IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 16, 2021, accepted March 8, 2021, date of publication March 17, 2021, date of current version March 26, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3066446

Fast Evolutionary Algorithm for Flow Shop Scheduling Problems

BILAL KHURSHID[1], SHAHID MAQSOOD[2], MUHAMMAD OMAIR[2], BISWAJIT SARKAR[3], MUHAMMAD SAAD[4], AND UZAIR ASAD[5]

[1]Department of Industrial Engineering, University of Engineering and Technology, Peshawar 25000, Pakistan
[2]Department of Industrial Engineering, University of Engineering and Technology, Jalozai Campus, Peshawar 25000, Pakistan
[3]Department of Industrial Engineering, Yonsei University, Seoul 03722, South Korea
[4]Department of Mechanical Engineering, Institute of Engineering and Fertilizer Research, Faisalabad 38090, Pakistan
[5]Department of Chemical Engineering, National University of Science and Technology, Islamabad 44000, Pakistan

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

ABSTRACT Being complex and combinatorial optimization problems, Permutation Flow Shop Scheduling Problems (PFSSP) are difficult to be solved optimally. PFSSP occurs in many manufacturing systems i.e. automobile industry, glass industry, paper industry, appliances industry, and pharmaceutical industry, and the generation of the best schedule is very important for these manufacturing systems. Evolution Strategy (ES) is a subclass of Evolutionary algorithms and in this paper, we propose an Improved Evolution Strategy to reduce the makespan of PFSSP. Two variants of the Improved Evolution Strategy are proposed namely ES5 and ES10. The initial solution is generated using the shortest processing time rule. In ES5, four offsprings are generated from one parent while in ES10, nine offsprings are generated from one parent. The selection pool consists of both the parents and offsprings. Quad swap mutation operator has been proposed to minimize computational time and for the maximum search of solution space. Also, a variable mutation rate is used for the fine-tuning of results, with the increasing number of iterations the mutation rate is reduced. The performances of both ES variants were tested on two test domains. First, it is applied to benchmark the PFSSP of Carlier and Reeves. Computational results are matched with other well-known techniques available in the literature, and the results show the effectiveness and robustness of the proposed techniques. Secondly, ES is applied to the real-life problem for the manufacturing of batteries to demonstrate its effectiveness. Data was taken from Pakistan Accumulator for NS30-40 Plates battery, the company is daily producing 1400 units of NS30-40 Plates battery. ES is applied to different batch sizes i.e. 35, 140, 1120 & 1400. Our results show that a Min %GAP of 1.25 is found using ES10. Hence the company can increase monthly 450 units of NS30 batteries using the ES10 algorithm.

INDEX TERMS Permutation flow shop scheduling, evolution strategy, Carlier problem, makespan, Reeves problem.

I. INTRODUCTION

Scheduling is the key factor for maintaining a competitive environment in manufacturing systems and production planning [1]. For large manufacturing systems, small improvements such as saving processing times and improving production efficiency can results in significant profit for the company. PFSSP is the key part of scheduling in manufacturing systems for one piece of mass production. Qian *et al.* [2] demonstrated that makespan minimization for PFSSP is an NP-hard problem. This NP-hard distinctive and its extensive application in the engineering field makes it a hot topic in the engineering and research field.

The allocation of resources (e.g. Machines) to tasks (e.g. Jobs) over a period of time is termed as scheduling and is used to optimize one or more objectives [3] and provides a pivoting role for enhancing production performance in manufacturing systems, hence the development of efficient and robust technique is mandatory for manufacturing systems. PFSSP is a highly focused research area in the scheduling field, and

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

it signifies almost a quarter of problems of assembly lines, manufacturing systems, and information service facilities [2]. Minimization of makespan is a highly focused area for flow shop scheduling [3]. Recently numerous researchers have worked on PFSSP to minimize makespan [4]. The Objective of the proposed research is makespan minimization for PFSSP. An Improved Evolution Strategy with two variants namely ES5 and ES10 are proposed and their performance is tested on a standard set of Carlier and Reeves problems. In ES5, four offsprings are generated from a single parent while in ES10, nine offsprings are generated from a single parent. The ES10 algorithm is also tested on a real-life case from industry i.e. manufacturing of batteries. The Improved Evolution Strategy code is easily implementable to other scheduling problems.

The rest of the paper is organized as follows. The literature review is explained briefly in Section 2. Section 3 & 4 explain PFSSP and problem definition. Section 5 and 6 explain the methodology and numerical experiments. In section 7, ES has been applied to the real-life problem of battery manufacturing while the conclusion and future recommendations are suggested in Section 8.

II. LITERATURE REVIEW

Due to the practical implementation of PFSSP initially proposed by Johnson [5], many researchers have focused on PFSSP and have proposed several techniques which may be categorized as Exact Methods, Constructive Heuristics, and Meta-Heuristic Algorithms. Exact Methods i.e. Branch & Bound, Lagrangian Relaxation Algorithm, and Dynamic programming apply to small size problems due to the complexity of PFSSP's. As problem size increases, storage of algorithm raise with the rise of problem size, and hence computational time rises abruptly. According to Ruiz *et al.* [6], exact methods can be used only when the number of jobs is less than 20.

Constructive Heuristic starts from scratch and builds a feasible schedule for two and three machines. Although constructive heuristic is fast but its quality is not satisfactory [2]. Experimental studies [4] have proved that the NEH Heuristic proposed by Nawaz *et al.* [7] is the best constructive heuristic, it assigns priorities on each job based on total processing time and then jobs are inserted successively in the sequence to develop a complete schedule. Constructing priority sequence and job insertion are the two major steps of the NEH Heuristic. Several constructive Heuristics have been developed by researchers to solve flow shop problems. Constructive heuristics use local information to design scheduling rules. Constructive heuristics are fast but their final solution is local optimal instead of global optimal.

Over the years a lot of researchers have focused on PFSSP with improved meta-heuristic or innovative hybrid meta-heuristic due to the constraints of exact methods and constructive heuristics. With a reasonable computational time, meta-heuristic provide high-quality solutions. Using a large number of iterations and controlling operational parameters of the algorithm, meta-heuristics provide much-improved results as compared to the other two categories.

Some of the famous meta-heuristics are Evolutionary Algorithms (EA) comprising Genetic Algorithms (GA), Evolution Strategy (ES) and Evolutionary Programming (EP), Ant Colony Algorithm (ACO), Simulated Annealing (SA), Iterated Greedy Algorithm (IGA), Estimation of Distribution Algorithms (EDA), Tabu Search (TS), and Particle Swarm Optimization (PSO). Since each metaheuristic technique has its advantage, by combining the best features of various techniques in a correlative way, the result is optimized and robust.

For PFSSP, Ruiz and Stützle [8] Suggested an Iterated greedy algorithm (IGA) to minimize makespan and found six upper bounds for Taillard benchmark flow shop problems. The IGA algorithm works in two phases, in the destruction phase, some jobs are removed. While in the construction phase, removed jobs are re-inserted into all possible positions so that the makespan is minimized Andrade *et al.* [9] proposed a Biased Random Key GA featuring a shaking strategy for escaping local minima. In which, the Individuals from the elite set are disturbed while the remaining population is reset. In 2018, Wei *et al.* [10] Proposed a Hybrid Genetic Simulated Annealing (HGSA) Algorithm. The initial solution was generated by combining NEH Heuristic, MinMax, and MME Algorithm, while the solution was optimized by using hormone regulation scheme in the SA algorithm. Inspired by the Shallow-water wave theory, Zhao *et al.* [11] a Discrete water wave optimization (DWWO) algorithm having refraction, propagation, and breaking operators to minimize makespan. A hybrid GA was proposed by Zheng and Wang [12] in which SA replaced mutation and multiple crossover operators were applied to subpopulations. A new hybrid SA algorithm was presented by Nearchou [13], who combined features of GA and local search techniques. By implementing small perturbation schemes, the algorithm generated new neighborhood schedules, and an iterated hill-climbing procedure was used in the annealing process for improving the performance.

GA was hybridized with Variable Neighborhood Search (VNS) by Jarboui *et al.* [14] to minimize total flow time and makespan. The VNS is used as an improvement method in the last step of GA. The proposed technique outperformed the results of Grabowski. Zhang *et al.* [15] proposed an EDA, by integrating the longest common subsequence in the probability distribution model and a screening seed better individuals were generated. The EDA was integrated with VNS to improve the results Engin and Güçlü [16] proposed a Hybrid Ant Colony Optimization (HACO) Algorithm using mutation and crossover operators for minimizing the makespan. Zhang, *et al.* [17] Combined ACO with Cuckoo Search (CS) to minimize the makespan of PFSSP. The initial solution is generated using CS, Solution generated using ACO is compared with Levy Flight and is replaced with it once it is superseded. Then according to abandonment probability, the bird's nest position is changed using the CS algorithm. Ahmadizar [18] Proposed a new ACO algorithm named

NACA, with negligible computational time first, a solution is generated, and then based on the solution trail intensities are generated. The solution is optimized by job index local search procedure combined with a threshold probability for inserting jobs at other locations of the sequence. A hybrid discrete ABC algorithm was suggested by Liu and Liu [19], in which each solution is called a food source and is exemplified as a discrete job permutation. Initially, a population-based NEH Heuristic was generated from a greedy randomized adaptive search procedure. Secondly, discrete operators are used to generating new solutions.

