Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

An improved genetic algorithm for multi-AGV dispatching problem with unloading setup time in a matrix manufacturing workshop

Yuan-Zhuang Li[a], Jia-Zhen Zou[a], Yang-Li Jia[a*], Lei-Lei Meng[a] and Wen-Qiang Zou[a*]

[a]*School of Computer Science, Liaocheng University, Liaocheng 252000, P. R. China*

C H R O N I C L E

Article history:
Received March 1 2023
Received in Revised Format May 12 2023
Accepted July 5 2023
Available online July, 7 2023

Keywords:
Automated guided vehicle
Dispatching
Genetic algorithm
Setup time
Matrix manufacturing workshop

A B S T R A C T

This paper investigates a novel problem concerning material delivery in a matrix manufacturing workshop, specifically the multi-automated guided vehicle (AGV) dispatching problem with unloading setup time (MAGVDUST). The objective of the problem is to minimize transportation costs, including travel costs, time penalty costs, AGV costs, and unloading setup time costs. To solve the MAGVDUST, this paper builds a mixed-integer linear programming model and proposes an improved genetic algorithm (IGA). In the IGA, an improved nearest-neighbor-based heuristic is proposed to generate a high-quality initial solution. Several advanced technologies are developed to balance local exploitation and global exploration of the algorithm, including an optimal solution preservation strategy in the selection process, two well-designed crossovers in the crossover process, and a mutation based on Partially Mapped Crossover strategy in the mutation process. In conclusion, the proposed algorithm has been thoroughly evaluated on 110 instances from an actual electronic factory and has demonstrated its superior performance compared to state-of-the-art algorithms in the existing literature.

1. Introduction

The rise of the smart industry has brought an increasing interest in AGVs due to their flexible, automated, and intelligent characteristics (Lu et al., 2017). However, AGV dispatching has brought many challenges and troubles to the academia and industrial communities (Micieta et al., 2018). AGV dispatching refers to the process of allocating tasks to AGVs and determining the order in which the tasks are to be executed by each AGV. Effective AGV dispatching can improve production efficiency, reduce costs, and enhance enterprise competitiveness (Liu et al., 2022; Ng et al., 2009; Song, 2021; Wang et al., 2015; Hao et al., 2020; Yao et al., 2020). Especially with the expansion of scale, the AGV dispatching problem has become more complex and difficult. Therefore, the academic and industrial communities need more efficient strategies and dispatching algorithms to solve this crucial and meaningful problem. Current research in AGV dispatching primarily focuses on various scenarios or settings, including terminals, depots, manufacturing systems, underground parking lots and so on. As for automated container terminals, Jin and Zhang (2016) designed a dynamic multi-AGV dispatching model based on the genetic algorithm to minimize both the completion time and the standard deviation of handling time for quay cranes. Wang and Zeng (2022) conducted a study on the dispatching and routing problem of AGVs with multiple bidirectional paths, aiming to generate conflict-free routes. In the manufacturing system, Ren et al. (2012) designed a mathematical model that integrates double-buffered AGVs based on the dispatching problem of a flexible manufacturing system. And they proposed an improved genetic algorithm to rank the processing order of jobs and the movement of double-buffered AGVs. Niu et al. (2023) presented a multi-tasks chain dispatching algorithm to improve the heavy load ratio and reduce the makespan. Meanwhile, Liao et al. (2020) proposed a hybrid genetic algorithm-based AGV dispatching method and a time-space graph search algorithm-based

* Corresponding author
E-mail: zouwenqiang@lcu.edu.cn (W.-Q. Zou); jiayangli@lcu-cs.com (Y.-L. Jia)
ISSN 1923-2934 (Online) - ISSN 1923-2926 (Print)
2023 Growing Science Ltd.
doi: 10.5267/j.ijiec.2023.7.002

path optimization method to address the AGVDP in unmanned underground parking lots. The AGVDP is a challenging combinatorial optimization problem, and finding accurate and efficient solutions using traditional exact methods becomes difficult as the problem size increases (Xu et al., 2016). Scholars in existing research predominantly use heuristic or meta-heuristic algorithms to address the AGVDP. Hu et al. (2020) integrated the A* algorithm with the principle of time window to sequentially plan the path of each AGV in chronological order; Zou et al. (2021) presented an improves iterative greedy algorithm to address the multi-compartment AGVDP; Wang &Wu (2023) utilized an enhanced ant colony optimization-simulated annealing algorithm to tackle a multiload AGVs workshop dispatching problem with limited buffer capacity; Li et al. (2023) solved the AGVDP considering time and capacity constraints by employing a discrete invasive weed optimization algorithm.

A matrix manufacturing workshop is a production facility that utilizes a matrix structure to organize its equipment and workstations. It has become increasingly popular due to its versatility and adaptability (Zou et al., 2021). The workshop is divided into two primary segments: the production segment and the logistics segment. The logistics segment is responsible for the transportation of materials and products between workstations, which can be a complex process (Wu et al., 2023). Various considerations arise within the logistics link, including the coordination of multiple AGVs to prevent collisions and conflicts (Wang &Wu, 2023; Chen et al., 2022; Yuan et al., 2020), the development of a sound charging strategy to ensure uninterrupted AGV operation (Huang et al., 2018), timely resolution of abnormalities in AGV operations to maintain the continuity and stability of the logistics process (Sun et al., 2022), and more. At present, Meng et al. (2023) designed a population diversity checking method to solve the flexible job shop dispatching problem with a limited number of AGVs; Zou et al. (2022) investigated the energy-saving dispatching of AGVs with optimized energy consumption; Li et al. (2022) proposed a genetic algorithm to handle the dispatching problem of multiple AGV flexible manufacturing cells with charging constraints; Zou et al. (2021) conducted a study on AGVDP with pickup and delivery; Singh et al. (2022) proposed an adaptive large neighborhood search algorithm to address AGVDP with battery constraints; Eda et al. (2012) introduced a Petri net decomposition method to address the bi-objective optimization problem, which formulates the dispatching and conflict-free routing problem of AGVs as a Petri net bi-objective optimal firing sequence problem; Nishida et al. (2022) proposed a heuristic solution procedure to tackle the conflict-free route planning problem for AGVs with on-time delivery; Zou et al. (2023) conduct a study on the multi-AGV dispatching problem that incorporates charging and maintenance considerations. However, there has been a lack of research on $MAGVD_{UST}$ to date.

In the real manufacturing workshop, the computer numerically controlled (CNC) machines and other equipment in the workstations must undergo regular maintenance to ensure error-free production processes. For such workstations, safety checks, such as parking accuracy, robotic arm gripping goods, and buffer zone alarm, must be carried out before unloading materials. The time spent on safety checks is called unloading setup time. So, $MAGVD_{UST}$ is one of the specific MAGVDP, which involves the consideration of unloading setup time.

Obtaining optimal solutions is of utmost importance for a dispatching problem, especially when dealing with a new problem. Meta-heuristic algorithms are widely used in such cases to achieve optimal or near-optimal solutions (Meng et al., 2022). Genetic algorithm (GA) is a popular algorithm that simulates biological evolution (Jahanzaib et al., 2013). It searches for an optimal solution among many candidate solutions by using natural selection, crossover, and mutation. In this paper, IGA is proposed to solve $MAGVD_{UST}$. The main contributions can be summarized as follows:

(1) Formulate the $MAGVD_{UST}$ and establish a mixed-integer linear programming model.

(2) Propose an improved nearest-neighbor-based heuristic to generate a high-quality initial solution.

(3) Present an optimal solution preservation strategy, two well-designed crossovers, and a mutation based on Partially Mapped Crossover (PMX) strategy to balance local exploitation and global exploration of the algorithm.

The remainder of this paper is organized as follows. In Section 2, the $MAGVD_{UST}$ and its challenges are introduced in detail. Section 3 presents the proposed IGA and discusses its design and optimization strategies tailored for the $MAGVD_{UST}$. Section 4 is dedicated to presenting the experimental results and comparing the computational outcomes with well-known algorithms commonly employed for AGVDP. Lastly, in Section 5, a comprehensive summary of the paper is presented, highlighting the key findings and contributions. Additionally, this section provides valuable insights into potential future research directions in the field.

2. Problem description and formulation

2.1 Problem description

In a matrix layout of a general manufacturing workshop, workstations are arranged in a grid-like structure, as illustrated in Fig. 1. Each workstation is equipped with a material buffer and multiple CNC machines. The material buffer functions as a storage area for materials that are consistently consumed by the CNC machines. AGVs are responsible for transporting these materials to the respective workstations where the final products are produced. When the material level in the buffer reaches a pre-set minimum, the workstation will send an alarm signal for replenishment to the control system. Then the workstation is converted to a call workstation. Upon receiving instructions from the control system, the AGV leaves the depot with a full load of materials, follows the aisles to its assigned workstation to unload the materials and finally returns to the depot. Specifically, there are certain workstations that have just been maintained and are referred to as special workstations. When an AGV arrives at a special workstation, it will first undergo safety checks before unloading. It is important to note that this delivery process will incur transportation costs.

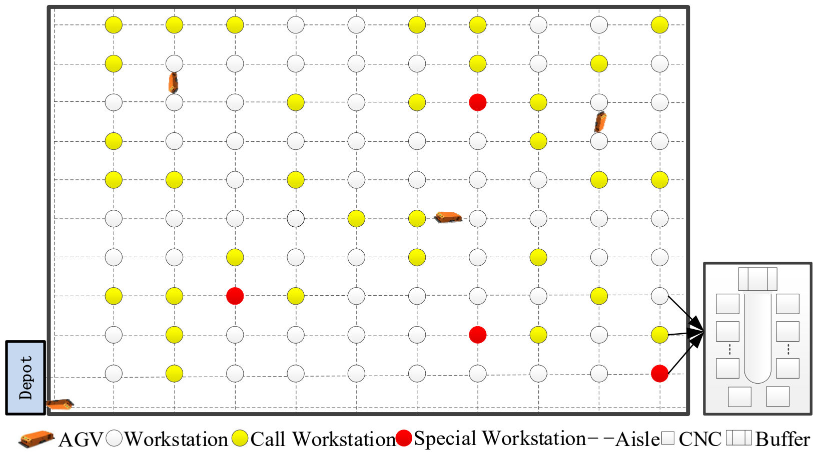

Fig. 1. The layout diagram of the matrix manufacturing workshop

Regular maintenance is crucial to ensure error-free production processes as workstations age. After maintenance, AGVs must undergo safety checks such as the AGV parking accuracy, the robotic arm gripping goods, or the buffer zone alarm system checks when they return to a workstation. The time for safety checking is defined as the unloading setup time, which is the amount of time required for the AGV to be ready to unload at the workstation. The unloading setup time of the AGV at a workstation may vary depending on the age of the workstation. If a safety check is required prior to dispatch, this information is known in advance.

