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

K.R. Anil Kumar¹,* and J. Edwin Raja Dhas²

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

Copyright © 2023 Kumar et al., licensed to EAI. This is an open access article distributed under the terms of the CC BY-NC-SA 4.0, which permits copying, redistributing, remixing, transformation, and building upon the material in any medium so long as the original work is properly cited.

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 (QoE) for monitoring the Quality of Experience (QoE) of the end user via 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 minute size and less unique features. The method contains some constraints like that the rule is engendered 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 Preemptive 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 (Jaya) 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 perturbation-based ensemble surrogates, 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.

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

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.
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 detailed study of proposed GOA is given below.

4. Proposed methodology for job shop scheduling

Let us 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 \( t^{th} \) job with its processing time, 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,

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

And the queue is expressed as,

\[
q = 1,2,3,...,j \times m, j \times m + 1 \tag{2}
\]

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

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

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

\[
F_j \geq 0, \quad j = 1,2,\ldots,n \times m + 1 \tag{5}
\]

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

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 Eqn. (5), and Eqn. (6) define the non-negative finishing times. Where, \( A(t) \) represents the total number of tasks processed at any given time, 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.

4.1. 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.

\[
X = \begin{bmatrix}
X_{1,1} & \ldots & X_{1,j} & \ldots & X_{1,Dim-1} & X_{1,Dim} \\
X_{2,1} & \ldots & X_{2,j} & \ldots & X_{2,Dim-1} & X_{2,Dim} \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\
X_{N-1,1} & \ldots & X_{N-1,j} & \ldots & X_{N-1,Dim-1} & X_{N-1,Dim} \\
X_{N,1} & \ldots & X_{N,j} & \ldots & X_{N,Dim-1} & X_{N,Dim}
\end{bmatrix} \tag{7}
\]

\[
x_{i,j} = r_1 \ast (UB_j - LB_j) + LB_j, \quad i = 1,2,...,N, j = 1,2,...,Dim \tag{8}
\]

Where, \( x_{i,j} \) signifies the position of the \( i \)th 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 \( j \)th 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.

\[ t = 1 - \frac{It}{T_{\text{max,iter}}} \]  
\[ a = 2 \times \cos (2 \times \pi \times r_2) \times t \]  
\[ b = 2 \times V (2 \times \pi \times r_3) \times t \]  
\[ V(x) = \begin{cases} \frac{1}{\pi} \times x + 1, & x \in (0, \pi) \\ \frac{1}{\pi} \times x - 1, & x \in (\pi, 2\pi) \end{cases} \]

Where, \( T_{\text{max,iter}} \) 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).

\[ \text{Capturability} = \frac{1}{R + t^2} \]  
\[ t^2 = 1 + \frac{It}{T_{\text{max,iter}}} \]  
\[ R = \frac{M \times \text{vel}^2}{L} \]  
\[ L = 0.2 + (2 - 0.2) \times r_6 \]

Where \( \text{vel} = 1.5 \text{m/s} \) is the gannet's speed in the water while ignoring the current level of water resistance, \( M = 0.5 \text{kg} \) 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).

\[ M_X(t + 1) = \begin{cases} t \times \text{delta} \times (X_i(t) - X_{\text{Best}}(t)) + X_i(t), & \text{Capturability} \geq c \\ X_{\text{Best}}(t) - (X_i(t) - X_{\text{Best}}(t)) \times P \times t, & \text{Capturability} < c \end{cases} \]

\[ \text{delta} = \text{Capturability} \times |X_i(t) - X_{\text{Best}}(t)| \]  
\[ P = \text{Levy} (\text{Dim}) \]

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

\[ \text{Levy} (\text{Dim}) = 0.01 \times \frac{\mu_{\text{levy}}}{v^\tau} \]

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.

\[ J = \{J_1, J_2, ..., J_n\} \quad \text{and} \quad M = \{M_1, M_2, ..., M_j\} \]

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.

\[ \text{Fitness} = \min F_{n=m+1} \]

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

**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, 1039, 1039, 1039, 1039, 1039, 1039, 1039, 1039, 1039, 1039, 1039, 1039, 1039, 1039, 1039, 1039, 1039, 1039, 1039, 1039, 1039.

