Communications in Mathematics and Applications
Vol. 16, No. 3, pp. 879–893, 2025
ISSN 0975-8607 (online); 0976-5905 (print)
Published by RGN Publications
DOI: 10.26713/cma.v16i3.3087

http://www.rgnpublications.com

Research Article

Minimization Makespan Problem With an Aging Effect Based on Improved Arithmetic Optimization Algorithms on Job Shop Machines

Tian-Meng Gan[1], Xin-Gong Zhang[2], Win-Chin Lin[3] and Chin-Chia Wu[3*]

[1] *Chongqing College of Architecture and Technology, Chongqing Shapingba, 400030, China*
[2] *College of Mathematics Science, Chongqing Normal University, Chongqing, China*
[3] *Department of Statistics, Feng Chia University, Taichung, 40724, Taiwan*
***Corresponding author:** cchwu@fcu.edu.tw

Received: March 14, 2025 **Revised:** August 6, 2025 **Accepted:** October 28, 2025

Abstract. The research delves into optimizing the *Job Shop Scheduling Problem* (JSSP) by considering aging effects and release time. This paper introduces an *Advanced Arithmetic Optimization Algorithm* (IAOA) designed specifically to minimize the makespan. IAOA operates by translating the continuous solution space into the discrete realm of JSSP through ROV transformation rules. It encodes and decodes the job shop problem using an insertion greedy decoding algorithm. To refine the conventional *Arithmetic Optimization Algorithm* (AOA), this study introduces a nonlinear *Mathematical Acceleration Function* (MOA) along with six neighborhood search strategies. Evaluating its performance involved a comparison of IAOA against AOA, *Grey Wolf Optimizer* (GWO), and *Arithmetic Trigonometric Optimization Algorithm* (ATOA) across 33 benchmark problems. The experimental results underscore IAOA's superior optimization efficacy and its ability to converge efficiently when dealing with JSSP. Notably, IAOA successfully mitigates AOA's limitations, particularly in achieving higher solution accuracy and faster convergence speed.

Keywords. Arithmetic Optimization Algorithm, Job-shop scheduling problems, Greedy decoding algorithm, Aging effect, Release time

Mathematics Subject Classification (2020). Primary: 68W50; Secondary: 68T20

1. Introduction

Scheduling problems represent crucial abstractions within modern production environments, encompassing industries like semiconductor manufacturing, automobile assembly, and mechanical manufacturing systems. In recent decades, extensive research has been devoted to these problems (Luo [9], Toksari and Atalay [13]). The primary objective of the *Job Shop Scheduling Problem* (JSSP) is to devise an optimal sequence for processing while adhering to various constraints, ultimately aiming to minimize the overall job completion time.

The JSSP has garnered substantial attention among scholars, serving not only as a framework for developing efficient scheduling strategies in manufacturing systems but also as a demanding benchmark for NP-hard combination optimization problems (Park *et al*. [10]). Presently, algorithms addressing JSSP are broadly categorized into accurate algorithms, heuristic algorithms, and meta-heuristic algorithms. These algorithms are aimed at solving the challenges posed by JSSP's complexity and constraints, offering different approaches to optimize job sequencing.

The category of exact algorithms encompasses the branch-and-bound method and dynamic programming. While these methods can yield theoretically optimal solutions for small-scale scheduling problems, their practical application is often hindered by their high time and space complexity. On the other hand, heuristic algorithms offer an approach to problem-solving that relies on inductive reasoning and empirical analysis of experience. By leveraging intuitive judgment or trial-and-error methods, heuristic algorithms provide suboptimal solutions. Despite their ability to swiftly address problems, ensuring the convergence of these algorithms can be challenging.

In contrast, meta-heuristic algorithms have gained widespread adoption for handling nonlinear optimization problems. Their straightforward principles, independence from initial values, and ease of implementation make them particularly advantageous. Notably, meta-heuristic algorithms do not rely on the gradient of the objective function, bolstering the credibility and applicability of their solutions. This characteristic further enhances their versatility across a range of problem domains. Zhang *et al*. [15] proposed an artificial swarm algorithm designed specifically to address scenarios involving total weighted late work under due-window constraints.

However, real-world manufacturing environments present a far more intricate landscape. Take, for instance, a renowned PCB circuit board processing factory where multiple factors intricately affect production efficiency. Three primary elements stand out: (1) The sequencing of job processing, (2) The timing of job release, and (3) The operational speed of machines. Among these factors, the release time of artifacts refers to the moment when raw materials arrive at the processing plant. Meanwhile, the operational speed of machines is not fixed due to varying utilization frequencies, which might result in aging effects, subsequently diminishing the processing rate. This phenomenon is commonly referred to as the 'aging effect'. These interdependent elements collectively contribute to the complexity of optimizing production processes within such manufacturing setups. Zhao [16] investigated single-machine scheduling problems with simultaneous considerations of due-window assignment, position-dependent and convex resource-dependent processing times, and a deteriorating rate-modifying activity.

To better align with the demands of practical enterprises, this study addresses the enhanced AOA algorithm tailored to tackle this specific challenge. AOA, a meta-heuristic algorithm rooted in the population model (Abualigah *et al*. [2]), operates through distinct stages: exploration and development. During exploration, multiplication and division operators are employed to broaden the scope of global search, while the development phase utilizes addition and subtraction operations to refine local search accuracy.

Over the past couple of years, the AOA algorithm has found widespread application in addressing practical problems, ranging from controller design (Elkasem *et al*. [4]), electric power systems (Kharrich *et al*. [6]), to data clustering (Abualigah *et al*. [1]). Gürses *et al*. [5] focused on a comparison of recent algorithms such as the arithmetic optimization algorithm, the slime mold optimization algorithm, the marine predators algorithm, and the slap swarm algorithm. Liu *et al*. [8] conducted a study constructing a reinforcement learning-based hybrid algorithm merging *Aquila Optimizer* (AO) with an enhanced *Arithmetic Optimization Algorithm* (IAOA). Their findings demonstrated that this hybrid approach exhibited superior convergence speed and accuracy compared to other algorithms. Wang and Mo [14] considered novel hybrid arithmetic optimization algorithm to address the inherent constraints of traditional numerical computing techniques, including increased computational complexity and excessive reliance on gradient information.

