Multidisciplinary | Rapid Review | Open Access Journal

Received January 24, 2020, accepted February 5, 2020, date of publication February 12, 2020, date of current version February 27, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2973336

An Effective Discrete Artificial Bee Colony Algorithm for Scheduling an Automatic-Guided-Vehicle in a Linear Manufacturing Workshop

WEN-QIANG ZOU[1], **QUAN-KE PAN**[1,2], **(Member, IEEE), AND M. FATIH TASGETIREN**[3]
[1]School of Mechatronic Engineering and Automation, Shanghai University, Shanghai 200072, China
[2]School of Computer Science, Liaocheng University, Liaocheng 252000, China
[3]International Logistics Management Department, Yaşar University, 35100 Izmir, Turkey

Corresponding author: Quan-ke Pan (panquanke@shu.edu.cn)

This research was partially supported by the National Science Foundation of China 61973203, 51575212, and Shanghai Key Laboratory of Power station Automation Technology.

ABSTRACT This paper deals with a new automatic guided vehicle (AGV) scheduling problem from the material handling process in a linear manufacturing workshop. The problem is to determine a sequence of Cells for AGV to travel to minimize the standard deviation of the waiting time of the Cells and the total travel distance of AGV. For this purpose, we first propose an integer linear programming model based on a comprehensive investigation. Then, we present an improved nearest-neighbor-based heuristic so as to fast generate a good solution in view of the problem-specific characteristics. Next, we propose an effective discrete artificial bee colony algorithm with some novel and advanced techniques including a heuristic-based initialization, six neighborhood structures and a new evolution strategy in the onlooker bee phase. Finally, the proposed algorithms are empirically evaluated based on several typical instances from the real-world linear manufacturing workshop. A comprehensive and thorough experiment shows that the presented algorithm produces superior results which are also demonstrated to be statistically significant than the existing algorithms.

INDEX TERMS Automated guided vehicle, heuristic, discrete artificial bee colony algorithm, scheduling, linear manufacturing workshop.

I. INTRODUCTION

Automatic Guided Vehicles (AGVs) are computer-controlled driverless vehicles that are used for transferring materials. Since AGVs were introduced in 1955, they have been successfully applied in many different applications such as warehouse, container terminals, transportation, and manufacturing systems [1]–[4]. Especially in recent years, with the rapid development of smart manufacturing industry, AGV is increasingly employed to transport materials in the manufacturing workshop due to its prominent features of simple operation, rapid response and high efficiency [2], [5]. In a linear manufacturing workshop, an AGV that transports materials for the designated cells under the control system's command starts from the warehouse, passes several cells, and finally returns to the warehouse after completing its mission. It can be concluded that it is a variant of the classical vehicle routing problem (VRP) in term of problem-specific characteristics. The effective scheduling of AGV can increase productivity and reduce the transportation cost [6]–[9]. Therefore, it is worthwhile for researchers to study an AGV scheduling problem (AGVSP) from a linear manufacturing workshop.

At present, the scheduling strategies adopted by almost all the plants are based on "First Come First Served (FCFS)", that is, the Cells that first send requests first obtain the service of AGV. This is not a reliable method because it may cause AGV to travel repeatedly from one end of the track to the other and make most of the time to be spent on the road. As a result, the production efficiency of enterprises is seriously affected and the transportation cost is considerably

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

increased. The AGVSP is an NP-hard problem [10], and it almost impossible to solve it by using exact solution methods in a limited amount of computing time [11]. Heuristic and meta-heuristic are the best choice for solving such a complex problem [12], [13] because they have been employed in many academic literatures. Zeng *et al.* [14] proposed a two-stage heuristic algorithm combining an improving timetabling method and a local search to solve the AGV scheduling problem transferring jobs between different machines by using a limited number of automated guided vehicles. Fazlollahtabar *et al.* [15] proposed an optimization method in two stages, namely searching the solution space and finding optimal solutions, to solve the scheduling problem for multiple automated guided vehicles in a manufacturing system. Miyamoto *et al.* [16] proposed the local/random search methods to address the dispatch and conflict-free routing problem of capacitated AGV systems. Saidi-Mehrabad *et al.* [17] proposed a two-stage ant colony algorithm to solve the job shop scheduling problem considering the transportation times of the jobs from one machine to another. Gen *et al.* [18] proposed a hybrid evolutionary algorithm to solve a variety of single or multi-objective scheduling problems in manufacturing systems to get the best solution with a smaller computational time. Yang *et al.* [2] proposed a rule-based bi-level genetic algorithm to solve the integrated scheduling problem of quay cranes, automated guided vehicles, and yard cranes in automated container terminals. Lyu *et al.* [19] proposed a genetic algorithm combined with the Dijkstra algorithm to solve the machine and AGV integrated scheduling problem in a flexible manufacturing system. Lu and Wang [20] designed a particle swarm optimization algorithm based on the graph theory model to solve the scheduling problem of two automated stacking cranes in an automated container terminal. Chen *et al.* [21] proposed a genetic algorithm to address the scheduling problem of a space-constrained AGV-based prefabricated bathroom units manufacturing system. Li *et al.* [22] proposed an improved harmony search algorithm to schedule AGVs to transfer production materials and cutting tools consumables in the manufacturing workshop.

As mentioned above, the AGVSP based on a linear manufacturing workshop is a variant of vehicle routing problem, so the methods for solving vehicle routing problems are highly appealed for solving the AGVSP under consideration. Andelmin and Bartolini [23] proposed a multi-start local search algorithm to solve the green vehicle routing problem. Yu *et al.* [24] developed a simulated annealing algorithm with a restart strategy to solve the hybrid vehicle routing problem. Poonthalir and Nadarajan [25] addressed a bi-objective fuel-efficient green vehicle routing problem with varying speed constraints by using particle swarm optimization with greedy mutation operator and time-varying acceleration coefficient. Gutierrez *et al.* [26] solved a vehicle routing problem with stochastic travel and service times by means of a multi-population memetic algorithm. Baradaran *et al.* [27] proposed a binary artificial bee colony (ABC) algorithm to address the vehicle routing problem with multiple hard prioritized time windows with a heterogeneous fleet of vehicles. Li *et al.* [28] proposed an improved ant colony optimization algorithm to solve the multi-depot green vehicle routing problem with multiple objectives. Various other methods that have been developed in existing literature are such as the evolutionary algorithms [29], genetic algorithm [30], tabu search algorithm [31], fruit fly optimization algorithm [32] and iterated local search algorithm [33].

From the short literature review above, we can see that the AGVSP is not only a very active research area but also the methods of solving AGVSP and VRP provide us with powerful references. Many studies on VRP take account into the travel distance (or travel time) as an objective because this criterion is important for evaluating the performance of the VRP. However, the travel distance (or travel time) is not the only factor that affects the performance of VRP, other factors such as the speed, load and type of the vehicle cannot be ignored. As for the AGVSP based on a linear manufacturing workshop, the standard deviation of the waiting time of material buffers and the total travel distance of AGV are two indicators for evaluating it, which respectively mean the overall capacity of CNC machines and the energy efficiency of AGV during the actual production. In the existing research, there is still no research on the AGVSP in a linear manufacturing workshop except for Ref. [10], so our research is to re-examine the AGVSP in linear manufacturing workshop in term of the VRP and find a more effective approach to solve it.

The discrete artificial bee colony (DABC) algorithm was first proposed by Ref. [34] for the lot-streaming flow-shop scheduling problem. The DABC algorithm is an extension of an artificial bee colony (ABC) algorithm that is developed by Ref. [35] to optimize multi-variable and multi-modal continuous functions. Many literatures have demonstrated that the performance of the ABC algorithm is competitive as other population-based algorithms [36]–[38]. Compared with the ABC algorithm, the DABC algorithm not only inherits the advantages of employing fewer control parameters but also makes up for its drawbacks of the discrete performance. To date, the DABC algorithm has been well applied to many practical application problems, such as flow shop problems [39], flexible job shop scheduling problems [40], reverse logistics problems [41] and traveling salesman problem [42].

This paper makes the following main contributions: First, we establish an integer linear programming model for the AGVSP in linear manufacturing workshop (hereafter called the AGVSP). Then, a constructive heuristic based on the problem-specific characteristics is presented to quickly generate a better solution. Next, we propose an effective discrete artificial bee colony algorithm, in which a solution generated by the proposed heuristic is used as an initial solution, six neighborhood operators are introduced to enhance its exploitation capability in the employed bee and onlooker bee phases and a new evolution strategy in the onlooker bee phase is presented to provide opportunities for further exploration to the potential solution.

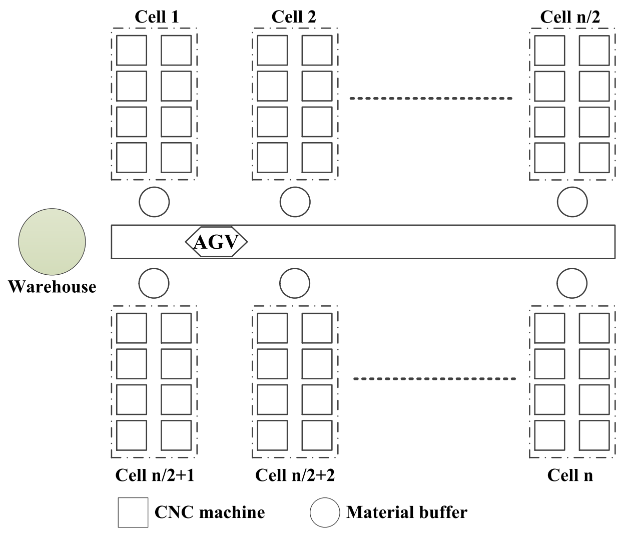


FIGURE 1. **The layout diagram of the linear manufacturing workshop.**

The rest of this paper is organized as follows. In section II, we descript the AGVSP and establish an integer linear programming model. Section III introduces the proposed heuristic in detail. In section IV, we present an effective DABC algorithm, whereas the experimental results and comparisons are reported in Section V. Finally, Section VI provides the concluding remarks and suggests some future work.

