IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 27 May 2024, accepted 6 July 2024, date of publication 9 July 2024, date of current version 22 July 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3425710*

RESEARCH ARTICLE

# Dual Resource Scheduling Problem of Machines and AGVs Based on Hybrid Discrete Salp Swarm Algorithm

**TIANRUI ZHANG AND GUANGHAO ZHU**
School of Mechanical Engineering, Shenyang University, Shenyang 110000, China
Corresponding author: Guanghao Zhu (Jupeki@outlook.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 52075088, and in part by the Graduate Education Teaching Reform Foundation of Liaoning Province under Grant LNYJG2022490.

**ABSTRACT** In a flexible manufacturing shop, the cooperation of Automated Guided Vehicles (AGVs) and machines is more in accordance with the real production situation. A scheduling problem of flexible manufacturing shop floor considering the transportation time is studied. The dual resource integrated scheduling issue with multiple AGVs and multiple machines is modeled by mixed-integer programming (MIP). An improved Hybrid Discrete Salp Swam Algorithm (HDSSA) is proposed to optimize the maximum makespan. A three-layer coding scheme based on workpiece, machine and AGV is adopted to realize the SSA adapted to the discrete combinatorial problem. The purpose of a designed heuristic initialization approach in decoding is to raise the quality of the algorithm's initial solution. The idea of differential variation is implemented into the HDSSA to increase its global search capability through variation and crossover operators. To enhance the HDSSA's local search ability, a variable neighborhood search depending on problem feature is introduced to the optimal individual of each iteration. To prove the effectiveness of the algorithm improvement strategy and the proposed HDSSA, comparative experiments are conducted with other algorithms through standard instances. The results of the experiments suggest the proposed approach is capable of successfully resolving the integrated scheduling problem considering transportation time.

**INDEX TERMS** Flexible manufacturing shop scheduling, multi-AGV scheduling, salp swarm algorithm, variable neighborhood search, integrated scheduling.

## I. INTRODUCTION

With the promotion of Industry 4.0, the production mode of multi-species, small batch and personalization has become the norm. With its advantages of high efficiency and reliability, and strong environmental adaptability, Automated Guided Vehicles (AGVs) realizes automation and flexible distribution of workshop logistics, which is the key aspect to enhance the adaptability of the flexible production workshop [1]. However, the traditional flexible job shop scheduling problems (FJSP) often ignore the handling time of workpieces and materials or consider the handling time in the processing time. In contrast to the FJSP, the flexible job shop scheduling problem with AGVs transportation times (FJSPT) necessitates cooperation between AGVs, machines, workpieces, and other resources to reflect the advantages of vehicles. The addition of the AGV allocation problem leads to more complex constraints between resources such as machines, workpieces, and processes in FJSPT. The integrated scheduling of AGV resources and equipment resources is harder to solve because it involves more complexity and uncertainty. This is because the processing order of the job, the matching relationship between the transportation task and the AGV, and the matching relationship between the job and the machine must all be considered simultaneously.

Within the context of a flexible manufacturing system (FMS), the problem of dual resource integrated scheduling of machines and AGVs is broken down into the following four subproblems: machine selection, process sequencing,

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

AGV assignment, and AGV path planning. Bilge et al. [2], [3], [4] developed the first integrated scheduling model for AGVs and machines considering transportation time and solved the model by an exact algorithm.

Although exact algorithms can solve the optimal solution of the problem, it is hard to obtain approximate optimal solutions to big-scale problems in reasonable periods. Therefore, several intelligent optimization algorithms have been used to address machine and AGV scheduling problems in recent years. Considering the use of AGVs and the maximum makespan, Liu et al. [5] provided an enhanced flower pollination algorithm (FPA) to deal with simultaneous scheduling problems of equipment and aerial vehicles. Zhu et al. [6] solved the hybrid flow-shop scheduling mathematical model with machine-AGV joints and renewable energy using an optimum foraging algorithm (OFA). However, the models in the above literature lack consideration of machine flexibility.

For the integrated scheduling problems of AGVs in flexible job shop, Bekkar et al. [7] camp up with a greedy heuristic algorithm based on iterative insertion method to solve the FJSP by taking workpiece handling time into account. To handle workpiece and transportation scheduling in the FMS, Homayouni et al. [8] suggested a local search-oriented heuristic algorithm. Wang et al. [9] proposed an improved hybrid discrete difference evolutionary algorithm to solve the integrated scheduling of multi-loaded AGVs considering power constraints. However, AGV unloaded and loaded states are not distinguished during transportation. In the actual transportation process, AGV completes a transportation task, which is divided into two stages: no-load and load. Thus, Hu et al. [10] proposed an improved iterative local search method, and it selects AGVs using a rule of first come, first served (FCFS). Liu et al. [11] proposed a distribution algorithm with a low-carbon scheduling heuristic strategy to solve a low-carbon scheduling optimization problem that integrated multispeed flexible manufacturing and multi-AGV transportation.

The problem of integrating AGVs and machines with scheduling has a high degree of complexity, and research on integrated scheduling has focused on the following types of approaches. In terms of exact algorithms, Ham [12] proposes a constrained planning method for dealing with the simultaneous scheduling problem of job shop production and material transportation. In the level of simulation methods, Erol et al. [13] proposed a joint scheduling method for AGVs and machines in manufacturing systems based on multi agent-based systems (MAS). As machine learning algorithms such as deep reinforcement learning (DL) and reinforcement learning (RL) as well as transfer learning (TL) have strong data processing and environment interaction capabilities in mining historical scheduling data [14], [15]. Sun et al. [16] proposed an integration algorithm framework based on convolutional neural network and deep reinforcement learning, which expresses the environment state by combining the information extraction of analytic maps by CNN and artificial features and provides a new solution for the integrated scheduling of machines and AGVs in manufacturing systems. At present, optimization algorithms of machine and AGV integration scheduling problem are mainly focused on intelligent optimization algorithms, such as A, such as genetic algorithms (GA) [17], [18], discrete whale optimization Algorithms (WOA) [19] and hybrid particle swarm algorithms (PSO) [20].

Although the above algorithms can obtain a better optimal solution, once the number of processes, machines, and AGVs in an instance have changed, the algorithms need to be to be designed according to the problem characteristics to obtain an optimal solution. Therefore, there is no generalized method that can solve all FJSP problems in an exact optimal way, which is the ''No Free Lunch Theorem'' [21]. Among them, Mirjalili et al. [22] introduced the salp swarm algorithm (SSA), a novel kind of group intelligence algorithm with the benefits of little computation, few control parameters, and easy operation. It has been effectively used to address scheduling problems in job shops. For instance, while Zhao et al. [23] proposed an improved SSA to solve the FJSP, and Niu et al. [24] offered an adaptive SSA to solve the flexible job shop scheduling problem with the transportation time. The above studies confirm that the SSA can effectively solve the shop floor scheduling problem.

The integrated scheduling of AGV and machine is more complicated than the traditional flexible shop scheduling problem. Its research has important academic significance and practical value and will become a research hotspot in the field of intelligent manufacturing. Based on clarifying the concept of integrated scheduling problems, this paper studies machine and AGV integrated scheduling problem in the FMS from two aspects of model and algorithm. A mixed integer programming (MIP) model is constructed, and an improved SSA algorithm is proposed, which is helpful to the research and development of AGV and machine-integrated scheduling. This work makes four primary contributions: First, an improved hybrid discretized slap swarm algorithm (HDSSA) is proposed to solve benchmark cases of the FJSPT that are small and medium in size. The discretization operation of the SSA is realized by a three-three-layer encoding based on the transformation of the solution space. Second, the algorithm's convergence speed and the quality of the initial solution are intended to be enhanced by a heuristic population initialization strategy. Third, to balance the algorithm's performance in both local search and global exploration, the two-step control factor and adaptive update weights are introduced to the position update formulation. Fourth, the mutation and crossover operations of differential evolution idea are introduced into the SSA to improve algorithm's global search capability. Meanwhile, to avoid the algorithm from falling into local optimality, the variable neighborhood search (VNS) algorithm is integrated into the SSA to perform variable neighborhood search for the optimal individual after each iteration.

Finally, simulation experiments are used to verify the feasibility of the algorithm improvement strategy and the effectiveness of the HDSSA for solving the FJSPT. The influence of population initialization method and different improvement strategies on the solution results of the algorithm is analyzed.

## II. PROBLEM DESCRIPTION AND MATHEMATICAL MODEL

### A. ABBREVIATIONS AND ACRONYMS

The dual-resource integrated scheduling model of AGVs and machines in a flexible manufacturing shop with AGV transportation time can be described as: The flexible manufacturing system is composed of a set $n$ of workpieces to be processed, a set $m$ of machines, and a restricted number of AGVs responsible for handling workpieces. Each workpiece contains one or more processes; the order between the processes is predetermined, and each process is processed by at least one optional machine. Each AGV can carry the workpiece between any two machines. AGVs can move workpieces between any two machines within a certain transportation time. Every machine has a buffer zone where AGV can be parked. The loading and unloading of workpieces on AGVs can be accomplished with the buffer zone.

The AGV transporting workpieces may be separated into two phases: no-load trip and load trip. In the case of unloaded travel, the AGV needs to travel from the present position to the destination location to pick up the workpiece, while in the case of loaded travel, the AGV picks up the workpiece and transports it from the station to the location of the processing machine.

To simplify the flexible manufacturing shop AGV and machine dual resource integration scheduling problem considering AGV transportation time, the following conditions are assumed:

1) At the initial moment, all workpieces are in a state to be machined, all machines are idle, and all workpieces can be machined at the zero moment.
2) The starting position of all workpieces and AGVs is in the load/unload (LU) area, and all AGVs and machines are accessible at the zero moment.
3) Each workpiece can only be processed on one machine at a time for each process, and no interruption is allowed once processing has begun. The time spent loading and unloading the workpieces is counted as part of the machining time, and the machining time for each operation on optional machines is known.
4) There are sequential constraints on the process of the same workpiece and no constraints on the machining sequence between different workpieces.
5) Each autonomous guided vehicle has a maximum unit load capacity, and AGV can only transport a single piece of equipment at a time.
6) AGV loading and unloading workpiece time is not considered, machine setup time processing is calculated in the processing time.
7) All AGVs have a unit load capacity, and each AGV can only carry one workpiece at a time.
8) The AGV transportation route between any two machines is fixed, and the travel time is simply proportional to the distance between the machines.
9) Not considering AGV variability, such as AGV transportation speed, single charge transportation distance, and charging time.
10) Path conflicts, collisions, and deadlocks between AGVs are not considered.
11) The loading and unloading times of the AGV can be ignored.

**TABLE 1. Parameters and descriptions.**

| Symbols | Descriptions |
|---|---|
| | Sets and indices |
| $J_i$ | The set of workpieces to be machined, $J=\{J_1,J_2,\mathrm{K},J_n\}$ |
| $M$ | The set of machines, $M=\{M_0,M_1,\mathrm{K},M_\mathrm{m}\}$ |
| $A$ | The set of AGVs, $A=(A_1,A_2,\mathrm{K},A_\mathrm{g})$ |
| $H_i$ | The set of processes for workpiece $i$, $H_i=\{1,2,3,\mathrm{K},h_i\}$ |
| $O_{i(j)}$ | The $j$th operation of workpiece $i\in J$, $j\in H_i$ |
| | Parametric variable |
| $T(i)$ | The completion time of the workpiece $i\in J$, $j\in H_i$ |
| p | Index of operations assigned to $M_k$ |
| $T^s_{(k,p)}$ | The machining starts time of the operation $O_{i(j)}$ on the machine $M_k$ |
| $T_{(k,p)}$ | The process time of the operation $O_{i(j)}$ on the machine $M_k$ |
| $T^e_{(k,p)}$ | The machining end time of the operation $O_{i(j)}$ on the machine $M_k$ |
| $ET^{M_\mathrm{k}}_{M_\mathrm{k'}}$ | The no-load time of the transportation task $A_v$ from machine k to machine k′ |
| $LT^{M_\mathrm{k}}_{M_\mathrm{k'}}$ | The load time of the transportation task $A_v$ from machine k to machine k′ |
| $q$ | Index of traveling tasks assigned to $A_v$ |
| $ET^s_{(v,q)}$ | The start time of the no-load state of transporting the operation $O_{i(j)}$ on $A_v$ |
| $ET_{(v,q)}$ | The time of completing the operation $O_{i(j)}$ in AGV no-load transportation |
| $ET^e_{(v,q)}$ | The end time of the no-load state of transporting the operation $O_{i(j)}$ on $A_v$ |
| $LT^s_{(v,q)}$ | The start time of the load state executing the operation $O_{i(j)}$ on $A_v$ |
| $LT_{(v,q)}$ | The time of completing the operation $O_{i(j)}$ in AGV load transportation |
| $LT^e_{(v,q)}$ | The end time of the load state of transporting the operation $O_{i(j)}$ on $A_v$ |
| | Decision variables (binary variable) |
| $X(k,ij)$ | The decision variable of machine processing |
| $XP(k,pqij)$ | The decision variable of machine processing order |
| $Y(v,ij)$ | The decision variable of AGV transportation |
| $YP(v,pqij)$ | The decision variable of AGV transportation sequence |

### B. MATHEMATICAL MODEL

#### 1) PARAMETER SYMBOLS AND DEFINITIONS

To establish the problem model, the following symbols and variables are introduced, as shown in Table 1.

