IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received March 23, 2021, accepted April 20, 2021, date of publication April 27, 2021, date of current version May 6, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3075948

An Adaptive Multi-Strategy Artificial Bee Colony Algorithm for Integrated Process Planning and Scheduling

YANG CAO [1,2,3,4,5,6] **AND HAIBO SHI** [2,3,5]
[1]College of Information Science and Engineering, Northeastern University, Shenyang 110819, China
[2]Shenyang Institute of Automation, Chinese Academy of Sciences, Shenyang 110016, China
[3]Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Shenyang 110169, China
[4]University of Chinese Academy of Sciences, Beijing 100049, China
[5]Key Laboratory of Network Control System, Chinese Academy of Sciences, Shenyang 110016, China
[6]Information and Control Engineering Faculty, Shenyang Jianzhu University, Shenyang 110168, China

Corresponding author: Yang Cao (caoyang@sia.cn)

This work was supported by the Liaoning Revitalization Talents Program, China, under Grant XLYC1808009.

ABSTRACT Traditionally, process planning and scheduling were performed sequentially, where scheduling depended on the result of process planning. Considering their complementarity, the two functions are more tightly integrated to improve the performance of job shop flexible manufacturing environment. This study proposes an adaptive multi-strategy artificial bee colony (AMSABC) algorithm to solve integrated process planning and scheduling (IPPS) problem. In AMSABC, two search strategies with different characteristics are introduced into employed bees and onlooker bees to take on the responsibility of both exploration and exploitation. The selection probability of each search strategy is dynamically adjusted according to previous experiences. To further improve the exploitation performance of the approach, a problem-specific multi-objective local search has been embedded in the proposed algorithm. Furthermore, AMSABC algorithm presents a unique solution representation where a food source is represented by three discrete vectors, and a well-designed decoding scheme is developed. Next, the corresponding neighborhood structure is adopted that it can directly generate feasible solutions in the search space. The proposed algorithm is tested on the well-known benchmark instances and compared with the state-of-the-art algorithms. Through detail analysis of experimental results, AMSABC algorithm is more beneficial in the quality and efficiency of solution.

INDEX TERMS Integrated process planning and scheduling, artificial bee colony algorithm, multi-objective optimization, multi-strategy collaboration, strategy adaptation.

I. INTRODUCTION

Process planning and scheduling are implemented as two sequential decision-making functions in product design and manufacturing process. Process planning needs to coordinate the precedence relationships between machines, operations, and other parameters to generate feasible manufacturing schemes. Scheduling will then attempt to optimize the sequence of operation according to the constraints such as makespan, critical machine workload, and machines total workload. Traditionally, the researchers independent consider the two separate systems. Due to recent advances in manufacturing system and increase the number of process plans during product manufacturing. The integrated process planning and scheduling (IPPS) problem [1] has received more and more attentions by researchers and engineers.

The classical job shop scheduling problem (JSP) consists of a set of jobs on a set of machines, each job having a sequence of operations which must be processed on a predefined machine. Meanwhile, JSP assumes only one operation sequence (feasible process plan) for each job. The problem is among the strongly NP-hard optimization problem [2]. The IPPS is an extension of inheriting JSP characteristics that allows an operation to be processed from among alternative machines, and from among alternative process plans of a job [3]. Obviously, the IPPS is more complicated than the general scheduling problem. It includes three flexibilities: sequencing flexible (SF), processing flexibility (PF), and

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

operation flexibility (OF) [4]. The first flexibility is the availability of alternative precedence relation for operations, the second is the availability of alternative process plans for each job, and the last is the availability of alternative machines for each operation. Meanwhile, many models have been proposed on the IPPS, as it is reviewed in the next section.