II. PROBLEM DESCRIPTION AND FORMULATION

A. PROBLEM DESCRIPTION

Fig. 1 shows the layout diagram in a linear manufacturing workshop, which is composed of an AGV and a number of Cells. Each Cell contains a material buffer and multiple computer numerical control (CNC) machines. The material buffer stores the materials for the CNC machines. Once the CNC machines are brought into production operations, the materials in the material buffer is constantly consumed by the CNC machines. When material buffer lacks the materials, the Cell sends a signal for replenishment to the control system. At this point, this Cell is called a Call Cell, and the time of sending a signal is called the Call time. After receiving the signal, the control system temporarily saves it. When a production cycle ends, the control system sorts all saved signals and dispatches AGV to the Call Cells in the generated sequence. The AGV departs from the warehouse and travels on the channel to the destination Cell. At the destination Cells, the AGV unloads the materials and then travels to other destination Cells, and finally returns to the warehouse.

The CNC machines and AGV are two key devices of the manufacturing system. Their efficient operation will improve the productivity of enterprises. The standard deviation of the waiting time of Call Cells is one of the most important factors determining the overall capacity of CNC machines. The total travel distance of AGV can directly reflect the efficiency of AGV. For solving the problem, we assume that all the devices operate normally, and there will be no shutdown, malfunction, and other accidents. The materials are stored in the warehouse. The velocity of AGV keeps constant. The AGV can travel forward and backward on the linear channel. The traveling path of AGV (*i.e.*, AGV route) originates and terminates at the warehouse. Each Call Cell is only visited once by the AGV. In this paper, we only study the Call Cells in one production cycle. The aim is to determine a sequence of scheduling AGV to these Call Cells with the objective of minimizing the objective function value. The objective function considers two indicators namely the standard deviation of the waiting time of Call Cells and the total travel distance of AGV.

B. PROBLEM FORMULATION

In this section, we introduce the parameters and decision variables employed in the model. The parameters and decision variables are defined as follows.

Parameters and constants:

i, j	the index number of Call Cell.
n	the total number of Call Cells.
p_i	the position of Call Cell i.
d_{ij}	the distance between Call Cell i and j.
t_{ij}	the travel time between Call Cell i and j.
v	the velocity of AGV.
t_i^c	the Call time of Call Cell i.
t_u	the unloading time of Call Cell.
t_r	the running time of the algorithm.
C	a production cycle.
f_1	the standard deviation of the waiting time of Call Cells.
f_2	the total travel distance of AGV.
w_1	the weight of f_1.
w_2	The weight of f_2.

Decision variables:

x_{ij}	$x_{ij} = 1$ if an arc (i, j) is traveled by AGV and 0 otherwise.
t_i^w	The waiting time of Call Cell i.

Let $G = \{V, E\}$ be a directed graph, where $V = \{0, 1, \ldots, n\}$ represents the set of vertices and $E = \{(i, j) | i, j \in V, i \neq j\}$ is the set of edges between each pair of vertices. Vertices $i = 1, \ldots, n$ represent the Call Cells, whereas vertex 0 denotes the warehouse. The d_{ij} (*i.e.*, edge (i, j)) and t_{ij} can be calculated by the following formula:

$$d_{ij} = |p_i - p_j| \tag{1}$$

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

Objective function:

$$\min \mathrm{F} = w_1 f_1 + w_2 f_2 \tag{3}$$

$$s.t: \sum_{i=0}^{n} x_{ij} = 1, \forall j \in V \backslash \{0\} \tag{4}$$

$$\sum_{j=0}^{n} x_{ij} = 1, \forall i \in V \backslash \{0\} \tag{5}$$

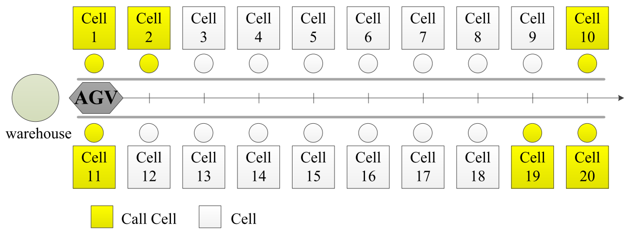


FIGURE 2. The distribution of the Call Cells in linear manufacturing workshop.

$$\sum_{i=1}^{n} x_{i0} = \sum_{j=1}^{n} x_{0j} = 1 \tag{6}$$

$$\sum_{i=0}^{n} x_{ij} - \sum_{i=0}^{n} x_{ji} = 0, \forall j \in V \backslash \{0\} \tag{7}$$

$$x_{ij}\left[t_i^c - t_j^c + t_{ij} + t_u - (t_j^w - t_i^w)\right] = 0, \forall i, j \in V \backslash \{0\} \tag{8}$$

$$t_1^w = C - t_1^c + t_r + t_{01} + t_u \tag{9}$$

$$f_1 = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(t_i^w - \frac{1}{n}\sum_{i=1}^{n} t_i^w)^2} \tag{10}$$

$$f_2 = \sum_{j=0}^{n}\sum_{i=0}^{n} d_{ij} \tag{11}$$

$$t_i^w > 0 \tag{12}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in V \tag{13}$$

As mentioned above, the objective function (3) is to obtain solutions with the minimum value of two indicators namely the total travel distance of AGV and the standard deviation of the waiting time of Call Cells. The constraints (4-6) impose that each Call Cell must be visited exactly once. The constraints (7) indicate that the route starts and ends at the warehouse. The constraints (8-9) represent the relationship of the waiting time between a Call Cell and its predecessor. The constraints (10-11) respectively present two indicators. The time constraint is represented by (12) and the constraints (13) impose restrictions on the decision variables.

III. THE PROPOSED HEURISTIC

It is difficult to obtain some good solutions by using an exact optimization method in a reasonable amount of computation time. In this section, we first detail the solution representation and then propose a constructive heuristic based on the problem-specific characteristics.

A. SOLUTION REPRESENTATION

To maintain the simplicity of the algorithm, a rather straightforward solution representation scheme is applied. Let us remind that the Cell that sends a signal for material replenishment to the control system is called the Call Cell. Suppose that there are m Call Cells in linear manufacturing workshop. The representation has the form of a vector of length m. In the vector, there are m integers between 1 and m inclusively representing the identity of Call Cells. Fig.2 shows the distribution of 6 Call Cells in a linear manufacturing workshop with 20 Cells. The instance information of 6 Call Cells is given in Table 1. As shown in Fig.2, assuming that the sequence of 6 Call Cells to be visited by AGV is (1, 10, 11, 19, 2, 20), its solution representation is represented as (4, 1, 2, 3, 6, 5) according to the instance information in Table 1.

TABLE 1. The instance information for call cells.

Identity	1	2	3	4	5	6
Cell number	10	11	19	1	20	2

B. PROPOSED INNH HEURISTIC

The nearest neighbor based heuristic (NNH) is generally used to solve the vehicle routing problem (VRP) [43]. It refers to that the vehicle first puts the warehouse as the starting point, searches for the customer closest to itself as the next starting point, and the rest may be deduced by analogy until the last customer is reached. However, For the AGVSP, we should not only consider the travel distance of AGV but also take into account the Call Time of Call Cells. Combined with the analysis of the FCFS method and the understanding of the problem, the Call Time may also be an important evaluation indicator. Therefore, we propose an improved nearest neighbor based heuristic (INNH), in which the evaluation is not the travel distance but the index function value we formulated. The index function is as follows:

$$F_i = \varphi \cdot d_{ij} + (1 - \varphi) \cdot T_i^c, \quad \varphi \in [0, 1] \tag{14}$$

where F_i denotes the index function value of arriving at the Call Cell i, and φ represents the weight parameter.

Algorithm 1 INNH Heuristic

```
Input: the set of Call Cells, $U = \{c_1, c_2, \ldots, c_n\}$
Output: the solution, $\pi = \{\pi(1), \pi(2), \ldots, \pi(n)\}$
01: Begin
02:     Let position $p = 1$ and Call Cell $j = 1$
        ($i.e.$,warehouse)
03:     While $U$ is not empty do
04:         For $i = 1$ to $size\ of(U)$
05:             Calculate $F_i$ from Call Cell $j$ to Call Cell $i$
06:         End for
07:         $F_{\min} = \min_{i=1,2,\ldots,N}(F_i)$, $N = size\ of\ (U)$
08:         Insert Call Cell $x$ with $F_{\min}$ into the $p$th position
            of sequence $\pi$
09:         Let $p = p + 1$ and Call Cell $j = x$
10:         Delete Call Cell $x$ from set $U$
11:     End While
12:     Output solution $\pi$
13: End
```

The INNH heuristic constructs a solution by inserting Call Cells one by one into the current sequence. The detailed procedure is as follows: Firstly, the procedure takes the warehouse as the starting point of AGV, evaluates the index function value between each Call Cell and the starting point by formula (14), and puts the Call Cell with the smallest index function value in the first position of the current sequence. Then, the procedure continues to find the second Call Cell in the same way as the previous Call Cell and inserts it into the second position of the current sequence. Finally, the procedure stops until all Call Cells have been chosen. In the above scheduling process, the set of Call Cells is denoted by $U = \{c_1, c_2, \ldots, c_n\}$, where n represents the number of Call Cells. The current sequence (*i.e.*, solution) is represented by $\pi = \{\pi(1), \pi(2), \ldots, \pi(n)\}$, where $\pi(1)$ refers to the Call Cell with the smallest index function value. The pseudo-code of INNH heuristic is depicted in Algorithm 1.