The constraints of the decision variables are as follows:

$$X(k, ij) = \begin{cases} 1, & \text{if the operation } O_{i(j)} \text{ is processed on machine } k \\ 0, & \text{else} \end{cases}$$

$$XP(k, pqij) = \begin{cases} 1, & \text{if operation } O_{i(j)} \text{ is processed on machine k after operation } O_{p(q)} \\ 0, & \text{else} \end{cases}$$

$$Y(v, ij) = \begin{cases} 1, & \text{if task } A_{i(j)} \text{ selects AGV } A_v \text{ for transportation} \\ 0, & \text{else} \end{cases}$$

$$YP(v, pqij) = \begin{cases} 1, & \text{if task } A_{i(j)} \text{ is transported on } A_v \text{ after task } A_{p(q)} \\ 0, & \text{else} \end{cases}$$

#### 2) TIME NODE MODEL

1) The machining starts time of the operation $O_{i(j)}$ on the machine $M_k$.

When the process and its immediately preceding process are machined on the same machine, the start time of the $O_{i(j)}$ is equal to the end time of the $O_{i(j-1)}$. In all other cases, the start time of the $O_{i(j)}$ is equal to the end time of the load state of transportation task $A_{i(j-1)}$. The start time of machining the $O_{i(j)}$ on the machine is calculated by Equation (1):

$$T^s(k, ij) = \begin{cases} T^e(k, i(j-1)), \sum_{k=0}^{m}\sum_{p=1}^{n}\sum_{q=1}^{h_p+1} XP(k, pqij) \times (i-p+1) \times (j-l) = 1 \\ \max\left\{ \begin{array}{l} LT^e(v, ij), \\ \sum_{k=0}^{m}\sum_{p=1}^{n}\sum_{q=1}^{h_p+1} T^e(k, pq) \\ \quad \times X(k, ij) \times X(k, pq) \times XP(k, pqij) \end{array} \right\}, else \end{cases} \tag{1}$$

2) The machining end time of the $O_{i(j)}$ on the machine $M_k$.

The end time of machining the $O_{i(j)}$ on the machine is equal to the sum of the start time of the $O_{i(j)}$ and its machining time on the corresponding machine, as shown in equation (2):

$$T^e(k, ij) = T^s(k, ij) + \sum_{k=0}^{m} X(k, ij) \times T(k, ij) \tag{2}$$

3) The start time of the no-load state of transporting the operation $O_{i(j)}$ on $A_v$.

The start time of the no-load trip of the AGV transportation task is divided into two cases, as shown in Equation (3): when is the first transportation task on the corresponding AGV, its start time of the no-load trip is 0; otherwise, the start time of the no-load trip is calculated according to the formula for the other cases:

$$ET^S(v, ij) = \begin{cases} 0, \sum_{v=1}^{g}\sum_{p=1}^{n}\sum_{q=1}^{h_p+1} YP(v, pqij) = 0 \\ \sum_{v=1}^{g}\sum_{p=1}^{n}\sum_{q=1}^{h_p+1} LT^e(v, pq) \times YP(v, pqij), else \end{cases} \tag{3}$$

4) The end time of the no-load state of transporting the operation $O_{i(j)}$ on $A_v$.

When a workpiece's initial process is being transported, which also happens to be the AGV's first transport job, or when the process being carried is being processed on the same machine as its immediate previous one, the AGV transport task does not have an unloaded transport status. In all other circumstances, the end of the unloaded state is larger than the unloaded AGV's arrival time and the prior process's completion time. The end time of the unloaded trip of the AGV transportation task is represented by equation (4):

$$ET^e(v, ij) = \begin{cases} ET^s(v, ij), \\ \sum_{k=0}^{m}\sum_{p=1}^{n}\sum_{q=1}^{h_p+1} XP(k, pqij) \times (i-p+1) \times (j-q) \\ = 1 or \left( \sum_{v=1}^{g}\sum_{p=1}^{n}\sum_{q=1}^{h_p+1} \times YP(v, pqij) = 0 \ \ and \ j = 1 \right) \\ \max\left\{ \begin{array}{l} ET^s(v, ij) + \\ \sum_{v=1}^{g}\sum_{p=1}^{n}\sum_{q=1}^{h_p+1}\sum_{k=0}^{m}\sum_{k'=0}^{m} X(k, i(j-1)) \\ \quad \times X\left(k', ij\right) \times YP(v, pqij) \times Y(v, ij) \\ \quad \times ET_{M_k}^{M_{k'}}, T^e(k, i(j-1)) \end{array} \right\}, else \end{cases} \tag{4}$$

5) The start time of the load state executing the operation $O_{i(j)}$ on $A_v$.

The running process of AGV can be divided into three states: no-load, load and waiting. When the AGV's no-load trip is over, if the process $O_{i(j-1)}$ immediately preceding the AGV's transportation task corresponding to the $O_{i(j)}$ is completed, the AGV immediately enters the load state; otherwise, it needs to wait for the completion of the $O_{i(j-1)}$ in the buffer zone of the machine before it can enter the load state. The start time of the load state of the AGV transportation task $A_{i(j)}$ is

represented by equation (5):

$$LT^s(v,ij) = \begin{cases} ET^e(v,ij), ET^e(v,ij) = T^e(k,i(j-1)) \\ ET^e(v,ij) + WT(v,ij), else \end{cases} \tag{5}$$

6) The end time of the load state of transporting the operation $O_{i(j)}$ on $A_v$.

As illustrated by Equation (6), the end time of the load state of the AGV transporting task A is separated into two cases: If the $O_{i(j)}$ is processed on the same machine as its immediately preceding process, there will be no load state of the AGV if process A is processed on the same machine as its immediately preceding process $O_{i(j-1)}$. Otherwise, there should be a load state of the AGV that is greater than the arrival time of the load and the completion time of the processing of process $O_{p(q)}$, which is processed before the $O_{i(j)}$, on the machine corresponding to the $O_{i(j)}$.

$$LT^e(v,ij) = \begin{cases} LT^s(v,ij), \sum_{k=0}^{m}\sum_{p=1}^{n}\sum_{q=1}^{h_p+1} XP(k,pqij) \\ \quad \times (i-p+1) \times (j-l) = 1 \\ \max\left\{ \begin{array}{l} LT^s(v,ij) + \\ \sum_{v=1}^{g}\sum_{k=0}^{m}\sum_{k'=0}^{m} X(k,i(j-1)) \\ \quad \times X\left(k',ij\right) \times Y(v,ij)\, LT_{M_k}^{M_{k'}}, T^e(k,pq) \end{array} \right\}, else \end{cases} \tag{6}$$

### 3) THE FORMULATION OF MIP MODEL

The following mathematical model, objective function, and constraint conditions are set up to address AGV and machine joint scheduling problem in a flexible manufacturing shop:

$$\textbf{\textit{Objective}}: \min C_{\max} = \min\left(\max_{i=1}^{n} \mathrm{T}(i)\right) \tag{7}$$

$$\sum_{k=1}^{m} X(k,ij) = 1 \tag{8}$$

$$T^e(k,ij) = T^s(k,ij) + \sum_{k=1}^{m} X(k,ij) \times T(k,ij) \tag{9}$$

$$\begin{aligned} &T^s(k,ij) \geq LT^e(v,ij), \\ &\quad T^s(k,ij) \geq T^e(k,pq) \cdot (1 - XP(k,pqij)), \\ &\qquad \forall i,p \in \{1,2,\ldots,n\}, \quad \forall j,q \in \{1,2,\ldots,h_i\} \end{aligned} \tag{10}$$

$$T^s(k,ij) \geq T^e(k,i(j-1)) \cdot (1 - XP(k,i(j-1))) \tag{11}$$

$$\sum_{v=1}^{g} Y(v,ij) = 1 \tag{12}$$

$$\begin{aligned} &ET^s(v,ij) \geq LT^e(v,pq) \cdot \left(1 - YP_{(v,pqij)}\right), \\ &\quad ET^s(v,ij) \geq T^s(k,ij) \\ &\qquad \forall i,p \in \{1,2,\ldots,n\}, \quad \forall j,q \in \{1,2,\ldots,h_i\} \end{aligned} \tag{13}$$

$$ET^e(v,ij) = ET^s(v,ij) + \sum_{v=1}^{g} Y(v,ij) \times ET(v,ij) \tag{14}$$

$$LT^s(v,ij) \geq LT^e(v,ij), LT^s(v,ij) \geq T^e(k,i(j-1)) \tag{15}$$

$$LT^e(v,ij) = LT^s(v,ij) + \sum_{v=1}^{g} Y(v,ij) \times LT(v,ij) \tag{16}$$

The objective function (7) represents the maximum completion time of all workpieces. Constraint (8) states that any one process can only be executed on one machine at the same moment. Constraint (9) assures that the machining completion time on the machine for each workpiece is equal to the sum of the start machining time and the necessary machining time. Constraint (10) demonstrates that the starting processing time of each workpiece on the machine depends on the arrival time of the AGV transporting the workpiece and the larger value of the end processing time of a process on the machine. Constraint (11) indicates that the processing start time of any process for each workpiece is not less than the end processing time of the previous process. Constraint (12) assures that no more than one workpiece can be traveled by an AGV at one time. Constraint (13) states that each AGV's the unloaded departure time is at least greater than its previous carried workpiece's load finish time and its start processing time. Constraint (14) ensures that the unloaded end transportation time of each AGV is equivalent to the total of the unloaded start time and the unloaded movement time of the AGV between machines. Constraint (15) represents that the start transportation time of each workpiece depends on the larger unloaded arrival time at the machine of each AGV and the machining completion time. Constraint (16) indicates that the AGV's entire end transportation time is equivalent to the total of its start transportation time and the load movement time between machines.

## III. METHODOLOGY

The flexible manufacturing shop machine and AGV dual resource integration scheduling problem is an extension of the FJSP, which contains three dimensions of process sequencing, machine selection, and AGV scheduling and belongs to the discrete combinatorial optimization problem. However, the basic SSA works on optimization problems in the continuous domain.

Therefore, this section presents an improved hybrid discrete salp swarm algorithm to solve the constructed MIP model. First, the continuous solution space of SSA is mapped to the discrete solution space through a conversion mechanism to accommodate the discrete search process of the algorithm. Meanwhile, the idea of differential variation is presented to perturb the process, AGV, and machine coding to enhance the algorithm's capacity for global search. Finally, the variable neighborhood search algorithm is fused into the improved SSA to perform a local search for the optimal individual in each iteration to avoid the HDSSA from falling into local optimality.

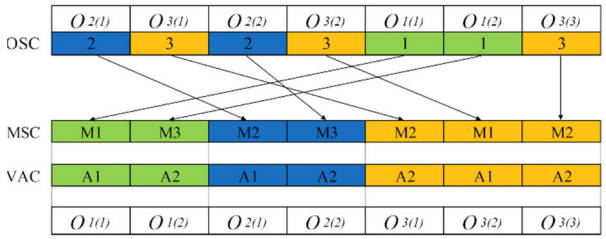



**FIGURE 1.** Coding scheme.

### A. ENCODING AND DECODING SCHEME

Given that the flexible manufacturing shop machine and AGV integration scheduling problem consists of three scheduling subproblems: process sequencing, machine assignment, and AGV assignment. A three-stage coding approach is used to construct the scheduling solution. The first row is the Operation Sequencing Code (OSC), while the Machine Selection Code (MSC) and AGV Assignment Code (VAC) are the second and third rows, respectively.

A flexible job shop scheduling coding scheme with 3 workpieces, 3 machines, and 2 AGVs is shown in Figure 1. Each vector in the OSC represents the number of the workpiece, and the times of occurrences of the job number from left to right corresponds to the number of processes of the workpiece.