For the purposes of the following description, the workstation that sends the alarm signal is referred to as a task, the moment the alarm signal is sent is recorded as the calling time, and the sequence of tasks that the AGVs is scheduled to serve is referred to as the AGV route. Additionally, it should be noted that each task can only be assigned to and served by a single AGV. and the control system requires timely delivery of the AGVs. Late deliveries will impact the production schedule, which is unacceptable. Conversely, early deliveries will result in a penalty and incur a time cost that varies based on the magnitude of the early delivery.

The aim of this study is to reduce the total transportation costs, including travel costs, time penalty costs, AGV costs, and unloading setup time costs. The analysis will be based on the manufacturing workshop's current situation.

The matrix manufacturing workshop involves multiple tasks, which would require a significant number of AGVs if each task had its own. However, it would result in congestion and inefficiencies (Yuan et al., 2021). To address this issue, a time-cycling strategy is suggested which divides the workshop production time into successive production cycles. In Fig. 2, tasks are created in each production cycle and assigned to AGVs for execution in the following cycle. The production cycle is comprised of two distinct phases - the calculation phase and the transport phase. In the calculation phase, the control system assigns all tasks to create a dispatching schedule. In the transport phase, AGVs are dispatched to transport materials to the assigned tasks based on the generated schedule. By implementing a time-cycling strategy, a single AGV can efficiently handle multiple tasks simultaneously, thus reducing the number of required AGVs and ultimately decreasing overall transportation costs.

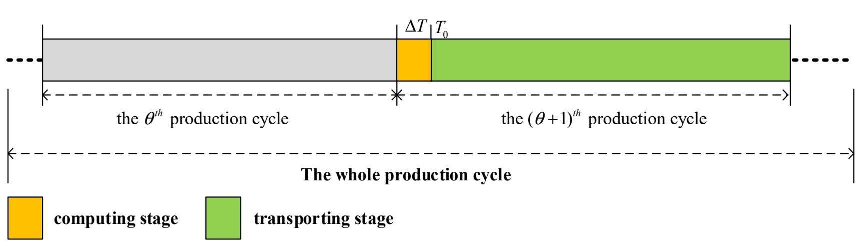

Fig. 2. The production cycles

2.2 Problem formulation

With the aforementioned information, a mathematical model is proposed utilizing the concept of $MAGVD_{UST}$. The model includes various decision variables and parameters, which are described as follows.

Parameters and constants:

i, j :Unique identification of the task

p_i : Position of the task i.

x_i :Size of the x coordinate of p_i .

y_i :Size of the y coordinate of p_i .

n : Overall quantity of tasks .

n' : Maximum number of tasks that an AGV can handle .

k : Current AGV (or AGV route) .

k' : Anticipated number of AGVs .

k'' : Number of AGVs available for dispatch .

v : Velocity of AGV .

Q : Capacity of AGV .

q_i : Material requirementof task i .

d_{ij} : Travel distance between tasks i and j .

t_{ij} :Travel time between tasks i and j .

C : Production cycle .

T_i^c : Call Time of tasks i .

T_i^l : Delivery time of task i .

T_i^u : Unloading setup time when AGV arrives at task i .

T_0 : Time when the AGV leaves the depot .

ΔT : Computation time for the computing stage in a production cycle .

t_u : Unloading time at each task .

t_m : Processing time per unit of production material .

S : Total amount of material in the buffer .

S_i^c : Inventory level of the material buffer at the call time .

g : Weight of each slice of production material .

c_t : Unit cost of traveling along an AGV route .

c_a : Cost of each AGV .

c_e : Penalty cost for earliness .

Decision Variables:

x_{ijk} : $x_{ijk} = 1$ if arc(i, j) is travelled by AGV and 0 otherwise.

T_i^r : Arrival time of task i .

Suppose $G = \{V, E\}$ is an undirected graph, where $V = \{1, 2, ..., n\}$ denotes the set of vertices and $E = \{(i, j) \mid i, j \in V, i \neq j\}$ denotes the set of edges connecting each pair of vertices. In this graph, vertex $i = 1, 2, ..., n$ represents a task, while vertex 0 denotes the depot. $K = \{1, 2, ..., m\}$ denotes the set of AGVs (or AGV routes). Each edge (i, j) represents the travel route of an AGV from tasks i (or the depot) to j . The travel distance and time along this route can be calculated using the following formula:

$$d_{ij} = |x_i - x_j| + |y_i - y_j| \tag{1}$$

$$t_{ij} = d_{ij} / v \tag{2}$$

Assuming an AGV departs from task i and arrives at task j , the time taken by the AGVs to reach task j is composed of the time taken to reach task i , the transport time, the unloading time, and the unloading setup time. This duration can be represented mathematically using formula (3):

$$T_j^r = T_i^r + T_i^u + t_{ij} + t_u, \forall i \in V, j \in V \setminus \{0\}, i \neq j_0 \tag{3}$$

The total distance of the AGV can be expressed as:

$$D_{ijk} = \sum_{k=1}^{m}\sum_{j=0}^{n}\sum_{i=0}^{n} x_{ijk} d_{ij} \tag{4}$$

The inventory in the material decreases as the CNC machine at the workstation consumes material for production. When task j is reached, the AGV unloads the material into the buffer. The amount of material unloaded by the AGV, which is the material requirement for task j, is calculated by formula (5):

$$q_j = \left[\left(S - S_j^c\right) + \left\lceil\left(T_j^r - T_j^c\right)\right\rceil / t_m\right] \bullet g, \forall j \in V \setminus \{0\} \tag{5}$$

Suppose that there are n tasks inside a production cycle, the minimum number of AGVs required is:

$$k' = \lceil n / n' \rceil, n' = 12 \tag{6}$$

Given the definitions mentioned above, the mixed-integer linear programming (MILP) formulation for $\text{MAGVD}_{\text{UST}}$ can be modeled as:

$$\min F(i,j,k) = c_t \sum_{k=1}^{m}\sum_{j=0}^{n}\sum_{i=0}^{n} x_{ijk} d_{ij} + c_a \sum_{k=1}^{m}\sum_{j=0}^{n}\sum_{i=0}^{n} x_{0jk} + c_e \sum_{k=1}^{m}\sum_{j=0}^{n}\sum_{i=0}^{n} x_{ijk}\left(T_j^l - T_j^r\right) + \sum_{k=1}^{m}\sum_{j=0}^{n}\sum_{i=0}^{n} x_{ijk} T_i^u \tag{7}$$

s.t:

$$\sum_{k=1}^{m}\sum_{i=0}^{n} x_{ijk} = 1, \forall j \in V \setminus \{0\} \tag{8}$$

$$\sum_{k=1}^{m}\sum_{j=0}^{n} x_{ijk} = 1, \forall j \in V \setminus \{0\} \tag{9}$$

$$\sum_{i=0}^{n} x_{ijk} - \sum_{i=0}^{n} x_{jik} = 0, \forall k \in K, j \in V \setminus \{0\} \tag{10}$$

$$\sum_{i=1}^{n} x_{i0k} = \sum_{j=1}^{n} x_{0ik} = 1, \forall k \in K \tag{11}$$

$$x_{ijk}\left(T_i^r + t_u + t_{ij} - T_j^r\right) = 0, \forall k \in K, j \in V \setminus \{0\}, i \in V \tag{12}$$

$$\sum_{i=1}^{n}\sum_{j=0}^{n} x_{ijk} \bullet q_j \le Q, \forall k \in K \tag{13}$$

$$T_i^c \sum_{k=1}^{m}\sum_{j=0}^{n} x_{ijk} \le T_i^r \le T_i^l \sum_{k=1}^{m}\sum_{j=0}^{n} x_{ijk}, \forall i \in V \setminus \{0\} \tag{14}$$

$$k' \le m \le k'', k'' = 6 \tag{15}$$

$$x_{ijk} \in \{0,1\}, \forall i, j \in V, \forall k \in K \tag{16}$$

$$x_{ijk} = 0, i, j \in V \text{ and } i = j \tag{17}$$

$$T_i^r = T_0, i = 0 \tag{18}$$

The model proposed in this paper aims to minimize transportation costs (constraint 7), including travel costs, time penalty costs, AGV costs, and unloading setup time costs. It is important to note that reducing one AGV route is always more advantageous than reducing other costs, even for large constant values. Additionally, constraints (8-10) require that each task be visited by an AGV at least once and must be left after the visit. In this subject paper, there are several constraints that have been imposed to ensure efficient and effective AGV operations. Constraint (11) specifies that every AGV route must commence and conclude at the depot. Constraint (12) defines a relationship between the arrival time of a task and its preceding task. In order to prevent overloading, Constraint (13) guarantees that the AGV's load does not exceed its maximum capacity. Constraint (15) maintains the total number of AGVs within an optimal range. Furthermore, Constraint (14) imposes a time constraint, while Constraints (16-18) impose restrictions on the decision variables.

3. The proposed improved genetic algorithm

In this paper, the IGA that effectively reduces transport costs is proposed to better solve the presented problem. The IGA is composed of five main parts: solution initialization, selection operation, crossover operation, mutation operation, and update operation. In the following sections, each part will be described in detail.

3.1 Solution representation

To simplify the representation of the solution for the $\text{MAGVD}_{\text{UST}}$, a straightforward approach has been adopted. Let there be

n tasks to be processed in the production workshop and m AGVs available to complete them. The solution vector has a length of $(m + n - 1)$ and consists of n different integers between 1 and n, representing the tasks. In the vector, the presence of zeros indicates the start of each AGV's route, effectively separating the routes of different AGVs. And the zero at the first position of the vector is usually omitted directly. For instance, if there are 3 AGVs and 9 tasks, and the first AGV is assigned to tasks 2, 5, and 8, the second AGV to tasks 1, 4, 6, and 9, and the third AGV to tasks 3 and 7, the solution would be represented as (2, 5, 8, 0, 1, 4, 6, 9, 0, 3, 7).

3.2 The improved nearest-neighbor-based heuristic

Zou et al., (2020) proposed a nearest-neighbor-heuristic (NNH) for task searching. The main idea behind NNH is to find the task that is closest to the current task based on the Manhattan distance. This task is then selected as the next task to be serviced. In this section, an improved NNH (INNH) is introduced, which utilizes the Chebyshev distance instead of the Manhattan distance. By considering distances in all directions between two tasks, the Chebyshev distance provides a more accurate measure in cases where the Manhattan distance may be over or underestimate distances. The calculation of the Chebyshev distance is as shown below:

$$D_{\mathrm{ij}} = \max(|x_j - x_i|, |y_j - y_i|) \tag{19}$$

where D_{ij} represents the Chebyshev distance between task i and task j, x and y represent the horizontal and vertical coordinates of the task.