IV. THE PROPOSED HEURISTIC

The discrete artificial bee colony (DABC) algorithm was first proposed by Ref. [34] for discrete optimization problems. In the DABC algorithm, its procedure is divided into four phases: initial population, employed bees, onlookers and scouts. When the algorithm starts, it generates several food sources (solution) by a certain rule in the initial population phase and assigns each employed bee to a food source. Then these food sources will be updated iteratively by the following three phases. Employed bees are responsible for exploiting potential food sources near its originally assigned food sources (old), if it finds a new food source with more nectar amount (fitness) than the old one, then the old one is replaced by the new one. In the onlooker bee phase, the onlookers will further explore the food sources shared by employed bees. If a food source has not been improved, it is abandoned by its employed bee, this employed bee becomes a scout bee that starts to search for a new food source near the hive. Then, the scout bee finding a new food source becomes an employed bee again. A new iteration of the DABC algorithm starts. The above process is repeated until a termination condition is satisfied. The detailed designs for the proposed DABC algorithm applied to the AGVSP are as follows.

A. INITIAL POPULATION PHASE

An initial population with a high level of quality and diversity always leads to outstanding outcomes. Many works of literature construct high-quality initial solutions by adopting effective heuristics, whereas other solutions are randomly generated to preserve the diversity of the initial population [44]–[46]. To solve the AGVSP, we will construct PS initial solutions, *i.e.*, $X = \{\pi_1, \pi_2, \ldots, \pi_{PS}\}$. In view of the above excellent achievements for the proposed heuristic in Section III.B, we present a simple initialization procedure as shown in Algorithm 2.

B. NEIGHBORHOOD OPERATOR

A neighborhood operator is used to get a new solution near the current solution. Different neighborhood operators play different roles in the exploration and exploitation of the proposed DABC algorithm. We consider six neighborhood operators as follows.

Algorithm 2 Initial Population

```
01: Begin
02:     Step 1: Generate a solution by using the INNH
            heuristic in Section III.B. Let counter $\alpha = 1$.
03:     Step 2: If $\alpha = PS$, go to Step 4; otherwise,
            randomly generate a solution.
04:     Step 3: If the generated solution is different from
            all of the existing solutions, place it into the
            initial population and let $\alpha = \alpha + 1$;
            otherwise, discard it.
05:     Step 4: Stop the procedure and output the initial
            solutions $X$.
06: End
```

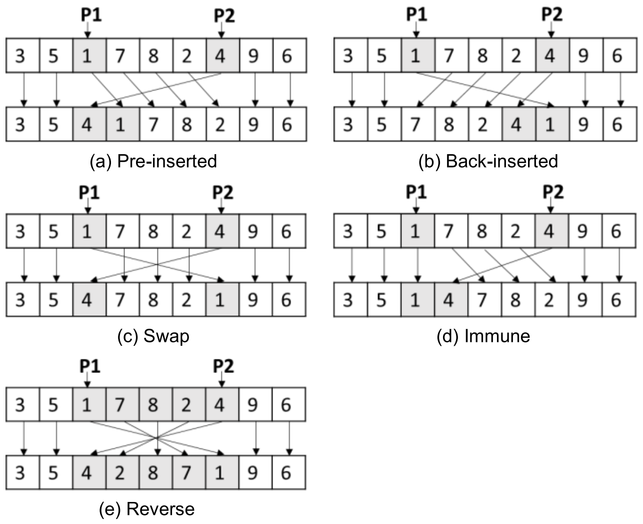


FIGURE 3. Neighborhood operator.

(1) Insertion

We randomly select two locations, namely P1 and P2 (suppose P1<P2), from the sequence of the current solution. Then, a random number m is generated between 0 and 1. If m<0.5, the Call Cell in the P2 is extracted from P2 and reinserted into location P1 (see Fig.3(a)); otherwise the Call Cell in the P1 is extracted from P1 and reinserted into location P2 (see Fig.3(b)).

(2) Swap

Two locations are randomly selected from the sequence of the current solution. The Call Cells in the two locations are exchanged (see Fig.3(c)).

(3) Immune

A location P1 is randomly selected from the sequence of the current solution. Suppose the Call Cell in location P1 is A. Then find the Call Cell B that has the shortest distance from A among the rest Call Cells. Next, extract B from its original location and reinsert it into location P1+1. (see Fig. 3(d)).

(4) Reverse

Two locations are randomly selected from the sequence of the current solution. The Call Cells between location P1 and P2 are reversed (see Fig. 3(e)).

(5) Two insert neighborhood operator

The insert operator is performed two times to the current solution.

(6) Two swap neighborhood operator

The new solution is obtained by applying the swap neighborhood operator two times to the current solution.

C. EMPLOYED BEE PHASE

All initial solutions are assigned to employed bees, then employed bees adopt a self-adaptive strategy to look for new solutions around their current solutions. At the beginning, an initial neighbor list (NL) with a specified length is generated by filling the list one by one randomly from six neighboring operators explained before. Then, the DABC algorithm is started. The current solution is assumed to be π_i, one operator from the NL is taken out and used to generate a new solution π_{new} during the evolution process. If the new solution π_{new} is better than the current solution π_i, then the current solution π_i is replaced by the new solution π_{new}, this operator is added to a winning neighboring list (WNL), and the counter cnt_i is set to 0, where cnt_i is used to count the number of times that π_i has not been improved. Otherwise, the new solution π_{new} is rejected and let $cnt_i = cnt_i + 1$. Once the NL is empty, it will be refilled by the following method: 75% of the NL is refilled from the WNL list, and then the rest of 25% is refilled by a random selection from six different operators. The above process is repeated until a termination criterion is satisfied. As a result, the proper operators can be gradually learned by the algorithm itself to suit to the problem under consideration. We assume that the updated initial population is X' and the best solution in X' is π_{best}. The pseudo-codefor the above procedure is shown in Algorithm 3.

D. ONLOOKER BEE PHASE

In this phase, all employed bees share the current solutions with the onlookers. To drive the selection process towards better solutions, each onlooker looks for a solution π_j by implementing the tournament selection method. Like the employed bee, the onlookers also adopts the same self-adaptive strategy. The solution π_j applies a operator that is taken out from the NL and generates a new solution π_{new}. If the new solution π_{new} is better than the solution π_k with the maximum cnt in the current population, then the solution π_k will be replaced by the new solution π_{new}, this operator is added to a winning neighboring list (WNL), and the counter cnt_k is set to 0. Otherwise, the new solution π_{new} is rejected and let $cnt_i = cnt_i + 1$. In this way, the potential solutions will be obtained more opportunities to be explored. Finally, the procedure updates the population and the best solution again. Let the population after updating be X''. The pseudo-code for this procedure is given in Algorithm 4.

E. SCOUT BEE PHASE

The employed bee becomes a scout if the current solution π_i has not been improved in a number θ of successive iterations.

Algorithm 3 Employed Bee

```
01:  Input: initial population, $X$
02:  Output: updated population and best solution $\pi_{best}$
03:  Begin
04:      For $i = 1$ to $PS$
05:          Take out a neighborhood operator from the NL
06:          Perform this operator and yield a new solution $\pi_{new}$
07:          If the fitness of $\pi_{new}$ is better than that of $\pi_i$ Then
08:              Update the current solution $\pi_i$
09:              Let $\pi_i = \pi_{new}$ and $cnt_i = 0$
10:              Add this operator to the WNL
11:          Else
12:              Let $cnt_i = cnt_i + 1$
13:          End if
14:          If the fitness of $\pi_{new}$ is better than that of $\pi_{best}$ Then
15:              Update the best solution $\pi_{best} = \pi_{new}$
16:          End if
17:      End for
18:  End
```

In order to maintain the diversity of the population and avoid the algorithm trap into a local optimum, the current solution π_i is replaced by a randomly generated solution.

F. FRAMEWORK OF THE PROPOSED ALGORITHM

After describing each component in regard to the proposed DABC algorithm, the complete steps are shown in Fig.4.

V. EXPERIMENTAL RESULTS

In order to verify the effectiveness of proposed model and evaluate the performance of the proposed INNH heuristic and DABC algorithm, we proceed with a comprehensive computational evaluation in which two types of instances are tested: one is a simple instance (Case 0) shown in Table 1, and the other is three instances (Case 1-3) described in Ref. [10]. Detailed information for Case 0-4 are reported in Appendix 1. The instances on Case 1-3 are from linear manufacturing workshop with 1 AGV and 30 Cells, the layout diagram of which is as shown in Fig.5. The proposed algorithms are coded in C++ programming language, and all experiments are implemented on an Intel Core i7-2620M 2.70 GHz PC with 8 GB memory in a Windows 10 Operation System. The proposed model is carried out in python programming language and solved by Gurobi 8.1.0 Solver.