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

<table>
<thead>
<tr>
<th>Problem</th>
<th>B.M</th>
<th>HGA</th>
<th>ACA</th>
<th>GRASP</th>
<th>DE</th>
<th>SSO</th>
<th>ESSO</th>
</tr>
</thead>
<tbody>
<tr>
<td>LA 12</td>
<td>1139</td>
<td>1039</td>
<td>1039</td>
<td>1039</td>
<td>1039</td>
<td>1039</td>
<td>1039</td>
</tr>
<tr>
<td>LA 13</td>
<td>1350</td>
<td>1150</td>
<td>1150</td>
<td>1150</td>
<td>1150</td>
<td>1150</td>
<td>1150</td>
</tr>
<tr>
<td>LA 14</td>
<td>1492</td>
<td>1292</td>
<td>1292</td>
<td>1292</td>
<td>1292</td>
<td>1292</td>
<td>1292</td>
</tr>
<tr>
<td>LA 15</td>
<td>1307</td>
<td>1207</td>
<td>1207</td>
<td>1207</td>
<td>1207</td>
<td>1207</td>
<td>1207</td>
</tr>
<tr>
<td>LA 17</td>
<td>794</td>
<td>784</td>
<td>789</td>
<td>789</td>
<td>789</td>
<td>788</td>
<td>785</td>
</tr>
<tr>
<td>LA 18</td>
<td>899</td>
<td>848</td>
<td>848</td>
<td>848</td>
<td>848</td>
<td>848</td>
<td>848</td>
</tr>
<tr>
<td>LA 19</td>
<td>887</td>
<td>844</td>
<td>842</td>
<td>842</td>
<td>842</td>
<td>842</td>
<td>842</td>
</tr>
<tr>
<td>LA 20</td>
<td>955</td>
<td>907</td>
<td>902</td>
<td>907</td>
<td>902</td>
<td>902</td>
<td>902</td>
</tr>
<tr>
<td>LA 22</td>
<td>972</td>
<td>935</td>
<td>938</td>
<td>960</td>
<td>934</td>
<td>934</td>
<td>927</td>
</tr>
<tr>
<td>LA 23</td>
<td>1088</td>
<td>1032</td>
<td>1032</td>
<td>1032</td>
<td>1032</td>
<td>1032</td>
<td>1032</td>
</tr>
<tr>
<td>LA 24</td>
<td>987</td>
<td>953</td>
<td>959</td>
<td>978</td>
<td>951</td>
<td>947</td>
<td>947</td>
</tr>
<tr>
<td>LA 25</td>
<td>997</td>
<td>981</td>
<td>977</td>
<td>1028</td>
<td>977</td>
<td>977</td>
<td>977</td>
</tr>
<tr>
<td>LA 26</td>
<td>1318</td>
<td>1218</td>
<td>1218</td>
<td>1271</td>
<td>1212</td>
<td>1212</td>
<td>1212</td>
</tr>
<tr>
<td>LA 28</td>
<td>1316</td>
<td>1216</td>
<td>1227</td>
<td>1293</td>
<td>1227</td>
<td>1217</td>
<td>1217</td>
</tr>
<tr>
<td>LA 29</td>
<td>1252</td>
<td>1160</td>
<td>1177</td>
<td>1293</td>
<td>1177</td>
<td>1170</td>
<td>1164</td>
</tr>
<tr>
<td>LA 30</td>
<td>1455</td>
<td>1355</td>
<td>1355</td>
<td>1368</td>
<td>1355</td>
<td>1355</td>
<td>1355</td>
</tr>
<tr>
<td>LA 32</td>
<td>1950</td>
<td>1850</td>
<td>1850</td>
<td>1850</td>
<td>1850</td>
<td>1850</td>
<td>1850</td>
</tr>
<tr>
<td>LA 33</td>
<td>1819</td>
<td>1719</td>
<td>1719</td>
<td>1719</td>
<td>1719</td>
<td>1719</td>
<td>1719</td>
</tr>
<tr>
<td>LA 34</td>
<td>1821</td>
<td>1721</td>
<td>1725</td>
<td>1753</td>
<td>1725</td>
<td>1723</td>
<td>1723</td>
</tr>
<tr>
<td>LA 35</td>
<td>1988</td>
<td>1888</td>
<td>1888</td>
<td>1888</td>