In this algorithm, the following notations are used: Let $U = \{1,\ 2,\ ...,\ n\}$ denote the set of unassigned tasks, R represent the current AGV route, π represent the generated solution, and j represent the current task. The algorithm assigns each task by prioritizing those that satisfy the time and capacity constraints. If there are no tasks that meet the constraints, the current AGV route is included in the solution, and a new AGV route is initiated for the subsequent task. This process repeats until all tasks have been assigned, leading to the termination of the algorithm. The flowchart for the INNH algorithm is shown in Algorithm 1.

Algorithm 1: INNH heuristic
Output : Solution π

```
Let the AGV route R = Φ and task j = 0
while U is not empty do
    for i = 1 to n
        Find the nearest task i from j in U
    endfor
    Test to append task i to route R
    if route R meets the capacity and time constraints then for
        Let task j = i and delete i from U
    else
        Append route R to solution π, and empty route R
        Add 0 at the end of π, and let j = 0
    endif
endwhile
if route R is unempty then
    Append route R to solution π
endif
return Solution π
```

3.3 Initial population phase

To enhance the quality and diversity of the initial population, this study employs three heuristic methods, INNH, NNH, and improved sweep-based heuristic (ISH) (Zou et al., 2021), to generate initial solutions. These methods are applied to produce three initial solutions, and the one with the minimum fitness value is selected and retained. Fitness in this context refers to the evaluation criterion for the quality of a solution. In this paper, a lower fitness value indicates a better solution quality. To further diversify the population, the remaining solutions are randomly generated. This approach achieves a balance between exploitation and exploration within the search space, thereby enhancing the quality and diversity of the initial population. Let P_i represent initialized populations, where PS represents the population size, $\pi_\sigma, \sigma \in [1, PS]$ represents the σth solution, π_{INNH} , π_{NNH} , and π_{ISH} represent the initial solutions generated by INNH, NNH, and ISH respectively. The flowchart of the initial population generation process is shown in Algorithm 2.

Algorithm 2 : Initial population
Output : the initial population P

Find the best solution $\pi_{bestIni}$ from π_{INNH} , π_{NNH} , π_{ISH}
Add solution $\pi_{bestIni}$ to the initial population P
for $\sigma = 2$ to $PSize$
  Randomly generate a permutation of all the tasks
  Let the AGV route $R = \Phi$
  for $i = 1$ to n
    Test to append task i to route R
    if router R meets the capacity and time constraints **then**
      Append task i to route R
    else
      Append route R to solution π_σ and empty route R
      Add 0 at the end of π_σ
    endif
  endfor
  if route R is unempty **then**
    Append route R to solution π_σ and empty route R
  endif
  Add solution π_σ to the initial population P
endfor
return the initial population P

3.4 Selection operation

The selection operation is a crucial aspect of GA as it improves the performance of populations by selecting well-adapted individuals. In traditional GA, the selection operation often directly selects individuals with better fitness from the initial population using either roulette wheel selection (Chen et al., 2019) or tournament selection (Routray & Ray, 2020). These selected individuals serve as the parents for the next generation.

In this paper, before the selection operation, an improved optimal preservation (IOP) strategy is applied. In each generation of a population, there are individuals which are better suited to the objective function than others. These individuals are known as 'elite individuals'. The IOP strategy is used to prevent the loss of elite individuals during iterations of the algorithm. Before each selection, the elite individuals are chosen. And a random sequence is generated for comparison with the elite individual. After that, the individual with the better fitness is selected as the new elite individual. This new elite individual is directly passed to the update stage without undergoing any crossover or mutation. The other individual that is not chosen as the new elite is replaced in the original position of the elite individual for subsequent operations. Let π_{best} denote the best individual of the current population, π_r denote the randomly generated individual, and f_b and f_r denote the fitness of π_{best} and π_r , respectively. The optimal preservation strategy flow is depicted in Algorithm 3.

Algorithm 3 : Optimal preservation strategy

Calculate the fitness f_b, f_r of π_{best}, π_r
if $f_b < f_r$ **then**
  Retain the solution π_{best}
  Let $\pi_{best} = \pi_r$
else
  Retain the solution π_r
endif
return Optimum conserved population P

For the populations updated by the IOP strategy, a roulette wheel approach is utilized to select individuals as the parental generation. The probability of selecting an individual is determined based on a fitness ratio calculation. This ratio is calculated by dividing the fitness value of the best individual in the population by the fitness value of the current individual. The resulting value is then used as the selection probability for the current individual. The calculation of the probability is as shown below:

$$P_i = \frac{f_{\min}}{f_i} \tag{20}$$

where f_{min} represents the fitness of the best individual in the population, f_i indicates the fitness of the current individual. The closer the value of P_i is to 1, the more similar the current individual is to the best individual in the population. The selection process is illustrated in Algorithm 4.

Algorithm 4 : Select operation
Output : Selected individuals of the parent generation seq_1, seq_2, seq_3

Let selection probability $P_i = f_{\min} / f_i$
do
  for $i = 1$ to 3
    select individuals seq_i from population P by roulette
  endfor
while $seq_1 \neq seq_2 \neq seq_3$
enddo
return Selected individuals of the parent generation seq_1, seq_2, seq_3

3.5 Crossover operation

In order to produce offspring chromosomes with improved fitness, the crossover operation combines advantageous traits from parent chromosomes. This paper proposes two crossover operations: the three-insertion store-optimal crossover (3-ISOC) operator and the three-parent random selection crossover (3-PRSC) operator, which are randomly selected using the variable *selectCross* with a certain probability. These methods have a dual benefit of increasing the diversity of the population and enhancing the convergence rate of offspring chromosomes. It ultimately leads to an overall improved performance of the IGA. The 3-ISOC process involves the insertion operation of genetic material in three parental individuals, while also evaluating the fitness of the resulting individuals. The individual with the superior fitness is then selected as the new offspring individual. To clarify the process, one must traverse all elements in the initial values of the offspring and determine their position, pos, in seq_i. Once pos is identified, all elements after pos in seq_i are inserted at the top of the seq_i. For example, supposing the initial sequence of offspring is (3, 6, 1, 5, 2, 4) and the seq_1 sequence is (1, 4, 2, 6, 3, 5). By comparison, it can be found that the first element of offspring, 3, is the fifth element in seq_1. Therefore, all elements after 3 are inserted to the top of seq_1 to get the sequence (3, 5, 1, 4, 2, 6). The second element of offspring, 6, is then moved on to, giving us the sequence (6, 3, 5, 1, 4, 2), and so on until all the elements of the offspring have been traversed, resulting in the sequence (4, 2, 6, 3, 5, 1). The same process is applied to seq_2 and seq_3. The detailed process is shown in Fig. 3, while the 3-ISOC process is shown in Algorithm 5.

The 3-PRSC operation involves randomly generating integers within the range of [0, 2]. The resulting number determines which parent individual the current element of the offspring individual will be taken from. Specifically, a random number of 0, 1, and 2 corresponds to seq_1, seq_2 and seq_3, respectively. Suppose the seq_1 sequence is (3, 6, 1, 5, 2, 4), the seq_2 sequence is (1, 4, 2, 6, 3, 5), the seq_3 sequence is (2, 5, 3, 4, 1, 6), and the generated sequence of random numbers is (1, 0, 2, 2, 0, 1). Then, the result generation process is shown in Fig. 4 and the 3-PRSC operation flow is shown in Algorithm 6. The overall flow of the crossover operation is illustrated in Algorithm 7.

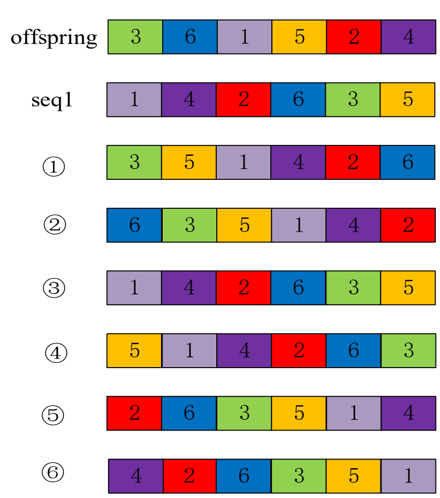

Fig.3. Example of 3-ISOC

Algorithm 5: Three-swap merit taking crossover operator
Output: Offspring individual *offspringseq*

Randomly select a sequence from seq_1, seq_2, seq_3 as seq
Let $offspringseq = seq$
for $i = 1$ to n
  for $j = 1$ to 3
    Crossover the $offspringseq$ and seq_j to obtain $cpyseq_j$
    Calculate the fitness of $cpyseq_j$ as f_j
  endfor
  Select the minimum value of fitness in f_1, f_2, f_3 as $f_k, k \in \{1,2,3\}$
  Let $offspringseq = cpyseq_k$
endfor
return Offspring individual *offspringseq*

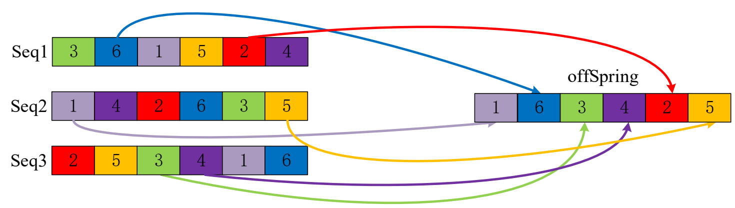

Fig.4. Example of 3-PRSC

Algorithm 6: Three-parent random selection crossover operator
Output: Offspring individual *offspringseq*

Randomly select a sequence from seq_1, seq_2, seq_3 as seq
Let $offspringseq = seq$
for $i =$ 0 to size of *offspring*
  Generate random integers from $[0,2]$ as j
  Update the i_{th} task in *offspringSeq* by the i_{th} task in Seq_j
endfor
return Offspring individual *offspringseq*

Algorithm 7: Cross operation
Output: Offspring individual *offspringseq*

Let *selectCrossCal* as the Algorithm selection probability
Randomly generate a decimal between 0 and 1 as *selectCross*
if $selectCross < selectCrossCal$ **then**
  3-ISOC
else
  3-PRSC
endif
return Offspring individual *offspringseq*

3.6 Mutation operation

The mutation operation is a random process that alters one or more gene positions within an individual's gene sequence. It results in the creation of a new individual. This paper proposes the use of PMX (Singh & Choudhary, 2009) to generate new offspring individual by mutating parent individuals. The process involves selecting three random positions, denoted as pos_1, pos_2 and pos_3, from the offspring individual. It is important to note that $pos_1 < pos_2 < pos_3$. The subsequence of tasks between positions pos_1 and pos_2 is extracted as *seqtemp* and inserted it after position pos_3 in the offspring individual. Then a new offspring individual is obtained. This approach allows for the creation of new and diverse offspring individual through the manipulation of parent individuals. The variation operation flow is illustrated in Algorithm 8.