A. EXPERIMENTAL SETTINGS

Case 0 is adopted to verify the effectiveness of the proposed model, while Case 1-3 are used to evaluate the performance of the proposed algorithm. To verify the effectiveness of the proposed model, we do an experimental evaluation in regard

Algorithm 4 Onlooker

```
01: Input: population, $X'$
02: Output: updated population $X''$ and best solution $\pi_{best}$
03: Begin
04:     For $i = 1$ to $PS$
05:         Select a solution $\pi_j$ by the tournament selection
06:         Take out a neighborhood operator from the NL
07:         Perform this operator and yield a new solution $\pi_{new}$
08:         Search for a solution $\pi_k$ with the maximum $cnt$
09:         If the fitness of $\pi_{new}$ is better than that of $\pi_k$ Then
10:             Replace $\pi_k$ with $\pi_{new}$ and let $cnt_k = 0$
11:             Add this operator to the WNL
12:         Else
13:             Let $cnt_i = cnt_i + 1$
14:         End if
15:         If the fitness of $\pi_{new}$ is better than that of $\pi_{best}$ Then
16:             Update the best solution $\pi_{best} = \pi_{new}$
17:         End if
18:     End for
19: End
```

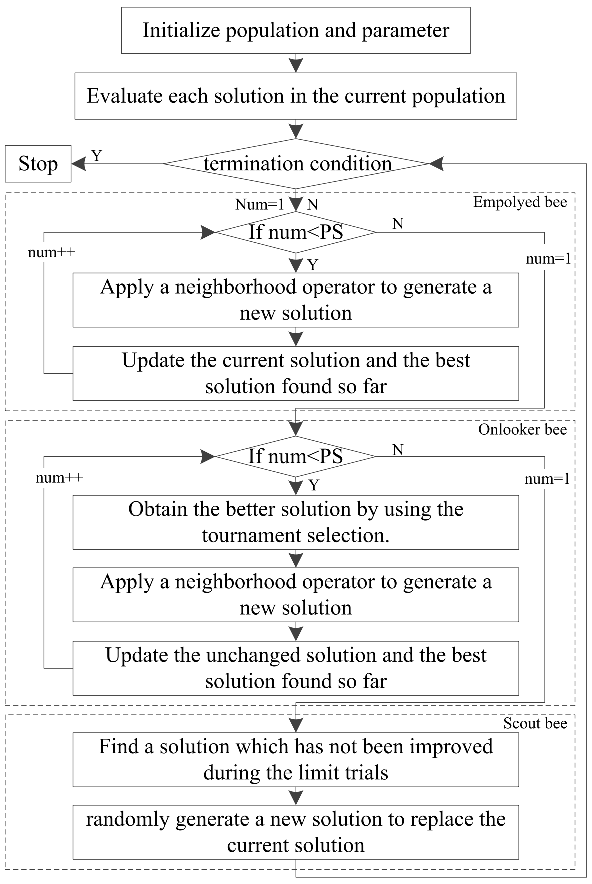

Initialize population and parameter
Evaluate each solution in the current population
Stop
Y
termination condition
Num=1
N
Empolyed bee
If num<PS
N
num++
Y
num=1
Apply a neighborhood operator to generate a new solution
Update the current solution and the best solution found so far
Onlooker bee
If num<PS
N
num++
Y
num=1
Obtain the better solution by using the tournament selection.
Apply a neighborhood operator to generate a new solution
Update the unchanged solution and the best solution found so far
Scout bee
Find a solution which has not been improved during the limit trials
randomly generate a new solution to replace the current solution

FIGURE 4. Flowchart of the proposed DABC algorithm.

to the comparison of INNH, FCFS and GUROBI, where GUROBI represents Gurobi 8.1.0 Solver. We also determine the weight parameter values when tests are implemented on Case 1-3. To evaluate the performance of DABC algorithm with different neighborhood operators, we compare the DABC algorithm with only one neighborhood operator and the DABC algorithm with six neighborhood operators. We also compare the DABC algorithm with and without the INNH heuristic in the initial population phase. Moreover, the DABC algorithm is compared with the existing algorithms, *i.e.*, the particle swarm optimization algorithm (PSO), the genetic algorithm (GA), the memetic algorithm (MA), the harmony search algorithm (HS) and the improved harmony search algorithm (IHS). Note that the control system has to generate a feasible solution within 10 seconds calculation time according to the production standard of manufacturing workshop, otherwise it will seriously affect the production plan of the manufacturing workshop. That is, for each qualified competing algorithm, its running time cannot exceed 10 seconds.

B. PARAMETER SETTINGS

To obtain algorithms with better performance, we need to determine the parameters for the proposed and competing algorithms. For the competing algorithms, their parameters are quoted from Ref. [10]. For the DABC algorithm, it has two parameters (controlled factors) to calibrate after the preliminary experiments: the population size (PS), tested at four levels: $\{5, 10, 15\}$ and the predetermined number of trials (θ), tested at four levels: $\{150, 200, 250\}$, and a response variable: the average fitness value. Cases 1-3 are used as calibration instances. As a result, we obtain the best configuration of the above two parameters through a full factorial Design of Experiments for a total of 3×3 configurations, which is respectively $PS = 10$ and $\theta = 200$. Therefore, the parameters of the proposed and competing algorithms are obtained as shown in Table 2.

The experimental parameters are set as follows: AGV velocity (v), unloading time (t_u), running time of the algorithm (t_r), production cycle (C), weight of the index function (φ), weight of f_1 (w_1), weight of f_2 (w_2) and the iteration (FEs), are set to 0.45, 30, 10, 1000, 0.7, 0.7, 0.3 and 10,000. For each Case, the parameters of Call Cell i include the number, Call time and location. The detailed information are available in Appendix 1.

C. RESULTS AND ANALYSIS

1) COMPARISION OF INNH, FCFS AND GUROBI

In this section, we first implement some preliminary tests on Case 0 so as to determine the optimal weight value of the index function and verify the effectiveness of the proposed

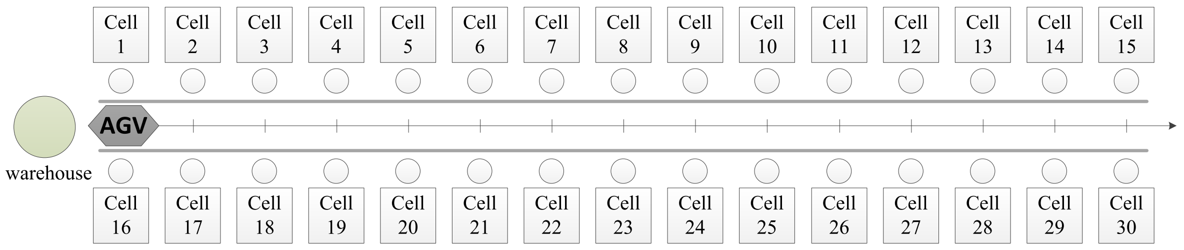


FIGURE 5. Diagram of the linear manufacturing workshop.

TABLE 2. Parameter settings.

Algorithm	Parameter
DABC	PS=10, θ=200
IHS	HMS=6, HMCR$_{min}$=0.2, HMCR$_{max}$=0.99
HS	HMS=6, HMCR=0.95, PAR=0.05
PSO	PS=100, $\omega_{max}=0.9$, $\omega_{min}=0.9$, $c_1=0.9$, $c_2=0.1$
GA	PS=100, $P_c=0.9$, $P_m=0.01$
MA	PS=100, $P_c=0.9$, $P_m=0.01$

DABC: discrete artificial bee colony algorithm; PS: population size; θ: predetermined number of trials; IHS: improved harmony search algorithm; HMS: harmony memory size; HMCR: harmony memory considering rate; PAR: pitch adjusting rate; HS: harmony search algorithm; PSO: particle swarm optimization algorithm; GA: genetic algorithm; MA: memetic algorithm; ω_{max} : upper limit of inertia weight; ω_{min} : lower limit of inertia weight; c_1 , c_2 : acceleration factors; P_c : crossover probability; P_m : mutation probability.

TABLE 3. Comparison of INNH and FCFS.

Methods	The fitness value	Standard deviation of waiting time of Call Cells	Total travel distance of AGV(m)
INNH	90.53	79.83	115.5
FCFS	175.91	131.09	280.5
GUROBI	90.53	79.83	115.5

model. However, other instances such as Case 1, Case 2 and Case 3 are not considered because they cannot be solved within 10 seconds by using Gurobi 8.1.0 Solver. Recall that 10 seconds refers to the running time of algorithms. The optimal weight value of the index function is tested as 0.7 on Case 0. The effectiveness of the proposed model will be demonstrated by the results in Table 3, where we compare three indicators: the fitness value, the standard deviation of waiting time of Call Cells and the total travel distance of AGV.

Table 3 shows the comparison results among INNH, FCFS, and GUROBI on Case 0 under the same experimental environment. From Table 3, we can see that: (1) the total travel distance of AGV obtained by INNH is equal to 115.5, much better than the value of 280.5 gained by FCFS, indicating that INNH shortens the total travel distance of AGV and improves the energy efficiency of AGV. (2) The waiting-time standard deviation of INNH is 79.83, far superior to the value of 131.09 gained by FCFS, which means that the overall capacity of CNC machines is increased by using INNH.