The single-objective IPPS problem, generally to minimize the makespan, has been extensively studied in the past decades [4]–[7]. Kim *et al.* are among the pioneers who study the single-objective IPPS problem, who devised a set of representative benchmark instances [8]. Compared to the single-objective IPPS, many real-world IPPS usually involve simultaneous optimization of several objectives, that is, the improvement in one objective will inevitably cause the deterioration in some other objectives. Thus, the Pareto-based multi-objective IPPS has shown a very good performance for the realistic production environments. For example, Y. W. Guo *et al.* [9] studied the particle swarm optimization (PSO) algorithm, and a re-planning method had been developed to address the conditions of machine breakdown and new order arrival. Jin *et al.* [10] proposed a multi-objective memetic algorithm (MOMA) with two local search method. Mohapatra *et al.* [11] reported a controlled elitist non-dominated sorting genetic algorithm (NSGA-II). Zhang and Wong [12] introduced an ant colony optimization(ACO) algorithm. Li *et al.* [13] studied the multi-objective problem with the Nash equilibrium in game theory, and employed a tabu search (TS) for improving the makespan criterion only. However, the above research articles concentrate on small- or medium-scale IPPS problem only, the optimum for large-scale instances have not been report.

Due to the complexity of the IPPS, meta-heuristic methods have been proven to have superior features than other traditional methods. The artificial bee colony (ABC) algorithm is proposed by Karaboga [14] to optimize multivariable and multimodal continuous functions in 2005. It has been applied successful to solve the multi-objective scheduling problem [15]–[17]. The ABC algorithm distributes information on food location throughout the population of bees. This characteristic makes ABC good at exploration but poor at exploitation. Actually, in order to solve the drawback, some attempts have already been made to improve its performance. The improvements can be classified as follows roughly: new search equation, hybrid algorithm, and multi-strategy adaption. Karaboga *et al.* [18] proposed more detailed discussions of the improvement research on ABC. Gao *et al.* [19], [20] provided analysis on the research trend of meta-heuristics and swarm intelligence algorithms for solving scheduling problem. To the best of our knowledge, there is no literature aimed at solving the IPPS problem by using ABC.

The multi-strategy adaption is designed to dynamic adjust the selection probabilities of the search strategies during different evolution stages for further improving the performance of ABC algorithm. Cui *et al.* [21] proposed ABC with depth-first search framework and elite_guided search equation (DFSABC_Elite) as low-level team cooperation, in which the solution was selected only from the top% elite solution. In literature [22], the best-so-far solution, inertia weight and acceleration coefficients were introduced to modify the search process of ABC. Inertia weight and acceleration coefficients were defined as functions of the fitness. Xiang *et al.* [23] proposed improved ABC algorithm, the individuals selected based on the gravity model in the employed bee phase. A random guiding search was introduced in the onlooker bee phase, and the candidate individuals were generated by three search equations with the different probabilities. ABCVSS algorithm with five search strategies and counters to update the solutions were proposed in [24]. At the initial stage, the five search strategies were assigned with the equal opportunity. During the search process performed by the artificial agents, these counters were used to determine the rule that was selected by the bees. Therefore, one or more search strategies were selected and were used during the iterations according to the characteristics of the optimization problems. Furthermore, the multi-species strategy had been adopted in ABC algorithm [25], in which the dynamic segmentation of the swarm and co-evolution strategy was designed. For serving the similar purpose, Li *et al.* [26] introduced discrete ABC algorithm with self-adaptive strategy, which could produce neighboring solutions in different promising regions.

When designing an ideal optimization algorithm, it is a key for determining the performance of optimization algorithm to how to dynamically adjust exploration and exploitation at different evolution stages. That is, good search strategies should promote initial exploration, and allow individuals to use the obtained information about the search space to improve and adjust solutions later. Another noteworthy enhancement is the introduction of multi-objective optimization. Objective-specific intensification search methods need to be developed to take every criterion in to account. Therefore, the optimization algorithm must be designed to select suitable search methods for exploiting the different objectives.