A parallel TS was suggested by Bożejko *et al.* [20] for hybrid FSSP's. The two methods suggested are the parallel calculation of the function and the local neighborhood search inserted in TS. Li and Yin [1] suggested a hybrid cuckoo search (HCS) algorithm, to change the position in the cuckoo search into discrete job permutation a largest ranked value-based random key is applied, and for population initialization, the NEH heuristic is combined with random initialization. By combining different neighborhood topologies with particle swarm optimization. A new nature-inspired approach was proposed by Marinakis and Marinaki [21] to minimize the makespan of PFSSP. Boumediene *et al.* [22] proposed a new hybrid GA to reduce makespan, GA was combined with an ACO based on pachycondyla apiaclis behavior to search prey.

GA, TS, and SA are the most famous meta-heuristic used for PFSSP. GA is fast, versatile, easily implementable, and has strong global optimization adaptability. However, they lack local search adaptability which reduces their searchability. In GA, selection and recombination operators are the main sources for search space; hence the GA very slowly converges to local optima [14]. SA has strong local searchability, and it avoids trapping in local search minima, unlike GA. TS is a trajectory method, and it proceeds aggressively to the local optimum as compared to SA and is best suited for fine-tuning of scheduling problems. ES is a subclass of Evolutionary Algorithms (EA) and is a randomized search technique suitable for combinatorial optimization problems. ES was developed for numerical optimization problems and is regarded for its efficiency and robustness. The main source of genetic variation in ES is a mutation, while reproduction and selection are background operators. In reproduction, offsprings (denoted as λ) are randomly generated from a single parent (denoted as μ) de Siqueira *et al.* [23] Applied ES for minimizing the makespan of PFSSP using $\lambda = 1$. Ahmad *et al.* [24] Used $\lambda = 4$ to classify mammograms using Cartesian Genetic Programming Evolved Artificial Neural Network (CGPANN) based on ES. For the grouping of arrhythmia types, Ahmad*et al.* [24] used CGPANN based on ES and used $\lambda = 9$. A computational model for image filters based on CGP was proposed by Paris *et al.* [25] using $\lambda = 16$. Limited researchers have used ES for optimization of FSSP's, while the application of ES on benchmark flow shop problems of Carlier, Reeves, and Taillard is still imminent. For Robust Permutation Flowshop problems, Khurshid *et al.* [26] used Hybrid Evolution Strategy and found better schedules as compared to other robust schedules available in the literature.

From the above-mentioned papers, it is obvious that different values of λ are used in literature i.e. 1, 4, 9, and 16. For $\lambda = 16$, from one parent sixteen offsprings are generated however it requires ample computational time. Hence in this research $\lambda = 4$ and $\lambda = 9$ will be used so that solution space can be thoroughly exploited and also computational time can be saved as compared to $\lambda = 16$. To exploit more solution space in less computational time, a quad swap mutation will be used instead of a single or double swap mutation operator. Moreover in ES, the variable mutation rate is used, unlike GA where a fixed mutation rate is used. Hence in ES, the mutation rate can gradually be reduced for fine-tuning of the results. **Table 1** shows the comparison of ES with other techniques available in the literature.

III. PROBLEM STATEMENT

In PFSSP the processing sequence of all the jobs is the same. Sequence change is not allowed in PFSSP, and the sequence assigned to the first machine remains the same till the last machine. The PFSSP scheduling orders n Jobs on m machines and the processing sequence of all jobs on each machine is the same. The processing time of Job J_i on machine M_j is given as P_{ij}. It is supposed that Zero time corresponds to a job and it has been executed on the machine in insignificant time. The objective is to minimize the total processing time also termed as makespan (C_{max}). Processing times are non-negative fixed values and their values are already known.

Following are a few assumptions of Permutation flow-shop problems:

- Each job has the same processing route on all machines, and its value is known.
- Each machine can execute only one job at a time.
- Each job can be started first as all jobs are independent.
- At time zero, all machines are available to process any job.
- Operation times of the job include the setup times.
- The machining process once started cannot be interrupted, and the processing time of every job on each machine is known.
- There is unlimited storage available between two successive machines.
- Machine breakdowns are not allowed.

The most common objective in flow shop scheduling problems is the minimization of makespan [8]. Minimization of makespan increases utilization of machines and reduces processing costs and is a useful criterion for large-scale machine shops. Carlier [31] proved that PFSSP are NP-hard problems and they are difficult to be solved especially large size problems, hence the solution for PFSSP is either constructive heuristic or metaheuristic. Due to two reasons, minimization of makespan is the most focused research area in PFSSP, first, it is a simple criterion for long-term utilization of machines and secondly, its analytical results are available.

TABLE 1. Comparison of ES with other techniques in literature.

Author	Technique	PFSSP	Problem Set	
			Carlier	Reeves
Ruiz and Stützle [8]	IGA	✓		
Andrade, Silva and Pessoa [9]	GA			
Wei, Li, Jiang, Hu and Hu [10]	HGSA			
Engin and Güçlü [16]	HACO			
Zhang, Yu, Zhang, Luo and Zhang [17]	ACOCS	✓	✓	
Zheng and Wang [12]	HGA		✓	✓
Jarboui, Eddaly and Siarry [14]	HGA		✓	
Ruiz, Maroto and Alcaraz [6]	GA	✓		
Reeves [27]	GA	✓		
Boumediene, Houbad, Hassam and Ghomri [22]	HGA	✓		
Nearchou [13]	HSA			
Li and Yin [1]	HCS	✓	✓	✓
Akhshabi, *et al.* [28]	GA	✓		
Pan and Huang [29]	GA			
Shao and Pi [30]	DE	✓	✓	✓
de Siqueira, Souza, de Souza, de França Filho	ES	✓		
Proposed Research	ES	✓	✓	✓

The objective of this research is to find the best sequence for processing jobs on machines so that the makespan is minimized. The makespan for job J_k at a machine i can be calculated through a set of recursive equations as under:

$$C_{i,j_1} = \sum_{l=1}^{i} P_l, J_1 \quad i = 1, \ldots m \tag{1}$$

$$C_{i,j_k} = \sum_{l=1}^{k} P_1, J_l \quad k = 1, \ldots n \tag{2}$$

$$C_{i,j_k} = max\left(C_{i-1,J_k}, C_{i,J_{k-1}}\right) + (\mathrm{P}_i, J_k) \quad where\ i = 2, \ldots, m \& k = 2, \ldots, n \tag{3}$$

The makespan for PFSSP is calculated as:

$$C_{max} = C\left(J_k, m\right)$$

Using the Evolution strategy makespan of PFSSP has been minimized in this paper. To validate the results, the technique has been tested on 29 Nos benchmark problems of Reeves and Carlier with machines ranging from 4, 5, 6, 7, 8, 9, 10, 15, 20 and Jobs ranging from 7, 8, 10, 11, 12, 13, 14, 20, 30, 50 & 75.

IV. METHODOLOGY

A. INTRODUCTION TO ES

ES is a subclass of evolution algorithms designed for parameter optimization problems and operates on floating-point numbers. Lin *et al.* [32] showed ES performs better than GA and SA for combinatorial problems. ES is an iterative procedure, and the individual population is searched to find a potential solution. In ES, each iteration is termed as a generation. The exploration of search space is accomplished with the help of recombination and mutation operators in each generation. The selection mechanism ensures that the best individual is selected to find the potential solution to the problem. ES has been successfully applied in digital circuits and mathematics. A variant of ES known as (1+λ) has been commonly used in Cartesian Genetic programming (CGP). Based on problem type, different values of λ (termed at the number of offspring) can be used i.e. 1, 4, 9, and 16.

The general framework of ES is as under:

General ES Code

Initialization

Repeat

Reproduction

Recombination

Mutation

Evaluation

Selection

Until termination criteria satisfied

In the initialization phase, the first generation is created consisting of one or more individuals, and then the fitness of the generation is evaluated. After initialization, the evolutionary loop consisting of recombination, mutation, evaluation, and selection operators are applied. The recombination

operator generates new offspring from the parent population. The mutation operator is the main source of variation in ES, and the property of the individuals are modified by sampling random variables. In ES, the mutation is parameterized; hence it can change its property during optimization. The newly created offsprings are then evaluated using fitness values. The selection operators then identify a list of individuals that constitute the new generation of the evolutionary loop. Based on the termination criteria such as fitness value, time-period, the maximum number of generations, etc., the evolutionary loop is terminated.

B. (1+1)-ES

(1+1)-ES was proposed by Rechenberg in 1994 and is considered the simplest evolution strategy. From 1 parent, 1 offspring is generated, and the best parent is selected from both these members for the next generation.

The algorithm for (1+1)-ES is as under:

Initialization
Reproduction
Mutation
Evaluation
Selection
Termination criteria

C. IMPROVEMENTS IN ORIGINAL ES

Following improvements are incorporated in the proposed algorithms to increase their exploration and exploitation abilities.

1. The initial solution is generated using the Shortest processing time rule.
2. Instead of $(1+1)$-ES, $(1+4)$-ES and $(1+9)$-ES are used. From one parent, 4 and 9 off-springs are generated respectively. Hence the solution space is thoroughly exploited by creating more off-springs from a single parent.
3. To exploit more solution space with minimum computational time, a quad swap mutation operator is used. Quad swap searches more solution space as compared to single and double swap mutation operators.
4. The mutation rate is also varied depending on the problem size. For large size problems (with machines of more than 30), a large mutation rate is used initially. For fine-tuning of results, the mutation rate is gradually reduced with the rise in the number of iterations. Initially, a 40% mutation rate is used, and after 1500 iterations the mutation rate is reduced to 20% for the fine-tuning of results.
5. The proposed ES algorithm is tested on Carlier and Reeves benchmark problems for the first time.