Building upon this, the current paper aims to introduce six distinct neighborhood search structures and incorporate a nonlinear *mathematical acceleration function* (MOA). These augmentations aim to further enhance the convergence and generalization capabilities of the AOA algorithm, pushing the boundaries of its efficiency and applicability in solving complex optimization problems.

2. Problem Statement

This paper delves into the job shop scheduling problem, accounting for both aging effects and release time. In this scenario, each job comprises m operations, denoted as $O_i = \{o_{i1}, o_{i2}, \ldots, o_{im}\}$, $i = 1,2,\ldots,n$, and the sequence of these operations is predefined. o_{ij} $(i = 1,2,\ldots,n, j = 1,2,\ldots,n)$ is specifically executed on a designated machine, and during its processing, it cannot be interrupted by any other operation until the job's processing is concluded.

The release time of each job, denoted as Job i $(i = 1,2,\ldots,n)$, is represented by r_i. Moreover, each machine exhibits an aging effect, wherein the duration of jobs extends as the machine's usage increases. This phenomenon results in the completion time of each job, denoted as $C_i = \{C_{i1}, C_{i2}, \ldots, C_{im}\}$, $i = 1,2,\ldots,n$, and the overall completion time of Job i, denoted as $C_{\max} = \max_i\{C_{im}\}$. Consequently, the maximum completion time among all jobs, often referred to as the makespan, is ascertained. This makespan represents the longest time taken for any job to complete within the given scheduling framework.

The objective of this paper is to minimize the makespan, focusing on optimizing both the aging effect and release time dynamics (Pinedo [11]). Here, the notation $x_{ijk} = 1$ equals 1 if the jth operation of the ith job is being processed on machine k; otherwise, it is set as 0. Similarly, the notation $y_{ijhlk} = 1$ equals 1 if operation o_{ij} precedes operation o_{hl} on machine k; otherwise, it is set as 0.

The mathematical model is established based on various parameters denoted as follows: p_{ij} and C_{ij} stand for the processing time and completion time, respectively, of the jth operation of the ith job. Moreover, β signifies the aging factor of the machine, while u_{ik} represents the position of the ith job on machine k,

$$\min C_{\max} = \max_i\{C_{im}\} \tag{2.1}$$

subject to

$$p_{ij} \geq 0 \tag{2.2}$$

$$s_i \geq r_i \tag{2.3}$$

$$s_{ij} + (1 + (u_{ik})^{\beta}) p_{ij} x_{ijk} \leq C_{ij} \tag{2.4}$$

$$s_{i(j+1)} - C_{ij} \geq 0 \tag{2.5}$$

$$C_{ij} \leq C_{\max} \tag{2.6}$$

$$C_{hl} - C_{ij} + L(1 - y_{ijhlk}) \geq p_{hl} \tag{2.7}$$

$$i,h = 1,2,\ldots,n;\ j,l,k = 1,2,\ldots,m.$$

The objective function $C_{\max}$ aims to minimize the maximum completion time, as expressed in formula (2.1). Formula (2.2) ensures that the processing time for each process remains greater than or equal to zero. In formula (2.3), the job is restricted to start processing only after its release time. Formula (2.4) considers the impact of the machine's aging effect. The sequence of processes for each job is indicated in formula (2.5). Additionally, formula (2.6) sets the constraint that the completion time of each job must not exceed the total completion time. Lastly, in formula (2.7), the variable L represents a sufficiently large integer, signifying the imposition of a specific job sequence on the same machine.

3. Improved Arithmetic Optimization

3.1 Basic Arithmetic Optimization Algorithm (AOA)

3.1.1 Initialization

The AOA algorithm initializes its population as $X_1, X_2, \ldots, X_N$, where everyone is denoted as $X_i = [x_{i1}, x_{i2}, \ldots, x_{iD}]$, where N represents the population size, and D signifies the dimensionality of everyone.

In the AOA algorithm, the selection of the subsequent search strategy is determined using the mathematical acceleration function known as the MOA function, as illustrated by equation (3.1):

$$\mathrm{MOA}(c) = \mathrm{Min} + c * \left(\frac{\mathrm{Max} - \mathrm{Min}}{T}\right), \tag{3.1}$$

where c denotes the current iteration number, T stands for the total number of iterations, $\mathrm{MOA}(c)$ represents the acceleration function of the cth generation, and Min and Max denote the minimum and maximum values of the acceleration function, respectively.

3.1.2 Exploration and Development

In the AOA method, when a randomly generated number r_1 is less than the MOA value, the algorithm initiates an exploration of the entire search space using multiplication and

division strategies to seek improved candidate solutions. If another randomly generated number r_2 is less than 0.5, the algorithm executes the division operation. Conversely, when r_2 is greater than or equal to 0.5, the multiplication operation is carried out. The updated position is determined according to equation (3.2).

On the contrary, if r_1 exceeds the MOA threshold, AOA directs its focus towards exploring the local space by employing addition and subtraction operations to enhance candidate solutions. When a randomly generated number r_3 is less than 0.5, the algorithm performs a subtraction operation. Conversely, if r_3 is greater than or equal to 0.5, the addition operation is executed as described in equation (3.3),

$$x_{i,j}(c+1) = \begin{cases} \text{best}(x_j) \div (\text{MOP}(c)+\varepsilon) \times ((\text{UB}_j - \text{LB}_j) \times \mu + \text{LB}_j), & r_2 < 0.5, \\ \text{best}(x_j) \times \text{MOP}(c) \times ((\text{UB}_j - \text{LB}_j) \times \mu + \text{LB}_j), & \text{otherwise}; \end{cases} \tag{3.2}$$

$$x_{i,j}(c+1) = \begin{cases} \text{best}(x_j) - \text{MOP}(c) \times ((\text{UB}_j - \text{LB}_j) \times \mu + \text{LB}_j), & r_3 < 0.5, \\ \text{best}(x_j) + \text{MOP}(c) \times ((\text{UB}_j - \text{LB}_j) \times \mu + \text{LB}_j), & \text{otherwise}, \end{cases} \tag{3.3}$$

where the notation $x_{i,j}(c+1)$ signifies the ith solution at the jth position during the $(c+1)$th iteration. best(x_j) denotes the jth position of the most optimal solution obtained up to the current iteration. The value ε stands for a small integer. UB_j and LB_j represent the upper and lower bounds, respectively, of the optimal solution within the first dimension. The parameter μ holds a specific value of 0.449. The mathematical optimization probability coefficient, denoted as MOP, is expressed as shown in equation (3.4),

$$\text{MOP}(c) = 1 - \frac{c^{\frac{1}{\alpha}}}{T^{\frac{1}{\alpha}}}, \tag{3.4}$$

where MOP(c) indicates the value during the cth iteration. The parameter α is a crucial factor with a value set at 5, signifying its sensitivity in the process.

