Mitigation of Make Span Time in Job Shop Scheduling Problem Using Gannet Optimization Algorithm

K.R. Anil Kumar[1,*] and J. Edwin Raja Dhas[2]

[1] Research Scholar, Department of Mechanical Engineering, Noorul Islam Centre for Higher Education, Kumarakovil Tamil Nadu- 629180, India.
[2] Head, Department of Automobile Engineering, Noorul Islam Centre for Higher Education, Kumarakovil Tamil Nadu- 629180, India.

Abstract

INTRODUCTION: Effective Job Shop Scheduling (JSS) was required in the manufacturing industry to satisfy demand productivity, reduce production costs, and increase competitiveness in a market that was becoming more active and demanding of a variety of goods.
OBJECTIVES: The Job Shop Scheduling Problem (JSSP) has gained importance in recent years as a result of rising customer demand across a variety of categories, shifting markets due to increased global competition, and the quick development of new technology. The proper scheduling and sequencing of jobs on machines was one of the fundamental and important issues that a shop or factory management faces.
METHODS: Different machines can be found in a shop, and depending on the task, one or more of these equipment may need to be used in a particular order. The aim in correcting this issue might be to reduce the make span. For each machine, the jobs sequencing must be done once the make span had been reduced.
RESULTS: To solve these issues, (GOA) was used to reduce make span time. Both jobs and machines were fed as an input to the proposed optimization and to found optimal job scheduling with low make span time. The outcome of the proposed work was compared the outcomes of various optimization strategies in JSSP in order to minimize the make span time. CONCLUSION: The objective of optimization was to reduce the total amount of time or duration required to complete a task. A proposed gannet optimization method was employed to reduce the make span time in various sectors to resolve the job shop scheduling problem.

Keywords: job shop scheduling problem, makespan, gannet optimization algorithm, machine and jobs.

Received on 07 December 2022, accepted on 12 June 2023, published on 26 June 2023

doi: 10.4108/eetsis.2913

*Corresponding author. Email: anilkumarniu742@gmail.com

1. Introduction

In order to improve production, various manufacturing tasks are allocated to machines at appropriate times. This is known as the job shop scheduling problem (JSSP), also termed as JSS [1]. The goal of JSSP is to determine the best timetable for distributing shared resources over time to competing tasks in order to decrease the total amount of time required to finish all tasks. Since 1950, scheduling has gained a lot of attention due to the direct impact it has on the productivity and expenses of a manufacturing system [23]. Scheduling is the process of allocating a group of resources to a group of activities over a predetermined period of time. In referring to machines, the tasks are referred to as operations and the resources as machines [2]. Thus, selecting a working order that optimises the production system is the process of scheduling. In a workshop, each task is carried out by each machine in a specific order, with each machine only able to handle one task at a time [3]. The vast majority of research focuses on decreasing make span time the total amount of time required to complete every operation on a schedule and improving flow, which is the amount of output or jobs produced per unit of time. The scheduling of shops is an illustration of a nondeterministic polynomial complete issue.

These issues are referred to be "Hard" issues since the processing time increases exponentially as the issue's size increases linearly [24]. The total duration indispensable for all shop floor machines to accomplish all operations on all tasks is referred to as make span time [4].

The main subject of present research in this field is the scheduling application of heuristics and meta-heuristics [5]. Genetic algorithm was first proposed by John Holland in 1975 [9]. For the purpose of solving an optimization problem, a population of possible solutions is generated. A genetic procedure known as crossover combines two chromosomes to create a new chromosome. Implementing the mutation operation should come after crossing. Mutation protects the method from regional extreme points [10]. Although the genetic algorithm performs well, it is expensive, difficult to optimize and consuming much computation time for complex scheduling. Afterward a PSO models is used to solve the JSSP. A flock of birds looking for food served as the model for the evolutionary algorithm known as PSO. A population of randomly generated particles with each having a velocity and position serve as the foundation for this population-based search procedure [11]. The enormously complicated nature of production scheduling problems is also addressed by fuzzy logic, genetic algorithms [6], simulated annealing [8] and particle swarm optimization with Cauchy distribution [7].

Convolutional approaches are the most efficient in cutting down on make-span time, but existing methods also have some minor drawbacks, such as slow convergence and a tendency to not provide an appropriate optimal solution, failure to identify the best optimal value, and inability to identify any additional local optima after the local optima have been attained [12].

Job scheduling is the process of managing and executing tasks on a cloud computing platform, such as Google Cloud. Job scheduling can help you optimize your resource utilization, automate workflows, and improve performance and reliability. Quality of experience framework for Cloud computing (QoC) for monitoring the Quality of Experience (QoE) of the end user using video streaming services in the cloud computing [32], fog computing [33] ,IOT platform [34] ,protect data in cloud computing [35], Quality Of Service (QoS)[36], Quality of Experience(QOE)[37],[38],[39] ,[40],[41] to ensure the performance of executing tasks environment in job shop scheduling but consume more time in execution.

To overcome these drawbacks, a novel proposed method is employed to determine optimal scheduling and minimize make span time. Jobs and machines are initialized to GOA in order to determine the optimal job shop scheduling with low make span time. The main contribution of a novel approach is discussed as follows.

- The advanced gannet optimization algorithm (GOA) is utilised to determine a typical job shop schedule which can reduce the make span time in a sector.
- At first, a number of machines and the number of jobs are given as an input to GOA.
- A single machine completes a task in a given period, and a GOA analyses each machine's process and the entire work volume to schedule task effectively.
- The performance of the proposed method is compared to existing approaches, and the more efficient average benchmark make span times provided by the proposed GOA are more in line with the real average benchmark make span times.

Rest of the manuscript structure are as follows; section 2 describes several researches related to job shop scheduling and minimize make span time. Section 3 describes proposed method problem formulation. Section 4 describes proposed methodology for job shop scheduling. Section 5 describes proposed experimental investigation and discussion. Section 6 describes conclusion.

2. Literature Review

Many strategies are developed to increase production while reducing make span time. The following techniques for job shop scheduling and minimising make span time were covered.

Kress and Müller [13] had presented a solution of FJSP via mathematical model. Two mathematical approaches were utilized to analyse the performance of the method namely, mixed-integer programming model and constraint programming model. An impact of the method is taking long time for calculating numerous number of machines and workers. Zhang, et al. [14] had presented an Evolving Scheduling Heuristics through Genetic Programming to solve flexible job-scheduling scheduling problem. The outcome showed the method was more efficaciously achieved interpretable scheduling heuristics with more minuscule size and less unique features. The method contains some constraints like that the rule is engender by CCGP up to the selective features. Due to the present of more features in rules, the method is arduous to explicate the rules.