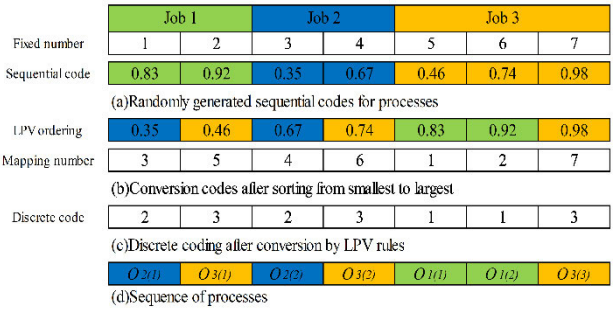



**FIGURE 2.** Schematic diagram of discretized process coding for LPV rule.

Each element of the machine selection code and AGV assignment code represents the machine number and AGV number selected for the workpiece process, respectively.

The solution space of the SSA algorithm is continuous, the process ranking codes are discrete points in the solution space. Therefore, the algorithm's continuous solution space needs to be transferred to the discrete solution space so as for the SSA to be effective in tackling discrete combinatorial optimization problems. For the flexible manufacturing shop floor integrated scheduling problem, each one-dimensional vector of each SSA individual location corresponds to one process. It can be represented by a [0, 1] random number, and a fixed sequence number starting from 1 is set from left to right to correspond to the random number one by one. Then, the random numbers are ordered from smallest to largest using the LPV (Largest Position Value) rule [25]. The discrete encoding of the individual is constructed based on the mapping number corresponding to each random number. Finally, according to the process number, each workpiece is converted into a process sequence for decoding. The specific transformation process is shown in Figure 2.

### B. HEURISTIC POPULATION INITIATION

The algorithm's convergence speed and solution quality are highly dependent on the quality of the initial solution. After determining the process sequencing code, it is necessary to confirm the MSC and the VAC. Since each process has a different set of optional machines, if the random selection method of sorting coding is adopted, a certain number of invalid solutions will be generated. Therefore, we design a heuristic initialization method based on the principles of the earliest arrival of the AGV to the workpiece to be handled and the earliest completion of processing by the machine.

This method first gets the next process to be manufactured in the scheduling sequence, the available time and position that each AGV has completed its last task assignment, and the time of completion at which each machine is able to handle the operation. Then, the AGV at the current position or the load/unload (LU) area is selected to load the workpiece $i$. Finally, the machine to be used for the process is selected. Repeat the above operation until all processes have been decoded, and the AGV assignment code and machine selection code can be determined. The decoding steps for the heuristic method are as follows:

1) **Step1:** Randomly initialize the OSC, traverse each process of the OSC, determine the codes of machine selection and AGV assignment.
2) **Step2:** Get the available time $T\,(v, q)$ and location $L\,(v, M_{k'})$ of each AGV and the available time of each machine $T\,(m, p)$.
3) **Step3:** Determine whether the $O_{i(j)}$ is the first process of $J_i$. If yes, skip to Step (1); otherwise, skip to **Step 4**.
   - *(1) Determine whether there is an idle AGV in the load/unload (LU) area. if so, select an AGV in the LU area to carry the workpiece. Otherwise, traverse all the AGVs and select the one with the shortest unloading time to the LU to transport the workpiece.*
   - *(2) After determining the AGVs for the handling process, the earliest machine to complete the process is selected under the conditions of satisfying the AGV resource constraints and the processing machine constraints.*
   - *(3) Calculate time of departure of this AGV from the LU:*

$$LT^{s}(v, q) = T\,(v, q) + ET^{LU}_{M_k} \tag{17}$$

   - *(4) Update the available time and position of AGVs:*

$$T^{*}(v, q) = LT^{s}(v, q) + ET^{M_{i(1)}}_{LU}, L^{*}(v, M_k) = L(v, M_{i(1)}) \tag{18}$$

- *(5) Update the available time of allocated machines and calculate the completion time of workpieces:*

$$T^{\mathrm{e}}(k, i1) = \max\left\{T(v^*, q), T(m, p)\right\} + T(k, i1) \tag{19}$$

4) **Step4:** Obtain the completion time of the process $T^{\mathrm{e}}(k, i(j-1))$ that is required to be completed immediately prior to the workpiece and the $M_{i(j-1)}$ that operates it.

- *(1) Determine whether there is an idle AGV on the buffer of the machine in the process immediately prior to the workpiece; if so, select that AGV for handling; otherwise, select the AGV that arrived at the machine earliest.*
- *(2) Calculate the time at which the vehicle leaves the machine $M_{i(j-1)}$:*

$$LT^{s}(v, q) = \max\left\{T(v, q) + ET_{M_k}^{M_{i(j-1)}}, T^{\mathrm{e}}(k, i(j-1))\right\} \tag{20}$$

- *(3) Select the earliest machine to complete the process $O_{i(j)}$ while satisfying the AGV resource constraints and processing machine constraints.*
- *(4) If the optional machine for the $O_{i(j)}$ is different from the processing machine of the preceding process ($M_{i(j)} \neq M_{i(j-1)}$), update the available time and position of the assigned AGV:*

$$\begin{aligned} T^*(v, q) &= LT^{e}(v, q) = LT^{s}(v, q) + ET_{M_{i(j-1)}}^{M_{i(j)}}, \\ L^*(v, M_k) &= L(v, M_{i(j)}) \end{aligned} \tag{21}$$

- *(5) Calculate the workpiece completion time and update the machine's available time:*

$$T^{\mathrm{e}}(k, ij) = \max\left\{T\left(v^*, q\right), T(m, p)\right\} + T(k, ij) \tag{22}$$

5) **Step5:** Iterate through each process of the OSC and output the AGV assignment code, machine selection code, AGV load time, no-load time, and job completion time.

Based on the coding scheme in Section III-A and the decoding approach in Section III-B, to facilitate the understanding of the FJSPT, a scheduling scheme containing five workpieces, eight machines, and 3 AGVs for transportation is randomly generated, and its Gantt chart is shown in Figure3.

The AGV1 is used as an example to illustrate the process of handling workpieces and the processing of workpieces. The transportation process of $A_1$: The $A_1$ loads workpiece $J_2$ from the LU and travels with the load to the $M_2$. ➔ After the $J_2$ is unloaded, the AGV drives with the empty to the LU (loading workpiece $J_1$) and then travels with load to the $M_1$. ➔ Waiting in the buffer zone of the $M_1$ for the $O_{1(1)}$ to be completed, then load the $J_1$ and drive to the buffer zone of the $M_4$ with the load. ➔ The $O_{2(1)}$ is transported to the $M_5$ by AGV after processing on the $M_2$, where it waits in the buffer zone for the $O_{2(2)}$ to be completed. ➔ Continue loading the $J_2$ from the $M_5$ to the $M_4$ with the load, wait in the buffer zone of the $M_4$ until the $O_{2(3)}$ is completed, and then load the $J_2$ to the LU.

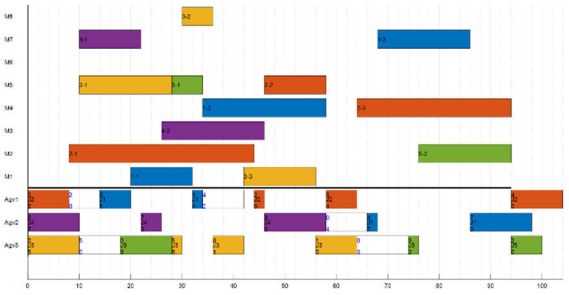


**FIGURE 3.** The Gantt chart for the 8-5 instance scheduling solution.

### C. DUAL ADAPTIVE STEP CONTROL WEIGHTING STRATEGY

The standard SSA algorithm has two flaws: it converges slowly and is prone to local optimization during the iterative phase. This section aims to improve the global search capabilities of the SSA by including two important step-length control factors($\omega_1$ and $\omega_2$) into the position update formula of leaders in salp population. The leaders' position update formula in the SSA with the introduction of two step-size control factors is shown in Equation (23):

$$x_j^1(t) = \begin{cases} \omega_1 \times F_j + c_1\left(\left(ub_j - lb_j\right)c_2 + lb_j\right), & c_3 \geq 0.5 \\ \omega_2 \times F_j - c_1\left(\left(ub_j - lb_j\right)c_2 + lb_j\right), & c_3 < 0.5 \end{cases} \tag{23}$$

where, $F_j$ denotes the $j$th dimensional food source location. The $w_1$ acts in the first half of the algorithm to expand the algorithm's global search range, and the $w_2$ acts in the second half of the algorithm to expand the algorithm's local search capability. The $w_1$ and $w_2$ cooperate with each other in the whole stage of the algorithm to improve the drawbacks of the inefficient search of the SSA. The parameter $c_1$ is the most important parameter in SSA, which gradually changes with the increase of the number of iterations and finally approaches 0, and its value is updated by the following formula model.

$$c_1 = 2e^{-(4t/T)} \tag{24}$$

The parameter $c_2$ and $c_3$ are random numbers distributed in [0, 1]. The $c_2$ controls the step size of the leader's movement while searching in the $j$th dimension, the $c_3$ determines the positive or negative direction of the leader's movement.

The formulas for the two critical step control factors $w_1$ and $w_2$ are as follows:

$$\omega_1 = (1 - t/T_{\max} + 1)^2 \exp\left(-2\pi(t/T_{\max})\right) \tag{25}$$

$$\omega_2 = (2 - 2t/T_{\max} + 1)^2 \exp\left(-2\pi(t/T_{\max})\right) \tag{26}$$

where, $t$ is the current number of iterations and $T_{\max}$ is the maximum number of iterations.

The position change of the follower in the standard SSA algorithm belongs to the behavior of undefined following, which is not affected by any random parameter. It depends entirely on the influence of the position of the *ith* individual and the *(i-1)th* individual of the last iteration. This updating mechanism leads to a local optimum stagnation problem and will limit the search efficiency of the algorithm in the later iteration.

To enable the follower to search locally in the vicinity of the better solution, this paper modifies the follower position update formula and proposes an adaptive inertial position update model. The formula for calculating the inertia weight coefficients proposed by adopting the literature [26] is as follows:

$$\omega_3 = (\omega_{\max} - \omega_{\min}) \cdot \frac{e^{10} - \mu t}{e^{10} - \mu t + \lambda} + \omega_{\min} \tag{27}$$

where, the $\omega_{\max}$ and $\omega_{\min}$ indicate the highest and values of the weights, respectively. The $\lambda$ and $\mu$ are constants, which are set to $\omega_{\max} = 0.9, \omega_{\min} = 0.2, \lambda = 3$, and $\mu = 0.04$ in this paper. The value of $\omega_3$ decreases down in a nonlinear way as the program iterates.

The new follower position update equation introducing the inertia weight adjustment strategy is as follows:

$$x_j^i(t) = \frac{1}{2}\left(x_j^i(t-1) + \omega_3 x_j^{i-1}(t-1)\right) \tag{28}$$

The overall search capacity of the algorithm is enhanced by using larger inertia weights in the early iterations. To reduce the impact of the previous follower on the subsequent follower and enhance the algorithm's convergence accuracy, lowering the inertia weights in later stages of the algorithm can cause the subsequent follower to explore more thoroughly in the domain of the superior solution.

### D. PERTURBATION MECHANISMS BASED ON MUTATION AND CROSSOVER OPERATIONS

Given that the genetic algorithm possesses superior global search capability, while the difference-based mutation and crossover operations reduce the complexity of the operation. Differential Evolutionary algorithms (DE) have also been applied by Wang et al. [9] to address machine and multi-load AGV scheduling in FMS. To increase the SSA's capacity for global search, the original SSA is improved by introducing the idea of differential variation and fusing variation and crossover operators, as shown in equation (29):

$$X_i^{t+1} = p_m \otimes F_1(X_i^t) + p_c \otimes F_2\left(V_i^t, X_i^t\right) + F_3\left(f\left(U_i^t\right), f\left(X_i^t\right)\right) \tag{29}$$

where, $p_m$ is the mutation probability; $F_1$ denotes the mutation operation. The mutated individual $V_i^t$ is obtained after performing the mutation operation on the target individual $X_i^t$. The $p_c$ is the crossover probability, and the $F_2$ is the crossover operation. The experimental individual $U_i^t$ is obtained by the crossover operation between the $V_i^t$ and the $X_i^t$. The $f(U_i^t)$ and $f(X_i^t)$ are the fitness of the $U_i^t$ and the $X_i^t$, respectively. $F_3$ is the selection operation, which compares the fitness of the two individuals and selects the one with higher fitness to enter the next generation.

Based on the characteristics of the three layers of coding above, variant operations are performed for process, AGV and machine code, respectively. For the operation sequence code (OSC), the IPOX [27] crossover operator is used in the following way: First, the set of workpieces $J_i$ is divided into two non-empty complementary subsets at random: $S_1$ and $S_2$. Second, the process position belonging to $S_1$ in the parent process code $P_1$ is reserved for the child code $C_1$. The position of the process belonging to $S_2$ in the parent process code $P_2$ is reserved in the same position of the child code $C_2$. Finally, the process number belonging to $S_2$ in $P_2$ is copied to the remaining position of $C_1$, and the process number belonging to $S_1$ in $P_1$ is copied to the remaining position of $C_2$, keeping the original order. Taking three jobs with different processes for each job as an example, as shown in Figure 4.