3.2 Improvement of the Algorithm

3.2.1 Chaos Initialization

Chaos embodies a form of motion observed within deterministic dynamic systems characterized by uncertainty, profound regularity, and rapid convergence rates. It encompasses qualities such as sensitivity to initial values and ergodic property. Leveraging the classical chaotic model of the Logistic chaotic mapping, the algorithm aims to mitigate interference arising from the instability of the AOA population, ultimately enhancing the accuracy of the algorithm's high-quality iterative positions. The chaotic mapping is expressed as follows:

$$X(t+1) = \lambda X(t)[1 - X(t)], \tag{3.5}$$

where $\lambda = 4$.

3.2.2 Convergence Strategy Based on Non-linear MOA Functions

In the AOA framework, the MOA function plays a pivotal role in harmonizing global exploration and local search. Notably, the linear nature of the MOA function depicted in equation (3.1) exhibits distinct characteristics. Initially, during the algorithm's early stages, the MOA function yields smaller values, prompting extensive exploration of the search space. This tendency, however, results in an inadequate capacity for global exploration within the algorithm.

Conversely, in the later phases of the iterative process, larger MOA values restrict the algorithm's local development of the algorithm, causing a slower convergence rate. To address this challenge, this paper introduces an enhanced MOA function formulated in equation (3.6) to mitigate these limitations,

$$\mathrm{MOA}(c) = \begin{cases} 1 - \frac{1}{1.6} \times \arctan\left(-\frac{T}{1.5} + t\right), & \frac{1}{1.6} \times \arctan\left(-\frac{T}{1.5} + t\right) > 0, \\ -\frac{1}{1.6} \times \arctan\left(-\frac{T}{1.5} + t\right), & \frac{1}{1.6} \times \arctan\left(-\frac{T}{1.5} + t\right) \leq 0. \end{cases} \tag{3.6}$$

Figure 3.1 presents a comparative analysis between the original MOA and the enhanced MOA. Observing Figure 3.1, it becomes evident that the improved MOA exhibits value close to 1 during the initial iterations of the algorithm. This characteristic emphasizes a stronger inclination towards global search, thus amplifying the algorithm's probability for comprehensive exploration. As the iterations progress, the improved MOA tends towards a value of 0. This shift increases the probability of local exploration within the algorithm, thereby enhancing its convergence rate.

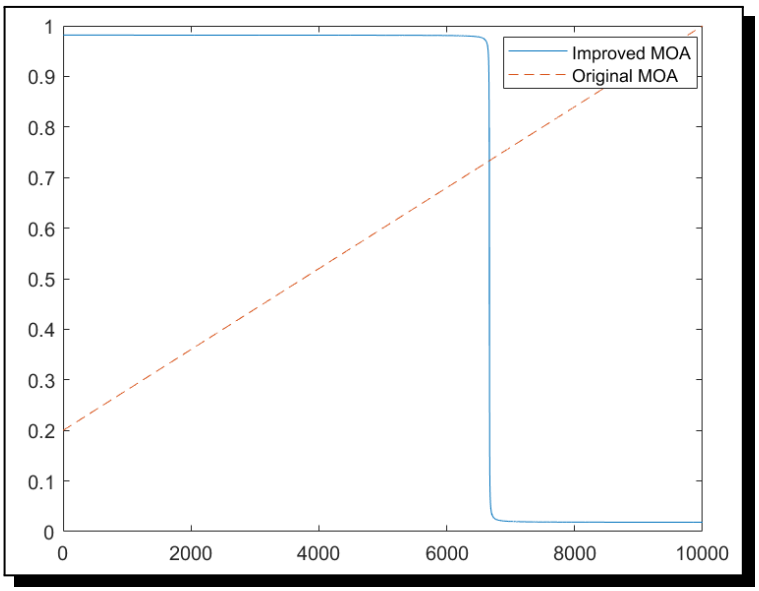


Figure 3.1. Comparison diagram of the MOA function

3.2.3 Neighborhood Search Strategy

To bolster the search capabilities of the AOA algorithm for job coding represented as $X_i = [x_{i1}, x_{i2}, \ldots, x_{iD}]$, six strategies have been employed for neighborhood search enhancement.

One of these strategies, termed Neighborhood search structure $N_1(1)$ or the 'two points cross', involves the selection of two elements within the process coding. These selected elements must correspond to different processes. Subsequently, an exchange operation is performed between these chosen elements.

Neighborhood search operations within the AOA algorithm involve several strategies designed to enhance the exploration of solutions:

Neighborhood search structure $N_2(2)$ – referred to as 'forward insertion': This strategy involves the selection of two elements within the operation encoding. Subsequently, the latter element is inserted into the position immediately before the former element.

Neighbor search structure $N_3(3)$ – *known as 'back insertion'*: This approach entails the selection of two elements in the operation encoding. Following this, the preceding element is inserted into the position immediately before the subsequent element.

Neighborhood search structure $N_4(4)$ – *termed 'random exchange of process blocks'*: This strategy encompasses the selection of a random number $\gamma_1 \in [0, \frac{D}{2}]$. It involves the swapping of two process blocks, specifically $[x_{i1}, x_{i2}, \ldots, x_{i\gamma_1}]$ and $[x_{i(n-\gamma_1)}, x_{i(n-\gamma_1+1)}, \ldots, x_{iD}]$. Refer to Figure 3.2 for visual representation and clarity.