The motivation of this article is to overcome the two problems above. We adopt an adaptive multi-strategy artificial bee colony algorithm to solve multi-objective IPPS problem, namely AMSABC algorithm. The proposed AMSABC has the following main contributions: 1) considering the problem features, a well-designed chromosome decoding scheme and the corresponding neighborhood structure are developed; 2) two novel search strategies with different characteristics are introduced into employed bees and onlooker bees to take on the responsibility of both exploration and exploitation; 3) in order to dynamical balance the exploration and exploitation abilities, an adaptive mechanism is proposed to adjust the selection probabilities of the different search strategies according to their success rates of generating solutions; 4) to further improve the exploitation ability, a problem-specific multi-objective local search is embedded in the proposed algorithm.

The remainder of this article is organized as follows. Section II provides the literature review of the IPPS.

The formulation and illustration of the IPPS are presented in Section III. Section IV develops the proposed AMSABC algorithm. And will compare with other algorithms and evaluate the experimental results in Section V. Finally, the conclusion and future work are drawn in Section VI.

II. PROBLEM FORMULATION

Over the last two decades, researchers have used various approaches to solve the IPPS problem. Many models have been proposed on the IPPS, the models can be classified as follows: nonlinear process planning (NLPP), closed loop process planning (CLPP), and distributed process planning (DPP).

A. NONLINEAR PROCESS PLANNING

NLPP is a basic model of the IPPS that is deployed by Jain *et al.* [27]. All the potential process plans would be considered and archived. The methodology of NLPP is to make the process with higher priority is always put into practice when the job is required. If the prior priority plan is not suitable for the current objective function, the next priority will be chosen. Because the problem is a case of combinatorial explosive, most of the current research works on the integration model focus on the improvement and implementation of the model. Later in Section III, the definition and formulation of the model are presented.

Kim *et al.* [4] presented a symbiotic evolutionary algorithm which adopted the strategies of localized interactions, steady-state reproduction, and random symbiotic partner selection for the IPPS. Li *et al.* [28] developed an evolutionary algorithm-based approach to solve this problem. Lihong and Shengping [5] and Shao *et al.* [29] gave a modified genetic algorithm (GA). Li and Mcmahon [30] used a simulated annealing (SA)-based approach.

B. CLOSED LOOP PLANNING

The process plan in this kind of approach is produced regarding the resources status and relying on the dynamic feedback from the workshop. The process planning mechanism generates process plans based on available resources. Production scheduling provides the information about which machines are available on the shop floor for an incoming job to process planning, so that every plan is feasible and respects to the current availability of production facilities. This dynamic simulation system can enhance the real-time, intuition and manipulability of process planning system and also enhance the utilization of alternative process plans. CLPP can also be called as dynamic process planning or online process planning.

Seethaler and Yellowley [31] presented a dynamic feedback system for finding the alternative process plans, and Wang and Shen [32] found the alternative process plans with the same method. Shrestha *et al.* [33] introduced a holonic manufacturing system (HMS) using dynamic programming-based method for the modification of process plans and GA-based method to select a combination of process plans. Zhao *et al.* [34] further discussed the architecture of HMS and process planning for multi-objective IPPS.

C. DISTRIBUTED PROCESS PLANNING

The latest approach is the distributed process plan including simultaneous process planning and scheduling. It divides process planning and scheduling tasks into two phases. In the initial planning phase, the characteristics of parts and the relationship between the parts are recognized, and then corresponding manufacturing processes and scheduling plan are determined. The second phase is final planning, which matches the required job operations with the operation capabilities of available production resource. The result is dynamic process planning and production scheduling constrained by real-time events. However, this approach requires ad-hoc software capabilities accompanied by a reliable hardware configuration. DPP can also be called as just-in-time process planning, concurrent process planning, or collaborative process planning.

