MUHAMMAD received the B.Sc. degree in mining engineering from the University of Engineering and Technology, Peshawar, Pakistan, in 1999, and the Ph.D. degree in earth sciences from the Camborne School of Mines, University of Exeter, U.K., in 2009.

Later, he pursued further research and development work in mining and earth sciences as an Assistant Professor with major focus on geostatistics, artificial intelligence, operations research, fuzzy logic, neural networks, and computer application softwares with the University of Engineering and Technology. He has authored more than ten research article, some of which are published in international journals, such as *Geostandards and Geoanalytical Research*, *Computers and Concrete*, *Archives of Mining Sciences*, *Applied Artificial Intelligence*, *ISPRS International Journal of Geo-Information*, *Applied Sciences*, *Resources Policy*, and *Intelligent Data Analysis and Minerals*.