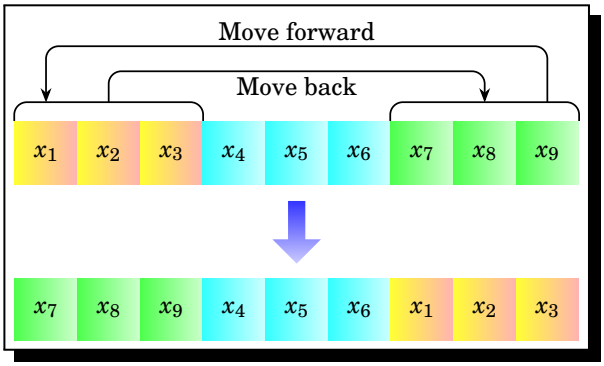


Figure 3.2. Random exchange of before and after process blocks

The AOA algorithm employs two additional neighborhood search strategies to further refine its exploration capabilities:

Neighborhood search structure $N_5(5)$ – *referred to as 'random process block swap'*: This strategy involves setting random integers $\gamma_2 \in [1, \frac{D}{2}]$, $\gamma_3 \in [\frac{D}{2}+1, D]$, $\gamma_4 \in [1, \min\{\frac{D}{2}-\gamma_2, D-\gamma_3\}]$. It then performs an exchange of job sequences between $[x_{i\gamma_2}, x_{i(\gamma_2+1)}, \ldots, x_{i\gamma_4}]$ and $[x_{i\gamma_3}, x_{i(\gamma_3+1)}, \ldots, x_{i\gamma_4}]$.

Neighborhood search structure $N_6(6)$: This approach involves deleting identical sections from the optimal solution and the worst solution, subsequently replacing them with the current solution. For further details, refer to Figure 3.3 for a visual representation.

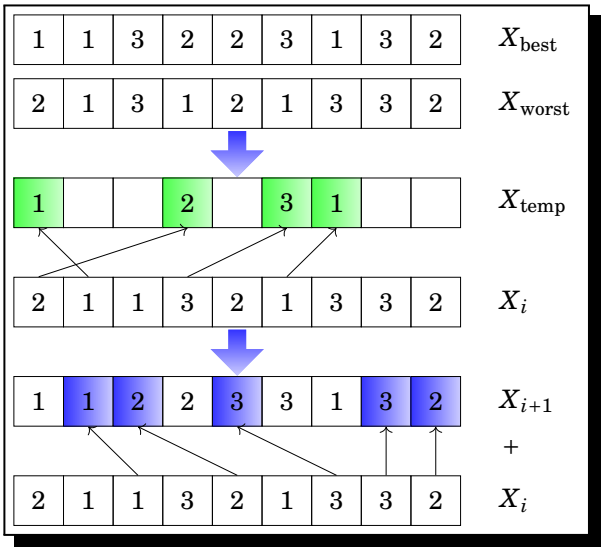


Figure 3.3. Neighborhood search structure N_6

3.3 Algorithm Process

The steps of IAOA are summarized as follows:

Pseudocode for the improved arithmetic optimization algorithm (IAOA)

Set IAOA parameters, $\alpha\mu$
Initialize the population
While ($C < T$)
Calculate the fitness value of everyone
Find the optimal solution
Update the MOA value using equation (3.5)
Update the MOP value using equation (3.4)
For ($i = 1 : N$)
For ($j = 1 : D$)
If $r_1 >$ MOA
If $r_2 < 0.5$
Execute division operation using the first rule in equation (3.2)
Else
Execute multiplication operations using the second rule in equation (3.2)
end
else
If $r_3 < 0.5$
Execute subtraction operation using the first rule in equation (3.3).
Else
Execute addition operation using the second rule in equation (3.3).
End
End
End
End
Execute six neighborhood search strategies
$C = C + 1$
End

4. The Application of IAOA in JSSP

4.1 Encoding

The *Job Shop Scheduling Problem* (JSSP) represents a discrete optimization challenge, while the IAOA algorithm primarily addresses continuous space problems. To effectively solve the JSSP using the IAOA algorithm, establishing a mapping relationship between continuous and discrete spaces becomes imperative. In this study, the ROV transformation method (Liu *et al*. [7]) was adopted for this purpose. It involves mapping the continuous solution space to the discrete space of the JSSP.

The process involves several steps: Initially, a set of continuous vectors $X_i = [x_{i1}, x_{i2}, \ldots, x_{iD}]$ and artifact number vectors $J = [j_1, j_2, \ldots, j_D]$ are generated. Subsequently, the smallest position within the continuous vector is identified, assigned the process number 1. Then, the second-smallest position in the continuous vector is allocated the process number 2, and this process continues until all continuous vectors are converted into corresponding process numbers.

For instance, consider a scenario involving operation workshop scheduling problems encompassing three machines and two jobs. For a continuous vector $X_i = [0.29,2.31,5.32,3.73,1.13,0.62]$, the resulting corresponding process numbers would be $[O_1, O_2] = [o_{11}, o_{12}, o_{13}, o_{23}, o_{22}, o_{21}]$ (refer to Table 1 for detailed illustration).

Table 1. ROV conversion

Position vector	0.29	2.31	5.32	3.73	1.13	0.62
Artifact serial number	j_1	j_1	j_1	j_2	j_2	j_2
ROV rule	1	4	6	5	3	2
Process serial number	o_{11}	o_{12}	o_{13}	o_{23}	o_{22}	o_{21}

4.2 Decoding

The process decoding method, influenced by Salido *et al*. [12] adopts the incentive greedy decoding algorithm. This algorithm follows distinct steps to decode the process:

Initially, the calculation involves determining the start time for each job. This computation entails setting the start time for each process as the maximum value between the completion time of the preceding job and the completion time of the job processed by the previous machine. Subsequently, identification of available idle time on the respective machine preceding a process occurs. This idle time is then inserted into the earliest feasible machining period on the corresponding machine without causing delays to the commencement of other scheduled processes.

Figure 4.1 visually presents the scheduling outcomes achieved by the insertive greedy decoding algorithm. Notably, this algorithm significantly reduces the maximum completion time of the job, demonstrating its efficacy in streamlining job completion.