Zhang *et al.* [35] developed imperialist competitive algorithm (ICA) for the districted IPPS, and GA-based method was adapted to maintain the robustness of the plan and schedule. Wang *et al.* [36] presented a framework of collaborative process planning system supported by a real-time monitoring system. Li *et al.* [13] presented a game theory-based hybrid algorithm for solving multi-objective IPPS problem.

Varela *et al.* [37] proposed more detailed discussions of these approaches and their features, advantages, and disadvantages. In this article, a mathematical model of IPPS is proposed base on the methodology of NLPP.

III. PROBLEM DESCRIPTION

The IPPS problem can be described as follows [38]: Given a set of jobs which are to be processed on machines with operations including alternative manufacturing resources, select suitable process plan, assign each operation to an eligible machine, and sequence the operations so as to determine a schedule in which the precedence relations among operations of each job can be satisfied and the corresponding objectives can be achieved.

In order to make the IPPS problem more understandable, there is an example of the grinding process plan of V-belt pulley, whose processing flexibility is typical in the gearing manufacture industry, as shown in Table 1. Fig. 1 gives the construction of V-belt pulley. The job is processed in eight stages and involves fourteen operations. Each job has its own route dynamically assigned according to the current device state; that is, the process plan differs for different jobs. It does not matter if the third stage is processed after the fourth, so the processing sequence can be rearranged. The twelfth and thirteen operations have the same situation. In addition, there are two sets of operations from which you can choose one and only one in the stage P_4; the tenth operation can be placed in any position between the first operation and the

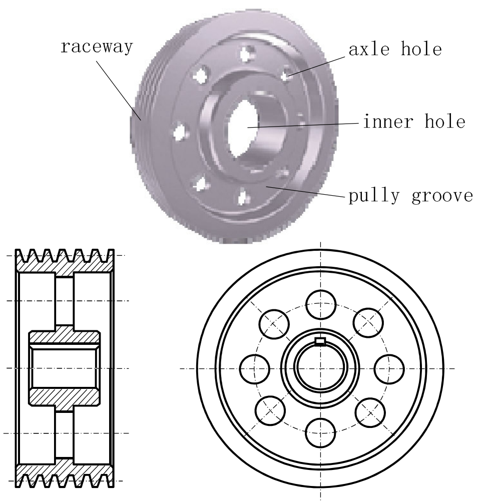


FIGURE 1. Construction of V-belt pulley.

TABLE 1. Processing information of V-belt pulley.

Processes	Process description	Operations	Operation description
P_1	casting	O_1	gravity die casting
P_2	overall dimensions	O_2	surface rough grinding
		O_3	pulley groove rough grinding
P_3	Φ30 inner hole	O_4	inner hole lathing
		O_5	inner hole rough grinding
P_4	Φ10 axle holes	O_6	axle holes boring
		O_7	axle holes rough grinding
		O_8	drilling machine boring
		O_9	axle holes rough grinding
P_5	raceway	O_{10}	raceway rough grinding
P_6	tempering	O_{11}	temper treatment
P_7	fine grinding	O_{12}	outside surface fine grinding
		O_{13}	inside surface fine grinding
P_8	brightening	O_{14}	electrolytic polishing

eleventh operation. Based on the above analysis of the gear making system, we have four alternative process plans for the V-belt pulley. Considering the alternative precedence relation and machine selection of each operation, we can model this scheduling as an IPPS problem. Specific formulation and illustration are detailed below.

A. PROBLEM FORMULATION

A multi-objective IPPS problem is considered in this study. The detailed assumptions and notation are given as follows:

1) ASSUMPTIONS

- All the machines are available at time zero.
- All the jobs can be started at time zero.
- Each machine can process only one operation at a time, and each operation cannot be interrupted.
- The transportation time on different machines and the setup time are negligible.
- There are no buffers and precedence constraints.