Algorithm 8 : Mutation operation
Output : Offspring individual *offspringseq*

Sort pos_1, pos_2, pos_3in descending order so that $pos_1 < pos_2 < pos_3$
Let *seqtemp* for the tasks between pos_1 and pos_2
Insert *seqtemp* after position pos_3 of *offspringseq*
return Offspring individual *offspringseq*

3.7 Update population operation

The update population operation involves inserting offspring individual obtained through crossover mutation into the population. The offspring individual is used to replace the worst individual in the population, thereby updating it. This paper focuses on using offspring individual to update the population through the following method. The method first compares the offspring individual with the least fit individual in the population. The individual with better fitness values is then preferentially retained to update the population. Next, the least fit individual in the updated population is compared with individual preserved in the optimal preservation strategy. The individual with better fitness values is retained to further update the population. Let π_{worst} be the individual with the worst fitness and f_{worst} be its corresponding fitness value. Similarly, $\pi_{offspring}$ and $f_{offspring}$ represent the offspring individuals generated after crossover mutation. Then π_{retain} represents the optimal solution preserved in the optimal preservation strategy, and f_{retain} denotes its fitness. Finally, π'_{worst} and f'_{worst} represent the individual with the worst fitness after updating $\pi_{offspring}$ to the population. The process for updating the population operation is shown in Algorithm 9.

Algorithm 9 : Update population operation
Output : the updated population P

if $f_{worst} > f_{offspring}$ **then**
  $\pi_{offspring}$ substitutes for π_{worst}
endif
if $f'_{worst} > f_{retain}$ **then**
  π_{retain} substitutes for π'_{worst}
endif
return the updated population P

3.8 Procedure of the proposed IGA

The IGA proposed in this paper follows a series of main phases. Firstly, the initial population phase is executed (section 3.3). Subsequently, the selection, crossover and mutation and update operations (sections 3.4, 3.5 and 3.6 respectively) are performed in a sequential manner to generate offspring individual. Lastly, the offspring individual is inserted into the population through an update operation (section 3.7). The overall flow of the IGA is illustrated in Fig. 5.

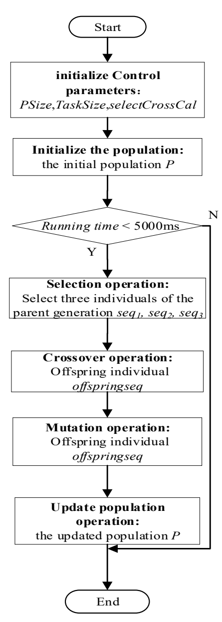

Fig. 5. Overall flow of the IGA

4. Computational and statistical experimentation

The efficacy of the proposed strategy and algorithm is validated in this section through thorough statistical experiments and calculations. The experimental test methods, data setup, and analysis methods are detailed. The optimal combination of all parameters is subsequently determined through extensive calibration experiments. Ultimately, the effectiveness of the proposed strategies and algorithms is demonstrated through a comparative analysis with other existing algorithms.

4.1 Experimental settings and test methods

In our experiments, instances are collected from Foxconn Technology Group, a Chinese electronic manufacturing company. One hundred instances of varying sizes, ranging from 10 to 50, are used. And these instances are categorized into test and calibration instances. Test instances are used for calculations and algorithm comparisons, while calibration instances are used to calibrate existing algorithms. To ensure the absence of experimental bias in calibration, the instances are segregated into two groups. The first group consists of 20 instances of the same size as the test instances, resulting in a total of 100 instances. The second group is composed of 10 instances, with two duplicate instances for each instance of the same size. The test instances are denoted as T and the calibration instances are denoted as C. In the test case, T3019 represents 30 tasks that need to be scheduled for completion, with 19 being the index of the specific instance. Each instance has unique details including identification, location, moment of invocation, material buffer inventory at the time of invocation, and the latest delivery time. For instance, {43, 5, 3, 53.9, 310, 28, 910} consists of the task number (43), the task's x-axis position (5), the y-axis position (3), the shortest distance to the depot point (53.9), the moment of call in the production cycle (310), the remaining stock in the buffer at the moment of call (28), and the latest delivery time in the producer's cycle (910). Due to space limitations, the data used in this paper is not presented, but interested readers can obtain it from the authors. The parameters involved in the model are shown in Table 1.

Table 1
Parameter settings

Items	Values	Items	Values
ΔT	$5s$	n'	12
C	$360s$	t_m	$30s/slice$
T_0	$365s$	g	$0.75kg/slice$
Q	$250kg$	S	$48slice$
v	$1m/s$	C_t	1
t_u	$15s$	c_e	0.1
n	100	c_a	200

The $\text{MAGVD}_{\text{UST}}$ is a novel problem for dispatching AGVs in matrix manufacturing workshops. To compare several algorithms that are suitable for the problem and popular in AGV dispatching, we select the Discrete Artificial Bee Colony algorithm (DABC) (Zou et al., 2020), the Discrete Invasive Weed Optimization algorithm (DIWO) (Li et al., 2023), the Harmonic Search algorithm (HS) (Li et al., 2019), the Greedy Iterative algorithm (IG) (Zou et al., 2021), and the Improved Greedy Iterative algorithm (IIG) (Zhang et al., 2022). Each algorithm uses the maximum running time, ΔT, as the termination condition. And each algorithm is repeated independently for the calibration and test instances, 10 and 30 times, respectively. The strengths and weaknesses of all algorithms are compared using the relative percentage deviation (RPD). The RPD is expressed as follows:

$$RPD = \frac{(F - F_{best})}{F_{best}} \times 100\% \tag{21}$$

where F is the transport cost of an algorithm for a given case and F_{best} is the minimum transport cost of all the compared algorithms for the same case. The smaller the RPD value, the better the algorithm's performance. All comparison algorithms in this paper are implemented in C++ and compiled using Visual Studio 2019 with the x64 compiler. The experiments are conducted on a Windows 10 operating system, utilizing an Intel Core i7-900K 3.60GHz PC with 32GB of RAM.

4.2 Calibration of the proposed and competing methods

Metaheuristics typically have optional parameters that require fine tuning to achieve optimal performance (Meng et al., 2020). In the IGA, two parameters are proposed, one for population size ($PSize$) and the other for $SelectCross$, where $SelectCross$ is utilized to select two crossover operations during the crossover phase. Through previous experience and extensive experimentation, a general range of values has been determined for the parameters $PSize$ and $SelectCross$. $PSize$ has 5 levels of calibration, which are 10, 50, 90, 130, and 170. Meanwhile, $SelectCross$ has 5 levels of calibration, which are 0.1, 0.6, 0.7, 0.8, and 0.9. Consequently, there are 25 parameter combinations in the calibration process, and each combination is run independently 10 times in each calibration instance. This results in a total of 2500 results for the 10 calibration instances. To determine the optimal parameter combination for the algorithm, the experimental results were evaluated using analysis of variance (ANOVA) and design of experiments (DOE) techniques. The results of the experiments are presented in Fig. 6.

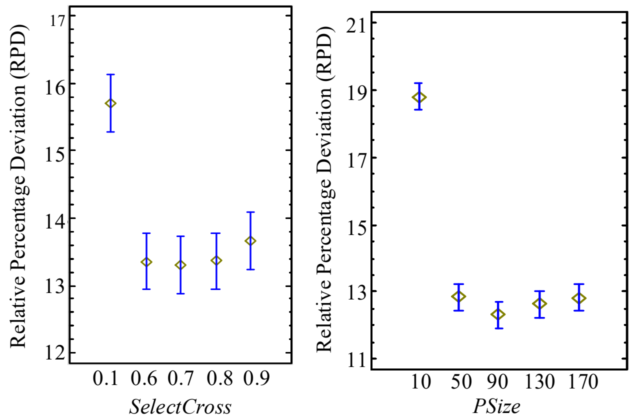

Fig. 6. Means plots of all parameters of the IGA

Fig. 6 displays the means along with their corresponding 95% confidence intervals for the two parameters of the IGA. The figure shows that the confidence intervals for the parameter *SelectCross* overlap at the levels of 0.6, 0.7, and 0.8, suggesting that there is no significant distinction between these three levels. Similarly, for the parameter *PSize* , there is no significant difference between the levels of 90, 130, and 170. However, after analyzing the results, it is discovered that *SelectCross*=0.7 and *PSize*=90 product the smallest RPD values. As a result, these parameter values are selected for the IGA. The same calibration process is applied to the other comparison algorithms, and the calibrated parameter values are presented in Table 2.

Table 2
Comparison of the algorithm's parameter calibration results

Algorithms	Parameters
IGA	$PSize = 90,\ \ SelectCross = 0.7$
HS	$HMS = 4,\ HMCRmin = 0.2,\ HMCRmax = 0.8$
DABC	$PSize = 150,\ l = 800,\ r = 80,\ \tau = 20$
DIWO	$PSize0 = 50,\ PSizemax = 70,\ Smax = 15,\ PLen = 2$
IG	$InitType = 0.8,\ T = 0.5,\ d = 5$
IIG	$d = 5,\ OperIter = 60$

4.3 Comparison of methods

To assess the effectiveness of the proposed IGA, a validation is conducted using 100 test instances with different sizes. Then cases of different sizes are analyzed to demonstrate the scalability of the IGA. The experimental setups in this section are identical to those in Section 4.1 and the evaluation metric used is RPD. And to obtain accurate experimental results, each algorithm is repeated 30 times for each of the 100 instances. The results, consisting of the minimum (Min), maximum (Max), and average (Ave) values of RPD, are presented in Tables 3-7. The best algorithm comparison results are highlighted in bold.

Table 3
Experimental results of 10 tasks.