Wang and Zhu [15] had suggested a method of mathematical model to resolve the flexible job shop scheduling problem (FJSP). A hybrid algorithm which combines genetic algorithm and tabu search is to solve the FJSP-SDST-LT. The method gives same outcome as another perpetual models, however, the method is not fit for astronomically immense number of variables and equation since it takes more time to give a result. Juvin et al. [16] had presented Logic-Based Benders Decomposition for the Pre-emptive Flexible Job-Shop Scheduling Problem. Mathematical and constraint programming models enable the resolution of this problem for small instances. However, as an NP-hard problem, the cost of solving grows rapidly when considering larger instances. In this regard, propose a logic-based Benders decomposition that relies on an efficient branch-and-bound procedure to solve the sub problem representing a pure (non-flexible) pre-emptive job-shop scheduling problem. Yet, the dimension of a number of jobs and machine in this method is not dependent on processing time so the scheduling time is erroneous.

Dehghan-Sanej, et al. [17] had presented a simulation model to resolve JSSP as well as reducing make span time. The model had been minimizing the make span time using meta-heuristic algorithms. The method outcome is very well, yet it is only suited for diminutive quandaries since the simulation of immensely quandary take more execution time. Dai et al. [18] had developed an optimization model with numerous objectives was developed with the aim of reducing energy consumption and to solve a problem of make span with flexible workshop scheduling based on by transportation constraints. An improved genetic algorithm was used to completely resolve the problem. This model offers high precision and quick computation, but it falls short of effectively reducing energy usage since it does not account for unforeseen occasions like task cancellation, machine failure, and rush orders.

Meng et al. [19] presented a method for solving the JSSP that makes use of six dynamic MILP models and an On/Off switching strategy. Numerical tests were performed using CPLEX SLOVER to confirm the accuracy and efficiency of each MILP model. But it is less effective and more difficult. Mihoubi, et al. [20] had represented that in order to the realistic FJSSP and resolve the RS, a GA-based computational technique be used to control the employment of a hybrid neural component and a DES model. The system, however, was unable to gather accurate and reliable data, which was necessary for it to produce results that were satisfactory.

Caldeira and Gnanavelbabu [21] had suggested that an improved Jaya algorithm be created to address the issue of job shop scheduling. 203 benchmark instances were used to evaluate this method's performance. However, this approach is inappropriate for the industry's multi-objective job scheduling. Li, et al. [22] had presented to reduce the JSSP with transportation and setup times, the improved Jaya (IJaya) algorithm has been improved. An integer programming technique was created to reduce energy usage and achieve goals. However, this is overly intricate and does not offer greater precision.

Wang, et al. [25] had developed an improved GA which utilised local search to efficiently and effectively solve Sudoku puzzles, but it is still needed to increase the performance of LSGA by integrating it with other Sudoku-solving techniques. Li, J. Y et al. [26] had developed a novel three-layer DDE framework with adaptive resource allocation (DDE-ARA), including the algorithm layer for evolving various differential evolution (DE) populations, the dispatch layer for dispatching the individuals in the DE populations to different distributed machines, and the machine layer for accommodating distributed computers.

Author name	Techniques	Advantage	Disadvantage
Ge, Yu et al. [27]	distributed memetic algorithm	Enhancing database privacy and utility	Consume more time
Li, Zhan et al. [28]	DDEA framework with perturbation-based ensemble surrogates	Efficient mechanism to enhance performance	But its expensive
Ge, Caoa et al. [29]	set-based adaptive distributed differential evolution algorithm	An island model to maintain population diversity	Hybrid mechanism is not work out.
Ge, Wang et al. [30]	information-driven genetic algorithm	To achieve optimal anonymization based on attribute generalization and record suppression.	Consume more time
Ge, Orlowska et al. [31]	similarity-based alignment operator	To adjust the fragment orders in different database fragmentation solutions.	complication

The above methods minimizes make-span time most effectively [17], [26] but also have some major drawbacks, take more time [13], [25] like slowdown convergence [21], [29] a tendency to not provide an appropriate optimal solution [16], difficulty to determine the best present optimal value [22], [31].To overcome these drawback proposed method is employed which find the optimal scheduling process with low make span time.

3. Problem formulation

The task of assigning a number of jobs to machines so that one or more criteria are optimised is termed as JSSP. A particular machine must process each piece of work continuously for a predetermined period of time. Arranging j^{th} job on m^{th} machine with goal of abbreviating the scheduled low make span time of all jobs. Each of the j^{th} jobs consists of numerous tasks that must be completed on m^{th} various machines within a specified amount of time. Each work should be performed exactly once on each machine, and only one machine can perform the same task at once. To improve production efficiency, reduce expenses, and/or improve product quality, effective solutions to the JSSP are essential. Modelling of job shop scheduling is illustrated in figure 1.

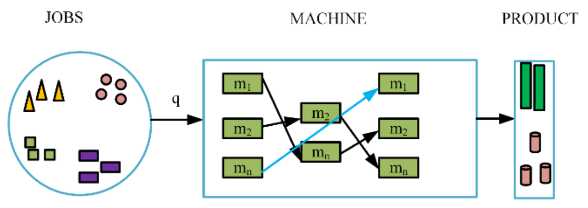


Figure 1. Job scheduling in an industry.

Let as consider, j as task/jobs available in a company/industry, m denote machines, n and q denotes number of machine and queue, and queues respectively. As $j_i(t)$ denotes the i^{th} job with its processing time$t$, and q_i is termed as m^{th} machine j^{th} queue. Each machine has various execution time and scheduling time for complete a task. The jobs and machines of an industry is stated as below,

$$J = 1,2,3, \dots .j, \text{ and } M = 1,2,3, \dots .m \tag{1}$$

And the queue is expressed as,

$$q = 1,2,3, \dots ., j \times m, j \times m + 1 \tag{2}$$

Where, $j \times m + 1$ denotes the last operation of a machine. The make span time of a machine is stated as,

$$\min F_{n \times m+1} \tag{3}$$

$$s.t;\ F_k < F_j - T_j,\ \ j = 1, 2, \dots \dots, n \times m + 1; k \in P_j \tag{4}$$

$$F_j \geq 0,\ \ j = 1, 2, \dots \dots, n \times m + 1 \tag{5}$$

$$\textstyle\sum_{j \in A(t)} e_{jm} \leq 1, \quad m \in M; t \geq 0 \tag{6}$$

The objective function used to reduce the make span time is indicated by Eqn. (3). Eqn. (4) denotes the order of priority among the process. A machine can only process one task at a time, according to equation (5), and Eqn. (6) define the non-negative finishing times. Where, $A(t)$ represents the total number of tasks processed at any given time$t$, and F_j and T_j stand for the completion time and operation time of operation$j$, correspondingly. When a job j needs to be processed on a machine$m$, e_{jm} is set to 1, else it is set to zero. The activities are interconnected, and due to given constraints, each action j must be scheduled after all earlier operations P_j have finished. Operation j can only be planned if the relevant machine is idle. To obtain the optimal make span quickly and with fewer iterations, the estimation of an appropriate value and a quick solution to satisfying the problems are important. Utilize the fact that using a novel optimization to solve the problem quickly and minimizing the make span time.