Pseudocode for the Proposed ES Algorithm is shown in **Figure 1**.

1) PARENT POPULATION

Parent population is randomly generated as per population size. For a population size of 10, the randomly generated parent population is as under:

Parent	9	5	10	2	4	1	8	6	3	7

2) REPRODUCTION

The reproduction operator generates offsprings from their parents; in this paper (1+4) and (1+9) reproduction operators are used. In $(1+4)$, four offsprings are generated from one parent while in $(1+9)$ nine offsprings are generated from one parent. Although (1+4) is faster however it cannot thoroughly exploit solution space, to thoroughly exploit solution space $(1+9)$ is used. The procedure of the reproduction operator is shown in **Figure 2**, where the number of offsprings is nine. Hence nine offsprings are generated from one parent as shown in Figure 2.

3) RECOMBINATION

Recombination operator brings good characteristics of parents to their offsprings and reduces uncorrelated characteristic part of parents to its offsprings. Recombination is useful when it is used along with selection and mutation operators. In this paper, discrete recombination is used.

4) MUTATION

The performance of ES normally depends on the strength of its mutation operator. Mutation operator preserves genetic diversity from one generation to the next. The probability of mutation should be minimum, otherwise, it will become a search operator. If the parents are too similar to their offsprings then the algorithm will move towards local minima; hence, the mutation operator is used to avoiding local minima. Different mutation operators are suggested for PFSSP however Han *et al.* [33] showed that the best mutation operator for PFSSP is the swap operator. Quad swap mutation operator is used in this paper to save computational time. In quad swap mutation operator twice space is searched with the same number of iterations, hence significant computational time is saved. The procedure of quad swap mutation operator is shown in **Figure 3**, for a population size of 20. Four genes from the parent population are randomly chosen, and their values are interchanged. Gene 2, 4, 6 & 8 are swapped with gene 18, 16, 13 & 10 simultaneously.

5) SELECTION

Based on the selection procedure, the following are the types of selection operators in ES.

i. $(\mu + \lambda)$-ES
ii. (μ, λ)-ES

Where μ represents the number of parents and λ represents the number of offsprings. The difference between these strategies is illustrated in **Figure 4**.

In $(\mu + \lambda)$-ES, from parents λ offsprings are created at any given generation and based on the objective function values of $\mu + \lambda$ are sorted. The selection takes place between $\mu + \lambda$ and the best μ from all the $\mu + \lambda$ become parent for the next generation. In (μ, λ)-ES, from parents λ offsprings are created at any given generation and based on the objective function values of λ are sorted. The selection takes place

ES Algorithm: Pseudocode for the ES (1+9) is as under:

Input:
Number of offspring, λ (λ is 9)
Population size, μ
Total generations, k
Mutation rate, m

Generate parent population using shortest processing time rule, P
Compute Cmax of the parent population
for *gen =1: k*
Evaluate sequence
If *the total number of generations (k) is not reached go to step 7*
else
go to step 19
for *v= 2: μ*
Generate offsprings OS1-OS9 using reproduction operator (Since λ is 9, hence from one parent nine offsprings are randomly generated)
Update population, Pi
Perform Quad swap Mutation Operator, (Randomly select four different jobs, and interchange their positions simultaneously)
Compute Cmax for all newly generated offsprings
end for
Evaluate parent population to select the fittest candidate (C) for next-generation using the following rules:
If *an offspring has a minimum makespan it is termed as a candidate (OS1-OS9=C)*
else
Parent is selected as a candidate (P=C)
end
repeat *until loop (Go to step 2)*
Output Cmax and Total generations

FIGURE 1. Pseudo code for ES.

Parent	9	5	10	2	4	1	8	6	3	7

Off-01	9	10	5	2	4	1	8	6	3	7
Off-02	10	5	9	2	4	1	8	6	3	7
Off-03	6	5	10	2	4	1	8	7	3	9
Off-04	9	5	1	2	10	4	8	6	3	7
Off-05	9	5	1	2	4	8	10	6	3	7
Off-06	9	5	4	2	10	7	8	6	3	1
Off-07	9	5	10	2	8	1	4	6	3	7
Off-08	3	5	10	2	7	4	1	8	6	9
Off-09	9	10	5	2	4	1	8	7	3	6

FIGURE 2. The procedure of reproduction operator.

between λ offsprings and the best μ become a parent for the next generation.

In this paper $(\mu + \lambda)$-ES is used as it is recommended for numerical optimization problems, hence a parent once superseded by its offspring cannot be selected again.

6) TERMINATION

Different termination criteria are used keeping in view the complexity of the problem. For PFSSP mostly the number of iterations is used as the termination criteria however other criteria are also available, i.e. processing time, fitness value, no change results, etc. In this research, the Maximum number of iterations is set to 2,000.

V. NUMERICAL EXPERIMENTS

To evaluate the performance of ES, it has been compared with some famous benchmark algorithms for PFSSP. The proposed technique has been tested on 29 Benchmark PFSSP's. Carlier eight problems are denoted as Cr1, Cr2 …Cr8 designed by Carlier [31], while Reeves 21 problems are denoted as Rc01, Rc03, Rc05 …Rc41, designed by Reeves [27]. Several researchers have tested their technique on these benchmark problems to verify the effectiveness and robustness of their techniques; however, the application of ES for the solution of these benchmark problems is still imminent. The test problems have been downloaded from OR-library (http://people.brunel.ac.uk/~mastjjb/jeb/orlib/flowshopinfo.html).

The ES5 and ES10 algorithms are coded in MATLAB, and numerical experiments were performed on a PC with Core™ I5, 2.6 GHz processor, and 4.0 GB RAM. Each instance was

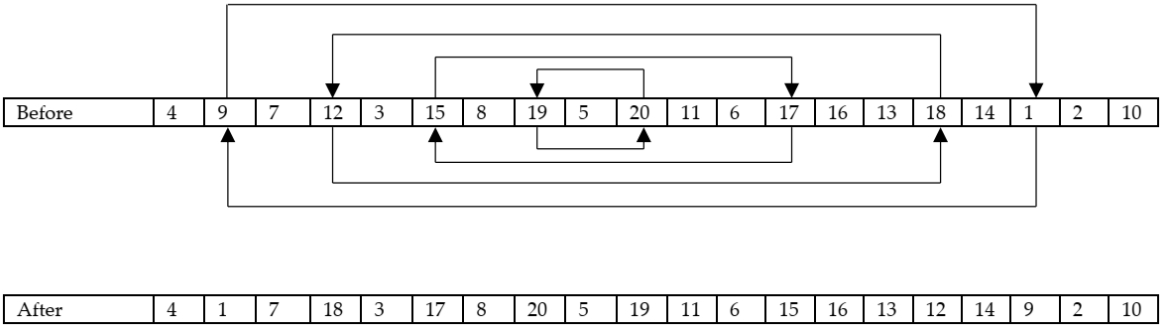

FIGURE 3. The quad swap mutation operator.

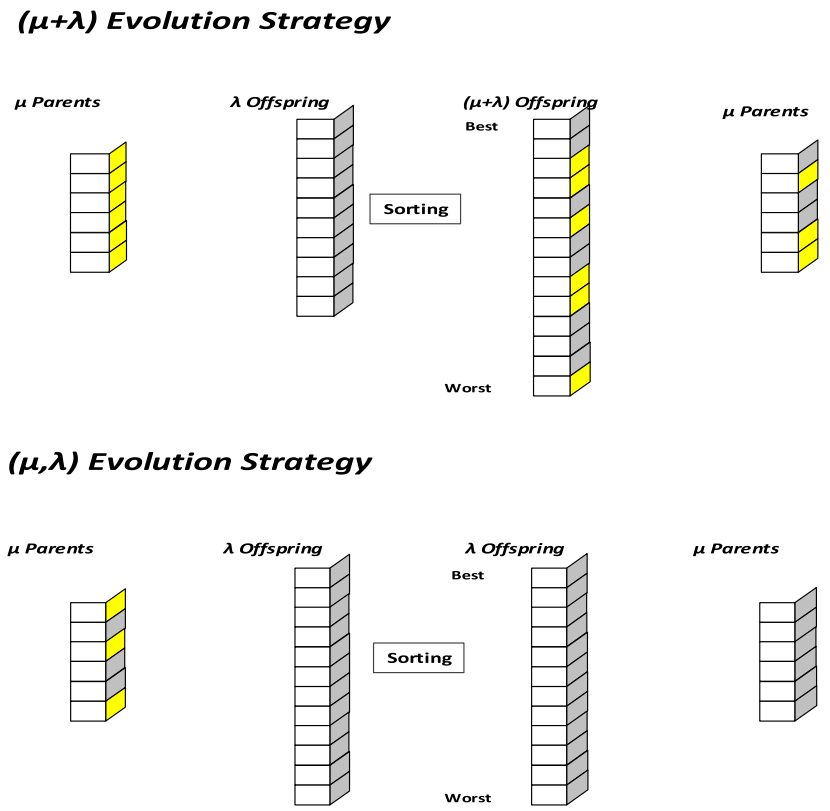

FIGURE 4. The procedure of $(\mu + \lambda)$-ES & (μ, λ)-ES selection operators.