Instance	DABC			IG			IIG			HS			DIWO			IGA		
	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave
T10I1	72.54	72.57	72.55	72.54	72.54	72.54	72.54	72.54	72.54	72.57	75.10	73.56	31.57	31.90	31.72	0.00	10.08	**4.82**
T10I2	30.51	30.65	30.56	30.46	30.47	30.46	30.46	30.55	30.49	30.51	76.57	63.67	22.98	23.33	23.21	0.00	6.76	**3.59**
T10I3	36.01	40.29	36.60	36.01	36.02	36.01	36.03	36.18	36.12	36.02	91.41	78.38	25.23	25.41	25.25	0.00	6.72	**3.49**
T10I4	28.89	29.34	29.16	28.69	28.70	28.69	28.72	29.19	28.99	28.76	33.05	29.38	23.99	24.36	24.11	0.00	11.78	**3.82**
T10I5	69.24	69.42	69.29	69.24	69.24	69.24	69.24	69.29	69.26	69.26	73.17	70.47	27.33	28.13	27.78	0.00	6.12	**3.45**
T10I6	61.63	61.64	61.63	61.63	61.63	61.63	61.63	61.63	61.63	61.65	62.59	62.07	18.72	19.04	19.01	0.00	7.78	**3.69**
T10I7	81.51	81.62	81.55	81.50	81.51	81.50	81.52	81.82	81.65	81.54	83.30	82.20	31.39	31.39	31.39	0.00	12.37	**7.56**
T10I8	75.43	76.98	75.95	75.33	75.34	75.33	75.34	75.49	75.40	75.34	79.76	76.75	24.67	24.84	24.78	0.00	12.29	**7.65**
T10I9	16.55	16.68	16.59	16.53	16.53	16.53	16.53	16.54	16.53	16.71	63.15	49.63	23.18	23.34	23.22	0.00	6.36	**3.34**
T10I10	18.44	59.20	24.29	18.25	20.92	19.10	18.35	21.21	19.42	20.95	61.72	53.48	19.97	19.97	19.97	0.00	4.93	**2.46**
T10I11	34.19	34.94	34.51	34.13	34.14	34.13	34.15	34.51	34.26	34.50	72.10	61.68	27.51	29.10	27.61	0.00	11.51	**5.40**

T10I12	13.99	14.16	14.08	13.99	13.99	13.99	13.99	14.01	14.00	14.26	70.55	26.31	20.54	20.54	20.54	0.00	9.00	**5.09**
T10I13	35.21	35.35	35.28	35.20	35.21	35.20	35.20	35.23	35.21	35.24	77.93	58.28	36.99	37.17	37.02	0.00	9.04	**3.80**
T10I14	29.73	30.01	29.84	29.72	29.72	29.72	29.72	29.77	29.75	29.74	74.96	53.15	28.98	29.33	29.16	0.00	6.83	**3.04**
T10I15	28.83	30.88	29.50	28.74	28.78	28.75	28.78	29.19	28.97	28.79	79.47	55.35	32.71	32.88	32.75	0.00	12.65	**6.97**
T10I16	31.32	67.72	45.99	31.13	31.30	31.18	31.32	31.85	31.74	31.23	75.72	63.48	26.09	26.09	26.09	0.00	10.56	**6.38**
T10I17	32.49	32.60	32.54	32.47	32.51	32.49	32.50	32.67	32.57	32.50	75.74	37.76	34.69	34.88	34.86	0.00	13.36	**8.24**
T10I18	34.97	35.41	35.14	34.91	34.92	34.92	34.92	34.98	34.94	35.85	91.12	81.16	24.76	24.94	24.84	0.00	8.69	**4.20**
T10I19	16.42	17.46	16.99	16.39	16.46	16.41	16.53	17.39	17.09	17.37	66.68	35.82	18.96	19.12	18.97	0.00	5.85	**2.64**
T10I20	72.38	72.68	72.51	72.37	72.38	72.38	72.38	72.81	72.57	72.50	73.51	72.90	25.48	25.48	25.48	0.00	7.48	**3.94**
Average	41.01	45.48	42.23	40.96	41.12	41.01	40.99	41.34	41.16	41.27	72.88	59.27	26.29	26.56	26.39	0.00	9.01	**4.68**

Table 3 presents the average Ave values of RPD for 10 tasks, which are 42.23%, 41.01%, 41.16%, 59.27%, 26.39%, and 4.68% for DABC, IG, IIG, HS, DIWO, and IGA, respectively. The IGA outperforms the other algorithms with the highest overall average RPD, followed by DIWO, DABC, IG, and IIG, while HS performs the worst. It is noteworthy that the IGA achieved the minimum value in Ave for all 20 arithmetic instances with 10 tasks.

Table 4
Experimental results of 20 tasks

Instance	DABC			IG			IIG			HS			DIWO			IGA		
	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave
T20I1	20.33	23.00	21.88	19.96	21.87	20.71	20.66	22.23	21.61	21.25	62.03	48.03	21.59	26.60	24.19	0.00	10.29	**4.95**
T20I2	21.91	23.93	23.05	20.33	22.23	21.77	21.93	24.48	23.46	23.41	60.28	46.04	21.08	28.08	24.84	0.00	6.45	**2.99**
T20I3	17.43	20.09	18.62	17.22	17.40	17.31	18.34	21.89	20.24	17.54	54.39	43.47	12.58	21.01	17.19	0.00	9.82	**5.47**
T20I4	16.18	18.76	16.95	15.90	17.06	16.45	17.21	17.56	17.41	16.91	54.85	36.29	13.40	21.73	17.85	0.00	6.46	**3.51**
T20I5	18.09	42.24	19.98	17.41	19.19	17.72	18.38	21.15	19.74	42.16	58.02	49.01	20.33	32.52	27.47	0.00	14.34	**7.52**
T20I6	12.71	15.82	14.16	12.67	13.51	13.05	13.29	13.65	13.45	14.49	49.90	37.24	14.14	21.89	18.46	0.00	8.98	**4.94**
T20I7	22.26	27.17	23.99	21.56	42.58	24.07	23.24	23.75	23.57	42.71	61.95	51.02	25.10	32.31	28.55	0.00	10.46	**4.53**
T20I8	16.19	18.44	16.77	16.04	16.10	16.06	16.27	18.09	17.24	17.40	62.01	46.46	20.81	28.99	24.81	0.00	4.36	**2.49**
T20I9	17.62	22.14	19.70	17.57	19.91	18.63	18.13	18.67	18.44	40.54	58.28	46.40	17.98	26.32	21.60	0.00	5.34	**2.99**
T20I10	14.23	15.78	15.06	13.80	15.34	14.64	14.85	16.07	15.63	17.01	49.99	38.98	7.90	16.47	11.93	0.00	3.79	**1.91**
T20I11	20.74	24.42	22.27	20.54	23.27	21.65	22.14	23.77	22.91	41.58	60.79	50.51	19.75	26.56	22.99	0.00	5.26	**2.88**
T20I12	13.27	14.47	14.02	12.70	13.08	12.85	13.66	15.16	14.53	13.91	52.84	38.09	13.81	19.62	16.59	0.00	4.31	**2.41**
T20I13	26.27	27.53	26.99	25.16	26.41	25.74	26.92	28.58	27.71	26.78	62.51	53.03	25.08	30.60	27.59	0.00	9.24	**5.03**
T20I14	27.12	30.38	28.71	27.34	28.14	27.50	28.37	30.31	29.37	29.95	70.62	56.40	23.78	30.62	27.16	0.00	13.74	**5.74**
T20I15	21.55	24.80	23.34	19.82	42.30	22.89	21.63	23.98	22.90	44.49	57.31	47.81	17.51	25.71	21.29	0.00	6.37	**4.30**
T20I16	17.49	43.68	23.17	17.15	18.39	17.65	17.98	20.87	19.27	18.09	57.26	47.20	21.16	30.39	25.45	0.00	9.40	**6.41**
T20I17	21.04	22.99	21.85	20.98	21.39	21.12	21.61	23.58	22.63	22.71	65.29	46.99	23.72	31.67	28.40	0.00	11.28	**5.41**
T20I18	17.95	20.02	18.45	16.90	17.97	17.72	18.31	18.75	18.49	18.42	63.55	40.12	18.50	26.30	22.29	0.00	7.23	**3.70**
T20I19	13.05	15.36	13.64	11.97	13.01	12.70	13.41	14.40	13.82	14.89	48.96	36.47	16.53	22.64	20.35	0.00	10.42	**3.30**
T20I20	15.29	18.03	16.33	15.23	16.34	15.49	17.16	18.95	18.08	17.18	55.49	43.44	13.65	17.38	15.30	0.00	3.98	**2.57**
Average	18.54	23.45	19.95	18.01	21.27	18.79	19.18	20.79	20.02	25.07	58.32	45.15	18.42	25.87	22.22	0.00	8.08	**4.15**

According to the data presented in Table 4, there was a noticeable change in the performance of the algorithms in relation to the average Ave values of RPD as the number of tasks increased from 10. In this case, IGA remains the most efficient algorithm with the lowest average Ave value for all cases, IG outperforms DIWO, followed closely by IIG and DABC. DIWO performed only slightly better than HS, which is the worst performer. It indicates that IGA is highly effective in solving $MAGVD_{UST}$ for instance with 20 tasks, outperforming other comparative algorithms.