<td>1888</td>
<td>1888</td>
<td>1888</td>
</tr>
<tr>
<td>LA 37</td>
<td>1497</td>
<td>1407</td>
<td>1412</td>
<td>1457</td>
<td>1408</td>
<td>1403</td>
<td>1397</td>
</tr>
<tr>
<td>LA 38</td>
<td>1296</td>
<td>1196</td>
<td>1196</td>
<td>1267</td>
<td>1196</td>
<td>1196</td>
<td>1196</td>
</tr>
<tr>
<td>LA 39</td>
<td>1333</td>
<td>1233</td>
<td>1240</td>
<td>1290</td>
<td>1238</td>
<td>1238</td>
<td>1233</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Problem</th>
<th>GA</th>
<th>OGA</th>
<th>PSO</th>
<th>OPSO</th>
<th>GOA</th>
</tr>
</thead>
<tbody>
<tr>
<td>LA 12</td>
<td>1039</td>
<td>1039</td>
<td>1039</td>
<td>1039</td>
<td>1038</td>
</tr>
<tr>
<td>LA 13</td>
<td>1150</td>
<td>1150</td>
<td>1150</td>
<td>1150</td>
<td>1149</td>
</tr>
<tr>
<td>LA 14</td>
<td>1292</td>
<td>1292</td>
<td>1292</td>
<td>1292</td>
<td>1291</td>
</tr>
<tr>
<td>LA 15</td>
<td>1207</td>
<td>1207</td>
<td>1207</td>
<td>1207</td>
<td>1206</td>
</tr>
<tr>
<td>LA 17</td>
<td>787</td>
<td>787</td>
<td>787</td>
<td>786</td>
<td>783</td>
</tr>
<tr>
<td>LA 18</td>
<td>848</td>
<td>848</td>
<td>848</td>
<td>848</td>
<td>843</td>
</tr>
</tbody>
</table>
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>
<thead>
<tr>
<th>Problem</th>
<th>Actual make span time</th>
<th>Best GOA</th>
<th>Worst</th>
<th>Average</th>
<th>Std. Deviation</th>
<th>ARPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>LA 12</td>
<td>1139</td>
<td>1038</td>
<td>1225</td>
<td>1082</td>
<td>79.8289</td>
<td>0.237247</td>
</tr>
<tr>
<td>LA 13</td>
<td>1350</td>
<td>1149</td>
<td>1236</td>
<td>1193</td>
<td>35.5183</td>
<td>0.295217</td>
</tr>
<tr>
<td>LA 14</td>
<td>1492</td>
<td>1291</td>
<td>1365</td>
<td>1328.5</td>
<td>30.2112</td>
<td>0.214009</td>
</tr>
<tr>
<td>LA 15</td>
<td>1307</td>
<td>1206</td>
<td>1347</td>
<td>1277</td>
<td>57.5634</td>
<td>0.528998</td>
</tr>
<tr>
<td>LA 17</td>
<td>794</td>
<td>783</td>
<td>952</td>
<td>868.5</td>
<td>68.9955</td>
<td>0.892994</td>
</tr>
<tr>
<td>LA 18</td>
<td>899</td>
<td>843</td>
<td>989</td>
<td>918.5</td>
<td>59.6159</td>
<td>0.709316</td>
</tr>
<tr>
<td>LA 19</td>
<td>887</td>
<td>841</td>
<td>987</td>
<td>914.5</td>
<td>59.6047</td>
<td>0.780285</td>
</tr>
<tr>
<td>LA 20</td>
<td>955</td>
<td>900</td>
<td>1058</td>
<td>980</td>
<td>64.5049</td>
<td>0.679047</td>
</tr>
<tr>
<td>LA 22</td>
<td>972</td>
<td>925</td>
<td>1049</td>
<td>988</td>
<td>50.6249</td>
<td>0.582524</td>
</tr>
<tr>
<td>LA 23</td>
<td>1088</td>
<td>1029</td>
<td>1153</td>
<td>1092.5</td>
<td>50.6277</td>
<td>0.460756</td>
</tr>
<tr>
<td>LA 24</td>
<td>987</td>