4. Proposed methodology for job shop scheduling

Job shop scheduling and make span time reduction are significant issues for the industrial sector. A benchmark job schedule is designed to reduce make span time while maintaining high production. In order to reduce the make span time and assign the suitable job of a machine, a novel gannet optimization approach is introduced to schedule a job. The schedules are generated using a priority rule, where the novel gannet optimization algorithm determines the priorities. Both machine and jobs are initialized to find the optimal schedule which have low make span time. GOA analyses the input to plan the task at a low make span time by taking into account the total number of machines and jobs. Through GOA, the solutions are updated to discover the best one. Until the best ideal option is identified, the procedure is repeated. Effectively reducing the make span time is the suggested way. A detail study of proposed GOA is given below.

4.1. Background of Gannet optimization algorithm

This section will discuss the gannet optimization algorithm, a new meta-heuristic optimization technique that was inspired by gannets' predatory behaviour. To imitate the predatory behaviour of pond geese, two stages of exploration and exploitation are proposed in the pond goose optimization method. There are a total of four different predation behaviours present in the exploration and exploitation stages: V-shaped dive mode, U-shaped dive mode, rapid rotation, and random wandering.

Initialization phase

It starts with the collection of random results represented in Eq. (7), at that time the best solution is considered to be the optimal best solution.

$$X = \begin{bmatrix} x_{1,1} & \dots & x_{1,j} & \dots & x_{1,Dim-1} & x_{1,Dim} \\ x_{2,1} & \dots & x_{2,j} & \dots & x_{2,Dim-1} & x_{2,Dim} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N-1,1} & \dots & x_{N-1,j} & \dots & x_{N-1,Dim-1} & x_{N-1,Dim} \\ x_{N,1} & \dots & x_{N,j} & \dots & x_{N,Dim-1} & x_{N,Dim} \end{bmatrix} \tag{7}$$

$$x_{i,j} = r_1 * \left(UB_j - LB_j\right) + LB, i = 1,2 \dots, N, j = 1,2, \dots, Dim \tag{8}$$

Where, x_i signifies the position of the ith gannet. Each $x_{i,j}$ in the matrix X can be calculated by equation (8). N is the number of individuals in the population. UB_j and LB_j are the lower and upper bounds of the jth dimension of the problem. r_1 is a random number between 0 and 1, and Dim signifies the dimensional size of the problem.

Exploration phase

The depth of the prey's dive is taken into account when adjusting the gannet's dive pattern after it has located its prey in the water. A long, deep U-shaped dive and a quick, shallow V-shaped dive are the two types of dives available.

$$t = 1 - \frac{It}{Tmax_{iter}} \qquad (9)$$

$$a = 2 * cos\,(2 * \pi * r_2) * t \qquad (10)$$

$$b = 2 * V\,(2 * \pi * r_3) * t \qquad (11)$$

$$\mathrm{V(x)} = \begin{cases} -\frac{1}{\pi} * \mathrm{x} + 1, \mathrm{x} \in (0, \pi) \\ \frac{1}{\pi} * \mathrm{x} - 1, \mathrm{x} \in (\pi, 2\pi) \end{cases} \qquad (12)$$

Where, $Tmax\_iter$ is the maximum number of iterations, It the number of current iterations, r_2 and r_3 are both random numbers between 0 and 1.

Exploitation phase

The gannet rushes into the water in the methods mentioned above, and two further actions are needed to take advantage of it. In order to catch the fish desperately attempting to escape, the gannet also expends a great deal of energy. Explain the capture capacity in equation (13).

$$Capturability = \frac{1}{R * t2} \qquad (13)$$

$$t2 = 1 + \frac{It}{Tmax_{iter}} \qquad (14)$$

$$R = \frac{M * vel^2}{L} \qquad (15)$$

$$L = 0.2 + (2 - 0.2) * r_6 \qquad (16)$$

Where $vel = 1.5m/s$ is the gannet's speed in the water while ignoring the current level of water resistance, M=2.5kg is the bird's weight, and r_6 is a random number between 0 and 1. If the gannet's capturing range is within the catchable prey's range, the location is updated with a sudden turn; otherwise, if the gannet is unable to catch the flexible fish, it will wander in a Levy way to look for its next target at random Eq (17).

$$MX_i(t+1) = \begin{cases} t * delta * \left(X_i(t) - X_{Best}\,(t)\right) + X_i(t), & Capturability \geq c \\ X_{best}\,(t) - \left(X_i(t) - X_{Best}\,(t)\right) * P * t, & Capturability\ < c \end{cases} \qquad (17)$$

$$delta\ = Capturabilty * |X_i(t) - X_{best}(t)| \qquad (18)$$

$$P = Levy\,(Dim) \qquad (19)$$

Where $X_{Best}\,(t)$ is the best performing individual in the current population, c=0.2 is a constant whose value was determined after several experiment and $Levy$ () is the Levy fight function, which can be obtained from eq. (19).

$$Levy\,(Dim) = 0.01 * \frac{\mu * \sigma}{|v|^{\frac{1}{\beta}}} \qquad (20)$$

$$\sigma = \left(\frac{\Gamma(1+^{\beta}) * \ sin\left(\frac{\pi^{\beta}}{2}\right)}{\Gamma\left(\frac{1+^{\beta}}{2}\right) *^{\beta} * 2^{\frac{^{\beta}-1}{2}}} \right)^{\frac{1}{\beta}} \qquad (21)$$

4.2. GOA based optimal job shop scheduling for minimizing make span time

Step 1: Initialization
The GOA population's random solution is initiated at this moment, and the best solution is considered to be the best overall solution. Based on this initialize, the random selection of number of machine (m) and number of jobs(j) are initialized to find the best optimal scheduling solution.

$$J = \{J_1, J_{2,} \dots, J_n\} \text{ and } M = \{M_1, M_2, \dots, M_z\} \qquad (22)$$