Table 5
Experimental results of 30 tasks

Instance	DABC			IG			IIG			HS			DIWO			IGA		
	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave
T30I1	29.94	39.83	34.22	31.97	46.07	43.15	31.20	32.95	32.21	49.77	60.27	54.71	42.70	51.82	47.62	0.00	14.39	**9.72**
T30I2	30.78	36.83	33.81	30.38	47.11	43.76	33.58	34.75	34.40	51.43	63.51	57.84	40.24	50.22	46.80	0.00	16.95	**10.88**
T30I3	21.55	39.60	25.46	19.02	37.88	23.14	21.82	24.07	23.04	37.44	50.86	45.62	25.99	34.43	30.05	0.00	10.29	**4.97**
T30I4	18.76	25.88	20.92	17.37	24.28	18.71	20.99	23.18	21.72	36.27	48.82	43.27	26.44	35.17	32.45	0.00	4.81	**2.95**
T30I5	16.23	33.23	29.26	15.02	30.94	27.90	18.48	20.22	19.23	29.92	42.06	38.28	30.55	40.40	34.72	0.00	6.63	**4.13**
T30I6	22.84	39.16	26.40	21.64	38.26	35.59	24.49	26.32	25.57	39.50	51.32	46.56	39.03	45.10	41.30	0.00	12.72	**6.52**
T30I7	19.59	34.00	27.22	31.79	34.74	33.74	21.52	25.50	24.75	36.49	46.84	41.30	31.76	39.70	34.82	0.00	7.22	**4.80**
T30I8	22.37	39.65	36.01	37.21	40.03	38.34	31.00	33.76	32.36	41.47	50.42	46.74	30.05	40.15	35.79	0.00	12.81	**8.93**
T30I9	26.29	35.34	31.86	26.34	40.24	38.40	29.10	32.00	30.53	41.34	54.41	47.83	33.11	40.21	35.97	0.00	13.82	**8.21**
T30I10	30.97	39.22	33.52	43.85	46.09	44.74	31.67	35.00	33.17	45.55	59.36	53.96	39.30	47.50	44.17	0.00	15.47	**8.67**
T30I11	27.29	33.65	30.45	28.49	41.58	37.79	29.80	32.75	31.72	40.13	59.79	52.14	35.92	46.10	41.05	0.00	16.28	**9.73**
T30I12	12.73	29.30	15.06	11.59	28.01	17.41	12.91	14.18	13.39	30.27	41.92	36.07	24.74	32.44	29.00	0.00	9.32	**3.87**
T30I13	32.94	38.00	35.96	32.05	48.81	39.70	34.72	36.16	35.51	53.82	68.76	60.45	42.11	53.11	48.64	0.00	17.42	**10.56**
T30I14	25.90	40.95	29.89	37.53	40.17	39.19	29.39	33.18	32.03	42.77	56.79	48.59	36.30	45.88	40.70	0.00	10.49	**6.02**
T30I15	22.00	37.66	27.00	21.58	36.46	35.08	23.96	26.32	25.25	36.83	47.85	43.03	28.08	36.46	34.07	0.00	9.16	**5.12**
T30I16	33.19	35.91	34.62	32.67	34.10	33.47	22.46	36.94	34.39	35.76	44.77	40.21	33.89	40.33	37.14	0.00	11.04	**6.86**
T30I17	35.44	49.86	40.86	33.44	48.98	43.11	37.20	39.69	38.69	52.25	66.25	59.06	45.16	52.48	48.71	0.00	17.85	**12.32**
T30I18	23.90	29.59	27.22	22.08	41.61	30.70	25.67	27.67	26.69	44.99	57.92	50.71	36.70	46.04	41.76	0.00	15.05	**9.53**
T30I19	16.83	22.17	19.71	19.07	33.50	29.38	19.06	22.01	20.37	21.28	47.94	39.83	32.20	38.90	35.79	0.00	10.02	**6.72**
T30I20	25.89	31.99	28.82	24.46	42.65	38.14	25.30	27.47	26.73	44.60	58.12	50.43	31.12	40.98	37.59	0.00	13.48	**8.76**
Average	24.77	35.59	29.41	26.88	39.08	34.57	26.22	29.21	28.09	40.59	53.90	47.83	34.27	42.87	38.91	0.00	12.26	**7.46**

Based on the data provided in Table 5, it is clear that the IGA algorithm outperforms all other algorithms in terms of the minimum (Min) value, maximum (Max) value, and average (Ave) value. The IIG algorithm is the second-best performer, with HS and DIWO performing the worst. DABC and IG algorithms are placed in the middle level. This is a strong demonstration of the effectiveness of the IGA in solving $MAGVD_{UST}$ for instance with 30 tasks.

Table 6
Experimental results of 40 tasks

Instance	DABC			IG			IIG			HS			DIWO			IGA		
	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave
T40I1	24.14	28.44	27.14	24.19	35.29	33.27	24.81	27.02	26.18	37.51	45.87	42.12	37.86	47.39	43.17	0.00	10.44	**5.13**
T40I2	34.72	46.11	39.06	42.78	45.11	44.11	36.81	40.24	39.13	49.10	58.70	54.34	43.69	59.38	52.82	0.00	18.93	**9.91**
T40I3	27.21	31.53	29.54	24.50	40.81	31.75	29.59	31.67	30.82	44.50	54.88	49.46	36.88	46.58	42.34	0.00	11.40	**6.44**
T40I4	30.38	44.97	36.26	30.24	43.44	40.98	32.32	32.77	32.51	44.70	54.61	50.66	43.41	51.79	47.67	0.00	11.42	**5.12**
T40I5	20.47	32.23	26.35	28.27	30.58	29.60	21.42	23.43	22.44	34.52	41.24	37.88	36.08	43.77	40.08	0.00	13.87	**7.28**
T40I6	27.61	39.39	35.28	36.81	38.23	37.60	27.20	30.73	29.62	43.03	48.67	45.42	45.25	54.21	50.41	0.00	10.97	**5.23**
T40I7	27.29	29.62	28.26	35.63	38.98	37.77	26.98	27.53	27.23	42.54	50.36	46.04	39.31	52.33	47.27	0.00	17.50	**9.46**
T40I8	28.54	42.59	36.57	27.15	40.14	38.95	33.79	42.11	37.54	42.00	52.12	47.19	43.72	51.04	47.15	0.00	17.04	**12.39**
T40I9	26.13	38.33	30.91	35.63	37.37	36.54	27.91	29.45	28.63	39.85	49.90	45.06	37.68	48.33	45.66	0.00	18.32	**10.12**
T40I10	21.47	32.76	24.94	29.96	31.96	31.12	23.40	24.70	24.18	34.90	43.35	39.85	33.82	43.81	40.01	0.00	7.98	**4.71**
T40I11	25.85	37.79	29.13	27.05	36.48	35.47	28.27	29.41	28.87	41.40	48.26	44.37	38.45	47.78	44.42	0.00	12.16	**5.82**
T40I12	17.30	29.92	22.74	15.94	28.81	26.14	20.37	23.80	22.28	32.10	39.78	35.78	31.50	40.13	36.24	0.00	10.42	**4.21**
T40I13	31.76	44.54	35.90	29.58	41.38	38.14	31.83	33.35	32.69	44.89	58.02	50.11	44.01	56.18	51.55	0.00	8.33	**3.70**
T40I14	31.41	38.86	35.09	32.00	42.97	40.69	33.97	36.81	35.95	46.11	55.42	51.76	46.33	57.18	52.01	0.00	14.87	**8.78**
T40I15	21.59	33.97	26.45	31.04	32.76	31.99	22.33	24.21	23.41	35.24	42.44	39.45	32.84	42.17	38.67	0.00	13.20	**5.01**
T40I16	19.70	24.39	21.77	27.90	30.38	29.39	20.55	22.46	21.42	30.91	39.40	36.18	38.36	45.19	40.84	0.00	7.88	**4.47**
T40I17	39.77	51.68	49.20	47.88	50.88	49.42	39.92	44.10	41.93	51.57	62.28	58.34	52.23	61.94	57.62	0.00	17.30	**9.81**
T40I18	31.00	36.13	32.67	29.12	43.00	40.01	30.54	32.36	31.62	45.62	55.43	49.88	47.97	56.46	52.68	0.00	12.03	**8.20**
T40I19	19.94	31.78	22.47	19.49	31.66	30.34	20.27	21.50	20.90	30.50	42.53	38.57	37.99	45.05	41.65	0.00	14.31	**6.77**
T40I20	22.72	25.32	24.13	22.07	34.10	32.32	23.67	24.88	24.25	35.79	45.72	41.49	34.48	43.42	39.05	0.00	10.24	**5.10**
Average	26.45	36.02	30.69	29.86	37.72	35.78	27.80	30.13	29.08	40.34	49.45	45.20	40.09	49.71	45.56	0.00	12.93	**6.88**

Based on Table 6, it is evident that the IGA algorithm outperforms the other five algorithms. The performance of IIG, IG, and DABC is still relatively similar, while DIWO does not perform as well as HS and is the worst performing algorithm.

Table 7
Experimental results of 50 tasks.

Instance	DABC			IG			IIG			HS			DIWO			IGA		
	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave
T50I1	28.67	36.74	30.38	33.29	36.09	34.94	27.70	29.71	29.10	38.06	45.37	41.71	43.54	55.59	49.91	0.00	9.37	**5.11**
T50I2	45.42	47.85	46.91	43.42	45.52	44.65	37.32	49.14	43.55	49.78	69.66	53.07	52.31	64.71	59.10	0.00	23.22	**8.99**

T50I3	38.71	49.26	44.22	43.58	46.26	44.86	39.36	41.55	40.60	46.72	56.54	51.61	50.16	62.82	57.82	0.00	20.48	**11.09**
T50I4	40.06	49.01	45.56	45.79	47.42	46.65	39.30	41.71	40.55	51.57	59.96	53.99	57.73	69.48	65.10	0.00	22.87	**8.61**
T50I5	28.05	30.88	29.60	27.03	28.43	27.67	21.54	24.96	23.45	30.95	48.57	34.35	36.38	48.98	44.31	0.00	11.61	**5.23**
T50I6	37.04	44.77	43.05	40.24	42.11	41.50	40.30	48.65	46.81	44.89	51.32	48.70	52.29	67.51	61.91	0.00	22.22	**10.52**
T50I7	24.94	34.57	28.47	30.13	32.83	31.75	25.18	27.84	26.95	35.35	42.10	38.58	36.20	55.30	50.74	0.00	11.12	**5.50**
T50I8	25.83	31.94	30.38	26.79	29.39	28.49	23.89	30.79	27.32	31.31	38.20	34.80	33.77	49.56	43.61	0.00	7.92	**4.46**
T50I9	24.40	33.95	31.85	28.46	30.98	29.88	24.05	27.11	25.99	33.94	54.05	37.48	36.70	51.90	47.01	0.00	9.79	**4.95**
T50I10	35.72	42.24	41.02	37.06	39.32	38.72	34.11	37.12	35.62	42.68	66.16	46.75	48.94	58.43	54.14	0.00	14.15	**7.96**
T50I11	29.91	39.39	36.23	34.65	37.10	36.18	30.79	33.80	32.47	38.38	48.18	42.72	43.37	56.70	50.39	0.00	13.31	**6.46**
T50I12	24.14	33.16	31.77	28.88	32.17	31.14	25.04	27.01	26.32	33.82	39.99	37.19	36.24	53.33	46.87	0.00	10.82	**5.54**
T50I13	47.86	55.32	53.47	50.10	52.53	51.42	46.97	56.14	50.27	57.04	63.38	59.87	58.77	78.13	70.30	0.00	19.42	**10.25**
T50I14	45.38	48.05	46.70	42.92	45.03	44.35	40.90	50.20	42.95	49.48	55.56	52.25	42.27	69.09	63.26	0.00	17.28	**8.29**
T50I15	36.58	38.92	38.10	35.46	37.28	36.31	29.77	32.61	31.01	39.58	59.02	43.14	45.76	55.85	52.01	0.00	12.03	**6.52**
T50I16	22.32	33.35	30.39	29.24	31.05	30.14	22.44	23.60	23.01	33.57	40.86	36.37	42.78	56.18	50.26	0.00	8.72	**3.96**
T50I17	53.24	56.05	54.88	52.17	54.48	53.46	47.13	51.22	48.73	56.24	65.34	60.80	61.41	77.39	70.21	0.00	21.74	**11.04**
T50I18	29.28	32.32	31.07	35.76	38.16	36.88	30.57	32.68	31.66	40.74	47.21	43.08	39.28	58.92	52.54	0.00	11.03	**3.29**
T50I19	31.27	34.10	32.63	30.50	32.41	31.33	24.81	32.33	29.65	33.04	41.45	37.22	42.85	55.39	50.42	0.00	12.59	**6.47**
T50I20	29.86	42.48	38.83	37.08	39.85	38.48	30.16	33.47	31.97	43.73	50.86	46.61	44.75	59.08	54.91	0.00	12.67	**7.95**
Average	33.93	40.72	38.28	36.63	38.92	37.94	32.07	36.58	34.40	41.54	52.19	45.01	45.27	60.22	54.74	0.00	14.62	**7.11**

Table 7 presents the results obtained by all algorithms when solving instances with task numbers of 50. Table 7 shows that the average Ave values of RPD for DABC, IG, IIG, HS, DIWO, and IGA are 38.28%, 37.94%, 34.4%, 45.01%, 54.74%, and 7.11%, respectively. In terms of overall average Ave values, the best performing algorithm is still IGA, followed by IIG, IG, DABC, HS, with DIWO being the worst. IGA achieves the minimum Ave value for all 20 instances, with considerably smaller values than the Ave values of the other algorithms. The fact that IGA consistently outperforms other algorithms across all 20 instances indicates that it is capable of finding the best solution.