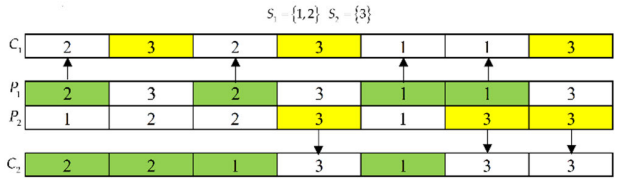



**FIGURE 4.** IPOX crossover operator for OSC.

For AGV assignment code (VAC), a two-point crossover operation is used. First, the two selected parents randomly generate two crossover points. Then, the corresponding AGV numbers in the parent's code between the two set crossover points are exchanged.

For machine selection code (MSC), the MPX [28] crossover operator is used. The procedure is as follows: First, a set M consisting of integers 0 and 1 with length equal to the parent's encoding is randomly generated. Sequentially, the machine serial number of the parent code $P_1$ at the position corresponding to 1 in M is copied to the same position in the child code $C_1$. Then, the machine serial number of the parent code $P_2$ at the position corresponding to 1 in M is copied to the same position in the child code $C_2$. Finally, the other machine serial numbers in the parent codes $P_1$ and $P_2$ are retained in the child codes $C_1$ and $C_2$. The detailed procedure is shown in Figure 5.

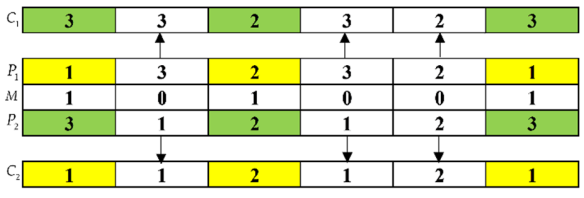



**FIGURE 5.** MPX crossover operator for MSC.

### E. LOCAL SEARCH STRATEGY

The local search strategy is designed to prevent premature convergence and improve the SSA's neighborhood search capability. By altering the neighborhood structure in response to the characteristics of the problem, the VNS algorithm can avoid the algorithm falling into local optimum and expands the search space of the algorithm. Therefore, this section designs three neighborhood structures to search for neighborhood solutions that can change the maximum completion time. After each iteration, the neighborhoods of operation sequence code, machine selection code and AGV arrangement code of individual are explored, respectively. Based on the features of the FJSPT, three neighborhood structures are constructed, as shown in Figure 6.

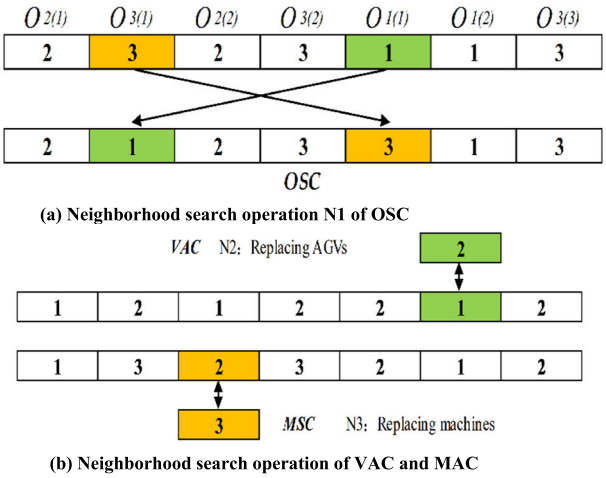



**FIGURE 6.** Example of N1, N2 and N3 for OSC, VAC and MSC, respectively.

1) Neighborhood structure of operation sequence code $\mathbf{N_1}$**:** Randomly select two coding positions, the two positions correspond to the *ith* and *jth* two processes. Then, the codes corresponding to the two positions are exchanged.
2) Neighborhood structure of AGV assignment code $\mathbf{N_2}$**:** First, process $j$ in the process of workpiece $i$ is randomly selected. Then, the AGV number at the original process location is replaced with another AGV that is chosen at random from the group of optional AGVs.
3) Neighborhood structure of machine selection code $\mathbf{N_3}$**:** Randomly selects process j among the processes of workpiece $i$. Then, other machine numbers are randomly selected from the set of selectable machines to replace the machine numbers at the original process locations.

Based on the above three neighborhood structures, the specific steps of variable neighborhood search are as follows:

**Step1:** The optimal solution in the population is taken as the initial solution S. Select the set of neighborhood structures $N_k, (k = 1, \cdots, k_{max})$ for shaking phase and the set of neighborhood structures $N_l, (l = 1, \cdots, l_{max})$ for local search. Initialize the parameter and choose stopping criteria: $k = 1, k_{max} = 3, l_{max} = 40$.

**Step2:** Repeat the following sequence until the termination condition is satisfied: If $k > k_{max}$, then the current solution S is taken as the currently searched optimal solution $S_{best}$, and the optimal solution in the original population is updated; otherwise,it enters Step3.

**Step3: S**harking: a perturbation solution $S'$ is generated by passing the current solution $S$ through the $k^{th}$ neighborhood $N_k(S)$.

**Step4:** Local search:

*(1) The perturbed solution $S'$ is taken as the initial solution; set $l = 1$.*
*(2) If $l > l_{max}$ then output the local optimal solution $S'$; otherwise, go to **step (3)**.*
*(3) Find the best neighbor $S''$ of $S$ in $N_l(S)$. If $f(S'') < f(S')$, then $S' = S'', l = 1$; otherwise, set $l = l + 1$ and go to **step(2)**.*

**Step5:** Move or not: if $f(S'') < f(S)$, then $S = S'$, go to **step4**; otherwise, set $k = k + 1$, go to **step2.**

### F. THE IMPLEMENTATION OF HDSSA

The hybrid discrete salp swarm algorithm that was developed by improving the fundamental SSA, and the overall process for the proposed HDSSA is as follows:

**Step1** Input scheduling information: workpiece processing information, AGV number and workshop system layout. Initialize the control parameters of the HDSSA: the population size *Pop*, maximum number of iterations $T_{max}$, probability of mutation $P_m$, and probability of crossover $P_c$.

**Step2** Evaluation: According to the design coding scheme and the priming approach proposed in Section III-B, the ***Pop*** individuals are coded by the three layers of process, AGV, and machine coding vectors as into an initial population. The historical optimal solution for each target individual and the global optimal solution for the target population are computed by decoding.

**Step3** Update population: According to the improved SSA position update method in Section III-C to update the position of the leader and the follower respectively: introduce double-step control coefficients and adaptive parameters to adjust and optimize the updating of the leader and the leader's position method respectively, update the fitness of the target population ***Pop***, and record the current optimal solution.

**Step4** Mutation and cross perturbation: according to the mutation probability, the mutation operation is carried out on the individual process vector, AGV vector, and machine vector, respectively, to obtain the mutation population ***Pop1***. According to the crossover probability, the crossover operation is conducted on the target population ***Pop*** and the variant population ***Pop2*** to obtain the test population ***Pop3***.

**Step5** Selection: The fitness values of the test individuals and the target individuals are judged, and the better individuals are selected to enter the next generation for local search using the VNS algorithm.

**Step6** Local search: The optimal individual in the population is used as the initial solution. Then, the current optimal individual is obtained, and the target population is updated by performing variable neighborhood search based on the neighborhood structure $N_k$ for individual's process sequencing, machine selection, and AGV assignment, respectively.

**Step7** Termination: judge whether the algorithm reaches the termination condition; if the algorithm termination condition has not been met, go to **Step3**; otherwise, output the global optimal solution.

Algorithm 1 is the pseudocode of the HDSSA algorithm, and Figure. 7 is its flowchart.

**Algorithm 1** The Pseudo-Code of SSA

*Initialize the swarm size and positions of population considering ub and lb*
***While*** *(end condition is not satisfied)* ***do***
  *Calculate the fitness of each search salp, save F as the location of food sources*
  *Update* $c_1$ *by Eq. (24)*
  ***for*** *each salp(xi)* ***do***
    ***if*** $i == 1$ *(leder)* ***then***
    *Update the position of leading salp by Eq.(23)*
    ***else if (follower)***
    *Update the position of follower salp by Eq.(28)*
    ***end if***
  ***end for***
  *Modify positions of salps base the upper and lower bounds of variables*
  *Apply the greedy search corresponding to each salp position*
  ***if*** rand $<$ *Pm* ***then***
  *Execute the differential variation mechanism by Eq.(29)*
  *Update F if there is a better solution*
  ***end if***
  *Choose the* optimal *individuals for variable neighborhood search*
  *Update the food source location*
***end while***
*Return the food position F*

### G. COMPUTATIONAL COMPLEXITY

The computational complexity is an important property that reflects the operational efficiency of algorithms. The computational complexity of the classical SSA depends on the population size($N$), dimensions($D$), and the computational amount of objective function($Cof$). Therefore, according to the solving steps of the SSA algorithm and the calculation method of time complexity, the time complexity of basic SSA algorithm is $O(T_{max}(N \times D) + N \times Cof(D)))$.

For HDSSA, it has been discussed in Section III that the heuristic population initialization, adaptive strategy, differential evolutionary mechanism, and local search are the main computational steps of HDSSA. There are changes in the exploitation phase and hence the updated computational complexity for all procedures is given by:

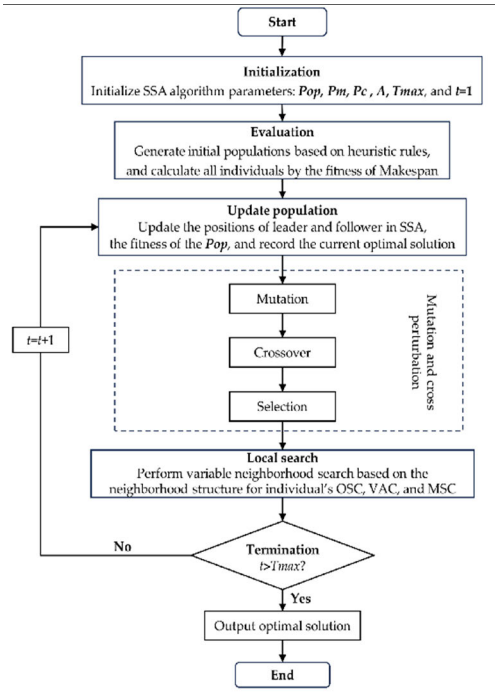



**FIGURE 7.** The flowchart of the improved HDSSA.

1) Assuming that the number of machines is m and the time to calculate the load time of machines is $t_1$, the complexity for heuristic initialization is given by $O(0.5N \times D + 0.5N \times D \times (m \times t_1) = O(N \times D)$.
2) After initialization, the run time complexity of evaluating the current population is given by $O(N \times Cof(D) + t_1 \times D) = O(N \times (Cof(D) + D))$.
3) By assuming that the computation times of the two-step control factor ($\omega_1$ and $\omega_2$) and the adaptive inertia weights $\omega_3$ are $t_2$ and $t_3$, the computational complexity of the HDSSA update position for this process is given by $O(N \times D \times t_2 + N \times t_3) = O(N \times D)$.
4) In the iterative search, the computing time for executing the DE mechanism is $t_4$, then the complexity of the mutation scheme is $O(N \times D \times t_4) = O(N \times D)$, and the computational time complexity of updating the global optimal solution is given by O($N$).
5) The time to construct the neighborhood is assumed to be $t_5$ and the complexity of the local search is given by $O(k_{\max} \times l_{\max} \times Cof(D) \times N \times t_5) = O(Cof(D))$.

Based on the above aspects, ignoring the coefficients of the higher order terms and the lower order terms,the total complexity of the HDSSA for $T_{max}$ number of iterations is given by $3O(N \times D) + O(N \times (D + Cof(D))) + O(N) + O(Cof(D)) = O(T_{max}(N \times D + Cof(D)))$, so, its time complexity belongs to the same order of magnitude and is the same as that of the classical SSA.