Where, n and z denotes number of populations.
Step 2: Fitness function
The objective function is a way to find the optimum solution by maximising or minimising the functions. In the proposed model, the fitness function is considered as a make span time.

$$Fitness = \min\ F_{n*m+1} \qquad (23)$$

Step 3: Update the solution
The two phases of GOA's design are exploration and development. The exploration process contains the U- and V-shaped diving patterns of gannets to determine the best area, while the development phase uses a sudden rotation and random walk to develop a better solution. The fitness value of each iterations are updated to find the best solution. Using eq. (17) to update the solution to find the best one. Figure 2 shows the Flow chart of proposed job scheduling model.
Step 4: Process will stop to get a best global optimal value

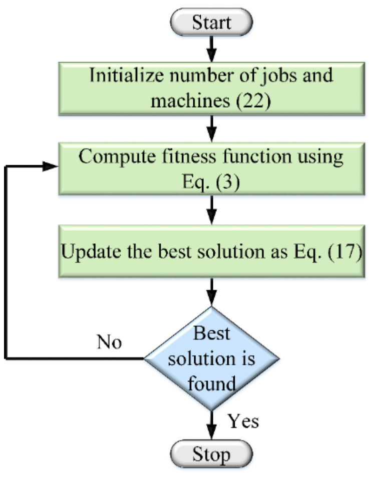

Start
Initialize number of jobs and machines (22)
Compute fitness function using Eq. (3)
Update the best solution as Eq. (17)
Best solution is found
No
Yes
Stop

Figure 2. Flow chart of proposed job scheduling model.

5. Experimental investigation and discussion

This section examines 23 different benchmark problems and the impact of three different optimization approaches, including DE, SSO, and ESSO. Comparing the GOA methodology to other JSSP algorithms, it takes the least amount of time to complete the task. When compared to other conventional approaches, it is clear that soft computing techniques have a larger potential for solving Job Shop Scheduling Problems (JSSP). Proposed methodology seems to have a faster calculation time than traditional problem-solving techniques, which is its main advantage. The complete proposed model is implemented, and the performance is analysed using MATLAB platform. Table -1 displays the computational outcomes of several techniques on benchmark problems. It shows that GOA method outperforms other algorithms.

23 benchmark problems are taken from table[1] (a) and [1](b) to calculate the make span time in various algorithms. A comparison is made between the ant colony optimization algorithm (ACA), the hybrid generic algorithm (HGA), the differential evolutionary algorithm (DE), and the social spider optimization algorithm (SSO), genetic algorithm (GO), particle swarm optimization (PSO), opposition based genetic algorithm, the greedy randomised adaptive search procedure algorithm (GRASP), and opposition-based particle swarm optimization with Cauchy distribution. Only a low span is provided by the suggested solution, which also resolves all JSSP quandaries. The appropriate GOA solution for the benchmark problems LA 12, LA 13, LA 14, LA 15, LA 17, LA 18, LA 19, LA 20, LA 22, LA 23, LA 24, LA 25, LA 26, LA 28, LA 29, LA 30, LA 32, LA 33, LA 34, LA 35, LA 37, LA 38, and LA 39 is 1038, 1149, 1291, 1206, 783, 843, 841, 900, 925, 1029, 944, 975, 1214, 1215, 1160, 1353, 1848, 1717, 1720, 1886, 1395, 1190, 1232.

Table 1 (a). Investigating state -of -the art methodologies on benchmark problems

Problem	B.M	HGA	ACA	GRASP	DE	SSO	ESSO
LA 12	1139	1039	1039	1039	1039	1039	1039
LA 13	1350	1150	1150	1150	1150	1150	1150
LA 14	1492	1292	1292	1292	1292	1292	1292
LA 15	1307	1207	1207	1207	1207	1207	1207
LA 17	794	784	789	784	789	788	785
LA 18	899	848	848	848	848	848	848
LA 19	887	844	842	842	842	842	842
LA 20	955	907	902	907	902	902	902
LA 22	972	935	938	960	934	934	927
LA 23	1088	1032	1032	1032	1032	1032	1032
LA 24	987	953	959	978	951	947	947
LA 25	997	981	977	1028	977	977	977
LA 26	1318	1218	1218	1271	1218	1218	1218
LA 28	1316	1216	1227	1293	1227	1217	1217
LA 29	1252	1160	1177	1293	1177	1170	1164
LA 30	1455	1355	1355	1368	1355	1355	1355
LA 32	1950	1850	1850	1850	1850	1850	1850
LA 33	1819	1719	1719	1719	1719	1719	1719
LA 34	1821	1721	1725	1753	1725	1723	1723
LA 35	1988	1888	1888	1888	1888	1888	1888
LA 37	1497	1407	1412	1457	1408	1403	1397
LA 38	1296	1196	1196	1267	1196	1196	1196
LA 39	1333	1233	1240	1290	1238	1238	1233

Table 1 (b). Investigating state-of -the art methodologies on benchmark problems

Problem	GA	OGA	PSO	OPSO	GOA
LA 12	1039	1039	1039	1039	1038
LA 13	1150	1150	1150	1150	1149
LA 14	1292	1292	1292	1292	1291
LA 15	1207	1207	1207	1207	1206
LA 17	788	787	787	786	783
LA 18	848	848	848	848	843

LA 19	842	842	842	842	841
LA 20	902	902	902	902	900
LA 22	934	933	932	931	925
LA 23	1032	1031	1032	1032	1029
LA 24	978	979	965	964	944
LA 25	980	979	978	977	975
LA 26	1229	1220	1219	1218	1214
LA 28	1293	1277	1267	1239	1215
LA 29	1293	1290	1287	1100	1160
LA 30	1368	1360	1358	1357	1353
LA 32	1850	1849	1849	1850	1848
LA 33	1719	1718	1719	1719	1717
LA 34	1753	1769	1755	1724	1720
LA 35	1888	1889	1887	1889	1886
LA 37	1399	1398	1396	1397	1395
LA 38	1198	1197	1195	1194	1190
LA 39	1256	1245	1244	1237	1232

Similar to this, different standard deviation conditions are used to compute the benchmark issues. As shown in Table[2], the terms actual make span time, best GOA, worst, average, standard deviation, and ARPD refer to the variables, respectively. The proposed model standard deviation denotes the consistency and dependability of each benchmark problem. A low standard deviation value indicates that the algorithm is more stable and dependable when searching for the best solution.