run for 30 independent runs and maximum iterations were set at 2000. For each instance, a Gantt chart is also generated in MATLAB to specify the available gaps as shown in Figure 5. To check the performance of the proposed technique and compare it with other techniques, the following parameters are set as standard to evaluate this technique, as shown in Eqs. (4)-(6). The best-known makespan is termed as C. The best relative error to C is termed as BRE. The average error to C is termed as ARE. The worst relative error to C is termed as WRE. The average computational time for each problem is calculated in seconds and is termed as T(s).

$$BRE = \frac{\min(solutions) - C}{C} \times 100\% \quad (4)$$

$$ARE = \frac{\text{average}(solutions) - C}{C} \times 100\% \quad (5)$$

$$WRE = \frac{\max(solutions) - C}{C} \times 100\% \quad (6)$$

A. COMPARISON OF ES5 AND ES10

To show a balance between global search and computational time, two variants named ES5 and ES10 are compared. Table 3 shows the computational results for ES5 and ES10. ES5 shows the value of makespan calculated for $\lambda = 4$ (from one parent four offsprings are generated, and the selection pool consists of both parent and offsprings), while ES10 shows the value of makespan calculated for $\lambda = 9$ (from one parent nine offsprings are generated and the selection

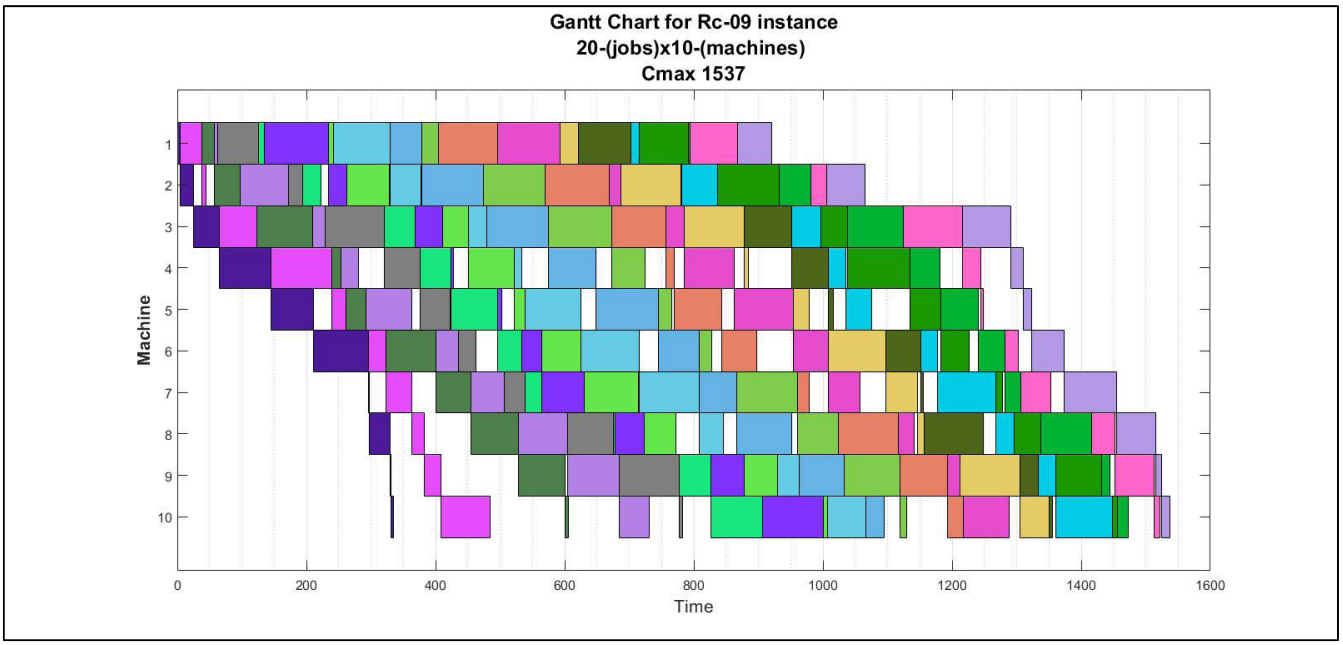

FIGURE 5. Gantt chart for Rc-09 instance.

pool consists of both parent and offsprings). The search space for $\lambda = 9$ is more than $\lambda = 4$, however, the more computational time is taken by $\lambda = 9$ as shown in Table 2. From Table 2, it can be seen that BRE, ARE, and WRE obtained by ES10 are better than ES5. ES5 has found eight best solutions for carlier problems and six best solutions for Reeves problem, while ES10 has found eight best solutions for Carlier problems and ten best solutions for Reeves problems. With the above analysis, it can be concluded that ES10 finds a better solution for PFSSP at the expense of computational time.

An important feature to validate the performance of the algorithm is to check convergence with an increase in the number of iterations. Hence, for Instance, Rc23 and Rcc33, the algorithm is run at several iterations and results are graphically shown in Figure 6-7. The comparison is made for both variants of ES i.e. ES5 and ES10. The makespan values of instance Rc23 for ES5 at 100, 200, 500, 1000, 1500 and 2000 iterations are 2369, 2303, 2268, 2230, 2150, 2098 and 2028 respectively. Makespan values for instance Rc23 for ES10 at 100, 200, 500, 1000, 1500 and 2000 iterations are 2325, 2268, 2188, 2131, 2119, 2076 and 2020 respectively. The makespan values of Rc33 for ES5 at 100, 200, 500, 1000, 1500 and 2000 iterations are 3621, 3591, 3538, 3443, 3355, 3224 and 3145 respectively. While makespan values of Rc33 for ES10 at 100, 200, 500, 1000, 1500 and 2000 iterations are 3506, 3413, 3364, 3262, 3258, 3197 and 3119. From all these values it is evident that by increasing the number of iterations the makespan is reducing. This depicts that the ES algorithm does not stuck in local minima and keeps improving the solution. The makespan values of ES10 are minimum as compared to ES5 for all instances, which depicts that by increasing the number of offsprings the solution improves.

FIGURE 6. Graph for makespan vs. No of iterations for Rc23.

B. COMPARISON WITH OTHER ALGORITHMS

To validate the results of ES5 and ES10 they are compared with algorithms of Lin *et al.* [32] and Shao and Pi [30] proposed a Hybrid Back Tracking Search Algorithm with makespan objective and evaluated their technique on carlier and reeves benchmark problems. The authors proposed two variants namely HSBA (Hybrid backtracking search algorithm with random insertion local search) and HSBA_NOLS (Hybrid backtracking search algorithm without local search). Both variants were equipped with discrete crossover and mutation operator and an SA based mechanism to avoid local minima. HSBA_NOLS performed better for small problems however for large problems its result was unsatisfactory, while HSBA was equally applicable to small and large-sized problems. Hence for computational comparison,

TABLE 2. Comparison of ES5 and ES10.

Problem	n x m	C	ES5 (λ =4)					ES10 (λ =9)				
			Cmax	T(s)	BRE	ARE	WRE	Cmax	T(s)	BRE	ARE	WRE
Cr1	11 x 5	7038	7038	1.56	0	0	0	7038	2.40	0	0	0
Cr2	13 x 4	7166	7166	1.68	0	0	0	7166	2.48	0	0	0
Cr3	12 x 5	7312	7312	1.58	0	0	0	7312	2.53	0	0	0
Cr4	14 x 4	8003	8003	1.94	0	0	0	8003	2.74	0	0	0
Cr5	10 x 6	7720	7720	1.52	0	0	0	7720	2.40	0	0	0
Cr6	8 x 9	8505	8505	1.38	0	0	0	8505	2.15	0	0	0
Cr7	7 x 7	6590	6590	1.37	0	0	0	6590	1.99	0	0	0
Cr8	8 x 8	8366	8366	1.57	0	0	0	8366	2.04	0	0	0
Rc01	20 x 5	1247	1247	2.23	0	0	0	1247	3.67	0	0	0
Rc03	20 x 5	1109	1109	2.19	0	0	0	1109	3.62	0	0	0
Rc05	20 x 5	1242	1242	2.20	0	0	0	1242	3.69	0	0	0
Rc07	20 x 10	1566	1566	2.51	0	0	0	1566	3.97	0	0	0
Rc09	20 x 10	1537	1537	2.59	0	0	0	1537	3.88	0	0	0
Rc11	20 x 10	1431	1431	2.52	0	0	0	1431	3.91	0	0	0
Rc13	20 x 15	1930	1935	2.49	0.259	0.259	0.259	1930	4.05	0	0.026	0.259
Rc15	20 x 15	1950	1962	2.53	0.615	0.615	0.615	1950	4.11	0	0.031	0.615
Rc17	20 x 15	1902	1911	2.57	0.473	0.526	0.999	1902	4.13	0	0.053	0.526
Rc19	30 x 10	2093	2112	3.22	0.908	0.956	1.386	2106	5.29	0.621	0.688	1.386
Rc21	30 x 10	2017	2036	3.64	0.942	1.011	1.636	2030	5.26	0.645	0.687	1.190
Rc23	30 x 10	2011	2028	3.31	0.845	1.042	1.989	2020	5.31	0.448	0.582	1.343
Rc25	30 x 15	2513	2536	3.24	0.915	1.015	2.905	2528	5.25	0.597	0.653	1.711
Rc27	30 x 15	2373	2384	3.06	0.506	1.895	3.499	2380	5.33	0.337	1.010	1.560
Rc29	30 x 15	2287	2301	3.21	0.612	0.700	2.361	2297	5.13	0.437	0.669	1.924
Rc31	50 x 10	3045	3076	4.26	1.018	1.248	3.317	3054	7.33	0.296	0.476	1.741
Rc33	50 x 10	3114	3145	4.22	0.996	1.024	1.381	3119	7.23	0.161	0.217	1.156
Rc35	50 x 10	3277	3285	4.31	0.244	0.244	0.244	3277	7.36	0	0.024	0.244
Rc37	75 x 20	4951	5054	6.48	2.080	2.170	3.878	5036	10.8	1.717	1.793	3.232
Rc39	75 x 20	5087	5150	6.51	1.238	1.287	2.202	5131	10.81	0.865	0.974	2.084
Rc41	75 x 20	4960	5071	6.50	2.238	2.411	4.173	5042	10.65	1.653	1.754	2.741