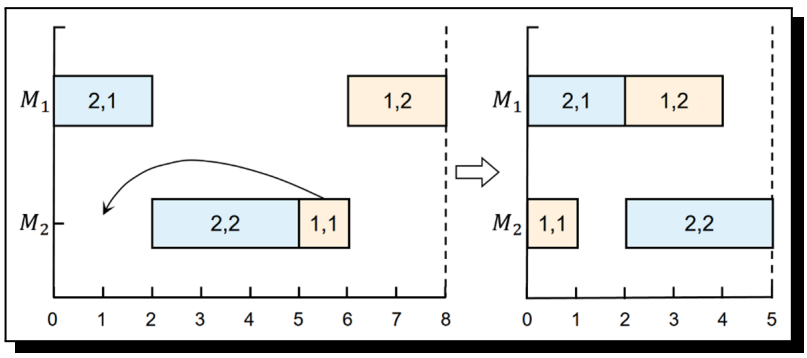


Figure 4.1. Incentive greedy decoding algorithm

5. Analysis of the Experimental Results

To validate the efficacy of the IAOA algorithm in addressing JSSP problems, a comparative study was conducted involving the IAOA algorithm, the Grey Wolf algorithm, the standard AOA algorithm, and the ATOA algorithm (Devan *et al*. [3]). These algorithms were assessed across 33 test sets, each executed with a population size of 30, 200 iterations, and repeated for 10 runs per algorithm.

The evaluation criteria employed were the minimum (Min) and mean (Mean) values. Notably, the release time of each artifact was randomly selected as an integer within the interval $[0, n]$, where 'n' denotes the total number of artifacts within the respective test set. The specifics of the parameters utilized for each algorithm are detailed in Table 2. This comparison aimed to highlight the performance and effectiveness of the IAOA algorithm in comparison to the other algorithms across various test scenarios.

Table 2. The algorithm parameters

Algorithm	Parameter
IAOA	Max = 1, Min = $0.2\alpha = 5\mu = 0.449\beta = 0.01$
AOA	Max = 1, Min = $0.2\alpha = 5\mu = 0.449$
GWO	Max = 2, Min = 0
ATOA	Max = 1, Min = $0.2\alpha = 5\mu = 0.449$

The experimental outcomes, displayed in Table 3, reveal noteworthy insights when comparing the performance of the AOA, GWO, ATOA algorithms against the IAOA algorithm. It becomes evident that the IAOA algorithm consistently yields optimal solutions across all 33 test sets, particularly excelling in minimizing the maximum completion time. This optimal solution highlights the algorithm's convergence prowess, while the average solution signifies its overall convergence trend.

Notably, the IAOA algorithm demonstrates superior performance, especially in addressing multi-artifact and multi-process scheduling challenges. For instance, in test sets such as la26 to la30, encompassing 20 artifacts processed across 10 machines, the IAOA algorithm showcases an increase in Mean and Min makespan by approximately 100 seconds. This increase underscores the algorithm's superiority and exceptional convergence ability in managing intricate scheduling problems involving multiple artifacts and processes.

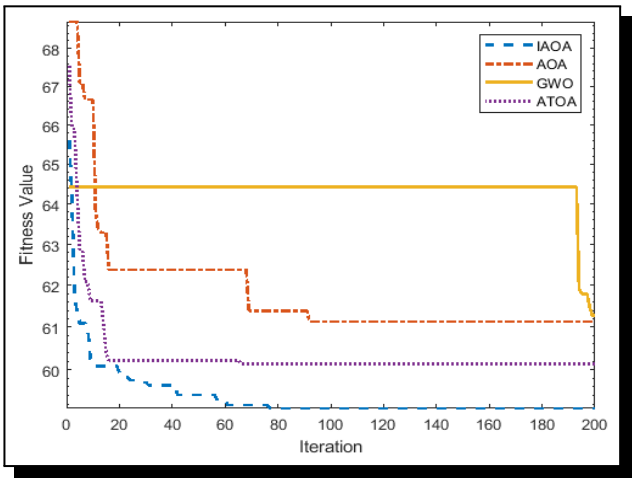
Table 3. The experimental results

Example	**IAOA**		AOA		GWO		ATOA	
	Mean	**Min**	Mean	Min	Mean	Min	Mean	Min
FT06	**59.03903**	**58.84815**	61.26766	60.55124	61.90548	60.14333	60.56529	59.64325
FT10	**1073.14**	**1054.793**	1152.796	1098.652	1165.283	1133.21	1136.402	1086.7
FT20	**1386.863**	**1336.542**	1546.64	1457.191	1546.141	1504.057	1504.727	1466.206
la01	**722.1742**	**715.7894**	771.9218	757.4616	763.3962	744.9031	766.2699	753.4501
la02	**717.2419**	**689.0908**	792.8702	756.7391	791.0079	771.8619	780.7476	724.2742
la03	**670.805**	**656.4138**	723.6617	718.7546	701.1846	684.2611	708.6668	689.9126
la04	**649.1781**	**639.1997**	680.9431	661.2704	693.8222	664.5617	676.4819	663.9503
la05	**622.4339**	**615.8835**	651.3664	641.9649	647.8458	638.179	640.1661	624.5375
la06	**999.6387**	**990.1751**	1062.451	1047.101	1048.112	1024.693	1044.95	1015.722
la07	**990.5363**	**981.3995**	1056.703	1039.724	1048.463	1030.441	1050.363	1033.103
la08	**942.2847**	**932.3141**	999.5664	989.0963	993.9732	960.076	997.3874	982.8415
la09	**1056.951**	**1044.475**	1110.597	1082.816	1098.424	1080.198	1099.41	1089.785

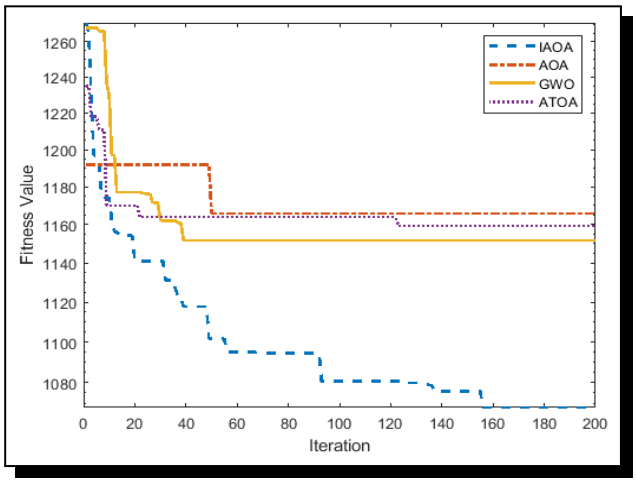
(Table Contd.)