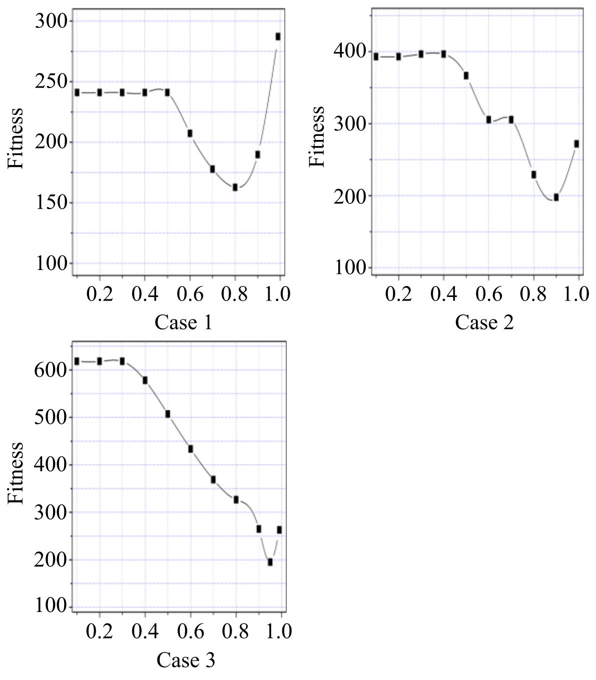


FIGURE 6. Determination of the optimal weight parameter φ.

(3) The fitness values obtained by INNH and FCFS is 90.53 and 175.91, respectively, the results produced by INNH over FCFS have decreased by 48.54%. (4) GUROBI get the same excellent results as INNH in three indicators in table 3, which not only proves the effectiveness of the proposed model but also proves the high performance of INNH.

2) DETERMINATION OF WEIGHT PARAMETER

The INNH has one parameter that might affect its performance. To determine the optimal value of weight parameter φ on each Case, we evaluate the performance of INNH by using different values φ in the interval [0.01,1]. It is found that the performance of INNH is sensitive over this range on Case 1 to 3. Fig.6 gives that the performance of INNH varies significantly in this interval for the instances on Case 1 to 3. From the experimental results, we can see that: (1) the optimal value of weight parameter φ on Case 1 is equal to 0.8, (2) the optimal value of the weight parameter φ on Case 2 corresponds to 0.9, and (3) when the number of Call Cells increases to 25, the optimal value of weight parameter φ on Case 3 becomes 0.95.

TABLE 4. Comparison of DABC and NDABC.

Instance	Standard deviation of waiting time of Call Cells		the total travel distance of AGV		the best fitness value	
	NDABC	DABC	NDABC	DABC	NDABC	DABC
Case 1	166.55	82.31	292.16	300.52	204.23	147.67
Case 2	169.58	152.7	161.7	115.5	167.21	141.54
Case 3	205.24	150.15	244.64	204.41	217.06	166.43
Mean	180.45	**128.39**	232.83	**206.81**	196.17	**151.88**

TABLE 5. Comparison of DABC and its variants.

Algorithm	Case 1			Case 2			Case 3		
	STD	TD	BFit	STD	TD	BFit	STD	TD	BFit
$DABC_1$	83.03	298.54	147.68	132.39	231	161.97	170.81	231	188.87
$DABC_2$	109.19	253.22	152.40	143.24	229.68	169.17	170.85	231	188.89
$DABC_3$	116.29	258.50	158.95	135.83	241.56	167.55	174.83	242	194.98
$DABC_4$	81.32	302.50	147.67	138.50	231	166.25	170.81	231	188.87
$DABC_5$	93.43	277.86	148.76	135.63	232.76	164.77	198.20	196.68	197.74
$DABC_6$	99.24	266.20	149.33	142.79	234.96	170.44	189.31	243.32	205.51
DABC	82.31	300.52	**147.67**	152.70	115.50	**141.54**	150.15	204.41	**166.43**
Mean	94.97	279.62	150.35	140.15	216.64	163.10	174.99	225.63	190.18

3) EFFICIENCY OF INNH HEURISTIC

To check the effectiveness of the proposed INNH heuristic, we implement a compared experiment between the proposed DABC and NDABC, *i.e.*, DABC for the algorithm with the INNH, and NDABC for the algorithm without the INNH. Two compared algorithms with the same parameters are implemented on the same experimental environment. The experimental results obtained such as the standard deviation of waiting time of Call Cells, the total travel distance of AGV and the best fitness value are collected for comparison, which are shown in Table 4.

It can be observed from Table 4 that: (1) the best fitness values gained by DABC on Case 1-3 are 147.67, 141.54 and 166.43, much better than the values of 204.23,167.21 and 217.06 gained by NDABC. (2) For the waiting-time standard deviation of Call Cells, it also shows similar characteristics to the best fitness value. But (3) for the total travel distance of AGV, except for Case 1, the results obtained by DABC are also superior to those gained by NDABC, indicating that the DABC sometimes has to sacrifice the energy efficiency of AGV to improve the overall capacity of CNC machines. In conclusion, the comparision results illustrate that the effect of INNH is remarkable for the proposed DABC algorithm.

4) EFFICIENCY OF PROPOSED NEIGHBORHOOD OPERATOR

To evaluate the effectiveness of the proposed neighborhood operators, we have tested six variants of the DABC algorithm on Case 1-3 by considering each neighborhood operator at a time, including insert, swap, immune, reverse, two-insert and two-swap neighborhood operators. The above variants are represented by $DABC_i$ ($i \in \{1, 2, \ldots, 6\}$), respectively. The DABC algorithm adopts a self-adaptive strategy concerning six neighboring operators, and the detailed procedure is shown in Algorithm 3 and Algorithm 4. In our experiments, the length of NL is set as 20. The experimental results including the standard deviation of the waiting time of Call Cells (STD), the total travel distance of AGV (TD) and the best fitness value (BFit) are reported in Table 5.

Table 5 reports the comparison results for DABC and its variants under the same experimental environment. It can be seen from Table 5 that the BFit values obtained by $DABC_1$ are respectively 147.68, 161.97 and 188.87 on Case 1, Case 2 and Case 3. Except that 147.68 is slightly less than 147.67 of $DABC_4$, its other Bfit values are the best of the six variants of DABC. It illustrates that the insert operator plays a key role in DABC. For $DABC_4$, its Bfit values on Case 1 and Case 3 are respectively 147.67 and 188.87, which are the best values in the corresponding Cases. Even if the Bfit values on Case 2 is not the best, its contribution to DABC cannot be denied. Hence, the reverse operator should also be an important part of DABC. $DABC_5$ performs better than $DABC_4$ on Case 2, while $DABC_3$ performs better than $DABC_5$ on Case 3 and $DABC_6$ performs better than $DABC_3$ and $DABC_2$ on Case 1, which indicate that the swap, immune, two-insert and two-swap neighborhood operators have different contributions for different Cases. For DABC, we observe from table 5 that the BFit values obtained by DABC on Case 1, Case 2 and Case 3 are respectively 147.67, 141.54 and 166.43, which are the best of all variants of DABC and improved by 1.78%, 13.22% and 12.49% compared with the corresponding mean BFit value. It means that DABC is the most effective algorithm to solve the AGVSP problem, especially for the instances with more Call Cells, such as Case 2 and Case 3.

5) COMPARISION OF DABC AND EXISTING ALGORITHMS

To evaluate the performance of the proposed algorithm, we compare it with the other five algorithms such as the PSO, GA, MA, HS and IHS algorithms [10]. For each instance on Case 1 to 3, each of the above algorithms is independently run 25 times, and each run carries out 10,000 iterations,

TABLE 6. Comparison of algorithms for Case 1.

Methods	Standard deviation of waiting time of Call Cells	Average waiting time (s)	Total travel distance of AGV(m)
DABC	**82.31**	**1035.83**	**300.52**
IHS	149.71	1146.89	313.06
HS	156.74	1155.43	316.58
PSO	228.28	1155.56	346.28
MA	154.56	1206.63	317.46
GA	150.96	1174.07	326.26
FCFS	150.77	1404.37	555.5

TABLE 7. Comparison of algorithms for Case 2.

Methods	Standard deviation of waiting time of Call Cells	Average waiting time (s)	Total travel distance of AGV(m)
DABC	**152.7**	**1026.64**	**115.5**
IHS	161.96	1061.03	138.38
HS	186.57	1143.38	180.18
PSO	326.83	1292.59	314.38
MA	215.99	1179.49	208.34
GA	194.4	1158.83	216.7
FCFS	256.22	1596.76	594

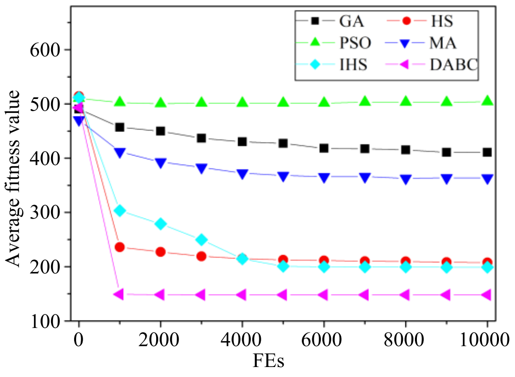
TABLE 8. Comparison of algorithms for Case 3.