**TABLE 2.** Comparison of optimization results of benchmark functions.

| Function | Algorithm | Mean | Std | Function | Algorithm | Mean | Std |
|---|---|---|---|---|---|---|---|
| | SSA | 6.45E-08 | 5.12E-08 | | SSA | 1.01E+03 | 2.01E+03 |
| | **HDSSA** | **5.26E-42** | **6.32E-42** | | **HDSSA** | **2.53E-13** | **3.67E-13** |
| $F_1$ | TACPSO | 5.55E-03 | 7.18E-03 | $F_3$ | TACPSO | 3.67E+00 | 3.80E+00 |
| | ESCA | 2.04E-06 | 6.38E-06 | | ESCA | 1.01E+01 | 2.61E+01 |
| | IGWO | 8.81E-33 | 8.91E-33 | | IGWO | 2.85E-07 | 2.95E-07 |
| | SSA | 1.09E+00 | 3.29E+00 | | SSA | 2.01E+03 | 3.10E+03 |
| | **HDSSA** | **7.21E-12** | **2.59E-12** | | **HDSSA** | **2.61E+01** | **2.11E-01** |
| $F_4$ | TACPSO | 2.01E+00 | 4.68E+00 | $F_5$ | TACPSO | 5.78E+01 | 1.57E+01 |
| | ESCA | 1.92E+00 | 2.87E+01 | | ESCA | 3.56E+01 | 1.30E+01 |
| | IGWO | 7.01E-06 | 3.82E-05 | | IGWO | 3.52E+01 | 2.67E-01 |
| | SSA | 1.13E+02 | 3.80E+02 | | SSA | 1.27E+01 | 4.37E+01 |
| | **HDSSA** | **0.00E+00** | **0.00E+00** | | **HDSSA** | **0.00E+00** | **0.00E+00** |
| $F_6$ | TACPSO | 2.91E-02 | 8.14E-02 | $F_9$ | TACPSO | 2.67E+00 | 6.81E+00 |
| | ESCA | 1.85E-01 | 6.81E-01 | | ESCA | 3.71E-04 | 2.71E-04 |
| | IGWO | 1.83E-04 | 5.89E-04 | | IGWO | 4.07E+01 | 2.85E+01 |
| | SSA | 3.37E+01 | 4.77E+01 | | SSA | 2.01E-01 | 3.10E+00 |
| | **HDSSA** | **1.27E-14** | **3.59E-14** | | **HDSSA** | **0.00E+00** | **0.00E+00** |
| $F_{10}$ | TACPSO | 1.04E+00 | 5.91E+00 | $F_{11}$ | TACPSO | 1.78E+00 | 1.45E+00 |
| | ESCA | 3.36E-04 | 2.16E-04 | | ESCA | 5.23E-02 | 1.94E-02 |
| | IGWO | 4.06E-14 | 5.14E-14 | | **IGWO** | **0.00E+00** | **0.00E+00** |
| | SSA | 1.14E+01 | 3.90E+01 | | SSA | 2.15E-03 | 3.12E-03 |
| | **HDSSA** | **7.24E-08** | **2.42E-08** | | **HDSSA** | **3.07E-04** | **2.81E-05** |
| $F_{12}$ | TACPSO | 1.83E+00 | 3.61E-01 | $F_{15}$ | TACPSO | **3.07E-04** | 6.19E-05 |
| | ESCA | 3.29E-01 | 5.19E-01 | | ESCA | 6.02E-03 | 6.87E-03 |
| | IGWO | 5.79E-06 | 5.24E-06 | | IGWO | **3.07E-04** | 5.35E-05 |

## IV. EXPERIMENTAL RESULTS

### A. EXPERIMENTS AND PARAMETERIZATION

To verify the effectiveness of the improved HDSSA, it was evaluated on various benchmark problems from the literature. Problem set 1 is a partial arithmetic example proposed by Fattahi et al. [29], which considers 2 to 12 different jobs and 4 to 48 operations processed in flexible job shop environments with 2 to 8 machines, named SFJS01-10 and MFJS01-10, respectively. Problem set 2 was proposed by Deroussi and Norre [30]. These instances make use of the FJSP1–10 job sets and four layouts that were initially proposed by Bilge and Ulusoy [2] for JSPT.

The difference from the original is that all machines are duplicated, so that each operation can be executed on two machines with the same processing time. Problem set 3 is proposed by Kumar et al. [31] Job sets in problem set provided by Bilge and Ulusoy have been modified by Kumar et al. by including alternative machines where each operation can be done by three alternative machines. The instances can be found at https://fastmanufacturingproject.wordpress.com. The test results are compared with previously published literature.

The simulation environment is the Win 10 system, AMD Ryzen 7-4800H CPU @ 2.9 GHz, RAM 16.0 GB, and programmed by MATLAB R2022b. The parameter settings have a significant impact on the performance of algorithm, the parameter selection criteria are measured by the quality of the solution and the running time of the algorithm. We conduct orthogonal experiments by Taguchi's method [32] to benchmark the mean deviation of the solution and the mean running time thus determining the relevant parameters. The proposed HDSSA algorithm consists of four main parameters such as population size, number of iterations, variance probability and crossover probability. The parameters of $Pop$, $T_{max}$, $P_m$ and $P_c$ are tried to be selected from $\{50, 100, 200\}$, $\{100, 200, 300\}$, $\{0.6, 0.7, 0.8\}$ and $\{0.5, 0.6, 0.7\}$ respectively. Based on the results of several calculations, the algorithm had the best performance when the parameter $Pop$ was set to 100, the number of iterations was set to 200, and the variation and crossover coefficients were set to 0.7 and 0.5, respectively, while other algorithmic parameter settings were in accordance with Ref.

### B. EXPERIMENTAL ANALYSIS OF BENCHMARK FUNCTIONS

To verify the performance of the HDSSA algorithm, the algorithm optimization search test is performed by 10 benchmark functions. In this section, the HDSSA, SSA, TACPSO(Time varying Acceleration Coefficients Particle Swarm Optimization) [33], ESCA(Exponentially Improved Exponential Sine Cosine Algorithm) [34], and IGWO (Improved Grey Wolf Optimizer) [35] are compared. To ensure the fairness and accuracy of the experiments, the same population size (50) and maximum number of iterations (500) were set in the comparison experiments. Table 2 gives the mean and standard deviation of the algorithms after 30 experiments, with the best results of the tests highlighted in bold. The fitness values of the algorithms are expressed as the mean of the results of, and the stability of the algorithms

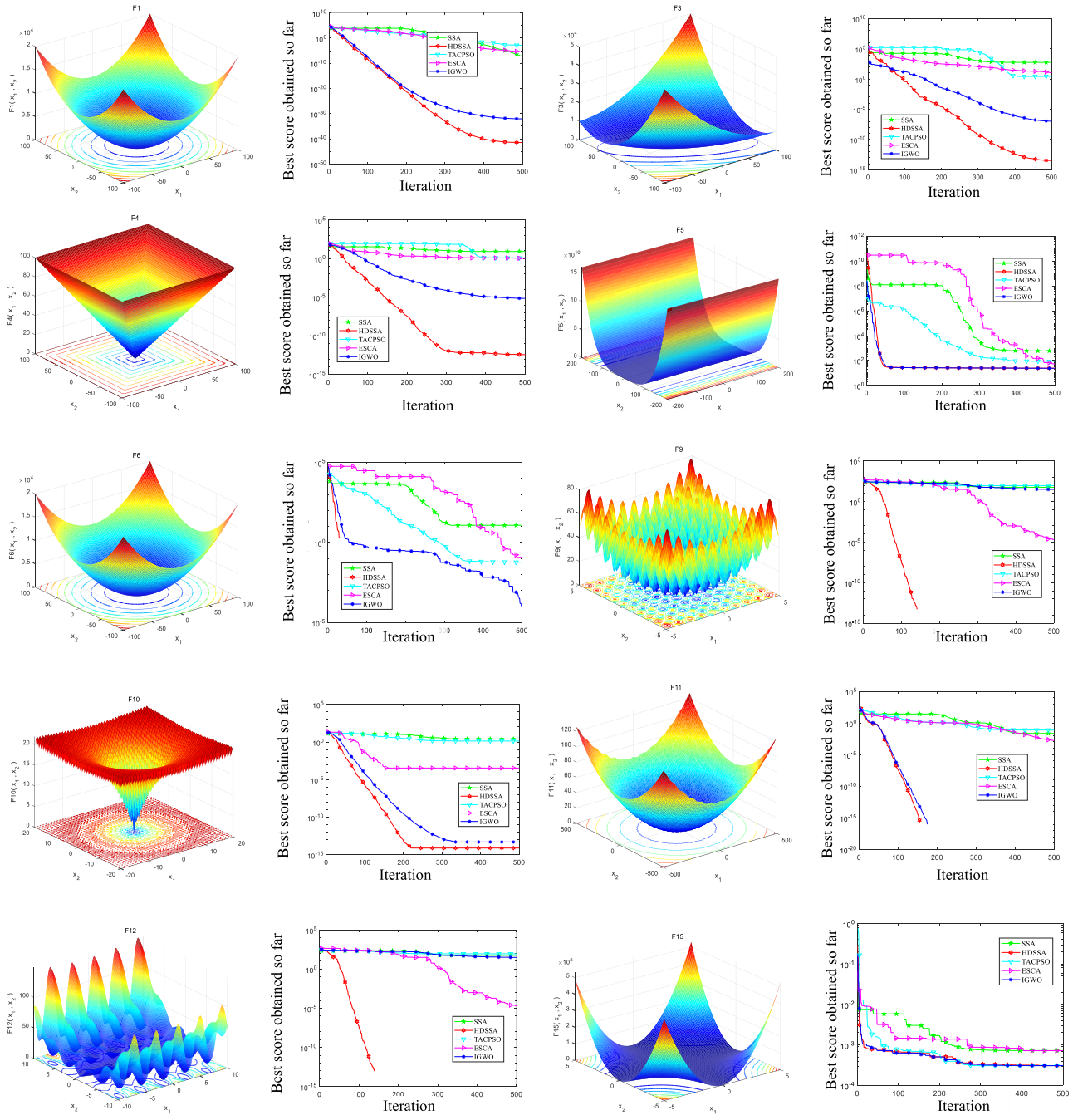



**FIGURE 8.** Convergence curves of selected functions.

is expressed as the standard deviation of the results of the 30 experiments.

For the five uni-modal functions (F1, F3, F4, F5, and F6), the HDSSA exhibits notable benefits in terms of mean and standard deviation, as shown in Table 10. For the multi-modal function functions F9, F10, and F12, HDSSA has the best performance among all the algorithms compared. For function F11, HDSSA algorithm converges better than IGWO in terms of convergence speed.

Additionally, HDSSA performs similarly to the TACPSO and IGWO algorithms in F15 mean value; however, HDSSA has the most stable solution performance in terms of standard deviation.

The convergence accuracy curves for each benchmark test function solved by the algorithm are plotted in Figure 8. Due to the required exploration phase, an algorithm's initial phase converges slowly. However, later in the iteration, it is important to fully utilize the algorithm's development phase, which

**TABLE 3. Comparison results of HSSA and RSSA.**

| Problem set 1 | *M-J-O-A* | Initial solution | | | Best solution | | |
|---|---|---|---|---|---|---|---|
| | | RSSA | HSSA | PEI | RSSA | HSSA | PEI |
| SFJST06 | 3-3-9-1.7 | 368.00 | **347.93** | 5.45 | 328.00 | **324.13** | 1.18 |
| SFJST07 | 5-3-9-2 | 444.57 | **423.30** | 4.78 | 409.20 | **409.00** | 0.05 |
| SFJST08 | 4-3-9-2 | 324.17 | **291.07** | 10.21 | 283.47 | **281.23** | 0.79 |
| SFJST09 | 3-3-9-2 | 263.80 | **227.90** | 13.61 | 227.20 | **224.10** | 1.36 |
| SFJST10 | 5-4-12-1.7 | 657.47 | **617.70** | 6.05 | 549.00 | **531.00** | 3.28 |
| MFJST01 | 2-2-4-2 | 725.37 | **671.50** | 7.43 | 519.23 | **508.10** | 2.14 |
| MFJST02 | 2-2-4-1.5 | 704.47 | **628.77** | 10.75 | 516.07 | **496.65** | 3.76 |
| MFJST03 | 2-3-6-1.7 | 836.13 | **739.13** | 11.60 | 539.77 | **525.80** | 2.59 |
| MFJST04 | 2-3-6-1.7 | 1032.63 | **901.83** | 12.67 | 659.73 | **629.00** | 4.66 |
| MFJST05 | 2-3-6-2 | 994.67 | **860.27** | 13.51 | 600.40 | **584.20** | 2.70 |
| Improve efficiency | | | 64.188 | | | 11.886 | |

results in a faster convergence to the optimal solution in the later stages compared to the exploration phase, allowing the algorithm to converge to the optimal solution more quickly in the later stages.

Though it performs some functions slightly less well than others, the HDSSA algorithm's convergence curve always converges to the optimal value at the end of the iteration. Furthermore, the proposed HDSSA algorithm's convergence curves exhibit a noteworthy downward trend across most test functions, suggesting that the enhanced HDSSA algorithm outperforms all other compared algorithms in terms of convergence speed. Therefore, the proposed HDSSA algorithm can produce better global optimal solutions through iterations, as evidenced by the convergence curve of the improved HDSSA algorithm being closer to the optimal solution on ten classical test functions.