Example	**IAOA**		AOA		GWO		ATOA	
	Mean	**Min**	Mean	Min	Mean	Min	Mean	Min
la10	**1032.355**	**1018.942**	1087.568	1072.954	1081.337	1067.306	1077.843	1056.847
la11	**1333.242**	**1313.47**	1424.125	1395.689	1403.59	1389.062	1416.441	1391.839
la12	**1176.029**	**1161.759**	1232.188	1210.728	1220.354	1193.637	1225.327	1214.551
la13	**1265.746**	**1251.057**	1335.754	1313.469	1324.363	1293.219	1325.927	1301.024
la14	**1365.826**	**1337.831**	1460.702	1439.425	1450.175	1410.745	1445.219	1408.014
la15	**1377.976**	**1340.821**	1484.786	1462.46	1470.372	1441.554	1456.594	1395.011
la16	**1068.996**	**1043.354**	1145.501	1106.106	1126.589	1103.542	1125.87	1082.223
la17	**850.4477**	**820.558**	926.6859	907.163	910.0429	889.1306	914.2798	883.7913
la18	**947.0173**	**918.8857**	1035.643	1005.382	1017.885	995.5214	1020.774	999.8471
la19	**953.3133**	**923.3705**	1026.486	1001.484	1030.188	1018.051	1024.22	1001.189
la20	**1008.332**	**988.0135**	1064.102	1046.258	1069.673	1054.392	1061.48	1049.291
la21	**1285.297**	**1267.786**	1395.601	1364.062	1376.87	1349.361	1379.707	1361.655
la22	**1125.943**	**1096.338**	1242.869	1220.364	1222.377	1172.188	1217.859	1191.025
la23	**1219.24**	**1193.852**	1310.305	1287.844	1289.619	1252.418	1303.432	1274.626
la24	**1144.004**	**1115.497**	1246.832	1224.754	1213.317	1179.175	1232.355	1202.162
la25	**1200.959**	**1186.457**	1273.081	1239.112	1313.805	1313.805	1265.05	1217.773
la26	**1503.126**	**1463.653**	1606.492	1557.226	1611.122	1569.835	1605.488	1567.964
la27	**1575.34**	**1537.241**	1656.844	1621.544	1654.096	1613.277	1652.463	1615.637
la28	**1539**	**1502.087**	1651.212	1591.77	1656.65	1614.344	1656.726	1637.227
la29	**1491.712**	**1451.86**	1592.696	1575.615	1607.077	1563.754	1579.595	1545.668
la30	**1629.962**	**1602.192**	1763.636	1739.7	1752.331	1713.883	1751.573	1725.973

To further substantiate the superior performance of the IAOA algorithm, four specific test cases — ft06, ft10, la10, la16, and la26 — were chosen for analysis. The iteration curves depicted in Figure 5.1 serve as evidence showcasing the algorithm's behavior across these selected cases. These results illustrate the IAOA algorithm's distinctive ability to achieve faster convergence during the early stages while mitigating the risk of getting trapped in local optimal solutions.

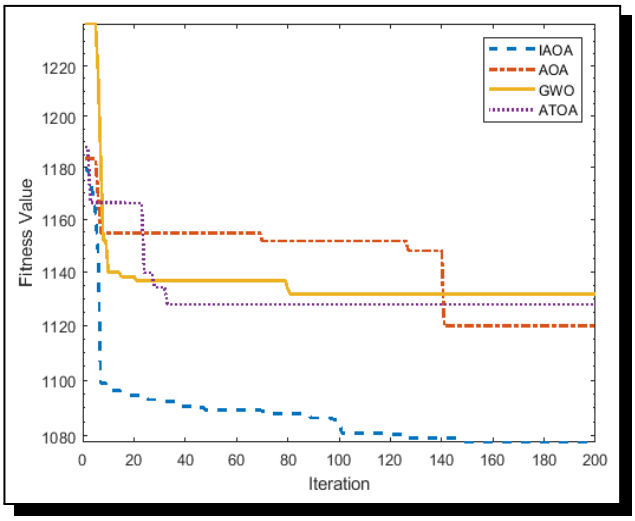


(a) FT06

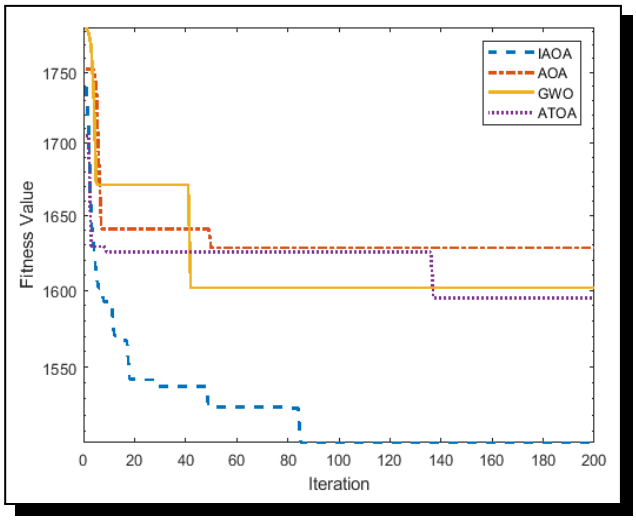


(b) FT10

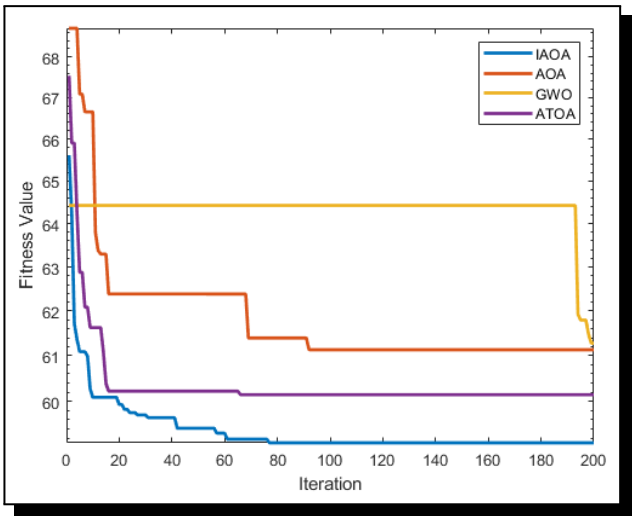
(Figure Contd.)