<td>944</td>
<td>1090</td>
<td>1018.5</td>
<td>59.6084</td>
<td>0.647307</td>
</tr>
<tr>
<td>LA 25</td>
<td>997</td>
<td>975</td>
<td>1098</td>
<td>1037.5</td>
<td>50.2167</td>
<td>0.467247</td>
</tr>
<tr>
<td>LA 26</td>
<td>1318</td>
<td>1214</td>
<td>1422</td>
<td>1320</td>
<td>84.9208</td>
<td>0.655993</td>
</tr>
<tr>
<td>LA 28</td>
<td>1316</td>
<td>1215</td>
<td>1569</td>
<td>1393</td>
<td>144.5206</td>
<td>1.211586</td>
</tr>
<tr>
<td>LA 29</td>
<td>1252</td>
<td>1160</td>
<td>1396</td>
<td>1280</td>
<td>96.3512</td>
<td>0.72122</td>
</tr>
<tr>
<td>LA 30</td>
<td>1455</td>
<td>1353</td>
<td>1765</td>
<td>1560</td>
<td>168.1989</td>
<td>1.05572</td>
</tr>
<tr>
<td>LA 32</td>
<td>1950</td>
<td>1848</td>
<td>2059</td>
<td>1954.5</td>
<td>86.1416</td>
<td>0.373514</td>
</tr>
<tr>
<td>LA 33</td>
<td>1819</td>
<td>1717</td>
<td>2068</td>
<td>1893.5</td>
<td>143.2959</td>
<td>0.86911</td>
</tr>
<tr>
<td>LA 34</td>
<td>1821</td>
<td>1720</td>
<td>2012</td>
<td>1867.5</td>
<td>119.2105</td>
<td>0.595183</td>
</tr>
<tr>
<td>LA 35</td>
<td>1988</td>
<td>1886</td>
<td>2225</td>
<td>2056.5</td>
<td>138.3969</td>
<td>0.637199</td>
</tr>
<tr>
<td>LA 37</td>
<td>1497</td>
<td>1395</td>
<td>1823</td>
<td>1610</td>
<td>174.7309</td>
<td>1.551181</td>
</tr>
<tr>
<td>LA 38</td>
<td>1296</td>
<td>1190</td>
<td>1426</td>
<td>1311</td>
<td>96.3569</td>
<td>0.627508</td>
</tr>
</tbody>
</table>
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:

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

Where, Best solution is the make span by GOA algorithm in each run, 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 at its 100 iteration. But existing method ESSO make span time 1209 on its 200 iteration, SSO and DE reach 240 iteration.

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 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](image)

Figure 6 present convergence graph for LA 24, LA 25, LA 26, 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 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 make span time 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](image)

Figure 7 present convergence graph for LA 28, LA 29, LA 30, 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 1722 on 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 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](image)
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 1882 on its 80 iteration and SSO reach on its 1886 on its 230 iteration, DE reach make span time 1888 on its 230 iteration.

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