### C. COMPARISON OF INITIALIZATION METHODS

To verify the viability and effectiveness of the proposed heuristic population initialization method, this section will conduct a comparison experiment between the heuristic population initialization method and the completely random initialization method based on the standard SSA algorithm. The SSA algorithm with completely random initialization is named RSSA, and the SSA algorithm with heuristic initialization is named HSSA.

The results of the comparison between the heuristic initialization method and the completely random initialization method are shown in Table 3. To evaluate the impact of the proposed heuristic initialization method on the initial solution quality, we recorded the completion times of the initial and optimal solutions for 2 groups of populations. Meanwhile, the percentage of efficiency improvement (PEI) is set as a comparison indicator, and the percentage of efficiency improvement was calculated by $\frac{c_{heuristic}-c_{random}}{c_{random}}\%$.

As shown in Table 3, the quality of the initial solutions obtained by heuristic initialization are all better than those obtained by random initialization. The optimal solutions obtained by the heuristic population through the SSA algorithm are all better than the optimal solutions obtained by the random population. The completion time of the initial solution of the heuristic population is 64.188 s less than that of the random population, and the completion time of the optimal solution of the heuristic population is 11.886 s less than that of the random population through HDSSA.

For instances SFJST07, HSAA can obtain almost the same optimal solution as RSSA in Figure 9. Comparing the mean value of solutions for 30 runs of the algorithm, RSSA gets an initial solution of 444.57 and HSSA can obtain its initial solution of 423.30. HSSA improves the quality of the initial solution by 4.78% compared to RSSA. In the MFJST01-MFJST05 instances, as the solution space of the instances increases, the enhancement efficiency of HSSA relative to RSSA in obtaining the initial and optimal solutions increases gradually. Therefore, the proposed heuristic initialization is effective and feasible to enhance the quality of the algorithm solution and accelerate the convergence speed of the SSA algorithm.

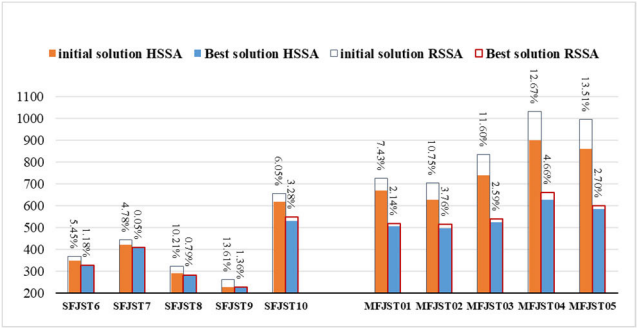



**FIGURE 9. The differences between random initialization and heuristic initialization methods.**

### D. ANALYSIS OF IMPROVEMENT STRATEGIES

To illustrate the effectiveness of the improved strategies for the standard SSA algorithm, this section will conduct

**TABLE 4.** Comparison results of SSA algorithms with different improvement strategies (SFJST01- SFJST10).

| Problem set 1 | *M-J-O-A* | LAHC-1 | | LAHC-2 | | SSA | | | ASSA | | | GSSA | | | VSSA | | | HDSSA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $C_{\max}^{\min}$ | CPU/s | $C_{\max}^{\min}$ | CPU/s | $C_{max}^{min}$ | $C_{\max}^{avg}$ | CPU/s | $C_{max}^{min}$ | $C_{\max}^{avg}$ | CPU/s | $C_{max}^{min}$ | $C_{\max}^{avg}$ | CPU/s | $C_{max}^{min}$ | $C_{\max}^{avg}$ | CPU/s | $C_{max}^{min}$ | $C_{\max}^{avg}$ | CPU/s |
| SFJST01 | 2-2-4-2 | 70 | 1.2 | 70 | 1.2 | 70 | 70.0 | 1.333 | 70 | 70.0 | 1.207 | 70 | 70.0 | 1.344 | 70 | 70.0 | 1.300 | 70 | 70.0 | 1.420 |
| SFJST02 | 2-2-4-1.5 | 111 | 1.3 | 111 | 1.4 | 111 | 111.0 | 1.237 | 111 | 111.0 | 1.210 | 111 | 111.0 | 1.425 | 111 | 111.0 | 1.316 | 111 | 111.0 | 1.295 |
| SFJST03 | 2-3-6-1.7 | 223 | 1.5 | 223 | 1.5 | 227 | 228.1 | 1.416 | 227 | 227.7 | 1.405 | 223 | 223.0 | 1.471 | 223 | 223.0 | 1.574 | 223 | 223.0 | 1.433 |
| SFJST04 | 2-3-6-1.7 | 359 | 1.5 | 359 | 1.4 | 367 | 367.0 | 1.456 | 363 | 363.3 | 1.442 | 359 | 359.0 | 1.475 | 359 | 359.0 | 1.465 | 359 | 359.0 | 1.562 |
| SFJST05 | 2-3-6-2 | 131 | 1.3 | 131 | 1.4 | 133 | 133.0 | 1.404 | 133 | 133.0 | 1.462 | 131 | 131.0 | 1.564 | 131 | 131.0 | 1.525 | 131 | 131.0 | 1.590 |
| SFJST06 | 3-3-9-1.7 | 324 | 1.9 | 324 | 1.9 | 328 | 328.5 | 1.638 | 324 | 328.6 | 1.605 | 324 | 324.1 | 1.671 | 324 | 327.4 | 1.898 | 324 | 324.1 | 1.675 |
| SFJST07 | 5-3-9-2 | 409 | 1.9 | 409 | 1.8 | 409 | 409.9 | 1.642 | 409 | 409.2 | 1.718 | 409 | 409.0 | 1.635 | 409 | 409.0 | 1.650 | 409 | 409.0 | 1.838 |
| SFJST08 | 4-3-9-2 | 269 | 5.0 | 269 | 2.0 | 280 | 282.1 | 1.654 | 280 | 281.7 | 1.640 | 280 | 281.5 | 1.641 | 280 | 281.2 | 1.759 | 270 | 271.6 | 1.864 |
| SFJST09 | 3-3-9-2 | 220 | 2.4 | 220 | 1.9 | 224 | 224.1 | 1.641 | 224 | 224.0 | 1.705 | 224 | 224.0 | 1.837 | 220 | 220.0 | 1.605 | 220 | 220.0 | 1.828 |
| SFJST10 | 5-4-12-1.7 | 531 | 4.6 | 531 | 2.4 | 531 | 532.3 | 1.879 | 549 | 531.9 | 1.830 | 531 | 531.1 | 1.823 | 531 | 531.1 | 1.890 | 531 | 531.0 | 2.143 |

**TABLE 5.** Comparison results of SSA algorithms with different improvement strategies (MFJST01-MFJST10).

| Problem set 1 | *M-J-O-A* | LAHC-1 | | LAHC-2 | | SSA | | | ASSA | | | GSSA | | | VSSA | | | HDSSA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $C_{max}^{min}$ | CPU/s | $C_{max}^{min}$ | CPU/s | $C_{max}^{min}$ | $C_{\max}^{avg}$ | CPU/s | $C_{max}^{min}$ | $C_{\max}^{avg}$ | CPU/s | $C_{max}^{min}$ | $C_{\max}^{avg}$ | CPU/s | $C_{max}^{min}$ | $C_{\max}^{avg}$ | CPU/s | $C_{max}^{min}$ | $C_{\max}^{avg}$ | CPU/s |
| MFJST01 | 6-5-15-2.2 | 485 | 9.9 | 485 | 2.7 | 510 | 533.1 | 2.019 | 511 | 540.6 | 1.992 | 502 | 524.9 | 2.338 | 495 | 508.1 | 2.486 | 493 | 502.4 | 2.918 |
| MFJST02 | 7-5-15-2.6 | 463 | 12.5 | 468 | 2.5 | 494 | 517.8 | 1.975 | 487 | 533.2 | 2.056 | 487 | 533.2 | 2.127 | 487 | 501.6 | 2.185 | 483 | 492.8 | 2.428 |
| MFJST03 | 7-6-18-2.7 | 482 | 30.9 | 482 | 2.8 | 528 | 538.0 | 2.146 | 524 | 564.9 | 2.208 | 511 | 540.6 | 2.247 | 500 | 537.1 | 2.296 | 493 | 516.3 | 2.768 |
| MFJST04 | 7-7-21-2.7 | 576 | 45.5 | 576 | 3.4 | 631 | 660.9 | 2.412 | 630 | 702.7 | 2.341 | 596 | 657.7 | 3.029 | 603 | 629.0 | 2.285 | 596 | 626.6 | 3.444 |
| MFJST05 | 7-7-21-2.6 | 532 | 47.3 | 532 | 3.5 | 582 | 601.4 | 2.562 | 587 | 654.4 | 2.362 | 583 | 619.0 | 2.499 | 565 | 584.2 | 3.093 | 550 | 587.6 | 3.365 |
| MFJST06 | 7-8-24-2.6 | 652 | 37.2 | 652 | 4.0 | 699 | 754.2 | 2.482 | 688 | 778.9 | 2.495 | 691 | 748.5 | 3.251 | 676 | 746.0 | 3.696 | 673 | 703.2 | 3.409 |
| MFJST07 | 7-8-32-2.4 | 898 | 109.4 | 907 | 5.8 | 1024 | 1071.9 | 2.936 | 1068 | 1167.2 | 2.727 | 998 | 1072.3 | 2.796 | 946 | 1063.2 | 4.359 | 935 | 987.8 | 4.637 |
| MFJST08 | 8-9-36-2.4 | 900 | 149.1 | 918 | 6.7 | 1060 | 1116.6 | 3.147 | 1054 | 1195.3 | 3.064 | 1031 | 1099.7 | 3.877 | 1028 | 1098.0 | 4.812 | 958 | 1040.2 | 4.837 |
| MFJST09 | 8-11-44-2.3 | 1120 | 202.1 | 1181 | 8.6 | 1367 | 1494.9 | 3.694 | 1290 | 1375.4 | 3.972 | 1216 | 1369.1 | 4.223 | 1242 | 1314.9 | 5.010 | 1197 | 1269.7 | 5.744 |
| MFJST10 | 8-12-48-2.3 | 1238 | 232.1 | 1310 | 9.5 | 1577 | 1685.7 | 3.794 | 1525 | 1742.4 | 3.753 | 1412 | 1480.7 | 4.791 | 1396 | 1560.2 | 5.628 | 1394 | 1536.9 | 6.209 |

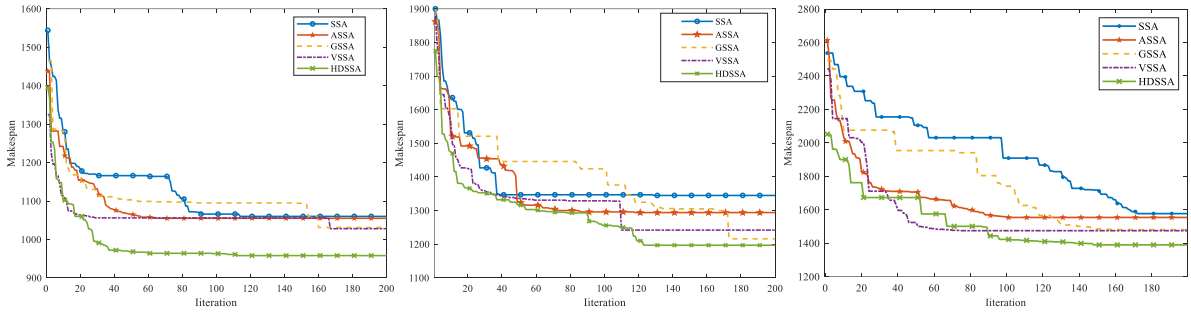
comparative experiments on HDSSA, SSA, and SSA variants. The slap swarm algorithm with double step size and inertia weighting strategy is named ASSA, the slap swarm algorithm with differential variational operator is named GSSA, the slap swarm algorithm with variable domain search strategy is named VSSA, and the synthesized and improved slap swarm algorithm is named HDSSA.

Problem set 1 consisting of twenty small and medium sized problems was selected for comparison experiment. The proposed SSA variant algorithms with different improvement strategies, HDSSA and the LACH (Late Acceptance Hill Climbing Algorithm) [8] are compared. The comparison results for the small-sized problems SFJST01-SFJST10 are shown in Table 4, and the comparison results for the medium-sized problems MFJST01-MFJST10 are shown in Table 5. The $C_{\max}^{\min}$ represents the optimal value obtained after 20 runs of each algorithm, and the $C_{\max}^{avg}$ represents the average value obtained after 20 runs of the algorithm. The CPU represents the running time of algorithms.