results of HSBA will be used Shao and Pi [30] proposed a self-guided differential evolution with neighborhood search (NS-SGDE) to minimize the makespan of PFSSP. The initial solution is generated by combining discrete harmony search [33] algorithm with NEH heuristic, Rajendran [34], and FRB1 [35]. The search is then guided using a probabilistic model of Estimation of Distribution Algorithm and using various crossover and mutation operators. Finally, a variable neighborhood search is used to improve solutions. The author suggested three variants namely NS-SGDE without initial solution and neighborhood search termed as SGDE-NOHS, NS-SGDE without neighborhood search termed as SGDE, and NS-SGDE having initial solution and neighborhood search. Comparison shown that NS-SGDE performed better than other two variants at the expense of computational time, hence we will use NS-SGDE for computational comparison.

For computational comparison, both HSBA and NS-SGDE were coded in MATLAB and executed on the same PC i.e. Core^{TM} I5, 2.6 GHz processor, and 4.0 GB RAM. For comparison number of iterations for HSBA and NS-SGDE is also set at 2000 iterations, and the result of each instance is run 30 times. In Table 3, BRE, ARE, WRE and the computational time of all techniques is compared.

For carlier benchmark problems, NS-SGDE, ES5, and ES10 found the best solution for all eight instances, and the BRE, WRE, ARE values are zero which shows that all these algorithms find the best solution in every iteration. HSBA find the best solution for seven instances while for Cr3 it found some non-optimal solutions. Computational time taken by each algorithm against carlier problems is also mentioned in Table 3. HSBA has taken more computational time as compared to NS-SGDE, ES5, and ES10. The average computational time taken by HSBA, NS-SGDE, ES5, and

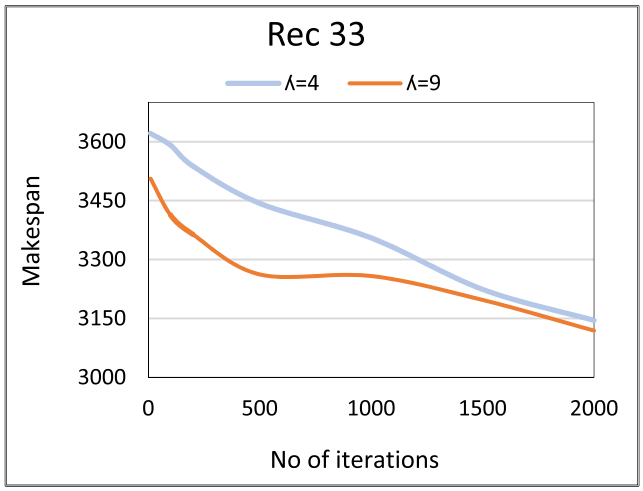

FIGURE 7. Graph for makespan vs. No of iterations for Rc33.

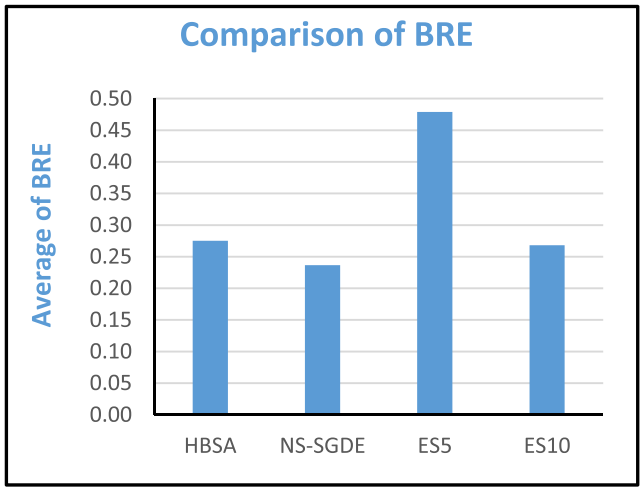

FIGURE 8. Comparison of BRE values for Cr and Rc problems.

ES10 for carlier instances is 19.12 sec, 0.02 sec, 1.58 sec, and 2.34 sec. Although NS-SGDE, ES5, and ES10 perform best for all carlier instances, the time taken by NS-SGDE is much lower than other techniques. ES5 and ES10 take much less computational time as compared to HSBA, however, their computational time is more than NS-SGDE for carlier instances.

For Reeves benchmark problems, the best solution found by HSBA, NS-SGDE, ES5, and ES10 are eight, eleven, six, and ten respectively. Time taken by each algorithm for Reeves problems is also mentioned in Table 3. Time taken by HSBA and NS-SGDE is much more than ES5 and ES10. For 30 machines and above, computational time taken by NS-SGDE rises abruptly. For instance, Rc37, the time taken by HSBA, NS-SGDE, ES5, and ES10 is 3615.84 sec, 385.30 sec, 6.48 sec, and 10.8 sec respectively. The average computational time by HSBA, NS-SGDE, ES5, and ES10 for Reeves instances is 674.19 sec, 77.80 sec, 3.51 sec, and 5.75 sec. From Table 3 it is evident that the ES10 performs better for Reeves benchmark problems than other techniques in terms of solution quality and computational time.

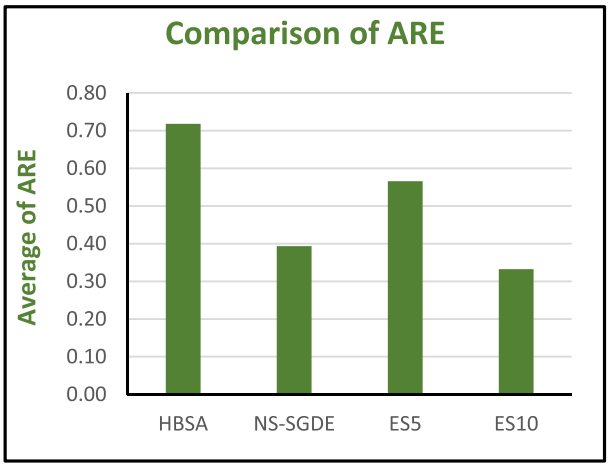

FIGURE 9. Comparison of ARE values for Cr and Rc problems.

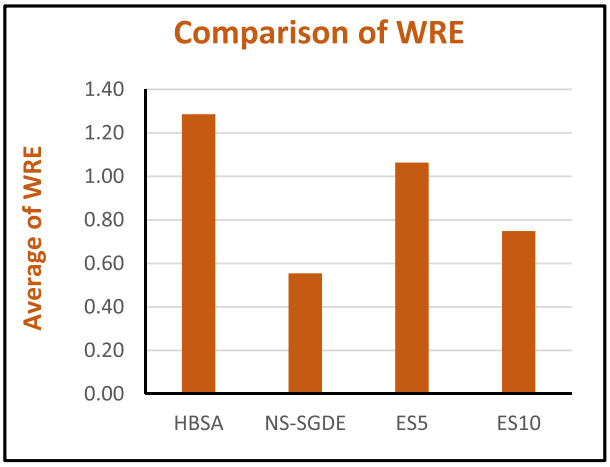

FIGURE 10. Comparison of WRE values for Cr and Rc problems.

Figure 8-10 provides a graphical representation for the average values of BRE, ARE, and WRE for HSBA, NS-SGDE, ES5, and ES10 algorithms. The average values of BRE for HSBA, NS-SGDE, ES5, and ES10 are 0.275, 0.237, 0.479, and 0.268. The average values of ARE for HSBA, NS-SGDE, ES5, and ES10 are 0.717, 0.393, 0.566, and 0.332. The average values of WRE for HSBA, NS-SGDE, ES5, and ES10 are 1.285, 0.55, 1.064, and 0.749. In terms of BRE values, ES10 performs better than HSBA and ES5 while NS-SGDE finds a better solution as compared to ES10. For ARE values, ES10 performs better than HSBA, NS-SGDE, and ES5. While for WRE values, ES10 performs better than HSBA and ES5, but inferior to NS-SGDE. In the light of all the above, ES10 performs better in-terms of solution quality keeping in view the minimal computational time required. ES10 found eight best solutions for Carlier problems and ten best solutions for Reeves problems.

VI. TEST CASE: APPLICATION OF EVOLUTION STRATEGY TO BATTERIES MANUFACTURING

Pakistan Accumulator (Pvt.) Ltd. is the manufacturer of Lead Acid batteries by the brand name "Volta and Osaka

TABLE 3. Comparisons of ES5, ES10 with other algorithms.