Table 2. Investigating the performance of GOA with different standard measures

Problem	Actual make span time	Best GOA	Worst	Average	Std. Deviation	ARPD
LA 12	1139	1038	1225	1082	79.8289	0.237247
LA 13	1350	1149	1236	1193	35.5183	0.295217
LA 14	1492	1291	1365	1328.5	30.2112	0.214009
LA 15	1307	1206	1347	1277	57.5634	0.528998
LA 17	794	783	952	868.5	68.9955	0.892994
LA 18	899	843	989	918.5	59.6159	0.709316
LA 19	887	841	987	914.5	59.6047	0.780285
LA 20	955	900	1058	980	64.5049	0.679047
LA 22	972	925	1049	988	50.6249	0.582524
LA 23	1088	1029	1153	1092.5	50.6277	0.460756
LA 24	987	944	1090	1018.5	59.6084	0.647307
LA 25	997	975	1098	1037.5	50.2167	0.467247
LA 26	1318	1214	1422	1320	84.9208	0.655993
LA 28	1316	1215	1569	1393	144.5206	1.211586
LA 29	1252	1160	1396	1280	96.3512	0.72122
LA 30	1455	1353	1765	1560	168.1989	1.05572
LA 32	1950	1848	2059	1954.5	86.1416	0.373514
LA 33	1819	1717	2068	1893.5	143.2959	0.86911
LA 34	1821	1720	2012	1867.5	119.2105	0.595183
LA 35	1988	1886	2225	2056.5	138.3969	0.673199
LA 37	1497	1395	1823	1610	174.7309	1.551181
LA 38	1296	1190	1426	1311	96.3569	0.627508

LA 39	1333	1232	1562	1397.5	134.7221	0.999189

The average relative percentage deviation (ARPD) is the objective of reducing the overall time delay by properly assigning and ordering the work on the same machine. The following equation is used to determine ARPD:

$$\text{ARPD} = \sum_{i=1}^{R} \frac{(\text{Best Solution}_i - \text{Well known Solution})}{\text{Well known Solution}} \times \left(\frac{100}{R}\right)$$

Where, $\mathit{Best\ solution}$ is the make spam by GOA algorithm in each run, $\mathit{Well\ known\ Solution}$ is the optimal or lowest known upper bound for instances and R is the number of runs.

Analysis of three optimization algorithm's convergence performance on 23 different benchmark issues. Through a continuous number of iterations, these investigations are carried out to determine three optimization strategies function. It is clear from the results that the proposed GOA outperforms existing methods in terms of obtaining values with a minimum make span time and a higher speed of convergence throughout the process. The effectiveness of the implemented optimization techniques is depicted in the following graphical representation. The convergence graph for LA 12, LA13, LA14, and LA15 is shown in Figure 3. At the start of the iteration in LA 12 proposed method reach make span time 1040 in 90 iteration. But the existing method ESSO, SSO reach 1040 make span time in 200 iteration and DE reach 1040 make span time in 250 iteration. In LA 13 proposed method reach 1150 at 55 iteration, but ESSO and SSO reach 240 and 260 iteration and DE reach 280 iteration. In LA14 proposed method reach 1295 make span time at 60 iteration, but existing method ESSO and SSO, DE reach 285 iteration. In LA 15 proposed method reach make span time 1208 its 100 iteration .But existing method ESSO reach make span time 1209 on its 200 iteration ,SSO and DE reach 240 iteration.

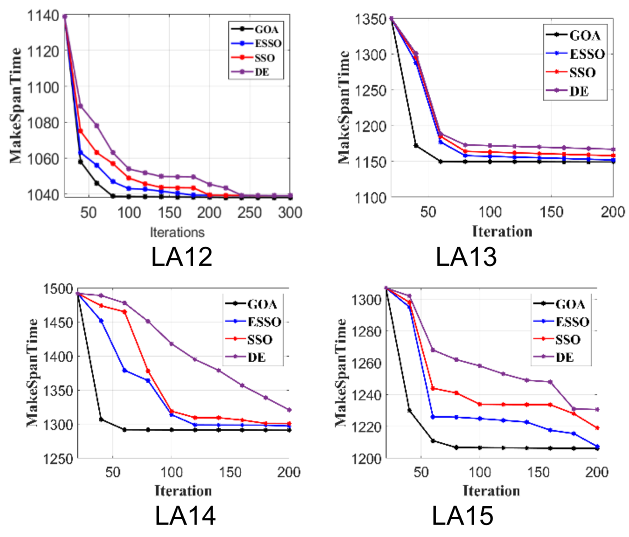


Figure 3. Convergence graph for LA 12, LA 13, LA 14, LA15.

Figure 4 present the convergence graph for LA 16, LA 17, LA 18, and LA 19. In LA16 proposed method reach make span time 783 on its 220 iteration. But existing method ESSO reach make span time 785 on its 260 iteration and SSO reach make span time 788 on its 290 iteration and DE reach make span time 789 on its 290 iteration. In LA17 proposed method reach make span time 843 on its 60 iteration. But existing method ESSO reach make span time 849 on its 220 iteration and SSO, DE reach 280 iteration. In LA 18 proposed method reach make span time 841 on its 140 iteration and existing method ESSO reach make span time 842 on its 220 iteration and SSO, DE reach make span time 843 on its 280 iteration .In LA 19 proposed method reach make span time 900 on its 160 iteration .But existing method ESSO reach make span time 902 on its 120 iteration and SSO reach make span time 903 on its 260 iteration, DE reach make span time 903 on its 280 iteration.

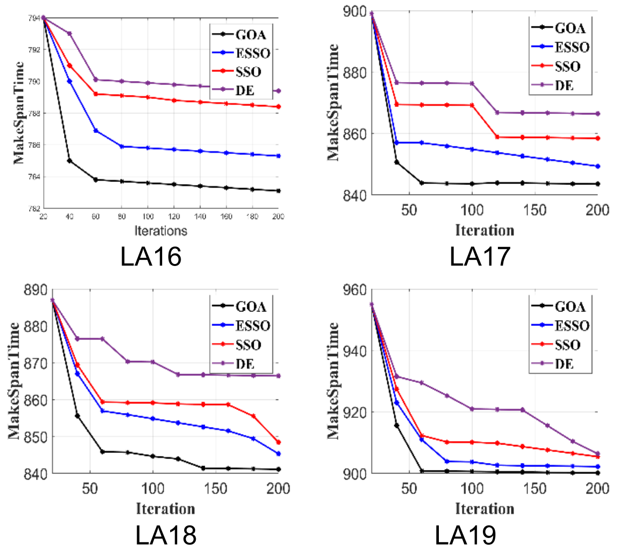


Figure 4. Convergence graph for LA 16, LA 17, LA 18, LA19

Figure 5 present convergence graph for LA 20, LA 21, LA 22, and LA23. In LA 20 proposed method reach make span time 926 on its 60 iteration. But existing method ESSO reach make span time 927 on its 100 iteration and SSO reach make span time 936 on its 100 iteration, DE reach make span time 937 on its 120 iteration. In LA 21 proposed method reach make span time 1030 on its 80 iteration. But existing method ESSO reach make span time 1033 on its 80 iteration and SSO reach make span time 1033 on its 260 iteration, DE reach make span time 1033 on its 260 iteration. In LA 22 proposed method reach make span time 945 on its 80 iteration. But existing method ESSO reach make span time 948 on its 240 iteration and SSO reach on its 240 iteration respectively ,DE reach make span time 951 on its 240 iteration. In LA 23 proposed method reach make span time 975 on its 220 iteration. But existing method