Methods	Standard deviation of waiting time of Call Cells	Average waiting time (s)	Total travel distance of AGV(m)
DABC	**150.15**	**1228.16**	**204.41**
IHS	160.85	1277.79	223.08
HS	228.17	1323.59	253
PSO	439.59	1491.77	436.7
MA	250.12	1325.58	284.24
GA	236.75	1308.04	272.58
FCFS	388.01	1780.56	687.5

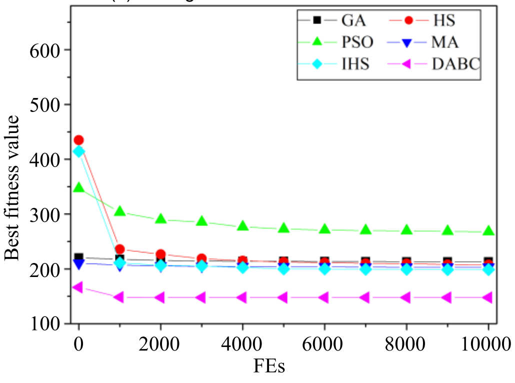
so there are $25 \times 10000 = 250000$ results in total for each Case. In fairness, the experimental results such as the average fitness value, the standard deviation of waiting time of Call Cells, the average waiting time and the total travel distance of AGV are the average value for 250000 iterations of each instance. The best fitness value is obtained by the best value of 250000 iterations of each instance. The experiments are implemented in same environment. The running time of each competing algorithm cannot exceed 10 seconds. The experimental results including the standard deviation of the waiting time of Call Cells, the average waiting time and the total travel distance of AGV are reported in Table 6-8. Whereas, Fig.7 to 9 shows the plot of the average fitness value and the best fitness value found by the competing algorithms on Case 1 to 3.

- Case 1

In this section, we test Case 1 with 15 Call Cells. The experimental results are obtained within 10 seconds. Fig.7(a) shows the plot of the average fitness value gained by the competing algorithms. It can be seen from the figure that DABC converges faster than the other five, and the overall



(a) Average fitness value versus FEs

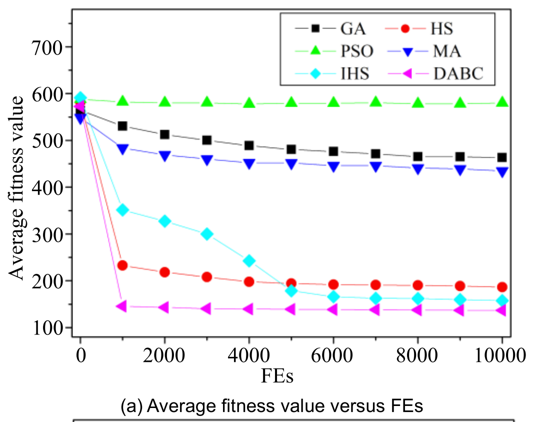


(b) Best fitness value versus FEs

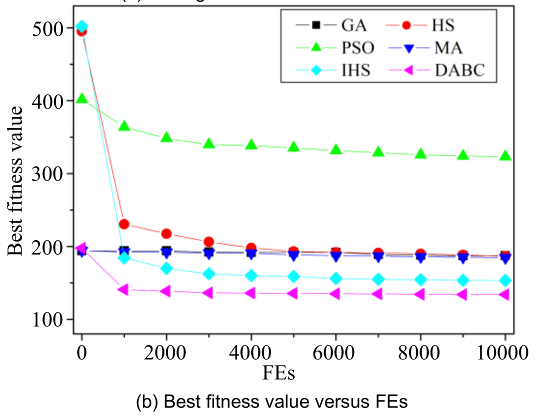
FIGURE 7. Convergence process for the competing algorithms on Case 1.

quality is also good as reflected by the average fitness value, which is the lowest among six algorithms. Around 1000 iterations, PSO has obtained the minimum average fitness value. However, DABC produces better results compared with PSO. Fig.7(b) illustrates the plot of the best fitness value gained by the competing algorithms. From the figure, we can see that the best fitness value obtained by DABC is far superior to the other competing algorithms at the beginning of the iteration, the reason is that the proposed heuristic generates a high-quality solution in initial population phase of DABC. Moreover, the convergence of these algorithms starts to slow down after 5000 iterations. However, DABC converges around 1000 iterations and obtains a better best fitness value than other five algorithms. Therefore, DABC performs the best than other competing algorithms on Case 1.

The standard deviation of the waiting time of Call Cells, the average waiting time of Call Cells and the total travel distance of AGV are the three main indicators for evaluating the performance of the competing algorithms, the statistical results of which are illustrated in Table 6. Here, we have not only compared with the competing algorithms but also compared with FCFS commonly used in the plant. It can be observed from Table 6 that the results gained by DABC are 82.31, 1035.83 and 300.52 respectively, much better those gained by other competing algorithms. Compared with those obtained by IHS and FCFS, the results obtained by DABC has decreased by 45.02% and 45.41% for the waiting-time standard deviation of Call Cells, indicating that the stability

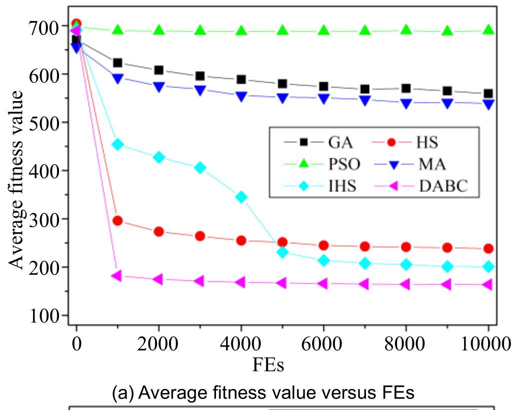


(a) Average fitness value versus FEs

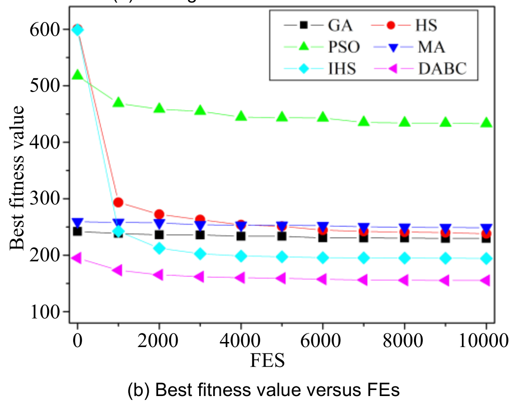


(b) Best fitness value versus FEs

FIGURE 8. Convergence process for the competing algorithms on Case 2.



(a) Average fitness value versus FEs



(b) Best fitness value versus FEs

FIGURE 9. Convergence process for the competing algorithms on Case 3.

of AGV transferring materials to Call Cells has a significant improvement. For the average waiting time of Call Cells, the results of DABC have decreased by 9.68% and 26.24% than those of IHS and FCFS, whereas, for the total travel distance of AGV, the results obtained by DABC has decreased by 4% and 45.9% than those obtained by IHS and FCFS, which means that DABC can shorten AGV travel distance and improve the energy efficiency of AGV.

For Case 1, the best sequence corresponding to the optimal solution obtained by DABC algorithm is {1 16 2 21 28 8 3 5 15 30 18 19 6 9 12}.

- Case 2

In this section, we test Case 2 with 20 Call Cells. The experimental results are obtained within 10 seconds. Fig.8(a) and Fig.8(b) show the plot of the average fitness value and the best fitness value for the competing algorithms. It can be seen that Fig.8(a) also shows the similar characteristics to Fig.7(a) such as the fastest convergence and the best average fitness value. For Fig.8(b), the best fitness value gained by DABC is slightly worse than that gained by MA algorithm at the beginning of the iteration, but DABC shows a powerful evolutionary performance in later iterations. Around 1000 iterations, DABC begins to converge and finally obtains the best fitness value. Therefore, DABC is the best performing among the competing algorithms for Case 2.

Table 7 gives the experimental results of the competing algorithms for Case 2. From Table 7, we can see that the standard deviation of waiting time of Call Cells, the average waiting time of Call Cells and the total travel distance of AGV gained by DABC algorithm are respectively 152.7, 1026.64 and 115.5, much better those gained by the competing algorithms. Compared with FCFS, the results gained by DABC have respectively decreased by 40.4%, 35.7%, and 81.56% on the above three indicators, while it has decreased by 5.72%, 3.24%, and 16.53% than IHS. It means that both the stability of AGV transferring materials to Call Cells and the energy efficiency of AGV have greatly improved by using the proposed DABC algorithm.

For Case 2, the best sequence corresponding to the optimal solution obtained by proposed DABC algorithm is {2 16 19 21 30 5 7 20 8 14 1 11 26 29 6 17 3 15 22 21}.

- Case 3

In this section, we test Case 3 with 25 Call Cells. The experimental results are obtained within 10 seconds. Fig.9(a) and Fig.9(b) show the plot of the average fitness value and the best fitness value for the competing algorithms. It can be seen from Fig.9(a) that it shows the similar characteristics to Fig.7(a) and Fig.8(a), *i.e.*, the fastest convergence and the best average fitness value. For Fig.9(b), the best fitness value obtained by DABC is better than other competing algorithms at the beginning of the iteration. Around 2000 iterations, the convergence rate of DABC gradually becomes stable. The best fitness value obtained is obviously superior to that obtained by other competing algorithms. Therefore, DABC is the best one among the existing algorithms for Case 3.