Instance	HSBA				NS-SGDE				ES5				ES10			
	BRE	ARE	WRE	T(s)	BRE	ARE	WRE	T(s)	BRE	ARE	WRE	T(s)	BRE	ARE	WRE	T(s)
Cr1	0	0	0	20.41	0	0	0	0.015	0	0	0	1.56	0	0	0	2.40
Cr2	0	0	0	23.00	0	0	0	0.020	0	0	0	1.68	0	0	0	2.48
Cr3	0	0.06	1.19	24.79	0	0	0	0.059	0	0	0	1.58	0	0	0	2.53
Cr4	0	0	0	25.11	0	0	0	0.013	0	0	0	1.94	0	0	0	2.74
Cr5	0	0	0	17.98	0	0	0	0.039	0	0	0	1.52	0	0	0	2.40
Cr6	0	0	0	15.39	0	0	0	0.018	0	0	0	1.38	0	0	0	2.15
Cr7	0	0	0	12.47	0	0	0	0.024	0	0	0	1.37	0	0	0	1.99
Cr8	0	0	0	13.77	0	0	0	0.008	0	0	0	1.57	0	0	0	2.04
Rc01	0	0.14	0.16	53.46	0	0	0	6.362	0	0	0	2.23	0	0	0	3.67
Rc03	0	0.08	0.18	59.29	0	0	0	2.321	0	0	0	2.19	0	0	0	3.62
Rc05	0.24	0.24	0.24	52.33	0	0.193	0.242	6.504	0	0	0	2.20	0	0	0	3.69
Rc07	0	0.46	1.15	56.21	0	0	0	1.504	0	0	0	2.51	0	0	0	3.97
Rc09	0	0.07	0.65	66.42	0	0	0	5.126	0	0	0	2.59	0	0	0	3.88
Rc11	0	0	0	61.07	0	0	0	2.463	0	0	0	2.52	0	0	0	3.91
Rc13	0.1	0.53	1.14	75.33	0	0.135	0.466	11.348	0.259	0.259	0.259	2.49	0	0.026	0.259	4.05
Rc15	0.05	0.64	1.18	80.35	0	0.041	0.051	12.965	0.615	0.615	0.615	2.53	0	0.031	0.615	4.11
Rc17	0	1	2.16	76.46	0	0.073	0.368	10.611	0.473	0.526	0.999	2.57	0	0.053	0.526	4.13
Rc19	0.29	0.81	1.29	201.04	0.287	0.497	0.86	28.849	0.908	0.956	1.386	3.22	0.621	0.688	1.386	5.29
Rc21	0.69	1.5	1.83	152.44	0.694	1.393	1.636	28.666	0.942	1.011	1.636	3.64	0.645	0.687	1.190	5.26
Rc23	0.45	1.28	3.08	187.76	0.448	0.522	0.746	28.930	0.845	1.042	1.989	3.31	0.448	0.582	1.343	5.31
Rc25	0.4	1.29	2.43	242.84	0.358	0.899	1.472	38.541	0.915	1.015	2.905	3.24	0.597	0.653	1.711	5.25
Rc27	0.25	1.27	2.57	237.65	0.253	0.943	1.391	39.388	0.506	1.895	3.499	3.06	0.337	1.010	1.560	5.33
Rc29	0.57	1.42	2.97	214.16	0.35	0.853	1.443	39.524	0.612	0.700	2.361	3.21	0.437	0.669	1.924	5.13
Rc31	0.43	1.91	2.66	685.10	0.296	0.929	1.149	117.856	1.018	1.248	3.317	4.26	0.296	0.476	1.741	7.33
Rc33	0	0.59	1.28	592.92	0	0.083	0.835	115.088	0.996	1.024	1.381	4.22	0.161	0.217	1.156	7.23
Rc35	0	0	0	526.82	0	0	0	5.081	0.244	0.244	0.244	4.31	0	0.024	0.244	7.36
Rc37	1.92	2.93	4.2	3615.84	1.434	1.818	2.096	385.309	2.080	2.170	3.878	6.48	1.717	1.793	3.232	10.8
Rc39	0.9	1.88	3.38	3638.52	0.924	0.995	1.108	381.898	1.238	1.287	2.202	6.51	0.865	0.974	2.084	10.81
Rc41	1.69	2.72	3.55	3282.12	1.815	2.036	2.218	365.614	2.238	2.411	4.173	6.50	1.653	1.754	2.741	10.65

Batteries''. The company was founded in 1992 with technical support from Hawker Batteries, UK, and is located in Hattar Industrial Estate, Pakistan. Its head office is located in Islamabad and has seven regional offices across the country. The company has got ISO 9001 and ISO 14001 certifications. With more than 2000 employees, the company is producing various types of batteries as mentioned below:

- Automotive Batteries (32-240 Amp).
- Motorbikes Batteries (4-10 Amp).
- Maintenance Free Batteries (32-200 Amp).
- Tubular Batteries (80-160 Amp).
- VRLA Batteries (80-200 Amp).

Pakistan Accumulator is the only local manufacturer of Tubular and VRLA batteries in Pakistan and is partially fulfilling the demands of our country.

In Pakistan, the Compound annual growth rate of batteries is more than 3% for 2020-2025. With the growth of the automobile sector especially hybrid and electric vehicles the demand is expected to rise during the forecast period. Presently the country is fulfilling only 4% of the total power requirement using renewable energy i.e. solar and wind power plants. By 2030, the government is planning to increase the share of renewable energy by up to 30%. Growth in the transport section of the country is increasing by 100% per year, however, most of the transportation sector is based on fossil fuel products. Shifting to electric vehicles will not only save the import bill for the country but will also lead to a cleaner and green environment [36]. Given all the above, the Pakistan battery market is expected to rise and there is a great need for an accurate scheduling environment to fulfill the demand of the country and also save significant resources.

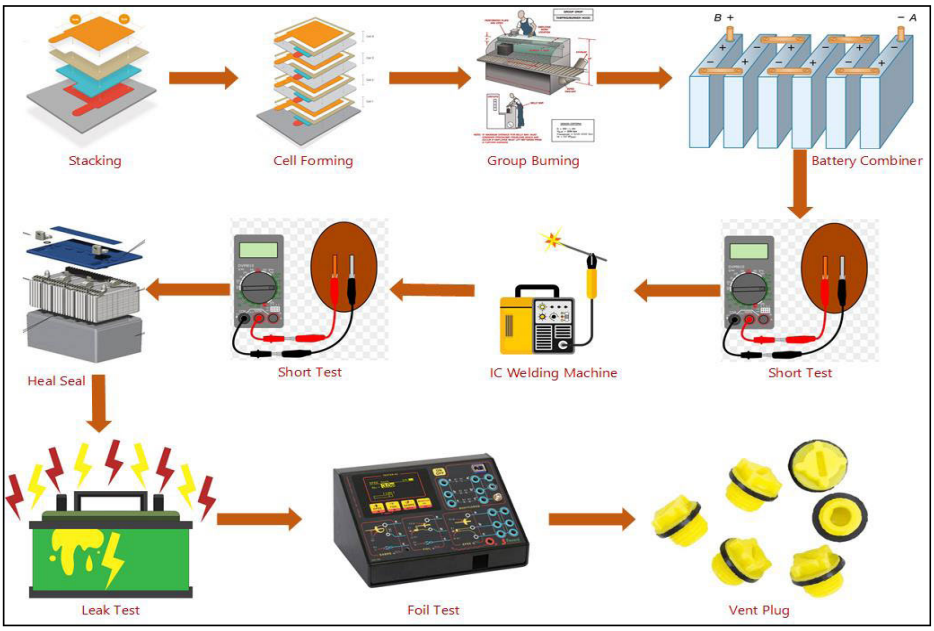

FIGURE 11. Flow chart of battery operations.

TABLE 4. Test case details.

Problem	Description	Size
NS40	NS40-30 Plate Battery	35 x 12

To raise the efficiency of manufacturing industries, the most effective way is to optimize their scheduling environment. In manufacturing scheduling, jobs are allocated to machines to optimize one or more objectives i.e. minimization of makespan, minimization of total tardiness, minimization of maximum lateness, and increase machine utilization. To increase production and machine utilization, the minimization of makespan is the most significant objective.

Batteries manufacturing is an energy-intensive environment and implementation of an accurate scheduling environment is a dire need at the moment. Evolution strategy can be easily applied to optimize real-life problems. For different problems, the optimization operators and parameters should be accurately designed. The framework proposed by ES operates in the major steps: initialization, selection, reproduction, mutation, and improvement. With reasonable computational time, the ES technique has been able to find the best solution for Volta and Osaka Batteries. Since the performance of ES10 is better than ES5, hence ES10 will be used in this test case to compute makespan. Following parameters are used in ES10: For reproduction λ is 9 (from 1 parent nine offspring's are randomly generated and the population size is 10) mutation rate is 40%, mutation type is quad swap mutation, and the Number of iterations is set to 2000. Detail of the test case is shown in Table 4.

A. TEST CASE- NS40-30 PLATE BATTERY

Table 5 shows the processing time for all 12 operations and 35 jobs for manufacturing of NS40-30 plate battery. The twelve operations involved are: Stacking, Cell forming, Group Burning, Battery combiner, manual Short Tester, IC Welding Machine, Short Tester-2, Heat Seal, Leak Tester Coding, Foil Seal, and Went Plug Installation. A graphical chart for various operations of battery manufacturing is shown in Figure 11. From these operations, Pakistan Accumulator is producing 1400 units of NS40-30 Plates batteries daily. Table 6 shows the results of ES10 for different batches of NS40-30 Plates.