In the comparison results in Tables 4 and 5, the proposed HDSSA algorithm achieves better optimal solutions in both small-scale and medium-scale arithmetic cases. In the SFJST01-10 and MFJST01–10 instances, the optimal solutions of HDSSA relative to the base SSA algorithm are all significantly improved. Compared to the SSA algorithm, the ASSA algorithm has the same optimal solution as the SSA for the 10 algorithms in Table 4. With the increase in the problem size of the examples in Table 5, the optimal solution of the ASSA algorithm is better than that of the SSA algorithm. This indicates that the dual adaptive step-size control weighting strategy can balance the global and local search ranges of the standard SSA algorithm. However, the relatively poor solution results of the ASSA algorithm indicate that the algorithm may fall into a local optimum in the later stages of the search.

Comparing the optimization results of GSSA and SSA algorithms, the optimal solutions of the GSSA algorithm are better than the SSA algorithm in 4 instances in Table 4, and the optimal solutions of all 10 instances in Table 5 are better than the SSA algorithm. This indicates that the perturbation mechanism based on variation and crossover can expand the scope of global search and make the algorithm's possibility of discovering a better solution larger.

Comparing the optimization results of the VSSA algorithm and the SSA algorithm, the optimal solution and average




(a) MFJST08 (b) MFJST09 (c) MFJST10

**FIGURE 10.** Convergence Curve of the algorithm using different improved strategy.

**TABLE 6.** Comparison results for the instances in Problem Set 2.

| Problem set 2 | *M-J-O-A* | TS [36] | | MSB [37] | | GAME [18] | | HDPSO [20] | | | HDSSA | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Best | Dev/% | Best | Dev/% | Best | Dev/% | Best | Mean | Dev/% | Best | Mean | CPU/s |
| FJSP1 | 8-7-19-2 | 160 | 8.11 | 156 | 5.41 | **144** | -2.70 | 148 | 154.4 | 0.00 | 148 | 152.5 | 2.229 |
| FJSP2 | 8-6-15-2 | 128 | 4.92 | 124 | 1.64 | **118** | -3.28 | 118 | 130.4 | -3.28 | 122 | 129.7 | 2.034 |
| FJSP3 | 8-6-16-2 | 162 | 30.65 | 140 | 12.90 | 124 | 0.00 | 130 | 135.4 | 4.84 | **124** | 127.4 | 2.464 |
| FJSP4 | 8-5-19-2 | 126 | 6.78 | 132 | 11.86 | 124 | 5.08 | 126 | 133.4 | 6.78 | **118** | 124.1 | 2.219 |
| FJSP5 | 8-5-13-2 | 100 | 6.38 | 96 | 2.13 | 94 | 0.00 | 94 | 95.2 | 0.00 | **94** | 94.8 | 1.914 |
| FJSP6 | 8-6-18-2 | 152 | 1.33 | 148 | -1.33 | **144** | -4.00 | 150 | 154.0 | 0.00 | 150 | 159.9 | 2.276 |
| FJSP7 | 8-8-19-2 | 132 | 15.79 | 132 | 15.79 | 120 | 5.26 | 126 | 132.0 | 10.53 | **114** | 122.5 | 2.244 |
| FJSP8 | 8-6-20-2 | 188 | 5.62 | 191 | 7.30 | 179 | 0.56 | 186 | 190.3 | 4.49 | **178** | 186.4 | 2.922 |
| FJSP9 | 8-5-17-2 | 162 | 5.19 | 154 | 0.00 | **146** | -5.19 | 152 | 154.8 | -1.30 | 154 | 160.6 | 2.164 |
| FJSP10 | 8-6-21-2 | 186 | 0.00 | 192 | 3.23 | **182** | -2.15 | 190 | 196.8 | 2.15 | 186 | 197.9 | 2.531 |

value of the VSSA algorithm are smaller than those of the SSA algorithm in both sets of algorithms. This indicates that the introduction of the variable neighborhood search strategy can effectively prevent the SSA from falling into local optimality in the late iteration, which enhances the local search ability of the SSA.

In summary, the HDSSA has faster convergence speed in the early iteration compared with the SSA and ASSA. Compared with GSSA, it can search for a better solution in the late iteration, effectively avoiding falling into local optimality. Meanwhile, the HDSSA also reflects the feasibility of hybrid improvements. It not only balances the algorithm's global optimization and local search ability but also can obtain better solutions in a shorter iteration period. Therefore, the improvement strategy proposed in this paper for the basic SSA algorithm is effective. The convergence curves of the SSA algorithm adopting different improvement strategies for solving some instances are shown in Figure 10.

### *E. ALGORITHM COMPARATIVE EXPERIMENT*

To further validate the feasibility of the proposed HDSSA algorithm, this section compares it with the previously published papers by means of 10 FJSPT instances in Problem Set 2. This problem set is the FJSPT problem with transportation constraints. The proposed HDSSA algorithm is compared with the following algorithms/heuristics:

1) TS [36] (A tabu search procedure by Zhang et al.)
2) MSB [37] (A modified shifting bottleneck heuristic method)
3) GAME [18] (The IGA as an add-in for Microsoft Excel® spreadsheet-based solution)
4) HDPSO [20] (A hybrid discrete particle swarm optimization, HDPSO)

The comparison results of the maximum completion time and percentage deviation of the above algorithms with respect to the improved HDSSA are shown in Table 6. The percentage deviation was calculated by equation (30). Positive values of the percentage deviation indicate that the proposed HDSSA is better than the compared algorithms, while negative values indicate that the proposed method performs worse.

$$\text{Dev} = \frac{MC_{HDSSA} - MC_{\text{Comparison Algorithms}}}{MC_{HDSSA}} \tag{30}$$

where, the $MC_{HDSSA}$ represents the best result in terms of make span obtained by repeating the HDSSA algorithm several times and the $MC_{Compared\ algorithms}$ represents the make span of the solved instances of the compared algorithms.

From the statistical results in Figure 11, the HDSSA algorithm outperforms the TS algorithm in solving Problem Set 2 with optimal solutions for all 10 instances. Except for the FJSP6, the HDSSA algorithm outperforms the MSB algorithm in solving the optimization results of the remaining 9 instances. Comparing the HDSSA algorithm with the GAME method, the FJSP3 and FJSP5 instances were able to achieve the same results. Though five sets of instances were slightly lower than the GAME method, the optimal solutions

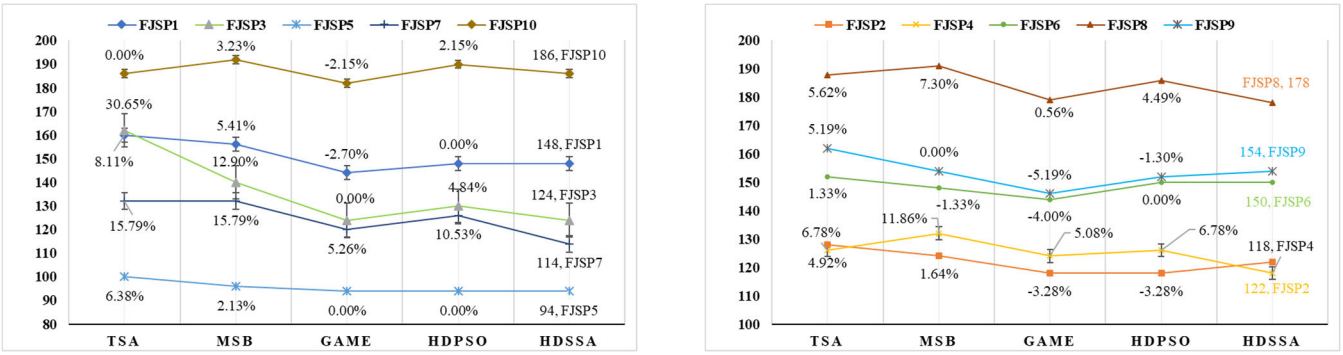



**FIGURE 11.** Comparative curves of experimental results for different algorithms.

of the remaining three instances outperformed the GAME method.

Comparing the results of the HDPSO algorithm and the HDSSA algorithm, HDSSA can obtain the same optimal solutions as HDPSO for the FJSP1, FJSP4, and FJSP5 instances. The optimization results for five sets of instances in Result Problem Set 2 show a more significant improvement with respect to the HDPSO algorithm. Although the completion times of FJSPT2 and FJSPT9 are slightly lower than those of the HDPSO algorithm, the mean values of the optimal solutions are better than those of the HDPSO.

According to the optimization results of Problem Set 1 and Problem set 2, the HDSSA can search for the optimal results of the problem better than the SSA and other comparative algorithms. The HDSSA has a good global search capability, which can avoid falling into the local optimal solution too early. For the flexible manufacturing shop machine and AGV integration scheduling problem, the HDSSA could find a shorter completion time efficiently. Meanwhile, it effectively proves that the improvement strategy proposed in this paper is feasible and effective. This indicates that the HDSSA used for the integration scheduling problem can find solutions that are more efficient and effective in terms of utilizing resources and optimizing the overall production process.

### F. INFLUENCE OF AGV NUMBER ON MAKESPAN

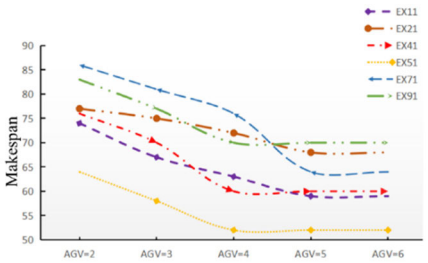
To study the influence of the number of AGVs in the workshop on the makespan of the system, HDSSA is used to solve the makespan when the number of vehicles A in problem set 3 is 2-6 respectively. The results are shown in Table 7.

From Table 7, the solution results of the HDSSA algorithm gradually decrease as the number of AGVs increases, but when the number of AGVs reaches 5, the decreasing trend of the maximum completion time tends to stabilize.

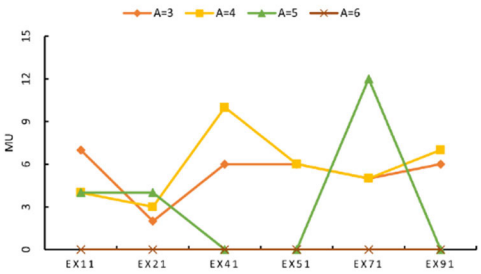
To explain the effect of AGV quantity change on scheduling efficiency, the marginal effect theory in economics is introduced to analyze the effect of AGV quantity on the completion time of the experimental case. Marginal utility (MU) is the utility of increasing or decreasing the revenue of a good or service for each new or reduced unit of the good or service. In this paper, goods are referred to as AGV,

**TABLE 7.** Experimental test results of different numbers of AGVs in Problem Set 3.

| Problem set 3 | A=2 | A=3 | A=4 | A=5 | A=6 |
|---|---|---|---|---|---|
| EX11 | 74 | 67 | 63 | 59 | 59 |
| EX21 | 77 | 75 | 72 | 68 | 68 |
| EX41 | 76 | 70 | 60 | 60 | 60 |
| EX51 | 64 | 58 | 52 | 52 | 52 |
| EX71 | 86 | 81 | 76 | 64 | 64 |
| EX91 | 83 | 77 | 70 | 70 | 70 |




(a)The Makespan VS. the number of AGV




(b) AGV marginal utility curve of each instance

**FIGURE 12.** The change diagram of completion time with the number of AGVs.

and benefits are referred to as makespan. The formula for evaluating the marginal effect of AGV is as follows:

$$MU = \frac{\Delta C_{\max}}{\Delta A} \tag{31}$$

where, the $\Delta C_{\max}$ is the difference in completion time, and the $\Delta A$ is the difference in the number of AGVs.

The graph of completion time with the number of AGVs in each case in Table 7 is shown in Figure 12(a). The results of Table 7 are transformed and calculated by equation (31), and the AGV marginal effect curve for each instance is plotted as shown in Figure 12(b).

Table 7 and Figure 12 show that with the increase in the number of AGVs in the workshop, the total completion time shows a decreasing pattern, but for every additional AGV, the marginal effect shows a decreasing pattern. When the number of AGVs increases to 6, the marginal effect decreases to 0. At this time, the completion time is no longer affected by the number of AGVs but is constrained by the machining process and machining time, so the increase in the number of AGVs cannot further improve the scheduling efficiency of the workshop.