According to Tables 3-7, it can be concluded that IGA outperforms the other five algorithms for task sizes ranging from 10 to 50. Therefore, it can be inferred that IGA is highly effective in solving the $\mathrm{MAGVD_{UST}}$. To enhance the analysis of the proposed algorithm and provide a more comprehensive understanding, a statistical approach utilizing multi-factor ANOVA analysis is employed in this section. The RPD data obtained from solving all algorithms is considered, with the influencing factors being the comparison algorithm and task size. In Fig. 7, mean plots with 95% Tukey HSD confidence intervals for the six comparison algorithms are presented. It can be clearly observed that the performance of IGA is significantly superior to the other five algorithms. Fig. 8 shows the interaction diagram between the six comparison algorithms and task scales. The horizontal coordinate n represents the task size, and the vertical coordinate represents the RPD value. The figure clearly indicates that IGA outperforms the other comparison algorithms for task sizes $n = 10, 20, 30, 40,$ and 50.

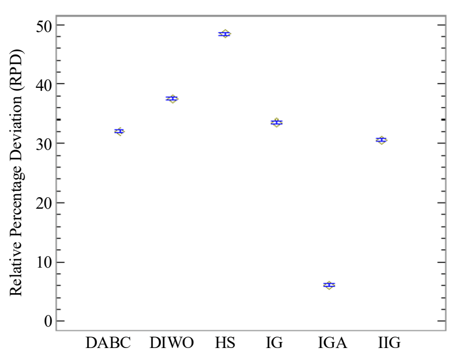

Fig.7. Means plots with 95% Tukey's HSD confidence intervals for all the comparison algorithms

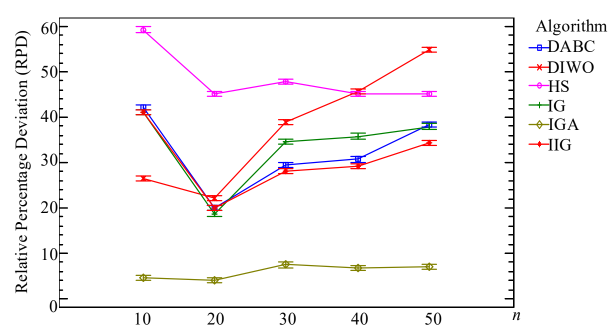

Fig. 8. Means plots of interaction between the six competitive methods and tasks size

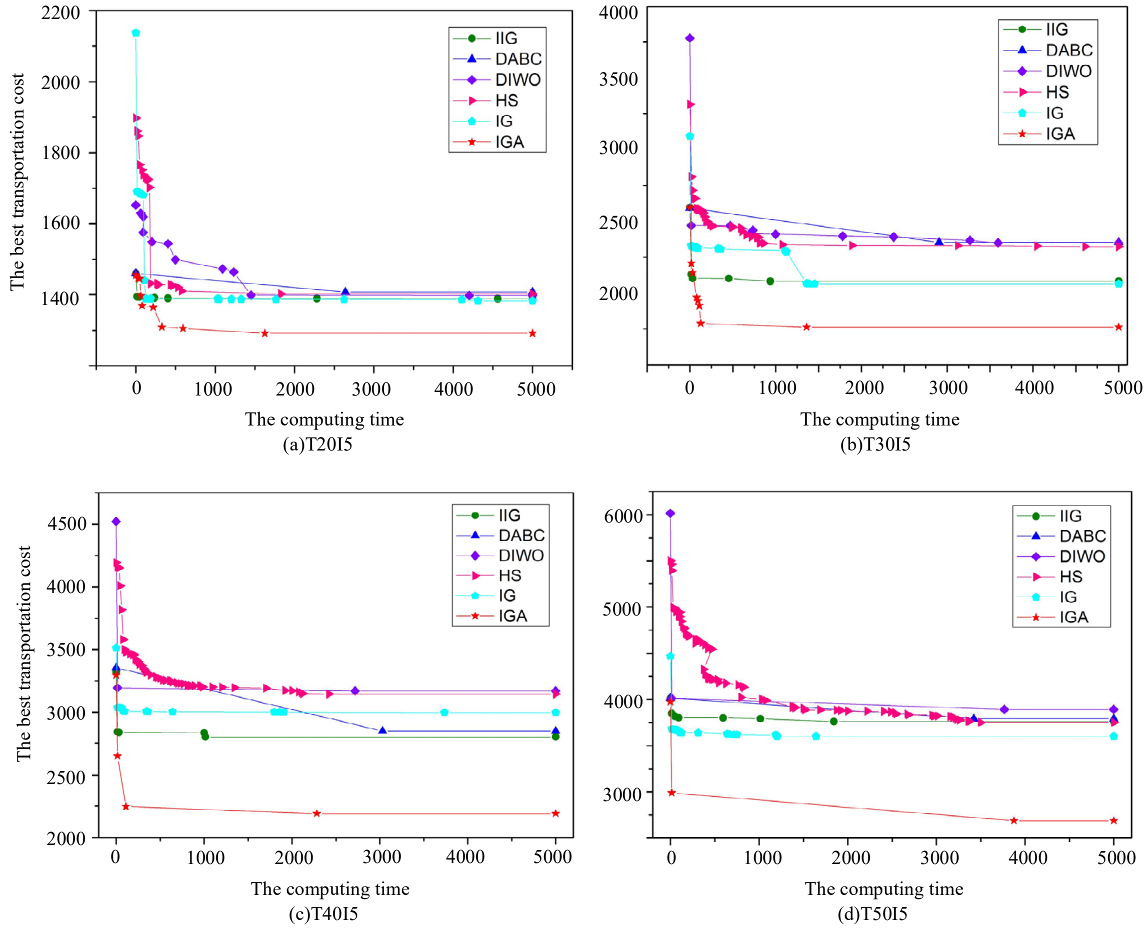

Fig.9. Convergence curves of algorithms.

To assess the convergence performance of the proposed algorithm, evolutionary graphs are presented to illustrate the variation of AGV transport costs for the problem at different time points in a cycle. Experiments are conducted on four different problem sizes - T20I5, T30I5, T40I5, and T50I5 - selected from the test set, and record the minimum total cost obtained for the six compared algorithms. Fig. 9(a) to Fig. 9(d) illustrate the evolution curves for the six algorithms in the four instances. The horizontal axis represents different time points within the same production cycle, while the vertical axis represents the minimum transport costs achieved. As seen in the figures, the IGA demonstrates the best initial results for each instance. The results indicate that the proposed heuristic algorithm and strategy exhibit superior performance for the problem at hand and prove to be more effective in solving the proposed problem.

The analysis presented above demonstrates that the proposed IGA is more effective than the other five algorithms in solving the given problem. It is evidenced by the results obtained for various task sizes, including average, minimum, and maximum values, as well as means plots, interaction plots, and evolutionary curves.

5. Conclusions and future research

This paper has studied the $MAGVD_{UST}$, a new problem with the objective of minimizing transportation costs. In our perception, it has not been addressed in the current research. In this paper, we establish a mixed-integer linear programming model at first, and then propose an effective IGA. To illustrate the effectiveness of IGA, we have selected five popular algorithms in existing literature for comparison. All of the algorithms have been thoroughly evaluated on 110 instances from an actual electronic manufacturing factory. The experimental analysis shows that the proposed IGA is more effective than the other five algorithms in solving the $MAGVD_{UST}$. The main contributions of IGA are as followed: Firstly, an INNH algorithm is utilized to enhance the quality of solutions generated in the initial population. Secondly, a meritocratic initial algorithm is used to further improve the initial population. Next, a selection operation employs the optimal solution preservation strategy to ensure the best solutions are retained. To increase population diversity, two crossover operations are designed and a control parameter is set to probabilistically choose between them. Besides, a mutation operation based on PMX is also used to further enhance the diversity of the population. Finally, two meritocratic choices are used to update the population. Overall, our proposed IGA offers a comprehensive solution to the problem.

This study focuses on the regular use of workshops, however, there exhibits variability and unpredictability in the production process such as AGV energy replenishment and conflicts. Therefore, the next step involves exploring problem-oriented strategies or leveraging problem-specific knowledge to enhance the performance of the IGA in effectively addressing and managing these uncertainties. Deep reinforcement learning (DRL) is an empirical approach to learning that can automatically determine optimal policies without explicit specification (Ha et al., 2021; Li et al., 2023; Wei et al., 2022). Therefore, it would be intriguing to explore the potential of combining DRL with AGV dispatching. This paper is expected to provide a fresh and thought-provoking perspective on AGVDP.

Acknowledgment

This research is partially supported by the National Natural Science Foundation of China under Grant (No. 52205529), partially supported by the Natural Science Foundation of Shandong Province (ZR2021QE195) and Research fund project of Liaocheng university under Grant (No. 318012110 and No. 318052150).

References

Cao, X., & Zhu, M. (2021). Research on global optimization method for multiple AGV collision avoidance in hybrid path. *Optimal Control Applications and Methods*, *42*(4), 1064-1080.

Chen, C., Tiong, L. K., & Chen, I.-M. (2019). Using a genetic algorithm to schedule the space-constrained AGV-based prefabricated bathroom units manufacturing system. *International Journal of Production Research*, *57*(10), 3003–3019.