In Table 6, the manufacturing time for different batches of NS40-30Plate battery is compared using the ES10 algorithm and manual scheduling of Pakistan Accumulator. Also, %GAP is calculated for each batch to indicate which technique is finding the best schedule against each batch. A positive value of the %GAP shows that ES10 is giving a good production schedule, while negative values show that manual scheduling is better. For a batch size of 35 batteries, the makespan for ES10 is 2510 sec while for the Pakistan accumulator it is 2583 sec. Similarly, the makespan of ES10 is also minimum for other batch sizes i.e. 140, 1120, and 1400 respectively. %GAP is also calculated for all four batches which are 2.83, 2.06, 1.84, and 1.25 respectively. For every batch %GAP value is positive, which shows that ES10 is finding better schedules. The minimum %GAP is 1.25 for a batch size of 1400. Hence results propose that ES is an effective tool for scheduling real-life cases. Pakistan Accumulator is daily producing 1400 batteries while using ES10 it can produce 1415 units daily. Additional production of 15 batteries daily, and an increase of 450 batteries monthly. Pakistan Accumulator can increase its monthly production with the same resources by using permutation schedules of ES10 and can increase their machine utilization. Since in Pakistan there is a great demand for batteries, hence this paper can serve as a reference for other battery manufacturers

TABLE 5. Processing times for 35 jobs and 12 operations of NS30-40 plates battery.

Job	O1		O2		O3		O4		O5`		O6		O7		O8		O9		O10		O11		O12	
	M	P.T	M	P.T	M	P.T	M	P.T	M	P.T	M	P.T	M	P.T	M	P.T	M	P.T	M	P.T	M	P.T	M	P.T
1	1	25	2	57	3	60	4	47	5	27	6	25	7	24	8	23	9	40	10	57	11	59	12	59
2	1	29	2	58	3	58	4	44	5	25	6	25	7	25	8	24	9	41	10	57	11	58	12	57
3	1	27	2	59	3	62	4	48	5	26	6	26	7	24	8	25	9	42	10	55	11	57	12	58
4	1	26	2	63	3	61	4	46	5	25	6	25	7	25	8	25	9	41	10	56	11	56	12	56
5	1	27	2	62	3	59	4	49	5	26	6	28	7	26	8	26	9	42	10	50	11	59	12	58
6	1	26	2	58	3	58	4	47	5	27	6	27	7	27	8	27	9	41	10	58	11	57	12	57
7	1	26	2	63	3	55	4	49	5	25	6	25	7	25	8	25	9	40	10	59	11	57	12	56
8	1	28	2	62	3	57	4	48	5	26	6	26	7	26	8	26	9	39	10	57	11	58	12	58
9	1	26	2	60	3	59	4	48	5	27	6	27	7	27	8	27	9	40	10	61	11	57	12	57
10	1	25	2	62	3	58	4	45	5	28	6	28	7	26	8	26	9	38	10	58	11	58	12	58
11	1	24	2	61	3	61	4	46	5	26	6	29	7	29	8	29	9	42	10	59	11	59	12	60
12	1	25	2	61	3	62	4	47	5	26	6	26	7	26	8	28	9	41	10	61	11	61	12	61
13	1	27	2	60	3	63	4	50	5	25	6	24	7	24	8	24	9	40	10	60	11	60	12	59
14	1	24	2	60	3	62	4	51	5	26	6	26	7	26	8	26	9	41	10	60	11	60	12	60
15	1	26	2	61	3	61	4	48	5	24	6	24	7	24	8	24	9	39	10	61	11	61	12	62
16	1	25	2	61	3	59	4	47	5	24	6	24	7	24	8	24	9	40	10	62	11	62	12	61
17	1	26	2	62	3	58	4	48	5	26	6	26	7	26	8	26	9	40	10	59	11	59	12	59
18	1	28	2	59	3	59	4	49	5	26	6	26	7	26	8	26	9	41	10	59	11	59	12	60
19	1	25	2	60	3	58	4	48	5	25	6	25	7	25	8	25	9	42	10	60	11	59	12	59
20	1	26	2	61	3	59	4	49	5	27	6	27	7	27	8	26	9	40	10	61	11	61	12	58
21	1	27	2	61	3	61	4	50	5	26	6	26	7	26	8	27	9	40	10	59	11	58	12	60
22	1	28	2	63	3	59	4	51	5	24	6	24	7	24	8	23	9	42	10	61	11	61	12	61
23	1	27	2	62	3	58	4	48	5	25	6	25	7	25	8	25	9	42	10	60	11	59	12	59
24	1	29	2	60	3	59	4	46	5	26	6	26	7	26	8	26	9	41	10	59	11	59	12	59
25	1	28	2	60	3	59	4	45	5	28	6	28	7	28	8	28	9	40	10	58	11	58	12	58
26	1	25	2	61	3	57	4	44	5	27	6	27	7	27	8	27	9	39	10	60	11	61	12	61
27	1	27	2	61	3	59	4	48	5	26	6	26	7	26	8	26	9	38	10	61	11	57	12	57
28	1	27	2	60	3	60	4	48	5	25	6	28	7	25	8	25	9	40	10	62	11	62	12	62
29	1	28	2	61	3	61	4	49	5	28	6	30	7	25	8	25	9	39	10	58	11	59	12	61
30	1	24	2	61	3	58	4	45	5	27	6	27	7	27	8	27	9	41	10	59	11	60	12	60
31	1	25	2	63	3	59	4	44	5	25	6	32	7	27	8	27	9	42	10	60	11	59	12	59
32	1	27	2	59	3	55	4	47	5	26	6	26	7	26	8	26	9	41	10	61	11	61	12	62
33	1	26	2	58	3	57	4	46	5	25	6	25	7	24	8	24	9	40	10	60	11	57	12	57
34	1	24	2	61	3	58	4	48	5	28	6	28	7	23	8	25	9	40	10	61	11	58	12	58
35	1	28	2	62	3	56	4	44	5	27	6	27	7	27	8	26	9	39	10	58	11	60	12	59

TABLE 6. The output of ES10 for different batches of NS40-30 plates.

Batch Size	Battery Model	PA Cmax (PA)	ES10 Cmax (ES10)	%GAP= (100*(PA–ES10)/PA)
35	NS40-30 Plates	2583 sec	2510 sec	2.83
140	NS40-30 Plates	9120 sec	8932 sec	2.06
1120	NS40-30 Plates	1160 min	1139 min	1.84
1400	NS40-30 Plates	1440 min	1422 min	1.25

to increase their production and machine utilization. It is recommended that in future studies energy consumption and material wastage may be incorporated.

VII. CONCLUSION AND FUTURE WORKS

In this paper, first, a comprehensive review is carried out of various techniques used to minimize the makespan of PFSSP. Then an improved evolution strategy is presented with two variants namely ES5 and ES10. Quad swap mutation operator is utilized to minimize the processing time. For fine-tuning of the results, the mutation rate is reduced with the increase in the number of iterations. The performance of the proposed technique is checked on benchmark problems of Carlier and Reeves. The results show that ES10 outperforms ES5, however, it takes more computational time. ES10 has found eight best solutions for Carlier problems and ten best solutions for Reeves problems. Comparison with other classical algorithms was carried out, and the result showed that ES10 performed better than some of the classical algorithms, as ARE values of ES10 were better than other algorithms and ES10 takes very

little computational time. ES10 is applied to real-life case for battery manufacturing. Pakistan Accumulators is daily producing 1400 units of NS30-40 Plates battery. Results show that by using ES10, monthly 450 batteries can be additionally produced with the same resources. After 1,500 iterations, the ES algorithm shows minimal improvement, hence to further improve the performance of the proposed algorithm, it should be combined with any local search technique to avoid local minima.

Further research can be focused on following directions. First, this technique can be tested on other scheduling problems, i.e. hybrid flow shop problems, job shop problems. Secondly, ES can be combined with any other local search technique to escape local minima and for fast convergence. Third, ES can be used to optimize multi-objective flow shop problems. Also solving the traveling salesman problem using ES is recommended.

Computational results show the robustness of ES as it is equally applicable to small and large size problems; hence it should be applied to real-life problems from the automobile, plastic, glass, and steel industry to minimize processing time and increase machine utilization.

REFERENCES

[1] X. Li and M. Yin, "A hybrid cuckoo search via Lévy flights for the permutation flow shop scheduling problem," *Int. J. Prod. Res.*, vol. 51, no. 16, pp. 4732–4754, Aug. 2013, doi: 10.1080/00207543.2013.767988.

[2] 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.

[3] J. N. D. Gupta and E. F. Stafford, "Flowshop scheduling research after five decades," *Eur. J. Oper. Res.*, vol. 169, no. 3, pp. 699–711, Mar. 2006, doi: 10.1016/j.ejor.2005.02.001.

[4] D. A. Rossit, F. Tohmé, and M. Frutos, "The non-permutation flow-shop scheduling problem: A literature review," *Omega*, vol. 77, pp. 143–153, Jun. 2018, doi: 10.1016/j.omega.2017.05.010.

[5] S. M. Johnson, "Optimal two- and three-stage production schedules with setup times included," *Nav. Res. Logistics Quart.*, vol. 1, no. 1, pp. 61–68, Mar. 1954, doi: 10.1002/nav.3800010110.

[6] 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.

[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] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *Eur. J. Oper. Res.*, vol. 177, no. 3, pp. 2033–2049, Mar. 2007, doi: 10.1016/j.ejor.2005.12.009.