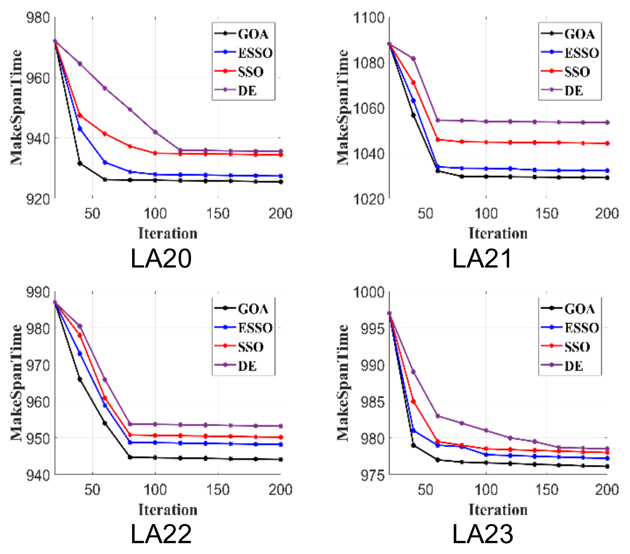
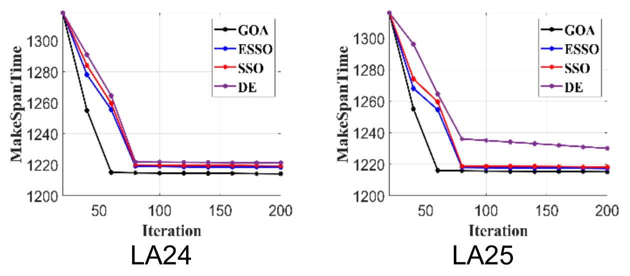


Figure 5. Convergence graph for LA 20, LA 21, LA 22, LA23

Figure 6 present convergence graph for LA 24, LA 25, LA 26, and LA27. In LA 24 proposed method reach make span time 1216 on its 60 iteration. But existing method ESSO reach make span time 1220 on its 80 iteration and SSO reach on its 80 iteration respectively, DE reach make span time 1222 on its 80 iteration. In LA 25 proposed method reach make span time 1216 on its 60 iteration. But existing method ESSO reach make span time 1218 on its 80 iteration and SSO reach make span time 1219 on its 240 iteration, DE reach make span time 1230 on its 260 iteration. In LA 26 proposed method reach make span time 1160 on its 80 iteration. But existing method ESSO reach make span time 1164 on its 100 iteration and SSO reach on its 1170 on its 100 iteration ,DE reach make span time 1178 on its 120 iteration. In L27 proposed method reach make span time 1354 on its 100th iteration. But existing method ESSO reach make span time 1356 on its 100 iteration and SSO reach 1356 on its 260 iteration and DE reach make span time 1358 on its 260 iteration.



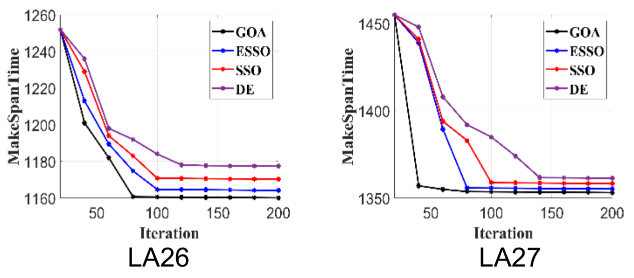


Figure 6. Convergence graph for LA 24, LA 25, LA 26, LA27

Figure 7 present convergence graph for LA 28, LA 29, LA 30, and LA31. In LA 28 proposed method reach make span time 1850 on its 100 iteration. But existing method ESSO reach make span time 1852 on its 280 iteration and SSO reach make span time 1858 on its 260 iteration respectively, DE reach make span time 1859 on its 260 iteration. In LA 29 proposed method reach make span time 1719 on its 60 iteration. But existing method ESSO reach make span time 1720 on its 100 iteration and SSO reach make span time 1722on its 120 iteration, DE reach make span time 1730 on its 140 iteration. In LA 30 proposed method reach make span time 1720 on its 160 iteration. But existing method ESSO reach make span time 1724 on its 80 iteration and SSO reach on its 1170 on its 100 iteration, DE reach make span time 1728 on its 200 iteration. In LA31 proposed method reach make span time 1388 on its 100 iteration. But existing method ESSO reach make span time 1389 on its 100 iteration and SSO reach 1404 on its 180 iteration and DE reach make span time 1409 on its 220 iteration.

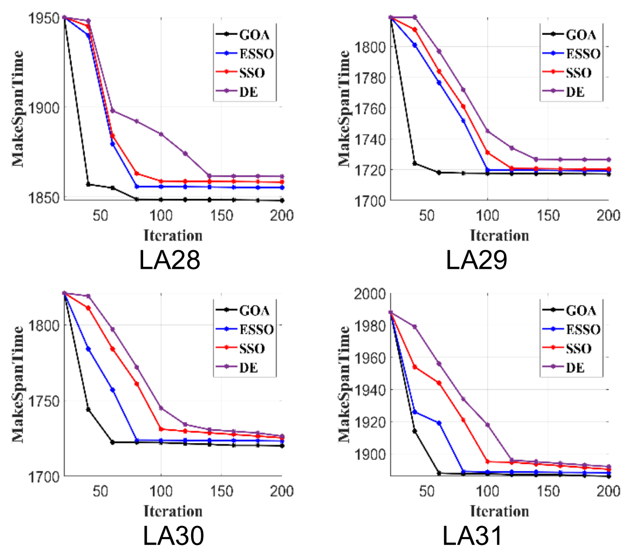


Figure 7. Convergence graph for LA 28, LA 29, LA 30, LA31

Figure 8 present convergence graph for LA 32, LA 33, and LA 34. In LA 32 proposed method reach make span time 1388 on its 100 iteration. But existing method ESSO reach make span time 1389 on its 100 iteration and SSO reach make span time 1404 on its 180 iteration respectively, DE reach make span time 1409 on its 240 iteration. In LA 33 proposed method reach make span time 1235 on its 100 iteration. But existing method ESSO reach make span time 1236 on its 240 iteration and SSO reach make span time 1240 on its 260 iteration, DE reach make span time 1240 on its 260 iteration. In LA 34 proposed method reach make span time 1880 on its 180 iteration. But existing method ESSO reach make span time 1882on its 80 iteration and SSO reach on its 1886 on its 230 iteration, DE reach make span time 1888 on its 230 iteration.