## V. CONCLUSION AND FUTURE WORKS

Aiming at the dual resource scheduling problem of machines and AGVs in flexible job shops, this paper proposes an improved hybrid discrete slap swam algorithm (HDSSA). The main work is reflected in the following three aspects:

1) A mixed integer planning model is made with constraints on process resources, machine resources, and AGV resources for the flexible manufacturing shop integrated scheduling problem that takes AGV transportation time into account. The MIP model considers the loaded and unloaded states of the AGV and minimizes the maximum completion time as the optimization objective.
2) A series of improvement strategies for the standard SSA algorithm are proposed. A 3-layer coding method based on process, machine and AGV is designed to realize the continuous solution space discretization of coding. A heuristic initialization based on the earliest arrival of AGV, or earliest end of machine processing is proposed to improve the quality of the initial solution. A double-step control factor is designed to adjust the leader position updating method, and an inertia weighting strategy is introduced in the follower position movement. The differential variation idea is introduced into the SSA to enhance the algorithm's global search capability by variation operator and crossover operator. A local search strategy based on problem features is designed to integrate the variational domain search algorithm into the improved Slap Swarm algorithm to avoid the algorithm falling into the local optimum.
3) The HDSSA algorithm is compared with other intelligent algorithms and their optimization results through numerical experiments on the benchmark problem. The results show that the proposed improvement strategy for the standard SSA algorithm is feasible. In most cases, the HDSSA algorithm can effectively solve the AGV and machine integration scheduling problems in a flexible manufacturing shop.

In future research, we will change the machine-AGV integration scheduling model and constraints to consider things like AGVs' power, multiple loads, and paths that conflict with each other. At the same time, multi-objective optimization of the machine-AGV integrated scheduling problem in flexible manufacturing systems will be carried out in conjunction with green scheduling theory.

## REFERENCES

[1] B. Wu, Y. C. Ding, and A. Basri, "Research status of AGV and machine integrated scheduling," *Comput. Eng. Appl.*, vol. 59, pp. 1–12, Jan. 2023, doi: 10.3778/j.issn.1002-8331.2207-0427.

[2] Ü. Bilge and G. Ulusoy, "A time window approach to simultaneous scheduling of machines and material handling system in an FMS," *Operations Res.*, vol. 43, no. 6, pp. 1058–1070, Dec. 1995, doi: 10.1287/opre.43.6.1058.

[3] G. Ulusoy and Ü. Bilge, "Simultaneous scheduling of machines and automated guided vehicles," *Int. J. Prod. Res.*, vol. 31, no. 12, pp. 2857–2873, Dec. 1993, doi: 10.1080/00207549308956904.

[4] G. Ulusoy, F. Sivrikaya-Şerifoğlu, and Ü. Bilge, "A genetic algorithm approach to the simultaneous scheduling of machines and automated guided vehicles," *Comput. Operations Res.*, vol. 24, no. 4, pp. 335–351, Apr. 1997, doi: 10.1016/s0305-0548(96)00061-5.

[5] E. Liu, X. Yao, T. Tao, and H. Jin, "Improved flower pollinaton algorithm for job shop scheduling problems integrated with AGVs," *Comput. Integr. Manuf. Syst.*, vol. 25, no. 9, pp. 2219–2236, Sep. 2019, doi: 10.13196/j.cims.2019.09.010.

[6] G. Zhu, W. Jia, and D. Li, "Research on the hybrid flow-shop scheduling considering the joint of machine and AGV with renewable energy," *J. Beijing Univ. Aeronaut. Astronaut.*, to be published, doi: 10.13700/j.bh.1001-5965.2023.0021.

[7] A. Bekkar, G. Belalem, and B. Beldjilali, "Iterated greedy insertion approaches for the flexible job shop scheduling problem with transportation times constraint," *Int. J. Manuf. Res.*, vol. 14, no. 1, pp. 43–66, Dec. 2018, doi: 10.1504/IJMR.2019.096746.

[8] S. M. Homayouni and D. B. M. M. Fontes, "Production and transport scheduling in flexible job shop manufacuring systems," *J. Global Optim.*, vol. 79, pp. 463–502, Feb. 2021, doi: 10.1007/s10898-021-00992-6.

[9] Z. Wang and Y. Wu, "An improved differential evolution algorithm for simultaneous scheduling of machines and multi-load AGVs in an FMS," *Control Decis.*, vol. 12, pp. 1–9, Aug. 2023, doi: 10.13195/j.kzyjc.2023.0597.

[10] X. Y. Hu, X. F. Yao, P. Huang, and Z. R. Zheng, "Improved iterative local search algorithm for solving multi-AGV flexible job shop scheduling problem," *Comput. Integr. Manuf. Syst.*, vol. 28, no. 7, pp. 2198–2212, Jun. 2022, doi: 10.13196/j.cims.2022.07.025.

[11] Z. Liu, Q. Luo, L. Wang, H. Tang, and Y. Li, "The low-carbon scheduling optimization of integrated multispeed flexible manufacturing and multi-AGV transportation," *Processes*, vol. 10, no. 10, p. 1944, Sep. 2022, doi: 10.3390/pr10101944.

[12] A. Ham, "Transfer-robot task scheduling in job shop," *Int. J. Prod. Res.*, vol. 59, no. 3, pp. 813–823, Feb. 2021, doi: 10.1080/00207543.2019.1709671.

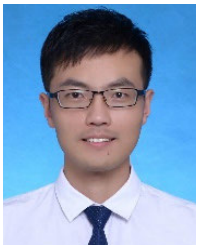
[13] R. Erol, C. Sahin, A. Baykasoglu, and V. Kaplanoglu, "A multi-agent based approach to dynamic scheduling of machines and automated guided vehicles in manufacturing systems," *Appl. Soft Comput.*, vol. 12, no. 6, pp. 1720–1732, Jun. 2012, doi: 10.1016/j.asoc.2012.02.001.

[14] M. Ishaq, M. Khan, and S. Kwon, "TC-net: A modest & lightweight emotion recognition system using temporal convolution network," *Comput. Syst. Sci. Eng.*, vol. 46, no. 3, pp. 3355–3369, 2023, doi: 10.32604/csse.2023.037373.

[15] M. Khan, W. Gueaieb, A. El Saddik, and S. Kwon, "MSER: Multimodal speech emotion recognition using cross-attention with deep fusion," *Expert Syst. Appl.*, vol. 245, Jul. 2024, Art. no. 122946, doi: 10.1016/j.eswa.2023.122946.

[16] A. H. Sun, Q. Lei, Y. C. Song, and Y. Yang, "Deep reinforcement learning for solving the joint scheduling problem of machines and AGVs in job shop," *Control Decis.*, vol. 39, no. 1, pp. 256–262, Jan. 2024, doi: 10.13195/j.kzyjc.2022.1821.

[17] L. Meng, W. Cheng, B. Zhang, W. Zou, W. Fang, and P. Duan, "An improved genetic algorithm for solving the multi-AGV flexible job shop scheduling problem," *Sensors*, vol. 23, no. 8, p. 3815, Apr. 2023, doi: 10.3390/s23083815.
[18] I. A. Chaudhry, A. F. Rafique, I. A.-Q. Elbadawi, M. Aichouni, M. Usman, M. Boujelbene, and A. Boudjemline, "Integrated scheduling of machines and automated guided vehicles (AGVs) in flexible job shop environment using genetic algorithms," *Int. J. Ind. Eng. Computations*, vol. 13, no. 3, pp. 343–362, 2022, doi: 10.5267/j.ijiec.2022.2.002.
[19] Y. Zou, Y. Song, Y. Wang, and X. Wang, "AGV and machine integrated scheduling method based on discrete whale optimization algorithm," *J. Chongqing Univ.*, vol. 45, no. 6, pp. 55–74, Jun. 2022, doi: 10.11835/j.issn.1000-582X.2021.03.
[20] K. Chen, L. Bi, and W. Wang, "Research on dual-resource integrated scheduling of AGVs and machines in flexible job shop," *J. Syst. Simul.*, vol. 34, no. 3, pp. 461–469, Mar. 2022, doi: 10.16182/j.issn1004731x.joss.20-0796.
[21] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997, doi: 10.1109/4235.585893.
[22] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017, doi: 10.1016/j.advengsoft.2017.07.002.
[23] W. Zhao, P. Guo, H. Wang, and K. Lei, "Improved slap swarm algorithm for scheduling of flexible job shop," *J. Intell. Syst.*, vol. 17, no. 2, pp. 376–386, Oct. 2022, doi: 10.11992/tis.202103036.
[24] H. Y. Niu, W. M. Wu, T. Q. Zhang, W. Shen, and T. Zhang, "Adaptive salp swarm algorithm for solving flexible job shop scheduling problem with transportation time," *J. Zhejiang Univ. Eng. Sci.*, vol. 57, no. 7, pp. 1267–1277, Jul. 2023, doi: 10.3785/j.issn.1008-973X.2023.07.001.
[25] J. C. Bean, "Genetic algorithms and random keys for sequencing and optimization," *ORSA J. Comput.*, vol. 6, no. 2, pp. 154–160, May 1994, doi: 10.1287/ijoc.6.2.154.
[26] H. Ding, X. Cao, Z. Wang, G. Dhiman, P. Hou, J. Wang, A. Li, and X. Hu, "Velocity clamping-assisted adaptive salp swarm algorithm: Balance analysis and case studies," *Math. Biosciences Eng.*, vol. 19, no. 8, pp. 7756–7804, 2022, doi: 10.3934/mbe.2022364.
[27] K. F. Geng and C. M. Ye, "Joint scheduling of machines and AGVs in green hybrid flow shop with missing operation," *Control Decis.*, vol. 37, no. 10, pp. 2723–2732, Aug. 2022, doi: 10.13195/j.kzyjc.2021.0318.
[28] C. Z. He, Y. C. Song, Q. Lei, X. F. Lv, S. X. Liu, and J. Chen, "Integrated scheduling of multiple AGVs and machines in flexible job shops," *China Mech. Eng.*, vol. 30, no. 4, pp. 438–447, Feb. 2019, doi: 10.3969/j.issn.1004-132X.2019.04.009.
[29] P. Fattahi, M. Saidi Mehrabad, and F. Jolai, "Mathematical modeling and heuristic approaches to flexible job shop scheduling problems," *J. Intell. Manuf.*, vol. 18, no. 3, pp. 331–342, Jul. 2007, doi: 10.1007/s10845-007-0026-8.
[30] L. Deroussi and S. Norre, "Simultaneous scheduling of machines and vehicles for the flexible job shop problem," in *Proc. Int. Conf. Metah. Nat. Ins. Comp.*, Djerba Island, Tunisia, 2010, pp. 1–2.
[31] M. V. S. Kumar, R. Janardhana, and C. S. P. Rao, "Simultaneous scheduling of machines and vehicles in an FMS environment with alternative routing," *Int. J. Adv. Manuf. Technol.*, vol. 53, nos. 1–4, pp. 339–351, Mar. 2011, doi: 10.1007/s00170-010-2820-2.
[32] C. T. Gray, "Introduction to quality engineering: Designing quality into products and processes, G. Taguchi, Asian productivity organization, 1986. Number of pages: 191. price: $29 (U.K.)," *Qual. Rel. Eng. Int.*, vol. 4, no. 2, p. 198, Apr. 1988, doi: 10.1002/qre.4680040216.
[33] G. Q. Bao and K. F. Mao, "Particle swarm optimization algorithm with asymmetric time varying acceleration coefficients," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Guilin, China, Dec. 2009, pp. 2134–2139, doi: 10.1109/ROBIO.2009.5420504.
[34] N. Li and L. Wang, "Bare-bones based sine cosine algorithm for global optimization," *J. Comput. Sci.*, vol. 47, Nov. 2020, Art. no. 101219, doi: 10.1016/j.jocs.2020.101219.
[35] M. H. Nadimi-Shahraki, S. Taghian, and S. Mirjalili, "An improved grey wolf optimizer for solving engineering problems," *Expert Syst. Appl.*, vol. 166, Mar. 2021, Art. no. 113917, doi: 10.1016/j.eswa.2020.113917.
[36] Y. Zheng, Y. Xiao, and Y. Seo, "A Tabu search algorithm for simultaneous machine/AGV scheduling problem," *Int. J. Prod. Res.*, vol. 52, no. 19, pp. 5748–5763, Oct. 2014, doi: 10.1080/00207543.2014.910628.
[37] Q. Zhang, H. Manier, and M.-A. Manier, "A modified shifting bottleneck heuristic and disjunctive graph for job shop scheduling problems with transportation constraints," *Int. J. Prod. Res.*, vol. 52, no. 4, pp. 985–1002, Feb. 2014, doi: 10.1080/00207543.2013.828164.



**TIANRUI ZHANG** received the Ph.D. degree in engineering from Northeastern University, Shenyang, Liaoning, China, in 2014.

He is currently an Associate Professor and the Master's Supervisor with Shenyang University. His research interests include optimization of manufacturing system engineering and production logistics systems management.

**GUANGHAO ZHU** received the B.S. degree in industrial engineering from Shenyang University, Shenyang, Liaoning, China, in 2022, where he is currently pursuing the M.S. degree in industrial engineering and management. His research interests include intelligent optimization algorithm and integrated scheduling of AGVs and machines.