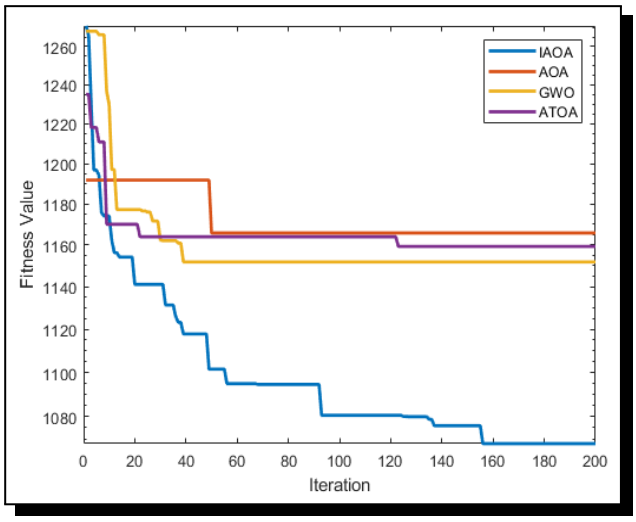
(c) la16



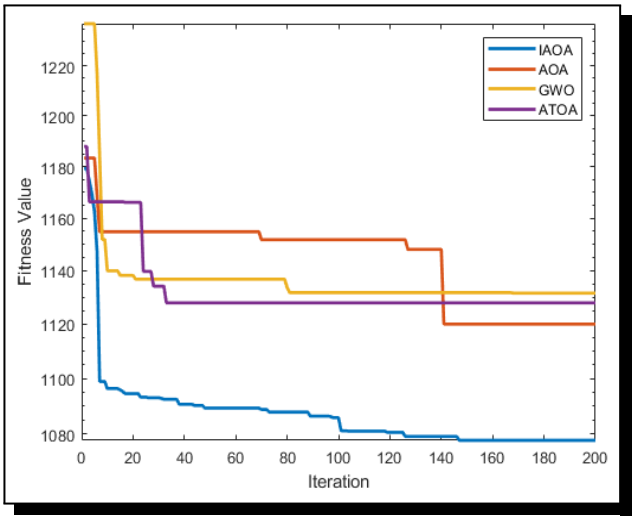
(d) la26



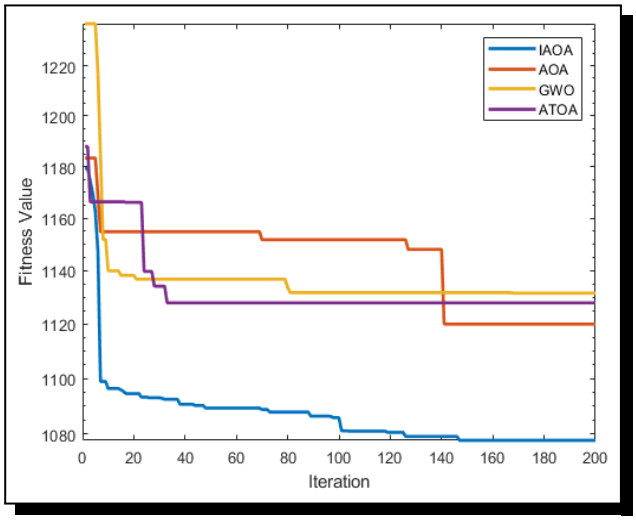
(e) FT06



(f) FT10



(g) la26



(h) la26

Figure 5.1. Iterative convergence comparison of the four algorithms

Moreover, examining the Gantt diagrams presented in Figure 5.2 and Figure 5.3 unveils the effectiveness of the IAOA algorithm in maximizing resource utilization and optimizing efficiency. The algorithm demonstrates an adept utilization of limited resources, significantly reducing processing times. This notable improvement underscores the efficacy of the proposed nonlinear MOA function strategy and the utilization of the six neighborhood search operators. These elements contribute to enhancing the algorithm's robustness, ensuring superior performance across diverse scenarios.

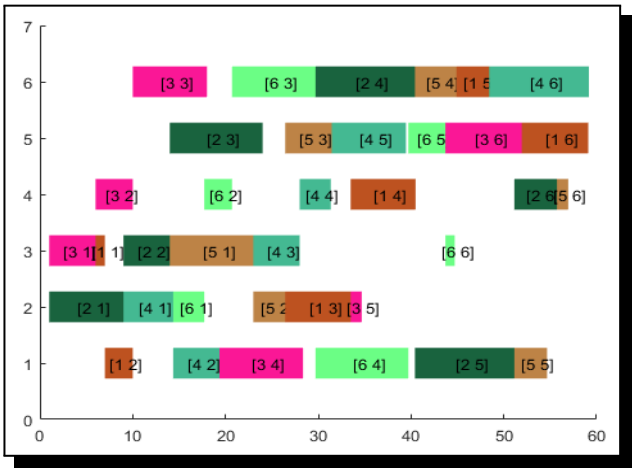


Figure 5.2. ft06 Gantt

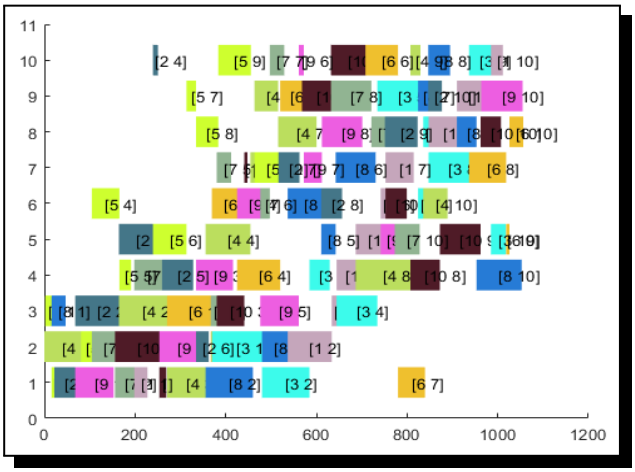


Figure 5.3. ft10 Gantt

6. Conclusion

This paper introduces an enhanced arithmetic optimization algorithm tailored to address the job scheduling problem while considering both the aging effect and release time, aiming to minimize the makespan. To tackle the unique characteristics of the JSSP problem, the algorithm facilitates the mapping between continuous and discrete spaces, leveraging the ROV conversion rule. This adaptation enables the IAOA algorithm to effectively resolve the complexities inherent in the JSSP.