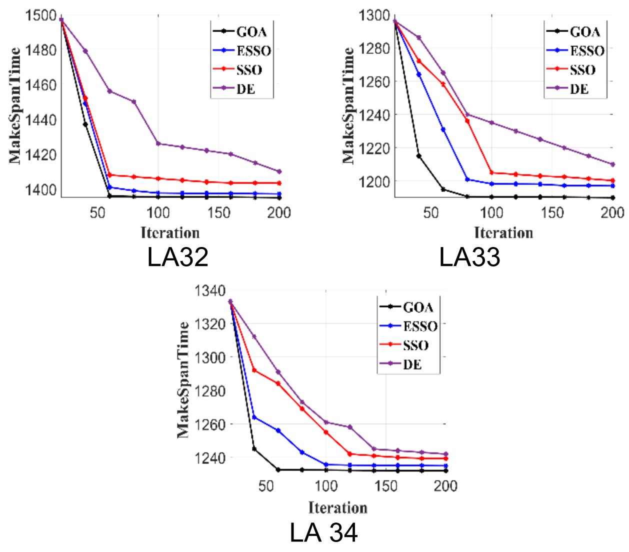


Figure 8. Convergence graph for LA 32, LA 33, LA 34

Compared to existing convergence graph proposed method is effectively convergence on high speed and it is minimizing the make span time. It is used for all industrial application for job scheduling and minimize make span time.

6. CONCLUSION

Scheduling is important in the industries since it determines to perform task at a machine in time. Giving a machine a suitable job requires more time and effort when scheduling manually. An optimization strategy is used to schedule a work in a successful machine at a specific time in order to reduce make span time. Here, the make span time is effectively truncated using an advanced GOA method. By comparing the performance of GOA in JSSP to that of other optimization techniques, standard benchmark values are evaluated. The research ability is promoted by this algorithm, which causes optimization to reveal a wide range of solutions. Additionally, this architecture ensures that the proposed GOA algorithm always directs search units to examine the most promising areas of the search space, which further enables this approach to produce excellent results. The experimental results provide strong support for the GOA in JSSP measure with benchmark data. This study shows the GOA dominance in JSSP and its ability to produce far better solutions for make span time minimization. The GOA algorithm in JSSP provides information it can be used to assess the solution to other combinatorial optimization issues. In addition, future work concentrate on minimize make span time effectively using new novel approach.

Acknowledgements.

Funding

The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Conflict of Interest

The authors declared that they have no conflicts of interest to this work. We declare that we do not have any commercial or associative interest that represents a conflict of interest in connection with the work submitted.

Availability of data and material

Not applicable

Code availability

Not applicable

Author contributions

The corresponding author claims the major contribution of the paper including formulation, analysis and editing. The co-author provides guidance to verify the analysis result and manuscript editing.

Compliance with ethical standards

This article is a completely original work of its authors; it has not been published before and will not be sent to other publications until the journal's editorial board decides not to accept it for publication.

References

[1] Viana MS, Contreras RC, Morandin Junior O. A New Frequency Analysis Operator for Population Improvement in Genetic Algorithms to Solve the Job Shop Scheduling Problem. Sens. 2022 Jun 17; 22(12):4561.

[2] Pourghaffari A, Barari M, Sedighian Kashi S. An efficient method for allocating resources in a cloud computing environment with a load balancing approach. Concurr. Comput. Pract. Exp. 2019 Sep 10; 31(17):e5285.

[3] Zhang M, Tao F, Nee AY. Digital twin enhanced dynamic job-shop scheduling. J. Manuf. Syst. 2021 Jan 1; 58:146-56.

[4] Gong X, De Pessemier T, Martens L, Joseph W. Energy- and labor-aware flexible job shop scheduling under dynamic electricity pricing: A many-objective optimization investigation. J. Clean. Prod. 2019 Feb 1; 209:1078-94.

[5] Park J, Chun J, Kim SH, Kim Y, Park J. Learning to schedule job-shop problems: representation and policy

learning using graph neural network and reinforcement learning. Int. J. Prod. Res. 2021 Jun 3; 59(11):3360-77.

[6] Tafakkori K, Tavakkoli-Moghaddam R, Siadat A. Sustainable negotiation-based nesting and scheduling in additive manufacturing systems: A case study and multi-objective meta-heuristic algorithms. Eng. Appl. Artif. Intell. 2022 Jun 1; 112:104836.

[7] Anuar NI, Fauadi MH. A Study on Multi-Objective Particle Swarm Optimization in Solving Job-Shop Scheduling Problems. Int. J. Comput. Inf. Syst. Ind. Manag. Appl. 2021; 13:051-61.

[8] Frausto-Solis J, Hernández-Ramírez L, Castilla-Valdez G, González-Barbosa JJ, Sánchez-Hernández JP. Chaotic multi-objective simulated annealing and threshold accepting for job shop scheduling problem. Math. Comput. Appl. 2021 Jan 12; 26(1):8.

[9] Zolpakar NA, Lodhi SS, Pathak S, Sharma MA. Application of multi-objective genetic algorithm (MOGA) optimization in machining processes. InOptimization of manufacturing processes 2020 (pp. 185-199). Springer, Cham.

[10] Park JS, Ng HY, Chua TJ, Ng YT, Kim JW. Unified genetic algorithm approach for solving flexible job-shop scheduling problem. Appl. Sci. 2021 Jul 13; 11(14):6454.

[11] Anil Kumar KR, Das ER. Genetic Algorithm and Particle Swarm Optimization in Minimizing MakeSpan Time in Job Shop Scheduling. InProceedings of ICDMC 2019 2020 (pp. 421-432). Springer, Singapore.

[12] Anil A, Venkateswarlu K, Srinivasan M, Kumar S. Solving Job Shop Scheduling Problem With the Aid of Evolution of Cub to Predator (ECP). Int. j. adv. res. sci. eng. technol. 2020;11(11).