Chen, X., Wu, W., & Hu, R. (2022). A Novel Multi-AGV Coordination Strategy Based on the Combination of Nodes and Grids. *IEEE Robotics and Automation Letters*, *7*(3), 6218-6225.

Eda, S., Nishi, T., Mariyama, T., Kataoka, S., Shoda, K., & Matsumura, K. (2012). Petri net decomposition approach for bi-objective routing for AGV systems minimizing total traveling time and equalizing delivery time. *Journal of Advanced Mechanical Design Systems and Manufacturing*, *6*(5), 672–686.

Ha, W. Y., Cui, L., & Jiang, Z.-P. (2021). A warehouse scheduling using genetic algorithm and collision index. 2021 *20th International Conference on Advanced Robotics (ICAR)*, 318–323. https://doi.org /10.1109/ICAR53236.2021.9659439

Hao, J., Wang, C., Yang, M., & Wang, B. (2020). Hybrid genetic algorithm based dispatch and conflict-free routing method of agv systems in unmanned underground parking lots. In *2020 IEEE international conference on real-time computing and robotics (RCAR)*, 475-480 https://doi.org/10.1109/RCAR49640.2020.9303275.

Hu, Y. J., Dong, L. C., & Xu, L. (2020). Multi-AGV dispatching and routing problem based on a three-stage decomposition method. *Mathematical Biosciences and Engineering*, *17*(5), 5150–5172.

Huang, S. J., Lee, T. S., Li, W. H., & Chen, R. Y. (2018). Modular on-road AGV wireless charging systems via interoperable power adjustment. *IEEE Transactions on Industrial Electronics*, *66*(8), 5918-5928.

Jahanzaib, M., Masood, S. A., Nadeem, S., Akhtar, K., & Shahbaz, M. (2013). Application of genetic algorithm (ga) approach in the formation of manufacturing cells for group technology. Life Science Journal, *9*(4), 799-809.

Jin, J., & Zhang, X. H. (2016). Multi AGV scheduling problem in automated container terminal. *Journal of Marine Science and Technology-Taiwan*, *24*(1), 32–38.

Li, G., Li, X., Gao, L., & Zeng, B. (2019). Tasks assigning and sequencing of multiple AGVs based on an improved harmony search algorithm. *Journal of Ambient Intelligence and Humanized Computing*, *10*(11), 4533–4546.

Li, H., Gao, K., Duan, P. Y., Li, J. Q., & Zhang, L. (2022). An Improved Artificial Bee Colony Algorithm With Q-Learning for Solving Permutation Flow-Shop Scheduling Problems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*. https://doi.org/10.1109/TSMC.2022.3219380

Li, J., Cheng, W., Lai, K. K., & Ram, B. (2022). Multi-AGV Flexible Manufacturing Cell Scheduling Considering Charging. *Mathematics*, *10*(19), 3417.

Li, Z. K., Sang, H. Y., Li, J. Q., Han, Y. Y., Gao, K. Z., Zheng, Z. X., & Liu, L. L. (2023). Invasive Weed Optimization for multi-AGVs dispatching problem in a matrix manufacturing workshop. *Swarm and Evolutionary Computation*, 101227.

Lu, S., Xu, C., Zhong, R. Y., & Wang, L. (2017). A RFID-enabled positioning system in automated guided vehicle for smart factories. *Journal of Manufacturing Systems*, *44*, 179-190.

Liu, L., Qu, T., Thurer, M., Ma, L., Zhang, Z., & Yuan, M. (2022). A new knowledge-guided multi-objective optimisation for the multi-AGV dispatching problem in dynamic production environments. *International Journal of Production Research*. https://doi.org/10.1080/00207543.2022.2122619

Meng, L., Gao, K., Ren, Y., Zhang, B., Sang, H., & Chaoyong, Z. (2022). Novel MILP and CP models for distributed hybrid flowshop scheduling problem with sequence-dependent setup times. *Swarm and Evolutionary Computation*, *71*, 101058.

Meng, L., Cheng, W., Zhang, B., Zou, W., Fang, W., & Duan, P. (2023). An Improved Genetic Algorithm for Solving the Multi-AGV Flexible Job Shop Scheduling Problem. *Sensors*, *23*(8), 3815.

Meng, L., Zhang, C., Ren, Y., Zhang, B., & Lv, C. (2020). Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem. *Computers & Industrial Engineering*, *142*, 106347.

Micieta, B., Edl, M., Krajcovic, M., Dulina, L., Bubenik, P., Durica, L., & Binasova, V. (2018). Delegate MASs for coordination and control of one-directional AGV systems: a proof-of-concept. *The International Journal of Advanced Manufacturing Technology*, *94*, 415-431.

Ng, P. P. W., Yucel, G., & Duffy, V. G. (2009). Modelling the effect of AGV operating conditions on operator perception of acceptability and hazard. *International Journal of Computer Integrated Manufacturing*, *22*(12), 1154–1162.

Nishida, K., & Nishi, T. (2022). Dynamic Optimization of Conflict-Free Routing of Automated Guided Vehicles for Just-in-Time Delivery. *IEEE Transactions on Automation Science and Engineering*. https://doi.org/10.1109/TASE.2022.3194082

Niu, H. Y., Wu, W. M., Xing, Z. C., Wang, X. K., & Zhang, T. (2023). A novel multi-tasks chain scheduling algorithm based on capacity prediction to solve AGV dispatching problem in an intelligent manufacturing system. *Journal of Manufacturing Systems, 68*, 130–144.

Ren, N., Zhao, Y., & Zhang, J. (2012). Scheduling research of AGV with double buffers based genetic algorithm in flexible manufacturing system. *Applied Mechanics and Materials*, *121*, 1630-1635.

Routray, M., & Ray, N. K. (2020). Remote homology detection using GA and NSGA-II on physicochemical properties. *International Journal of Computer Applications in Technology*, *64*(4), 393-402.

Singh, N., Dang, Q. V, Akcay, A., Adan, I., & Martagan, T. (2022). A matheuristic for AGV scheduling with battery constraints. *European Journal of Operational Research*, *298*(3), 855–873.

Singh, V., & Choudhary, S. (2009). Genetic algorithm for traveling salesman problem: using modified partially-mapped crossover operator. 2009 *International Conference on Multimedia, Signal Processing and Communication Technologies*, 20–23. https://doi.org/10.1109/MSPCT.2009.5164164

Song, J. (2021). Automatic guided vehicle global path planning considering multi-objective optimization and speed control. *Sensors and Materials*, *33*(6), 1999–2011.

Sun, P. Z., You, J., Qiu, S., Wu, E. Q., Xiong, P., Song, A., Zhang, H., & Lu, T. (2022). AGV-Based Vehicle Transportation in Automated Container Terminals: A Survey. *IEEE Transactions on Intelligent Transportation Systems*. https://doi.org/10.1109/TITS.2022.3215776.

Wang, C., Wang, L., Qin, J., Wu, Z., Duan, L., Li, Z., Cao, M., Ou, X., Su, X., Li, W., Lu, Z., Li, M., Wang, Y., Long, J., Huang, M., Li, Y., & Wang, Q. (2015). Path planning of automated guided vehicles based on improved a-star algorithm. *2015 IEEE International Conference on Information and Automation*, 2071–2076. https://doi.org/10.1109/ICInfA.2015.7279630.

Wang, Z., & Wu, Y. (2023). An Ant Colony Optimization-Simulated Annealing Algorithm for Solving a Multiload AGVs Workshop Scheduling Problem with Limited Buffer Capacity. *Processes*, *11*(3), 861.

Wang, Z., & Zeng, Q. (2022). A branch-and-bound approach for AGV dispatching and routing problems in automated container terminals. *Computers & Industrial Engineering*, *166*, 107968.

Wei, Q., Yan, Y., Zhang, J., Xiao, J., & Wang, C. (2022). A self-attention-based deep reinforcement learning approach for AGV dispatching systems. *IEEE Transactions on Neural Networks and Learning Systems*. https://doi.org/10.1109/TNNLS.2022.3222206.

Wu, S., Xiang, W., Li, W., Chen, L., & Wu, C. (2023). Dynamic Scheduling and Optimization of AGV in Factory Logistics Systems Based on Digital Twin. *Applied Sciences*, *13*(3), 1762.

Xu, L., Wang, Y., Liu, L., & Wang, J. (2016). Exact and Heuristic Algorithms for Routing AGV on Path with Precedence Constraints. *Mathematical Problems in Engineering*, *8*, 1-8.

Yao, F., Alkan, B., Ahmad, B., & Harrison, R. (2020). Improving just-in-time delivery performance of IoT-enabled flexible manufacturing systems with AGV based material transportation. *Sensors*, *20*(21), 6333.

Yuan, M. H., Li, Y. D., Pei, F. Q., & Gu, W. B. (2021). Dual-resource integrated scheduling method of AGV and machine in intelligent manufacturing job shop. *Journal of Central South University*, *28*(8), 2423-2435.

Yuan, Z., Yang, Z., Lv, L., & Shi, Y. (2020). A bi-level path planning algorithm for multi-AGV routing problem. *Electronics*, *9*(9), 1351.

Zhang, X. J., Sang, H. Y., Li, J. Q., Han, Y. Y., & Duan, P. (2022). An effective multi-AGVs dispatching method applied to matrix manufacturing workshop. *Computers & Industrial Engineering*, *163*, 107791.

Zou, W. Q., Pan, Q. K., Meng, T., Gao, L., & Wang, Y. L. (2020). An effective discrete artificial bee colony algorithm for multi-AGVs dispatching problem in a matrix manufacturing workshop. *Expert Systems with Applications*, *161*, 113675.

Zou, W. Q., Pan, Q. K., & Wang, L. (2021). An effective multi-objective evolutionary algorithm for solving the AGV scheduling problem with pickup and delivery. *Knowledge-Based Systems*, *218*, 106881.

Zou, W. Q., Pan, Q. K., & Tasgetiren, M. F. (2021). An effective iterated greedy algorithm for solving a multi-compartment AGV scheduling problem in a matrix manufacturing workshop. *Applied Soft Computing*, *99*, 106945.

Zou, W. Q., Pan, Q. K., Wang, L., Miao, Z. H., & Peng, C. (2022). Efficient multiobjective optimization for an AGV energy-efficient scheduling problem with release time. *Knowledge-Based Systems*, *242*, 108334.

Zou, W. Q., Pan, Q. K., Meng, L. L., Sang, H. Y., Han, Y. Y., & Li, J. Q. (2023). An effective self-adaptive iterated greedy algorithm for a multi-AGVs scheduling problem with charging and maintenance. *Expert Systems with Applications*, 119512.