Moreover, the study devises six distinct neighborhood search structures and incorporates nonlinear MOA function strategies to further elevate the algorithm's performance. Through comprehensive testing across 33 datasets, the IAOA algorithm demonstrates robust convergence capabilities and remarkable effectiveness in successfully solving the JSSP problem. Consequently, this research presents a novel solution addressing the challenges posed by machine aging effects and release time dynamics within the realm of job scheduling. Future work may focus on establishing a formal theoretical justification for the convergence and computational complexity of the IAOA algorithm, as well as performing statistical significance tests to assess the robustness of the results.

Acknowledgments

This work was supported in part by Project Supported by the National Natural Science Foundation of China (72074035), and in part by National Science and Technology Council of Taiwan (NSTC) under Grant No. NSTC 112-2221-E-035- 060-MY2.

Code Availability

The corresponding author will provide the Fortran programming code upon request.

Availability of Data and Material

The corresponding author will provide the relevant datasets upon request.

Competing Interests

The authors declare that they have no competing interests.

Authors' Contributions

All the authors contributed significantly in writing this article. The authors read and approved the final manuscript.

References

[1] L. Abualigah, K. H. Almotairi, M. A. Elaziz, M. Shehab and M. Altalhi, Enhanced flow direction arithmetic optimization algorithm for mathematical optimization problems with applications of data clustering, *Engineering Analysis with Boundary Elements* **138** (2022), 13 – 29, DOI: 10.1016/j.enganabound.2022.01.014.

[2] L. Abualigah, A. Diabat, S. Mirjalili, M. A. Elaziz and A. H. Gandomi, The arithmetic optimization algorithm, *Computer Methods in Applied Mechanics and Engineering* **376** (2021), 113609, DOI: 10.1016/j.cma.2020.113609.

[3] P. A. M. Devan, F. A. Hussin, R. B. Ibrahim, K. Bingi, M. Nagarajapandian and M. Assaad, An arithmetic-trigonometric optimization algorithm with application for control of real-time pressure process plant, *Sensors* **22**(2) (2022), 617, DOI: 10.3390/s22020617.

[4] A. H. A. Elkasem, M. Khamies, G. Magdy, I. B. M. Taha and S. Kamel, Frequency stability of AC/DC interconnected power systems with wind energy using arithmetic optimization algorithm-based fuzzy-PID controller, *Sustainability* **13**(21) (2021), 12095, DOI: 10.3390/su132112095.

[5] D. Gürses, S. Bureerat, S. M. Sait and A. R. Yildiz, Comparison of the arithmetic optimization algorithm, the slime mold optimization algorithm, the marine predators algorithm, the salp swarm algorithm for real-world engineering applications, *Materials Testing* **63**(5) (2021), 448 – 452, DOI: 10.1515/mt-2020-0076.

[6] M. Kharrich, L. Abualigah, S. Kamel, H. AbdEl-Sattar and M. Tostado-Véliz, An Improved Arithmetic Optimization Algorithm for design of a microgrid with energy storage system: Case study of El Kharga Oasis, Egypt, *Journal of Energy Storage* **51** (2022), 104343, DOI: 10.1016/j.est.2022.104343.

[7] B. Liu, L. Wang and Y.-H. Jin, An effective PSO-based memetic algorithm for flow shop scheduling, *IEEE Transactions on Systems, Man, and Cybernetics, Part B* (*Cybernetics*) **37**(1) (2007), 18 – 27, DOI: 10.1109/TSMCB.2006.883272.

[8] H. Liu, X. Zhang, H. Zhang, C. Li and Z. Chen, A reinforcement learning-based hybrid Aquila Optimizer and improved Arithmetic Optimization Algorithm for global optimization, *Expert Systems with Applications* **224** (2023), 119898, DOI: 10.1016/j.eswa.2023.119898.

[9] S. Luo, Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning, *Applied Soft Computing* **91** (2020), 106208, DOI: 10.1016/j.asoc.2020.106208.

[10] J. Park, J. Chun, S. H. Kim, Y. Kim and J. Park, Learning to schedule job-shop problems: Representation and policy learning using graph neural network and reinforcement learning, *International Journal of Production Research* **59**(11) (2021), 3360 – 3377, DOI: 10.1080/00207543.2020.1870013.

[11] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Springer Cham., xx + 670 pages (2016), DOI: 10.1007/978-3-319-26580-3.

[12] M. A. Salido, J. Escamilla, F. Barber, A. Giret, D. Tang and M. Dai, Energy efficiency, robustness, and makespan optimality in job-shop scheduling problems, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **30**(3) (2016), 300 – 312, DOI: 10.1017/S0890060415000335.

[13] M. D. Toksari and B. Atalay, Some scheduling problems with job rejection and a learning effect, *The Computer Journal* **66**(4) (2023), 866 – 872, DOI: 10.1093/comjnl/bxab201.

[14] H. Wang and Y. Mo, Adaptive hybrid optimization algorithm for numerical computing in engineering applications, *Engineering Optimization* **57**(4) (2025), 845 – 883, DOI: 10.1080/0305215X.2024.2337072.

[15] R. Zhang, S. Song and C. Wu, A hybrid artificial bee colony algorithm for the job shop scheduling problem, *International Journal of Production Economics* **141**(1) (2013), 167 – 178, DOI: 10.1016/j.ijpe.2012.03.035.

[16] X. L. Zhao, Bicriteria due-window assignment and resource allocation scheduling with aging effect and a deteriorating rate-modifying activity, *International Journal of Innovative Computing, Information and Control* **20**(4) (2024), 1169 – 1183, DOI: 10.24507/ijicic.20.04.1169.