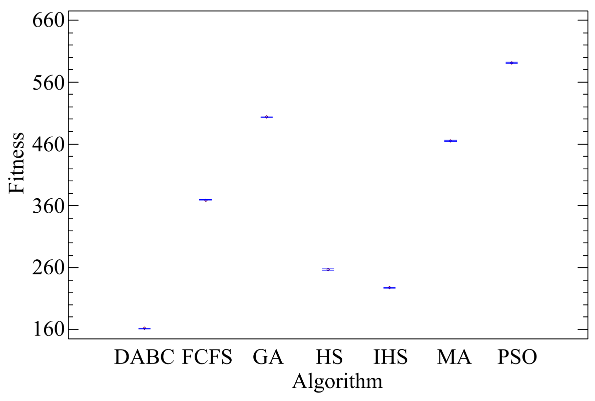


FIGURE 10. **Means plot for the competing algorithms. All means have Tukey's Honest Significant Difference (HSD) intervals at the 95% confidence level.**

Table 8 reports the experimental results of the competing algorithms for Case 3. It can be observed from Table 8 that the standard deviation of waiting time of Call Cells, the average waiting time of Call Cells and the total travel distance of AGV obtained by DABC are respectively 150.15, 1228.16 and 204.41, much better those obtained by other competing algorithms. Compared with FCFS, the results gained by DABC have decreased by 61.3%, 31.02%, and 70.27% on the above three indicators respectively, while it has decreased by 6.65%, 3.88%, and 8.37% than IHS. For Case 3 with 25 Call Cells, DABC has powerful performance to improve the stability of AGV transferring materials to Call Cells and the energy efficiency of AGV.

For Case 3, the best sequence corresponding to the optimal solution obtained by the proposed DABC algorithm is {12 20 6 19 2 24 8 1 28 3 30 25 13 29 14 16 22 15 23 9 10 5 7 18 26}.

From the above experiments, we can observe that the differences among competing algorithms are large enough, but it is still advisable to carry out statistical testing. We implement a multi-factor Analysis of Variance (ANOVA) to analyze the results of the experiments where the fitness value is the response variable and all the instance factors and the type of algorithm are controlled. Fig. 10 shows the means plots with 95% Tukey's Honest Significant Difference (HSD) confidence intervals for the competing algorithm. It has to be stressed that if there is no overlap for the confidence intervals among the means, the observed differences among the means is statistically significant in the response variable (Fitness). As shown in Fig. 10, we can see that DABC is the best in statistics, IHS, HS, FCFS, MA and GA followed, and PSO the worst. Meanwhile, it also verifies that DABC is the best algorithm to solve AGVSP.

VI. CONCLUSION

Since AGVSP plays a key role in improving productivity and reducing costs for manufacturing enterprises, it is necessary to develop effective methods for solving this problem. In this paper, an integer linear programming model was first formulated, which included two indicators namely the standard deviation of the waiting time of Call Cells and the total travel distance of AGV. Then, a constructive heuristic based on the problem-specific characteristics was presented to quickly generate an outstanding solution. Next, an effective discrete artificial bee colony algorithm was proposed. In order to settle this problem better, a solution generated by the proposed heuristic was used as an initial solution of the DABC algorithm. Six neighborhood operators were introduced to enhance the exploitation capability of the DABC algorithm in the employed and onlooker bee phases. A new evolution strategy in the onlooker bee phase was presented to provide opportunities for further exploration of the potential solution. The DABC algorithm was empirically evaluated by applying it to three cases from the real-world manufacturing system for producing the back cover of a smartphone. The experimental results show that the proposed algorithm is superior to competing algorithms.

For the future work, we will extend our research as the follows: (1) Consider additional characteristics such as travel time, return time, due dates and multi-objectives [47], [48]. (2) Improve the DABC algorithm by an information feedback method for the AGVSP [49]. (3) Apply the DABC algorithm for solving the AGVSP with multiple constrains.

APPENDIX

Case 0-4 are reported in Table 9-12. In the table, the Number of Call Cells is given in the first row. The Call time of Call Cells is given in the second row. The location of Call Cells is given in the third row.

TABLE 9. **Case 0.**

Number	10	11	19	1	20	2
Call time	20	60	100	180	220	300
Location	49.5	0	44	0	49.5	5.5

TABLE 10. **Case 1.**

Number	1	16	28	21	2	8	3	5
Call time	10	150	160	190	250	280	450	560
Location	0	0	66	33	5.5	38.5	11	22
Number	15	30	18	19	12	9	6	
Call time	650	760	820	880	950	980	990	
Location	77	77	11	16.5	60.5	44	27.5	

TABLE 11. **Case 2.**

Number	2	14	30	16	7	19	29	5	20	11
Call time	6	50	80	100	150	200	260	280	320	380
Location	5.5	71.5	77	0	33	16.5	71.5	22	22	55
Number	1	21	8	26	3	6	17	21	22	15
Call time	420	440	480	520	580	660	750	820	860	910
Location	0	27.5	38.5	55	11	27.5	5.5	27.5	33	77

TABLE 12. Case 3.

Number	8	19	28	6	12	20	24	2	30	25	13	3	1
Call time	10	48	82	110	146	180	212	250	278	325	356	388	421
Location	38.5	16.5	66	27.5	60.5	22	44	5.5	77	49.5	66	11	0
Number	29	22	14	7	26	10	23	16	5	15	9	18	
Call time	451	482	522	546	587	622	678	755	787	825	866	970	
Location	71.5	33	71.5	33	55	49.5	38.5	0	22	77	44	11	

REFERENCES

[1] I. Draganjac, D. Miklic, Z. Kovacic, G. Vasiljevic, and S. Bogdan, "Decentralized control of multi-AGV systems in autonomous warehousing applications," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 4, pp. 1433–1447, Oct. 2016.

[2] Y. Yang, M. Zhong, Y. Dessouky, and O. Postolache, "An integrated scheduling method for AGV routing in automated container terminals," *Comput. Ind. Eng.*, vol. 126, pp. 482–493, Dec. 2018.

[3] G. Cavone, M. Dotoli, and C. Seatzu, "A survey on Petri net models for freight logistics and transportation systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 6, pp. 1795–1813, Jun. 2018.

[4] F. Zhang and J. Li, "An improved particle swarm optimization algorithm for integrated scheduling model in AGV-served manufacturing systems," *J. Adv. Manuf. Syst.*, vol. 17, no. 03, pp. 375–390, Sep. 2018.

[5] R. Yan, S. J. Dunnett, and L. M. Jackson, "Novel methodology for optimising the design, operation and maintenance of a multi-AGV system," *Rel. Eng. Syst. Saf.*, vol. 178, pp. 130–139, Oct. 2018.

[6] L. Qiu, W.-J. Hsu, S.-Y. Huang, and H. Wang, "Scheduling and routing algorithms for AGVs: A survey," *Int. J. Prod. Res.*, vol. 40, no. 3, pp. 745–760, Jan. 2002.

[7] I. F. A. Vis, "Survey of research in the design and control of automated guided vehicle systems," *Eur. J. Oper. Res.*, vol. 170, no. 3, pp. 677–709, May 2006.

[8] H. Fazlollahtabar and M. Saidi-Mehrabad, "Methodologies to optimize automated guided vehicle scheduling and routing problems: A review study," *J. Intell. Robot. Syst.*, vol. 77, nos. 3–4, pp. 525–545, Mar. 2015.

[9] M. Mousavi, H. J. Yap, S. N. Musa, F. Tahriri, and S. Z. M. Dawal, "Multi-objective AGV scheduling in an FMS using a hybrid of genetic algorithm and particle swarm optimization," *PLoS ONE*, vol. 12, no. 3, Mar. 2017, Art. no. e0169817.

[10] G. Li, B. Zeng, W. Liao, X. Li, and L. Gao, "A new AGV scheduling algorithm based on harmony search for material transfer in a real-world manufacturing system," *Adv. Mech. Eng.*, vol. 10, no. 3, pp. 1–13, Mar. 2018.

[11] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse, "The vehicle routing problem: State of the art classification and review," *Comput. Ind. Eng.*, vol. 99, pp. 300–313, Sep. 2016.

[12] J. Brandão, "Iterated local search algorithm with ejection chains for the open vehicle routing problem with time windows," *Comput. Ind. Eng.*, vol. 120, pp. 146–159, Jun. 2018.

[13] Q.-K. Pan, L. Gao, L. Xin-Yu, and F. M. Jose, "Effective constructive heuristics and meta-heuristics for the distributed assembly permutation flowshop scheduling problem," *Appl. Soft Comput.*, vol. 81, Aug. 2019, Art. no. 105492, doi: 10.1016/j.asoc.2019.105492.

[14] C. Zeng, J. Tang, and C. Yan, "Scheduling of no buffer job shop cells with blocking constraints and automated guided vehicles," *Appl. Soft Comput.*, vol. 24, pp. 1033–1046, Nov. 2014.

[15] H. Fazlollahtabar, M. Saidi-Mehrabad, and J. Balakrishnan, "Mathematical optimization for earliness/tardiness minimization in a multiple automated guided vehicle manufacturing system via integrated heuristic algorithms," *Robot. Auton. Syst.*, vol. 72, pp. 131–138, Oct. 2015.

[16] T. Miyamoto and K. Inoue, "Local and random searches for dispatch and conflict-free routing problem of capacitated AGV systems," *Comput. Ind. Eng.*, vol. 91, pp. 1–9, Jan. 2016.

[17] M. Saidi-Mehrabad, S. Dehnavi-Arani, F. Evazabadian, and V. Mahmoodian, "An ant colony algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs," *Comput. Ind. Eng.*, vol. 86, pp. 2–13, Aug. 2015.