[9] 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.

[10] H. Wei, S. Li, H. Jiang, J. Hu, and J. Hu, "Hybrid genetic simulated annealing algorithm for improved flow shop scheduling with makespan criterion," *Appl. Sci.*, vol. 8, no. 12, p. 2621, Dec. 2018, doi: 10.3390/app8122621.

[11] F. Zhao, H. Liu, Y. Zhang, W. Ma, and C. Zhang, "A discrete water wave optimization algorithm for no-wait flow shop scheduling problem," *Expert Syst. Appl.*, vol. 91, pp. 347–363, Jan. 2018, doi: 10.1016/j.eswa.2017.09.028.

[12] D.-Z. Zheng and L. Wang, "An effective hybrid heuristic for flow shop scheduling," *Int. J. Adv. Manuf. Technol.*, vol. 21, no. 1, pp. 38–44, Jan. 2003, doi: 10.1007/s001700300005.

[13] A. C. Nearchou, "A novel Metaheuristic approach for the flow shop scheduling problem," *Eng. Appl. Artif. Intell.*, vol. 17, no. 3, pp. 289–300, Apr. 2004.

[14] B. Jarboui, M. Eddaly, and P. Siarry, "A hybrid genetic algorithm for solving no-wait flowshop scheduling problems," *Int. J. Adv. Manuf. Technol.*, vol. 54, nos. 9–12, pp. 1129–1143, Jun. 2011, doi: 10.1007/s00170-010-3009-4.

[15] Z.-Q. Zhang, B. Qian, B. Liu, R. Hu, and C.-S. Zhang, "Hybrid estimation of distribution algorithm for blocking flow-shop scheduling problem with sequence-dependent setup times," in *Proc. Int. Conf. Intell. Comput.*, 2018, pp. 628–637, doi: 10.1007/978-3-319-95930-6_63.

[16] O. Engin and A. Güçlü, "A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems," *Appl. Soft Comput.*, vol. 72, pp. 166–176, Nov. 2018, doi: 10.1016/j.asoc.2018.08.002.

[17] Y. Zhang, Y. Yu, S. Zhang, Y. Luo, and L. Zhang, "Ant colony optimization for cuckoo search algorithm for permutation flow shop scheduling problem," *Syst. Sci. Control Eng.*, vol. 7, no. 1, pp. 20–27, Jan. 2019, doi: 10.1080/21642583.2018.1555063.

[18] F. Ahmadizar, "A new ant colony algorithm for makespan minimization in permutation flow shops," *Comput. Ind. Eng.*, vol. 63, no. 2, pp. 355–361, Sep. 2012, doi: 10.1016/j.cie.2012.03.015.

[19] Y.-F. Liu and S.-Y. Liu, "A hybrid discrete artificial bee colony algorithm for permutation flowshop scheduling problem," *Appl. Soft Comput.*, vol. 13, no. 3, pp. 1459–1463, Mar. 2013, doi: 10.1016/j.asoc.2011.10.024.

[20] W. Bozejko, J. Pempera, and C. Smutnicki, "Parallel tabu search algorithm for the hybrid flow shop problem," *Comput. Ind. Eng.*, vol. 65, no. 3, pp. 466–474, Jul. 2013, doi: 10.1016/j.cie.2013.04.007.

[21] 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.

[22] F. Z. Boumediene, Y. Houbad, A. Hassam, and L. Ghomri, "A new hybrid genetic algorithm to deal with the flow shop scheduling problem for makespan minimization," in *Proc. Int. Conf. Comput. Intell. Appl.*, Oran, Algeria, 2018, pp. 399–410, doi: 10.1007/978-3-319-89743-1_35.

[23] E. C. D. Siqueira, M. J. F. Souza, S. R. D. Souza, M. F. D. F. 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.*, Jun. 2013, pp. 989–996, doi: 10.1109/CEC.2013.6557675.

[24] A. M. Ahmad, G. M. Khan, and S. A. Mahmud, "Classification of arrhythmia types using Cartesian genetic programming evolved artificial neural networks," in *Proc. Int. Conf. Eng. Appl. Neural Netw.*, 2013, pp. 282–291, doi: 10.1007/978-3-642-41013-0_29.

[25] 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.

[26] 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.

[27] C. R. Reeves, "A genetic algorithm for flowshop sequencing," *Comput. Oper. Res.*, vol. 22, pp. 5–13, Jan. 1995, doi: 10.1016/0305-0548(93)E0014-K.

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

[29] J. C.-H. Pan and H.-C. Huang, "A hybrid genetic algorithm for no-wait job shop scheduling problems," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 5800–5806, Apr. 2009, doi: 10.1016/j.eswa.2008.07.005.

[30] 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.

[31] J. Carlier, "Ordonnancements a contraintes disjonctives," *RAIRO-Oper. Res.*, vol. 12, pp. 333–351, Nov. 1978. [Online]. Available: http://www.numdam.org/item?id=RO_1978__12_4_333_0

[32] Q. Lin, L. Gao, X. Li, and C. Zhang, "A hybrid backtracking search algorithm for permutation flow-shop scheduling problem," *Comput. Ind. Eng.*, vol. 85, pp. 437–446, Jul. 2015, doi: 10.1016/j.cie.2015.04.009.

[33] Y.-Y. Han, Q.-K. Pan, J. J. Liang, and J.-Q. Li, "A hybrid discrete harmony search algorithm for blocking flow shop scheduling," in *Proc. IEEE 5th Int. Conf. Bio-Inspired Comput., Theories Appl. (BIC-TA)*, Sep. 2010, pp. 435–438, doi: 10.1109/BICTA.2010.5645164.

[34] C. Rajendran, "A no-wait flowshop scheduling heuristic to minimize makespan," *J. Oper. Res. Soc.*, vol. 45, no. 4, pp. 472–478, Apr. 1994, doi: 10.1057/jors.1994.65.

[35] S. F. Rad, R. Ruiz, and N. Boroojerdian, "New high performing heuristics for minimizing makespan in permutation flowshops," *Omega*, vol. 37, no. 2, pp. 331–345, Apr. 2009, doi: 10.1016/j.omega.2007.02.002.

[36] *Pakistan Battery Market—Growth, Trends, and Forecast (2020–2025)*. Accessed: Oct. 12, 2020. [Online]. Available: https://www.mordorintelligence.com/industry-reports/pakistan-battery-market

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 *Advances in Production Engineering and Management*. 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 Engineering Sciences 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*, the *International Journal of Intelligent Systems Technologies and Applications*, *Mathematics*, *The International Journal of Advanced Manufacturing Technology*, *Advances in Mechanical Engineering*, *Journal of Ergonomics*, and the *International Journal of Progressive Sciences and Technologies*. His main research interests include 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*, *The International Journal of Advanced Manufacturing Technology*, and the *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, India, in 2002 and 2004, respectively, the Master of Philosophy degree in the application of boolean polynomials from Annamalai University, India, in 2008, and the Doctor of Philosophy degree in operations research from the Jadavpur University, in 2010. He held a postdoctoral position with Pusan National University, South Korea, from 2012 to 2013. He is currently an Associate Professor with the Department of Industrial Engineering, Yonsei University, South Korea. He has dedicated his teaching and research abilities in various universities, including the Darjeeling Government College, India, from 2009 to 2010; Vidyasagar University, India, from 2010 to 2014, Hanyang University, South Korea, from 2014 to 2019; Yonsei University, South Korea (Since 2019). Under his supervision, 13 students have been awarded their Ph.D. and three students are awarded their master's. Since 2010, he has published 159 journal articles in reputed journals of *Applied Mathematics* and *Industrial Engineering*, and he has published one book. He has authored more than 150 research articles, some of which are published in international journals, such as the *International Journal of Production Research*, *Applied Soft Computing*, *Journal of Cleaner Production*, *Computers & Industrial Engineering*, *RAIRO-Operations Research*, *Mathematics*, *Journal of Retailing and Consumer Services*, *Energy Conversion and Management*, *The International Journal of Advanced Manufacturing Technology*, the *International Journal of Fuzzy Systems*, the *International Journal of Environmental Research and Public Health*, *Soft Computing*, *Journal of Manufacturing Systems*, *Robotics and Computer-Integrated Manufacturing*, *Journal of Intelligent & Fuzzy Systems*, *European Journal of Industrial Engineering*, the *International Journal of Applied and Computational Mathematics*, and *Journal of Industrial and Management Optimization*. He is also a member of several learned societies. He is also an editorial board member of some reputed *International Journal of Applied Mathematics* and *Industrial Engineering*. He has served as the Guest Editor for two special issues of two SCIE indexed journals named *Mathematics* and *Energies*. In 2014, his article was selected as the Best Research Paper in 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, an international award from the Korean Institute of Industrial Engineers in 2017 at KAIST, Daejon, South Korea, and the Hanyang University Academic Award as one of the most productive researchers, in 2017 and 2018, consecutively.

MUHAMMAD SAAD received the B.Sc. degree in mechanical engineering from the Institute of Engineering and Fertilizer Research, Faisalabad, in 2010. His main research interests include artificial intelligence, engineering optimization, machine learning, project management, and scheduling.

UZAIR ASAD received the B.Sc. degree in chemical engineering from the National University of Sciences and Technology, Islamabad, Pakistan, in 2019. His main research interests include engineering optimization, machine learning, programming, process improvement, process optimization, and search-based software engineering.