[13] Kress D, Müller D. Mathematical models for a flexible job shop scheduling problem with machine operator constraints. IFAC-PapersOnLine. 2019 Jan 1; 52(13):94-9.

[14] Zhang F, Mei Y, Nguyen S, Zhang M. Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling. IEEE Trans Cybern. 2020 Oct 20; 51(4):1797-811.

[15] Wang Y, Zhu Q. A hybrid genetic algorithm for flexible job shop scheduling problem with sequence-dependent setup times and job lag times. IEEE Access. 2021 Jul 9; 9:104864-73.

[16] Juvin C, Houssin L, Lopez P. Logic-Based Benders Decomposition for the Preemptive Flexible Job-Shop Scheduling Problem. Available at SSRN 4068164.

[17] Dehghan-Sanej K, Eghbali-Zarch M, Tavakkoli-Moghaddam R, Sajadi SM, Sadjadi SJ. Solving a new robust reverse job shop scheduling problem by meta-heuristic algorithms. Eng. Appl. Artif. Intell. 2021 May 1; 101:104207.

[18] Dai M, Tang D, Giret A, Salido MA. Multi-objective optimization for energy-efficient flexible job shop scheduling problem with transportation constraints. Robot. Comput.-Integr. Manuf. 2019 Oct 1; 59:143-57.

[19] Meng L, Zhang C, Shao X, Ren Y. MILP models for energy-aware flexible job shop scheduling problem. J. Clean. Prod. 2019 Feb 10; 210:710-23.

[20] Mihoubi B, Bouzouia B, Gaham M. Reactive scheduling approach for solving a realistic flexible job shop scheduling problem. Int. J. Prod. Res. 2021 Oct 2; 59(19):5790-808.

[21] Caldeira RH, Gnanavelbabu A. Solving the flexible job shop scheduling problem using an improved Jaya algorithm. Comput Ind Eng. 2019 Nov 1; 137:106064.

[22] Li JQ, Deng JW, Li CY, Han YY, Tian J, Zhang B, Wang CG. An improved Jaya algorithm for solving the flexible job shop scheduling problem with transportation and setup times. Knowl Based Syst. 2020 Jul 20; 200:106032.

[23]]Lu, Y., Lu, J., & Jiang, T. (2019). Energy-conscious scheduling problem in a flexible job shop using a discrete water wave optimization algorithm. IEEE Access, 7, 10156101574.

[24] Narayanan, P. S., Kumar, N. S., Potluru, R., & Mohanavelu, T. (2022). Job shop scheduling using heuristics through Python programming and excel interface. Decision Making: Applications in Management and Engineering, 5(2), 201-2018.

[25] Wang, C., Sun, B., Du, K. J., Li, J. Y., Zhan, Z. H., Jeon, S. W., ... & Zhang, J. (2023). A Novel Evolutionary Algorithm with Column and Sub-Block Local Search for Sudoku Puzzles. IEEE Transactions on Games.

[26] Li, J. Y., Du, K. J., Zhan, Z. H., Wang, H., & Zhang, J. (2022). Distributed differential evolution with adaptive resource allocation. IEEE transactions on cybernetics.

[27] Ge, Y. F., Yu, W. J., Cao, J., Wang, H., Zhan, Z. H., Zhang, Y., & Zhang, J. (2020). Distributed memetic algorithm for outsourced database fragmentation. IEEE Transactions on Cybernetics, 51(10), 4808-4821.

[28] Li, J. Y., Zhan, Z. H., Wang, H., & Zhang, J. (2020). Data-driven evolutionary algorithm with perturbation-based ensemble surrogates. IEEE Transactions on Cybernetics, 51(8), 3925-3937.

[29] Ge, Y. F., Cao, J., Wang, H., Chen, Z., & Zhang, Y. (2021). Set-based adaptive distributed differential evolution for anonymity-driven database fragmentation. Data Science and Engineering, 6(4), 380-391.

[30] Ge, Y. F., Wang, H., Cao, J., & Zhang, Y. (2022, November). An Information-Driven Genetic Algorithm for Privacy-Preserving Data Publishing. In Web Information Systems Engineering–WISE 2022: 23rd International Conference, Biarritz, France, November 1–3, 2022, Proceedings (pp. 340-354). Cham: Springer International Publishing.

[31] Ge, Y. F., Orlowska, M., Cao, J., Wang, H., & Zhang, Y. (2022). MDDE: multitasking distributed differential evolution for privacy-preserving database fragmentation. The VLDB Journal, 31(5), 957-975.

[32] Laghari, A. A., He, H., Khan, A., Kumar, N., & Kharel, R. (2018). Quality of experience framework for cloud computing (QoC). IEEE Access, 6, 64876-64890.

[33] Laghari, A. A., Jumani, A. K., & Laghari, R. A. (2021). Review and state of art of fog computing. Archives of Computational Methods in Engineering, 1-13.
[34] Laghari, A. A., Wu, K., Laghari, R. A., Ali, M., & Khan, A. A. (2021). A review and state of art of Internet of Things (IoT). Archives of Computational Methods in Engineering, 1-19.
[35] Ali, M., Jung, L. T., Sodhro, A. H., Laghari, A. A., Belhaouari, S. B., & Gillani, Z. (2023). A Confidentiality-based data Classification-as-a-Service (C2aaS) for cloud security. Alexandria Engineering Journal, 64, 749-760.
[36] Karim, S., He, H., Laghari, A. A., Magsi, A. H., & Laghari, R. A. (2021). Quality of service (QoS): measurements of image formats in social cloud computing. Multimedia Tools and Applications, 80, 4507-4532.
[37] Laghari, A. A., He, H., Shafiq, M., & Khan, A. (2018). Assessment of quality of experience (QoE) of image compression in social cloud computing. Multiagent and Grid Systems, 14(2), 125-143.
[38] Laghari, A. A., He, H., Karim, S., Shah, H. A., & Karn, N. K. (2017). Quality of experience assessment of video quality in social clouds. Wireless Communications and Mobile Computing, 2017.
[39] Laghari, A. A., & Laghari, M. A. (2021). Quality of experience assessment of calling services in social network. ICT Express, 7(2), 158-161.
[40] Laghari, A. A., He, H., Memon, K. A., Laghari, R. A., Halepoto, I. A., & Khan, A. (2019). Quality of experience (QoE) in cloud gaming models: A review. multiagent and grid systems, 15(3), 289-304.
[41] Laghari, A. A., He, H., Khan, A., Laghari, R. A., Yin, S., & Wang, J. (2022). Crowdsourcing platform for QoE evaluation for cloud multimedia services. Computer Science and Information Systems, (00), 38-38.