[18] M. Gen, W. Zhang, L. Lin, and Y. Yun, "Recent advances in hybrid evolutionary algorithms for multiobjective manufacturing scheduling," *Comput. Ind. Eng.*, vol. 112, pp. 616–633, Oct. 2017.

[19] X. Lyu, Y. Song, C. He, Q. Lei, and W. Guo, "Approach to integrated scheduling problems considering optimal number of automated guided vehicles and conflict-free routing in flexible manufacturing systems," *IEEE Access*, vol. 7, pp. 74909–74924, 2019.

[20] H. Lu and S. Wang, "A study on multi-ASC scheduling method of automated container terminals based on graph theory," *Comput. Ind. Eng.*, vol. 129, pp. 404–416, Mar. 2019.

[21] C. Chen, L. K. Tiong, and I.-M. Chen, "Using a genetic algorithm to schedule the space-constrained AGV-based prefabricated bathroom units manufacturing system," *Int. J. Prod. Res.*, vol. 57, no. 10, pp. 3003–3019, 2019.

[22] G. Li, X. Li, L. Gao, and B. Zeng, "Tasks assigning and sequencing of multiple AGVs based on an improved harmony search algorithm," *J. Ambient Intell. Hum. Comput.*, vol. 10, no. 11, pp. 4533–4546, Nov. 2019.

[23] J. Andelmin and E. Bartolini, "A multi-start local search heuristic for the green vehicle routing problem based on a multigraph reformulation," *Comput. Oper. Res.*, vol. 109, pp. 43–63, Sep. 2019.

[24] V. F. Yu, A. A. N. P. Redi, Y. A. Hidayat, and O. J. Wibowo, "A simulated annealing heuristic for the hybrid vehicle routing problem," *Appl. Soft Comput.*, vol. 53, pp. 119–132, Apr. 2017.

[25] G. Poonthalir and R. Nadarajan, "A fuel efficient green vehicle routing problem with varying speed constraint (F-GVRP)," *Expert Syst. Appl.*, vol. 100, pp. 131–144, Jun. 2018.

[26] A. Gutierrez, L. Dieulle, N. Labadie, and N. Velasco, "A multi-population algorithm to solve the VRP with stochastic service and travel times," *Comput. Ind. Eng.*, vol. 125, pp. 144–156, Nov. 2018.

[27] V. Baradaran, A. Shafaei, and A. H. Hosseinian, "Stochastic vehicle routing problem with heterogeneous vehicles and multiple prioritized time windows: Mathematical modeling and solution approach," *Comput. Ind. Eng.*, vol. 131, pp. 187–199, May 2019.

[28] Y. Li, H. Soleimani, and M. Zohal, "An improved ant colony optimization algorithm for the multi-depot green vehicle routing problem with multiple objectives," *J. Cleaner Prod.*, vol. 227, pp. 1161–1172, Aug. 2019.

[29] N. R. Sabar, A. Bhaskar, E. Chung, A. Turky, and A. Song, "A self-adaptive evolutionary algorithm for dynamic vehicle routing problems with traffic congestion," *Swarm Evol. Comput.*, vol. 44, pp. 1018–1027, Feb. 2019.

[30] T. Vidal, T. G. Crainic, M. Gendreau, N. Lahrichi, and W. Rei, "A hybrid genetic algorithm for multidepot and periodic vehicle routing problems," *Oper. Res.*, vol. 60, no. 3, pp. 611–624, Jun. 2012.

[31] J.-F. Cordeau and M. Maischberger, "A parallel iterated tabu search heuristic for vehicle routing problems," *Comput. Oper. Res.*, vol. 39, no. 9, pp. 2033–2050, Sep. 2012.

[32] C. L. Wang and S. W. Li, "Hybrid fruit fly optimization algorithm for solving multi-compartment vehicle routing problem in intelligent logistics," *Adv. Prod. Eng. Manage.*, vol. 13, no. 4, pp. 466–478, Dec. 2018.

[33] P. H. V. Penna, A. Subramanian, and L. S. Ochi, "An iterated local search heuristic for the heterogeneous fleet vehicle routing problem," *J. Heuristics*, vol. 19, no. 2, pp. 201–232, Apr. 2013.

[34] Q.-K. Pan, M. F. Tasgetiren, P. N. Suganthan, and T. J. Chua, "A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem," *Inf. Sci.*, vol. 181, no. 12, pp. 2455–2468, Jun. 2011.

[35] N. Karaboga, "A new design method based on artificial bee colony algorithm for digital IIR filters," *J. Franklin Inst.*, vol. 346, no. 4, pp. 328–348, May 2009.

[36] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Global Optim.*, vol. 39, no. 3, pp. 459–471, Oct. 2007.

[37] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Appl. Soft Comput.*, vol. 8, no. 1, pp. 687–697, Jan. 2008.

[38] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: Artificial bee colony (ABC) algorithm and applications," *Artif. Intell. Rev.*, vol. 42, no. 1, pp. 21–57, Jun. 2014.

[39] D. Gong, Y. Han, and J. Sun, "A novel hybrid multi-objective artificial bee colony algorithm for blocking lot-streaming flow shop scheduling problems," *Knowl.-Based Syst.*, vol. 148, pp. 115–130, May 2018.

[40] J.-Q. Li, Q.-K. Pan, and K.-Z. Gao, "Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems," *Int. J. Adv. Manuf. Technol.*, vol. 55, nos. 9–12, pp. 1159–1169, Aug. 2011.

[41] K. Guo and Q. Zhang, "A discrete artificial bee colony algorithm for the reverse logistics location and routing problem," *Int. J. Inf. Technol. Decis. Making*, vol. 16, no. 05, pp. 1339–1357, Sep. 2017.

[42] Y. Zhong, J. Lin, L. Wang, and H. Zhang, "Hybrid discrete artificial bee colony algorithm with threshold acceptance criterion for traveling salesman problem," *Inf. Sci.*, vol. 421, pp. 70–84, Dec. 2017.

[43] R. Tavakkoli-Moghaddam, N. Safaei, and Y. Gholipour, "A hybrid simulated annealing for capacitated vehicle routing problems with the independent route length," *Appl. Math. Comput.*, vol. 176, no. 2, pp. 445–454, May 2006.

[44] Y.-Y. Han, Q.-K. Pan, J.-Q. Li, and H.-Y. Sang, "An improved artificial bee colony algorithm for the blocking flowshop scheduling problem," *Int. J. Adv. Manuf. Technol.*, vol. 60, nos. 9–12, pp. 1149–1159, Jun. 2012.

[45] Q.-K. Pan, "An effective co-evolutionary artificial bee colony algorithm for steelmaking-continuous casting scheduling," *Eur. J. Oper. Res.*, vol. 250, no. 3, pp. 702–714, May 2016.

[46] H.-Y. Sang, P.-Y. Duan, and J.-Q. Li, "An effective invasive weed optimization algorithm for scheduling semiconductor final testing problem," *Swarm Evol. Comput.*, vol. 38, pp. 42–53, Feb. 2018.

[47] Q.-K. Pan, L. Gao, and L. Wang, "A multi-objective hot-rolling scheduling problem in the compact strip production," *Appl. Math. Model.*, vol. 73, pp. 327–348, Sep. 2019.

[48] B. Zhang, Q.-K. Pan, L. Gao, L.-L. Meng, X.-Y. Li, and K.-K. Peng, "A three-stage multiobjective approach based on decomposition for an energy-efficient hybrid flow shop scheduling problem," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published, doi: 10.1109/TSMC.2019.2916088.

[49] G.-G. Wang and Y. Tan, "Improving metaheuristic algorithms with information feedback models," *IEEE Trans. Cybern.*, vol. 49, no. 2, pp. 542–555, Feb. 2019.

WEN-QIANG ZOU received the M.S. degree in computer application technology from the Liaoning University of Science and Technology, in 2011. He is currently pursuing the Ph.D. degree with the School of Mechatronic Engineering and Automation, Shanghai University, Shanghai, China. His research interests include AGV scheduling and routing, intelligent optimization, and scheduling algorithms.

QUAN-KE PAN (Member, IEEE) received the B.Sc. and Ph.D. degrees from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 1993 and 2003, respectively. From 2003 to 2011, he was with the School of Computer Science Department, Liaocheng University, where he became a Full Professor, in 2006. From 2011 to 2014, he was with State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang, China. From 2014 to 2015, he was with the State Key Laboratory of Digital Manufacturing and Equipment Technology, Huazhong University of Science and Technology. He has been with the School of Mechatronic Engineering and Automation, Shanghai University, since 2015. His current research interests include intelligent optimization and scheduling algorithms. He has authored one academic book and more than 200 refereed articles. He acts as the Editorial Board Member for several journals, including *Operations Research Perspective* and *Swarm and Evolutionary Computation*.

M. FATIH TASGETIREN is currently a Full Professor with the International Logistics Management, Yaşar University, Turkey. His research focus is on modeling, analysis, and optimization of complex systems through the use of computational intelligence methods. He works on the design and development of modern meta-heuristic algorithms to solve discrete/combinatorial/binary as well as real-parameter unconstrained/constrained optimization problems. His main research interest has been on sequencing and scheduling problems (ranging from single machine problems to multimachine problems such as parallel machines, job-shops, flow shops, hybrid job-shops and flow shops, sequence-dependent, and distributed variants). Recently, he works on the development of energy-efficient production scheduling systems as well as architectural design optimization. His current Google academic citations are 6951 with an H-index of 38.