2) NOTATION

Indices and parameters

i, i'	:	index of jobs
j, j'	:	index of operations
k, k'	:	index of machines
l, l'	:	index of process paths
n	:	total number of jobs
N	:	set of jobs
M	:	set of machines
$O_{i,j,l}$	:	$j-th$ operation of job i on the job's l-th process plan
T_i	:	set of process plans of job i
no	:	total number of process paths
$NO_{i,l}$	:	set of operations for l-th process plan of job i
$R_{i,j,l}$	:	set of available machines for $O_{i,j,l}$
$p_{i,j,k,l}$	:	processing time of $O_{i,j,l}$ on machine k
A	:	a large positive integer
NP	:	total number of the population

Decision variables

C_m	:	Makespan
W_m	:	Critical machine workload
W_t	:	Machines total workload
$X_{i,l}$	:	taking value 1, if l-th process plan of job i is selected; 0, otherwise
$Z_{i,j,k,l}$	:	taking value 1, if $O_{i,j,l}$ is processed on machine k; 0, otherwise
$Y_{i,j,l,i',j',l'}$	:	taking value 1, if $O_{i,j,l}$ is processed directly or indirectly after $O_{i',j',l'}$; 0, otherwise
$C_{i,j}$	:	completion time of j-th operation of job i

The main mathematical model of the multi-objective IPPS problem is given as follows:

$$min\ C_m \tag{1}$$

$$min\ W_m = \max\left(\sum_{i\in N}\sum_{l\in T_i}\sum_{j\in NO_{i,l}} Z_{i,j,k,l}p_{i,j,k,l}\right) \tag{2}$$

$$min\ W_t = \sum_{i\in N}\sum_{l\in T_i}\sum_{j\in NO_{i,l}}\sum_{k\in R_{i,j,l}} Z_{i,j,k,l}p_{i,j,k,l} \tag{3}$$

subject to:

$$\sum_{l\in T} X_{i,l} = 1, \quad \forall i \in N \tag{4}$$

$$Z_{i,j,k,l} + \left(1 - X_{i,l}\right) = 1, \quad \forall i \in N, l \in T_i, j \in NO_{i,l} \tag{5}$$

$$C_{i,0} = 0, \quad \forall i \in N \tag{6}$$

$$C_{i,j} \geq C_{i,j-1} + p_{i,j,k,l} Z_{i,j,k,l},\quad \forall i \in N, l \in T_i, j \in NO_{i,l}, j \geq 1 \quad (7)$$

$$C_{i,j} \geq C_{i',j'} + p_{i,j,k,l} - A\left(3 - Y_{i,j,l,i',j',l'} - Z_{i,j,k,l} - Z_{i',j',k',l'}\right),\quad \forall i \in N, i < |N|, i' > i,\ l \in T_i,\ l' \in T_{i'}, j \in NO_{i,l},\ j' \in NO_{i',l'}, k \in R_{i,j,l} \cap R_{i',j',l'} \quad (8)$$

$$C_{i',j'} \geq C_{i,j} + p_{i',j',k',l'} - A\left(2 + Y_{i,j,l,i',j',l'} - Z_{i,j,k,l} - Z_{i',j',k',l'}\right), \forall i \in N, i < |N|, i' > i, l \in T_i, l' \in T_{i'}, j \in NO_{i,l}, j' \in NO_{i',l'}, k \in R_{i,j,l} \cap R_{i',j',l'} \quad (9)$$

$$C_m \geq C_{i,j}, \forall i \in N, l \in T_i, j \in NO_{i,l} \quad (10)$$

The objectives are described by Eqs. (1-3). Eq. (1) gives the first objective which is to minimize the maximum of the finishing time (makespan) considering all the operations. Eq. (2) gives the second objective which is to minimize the maximum of the critical machine workload. Eq. (3) gives the third objective which is to minimize the total workload. Constraint (4) ensures that only one process plan is selected for each job. Constraint (5) assigns operations to the corresponding machine according to each selected operation of job. Constraints (6) and (7) guarantee that an operation of a job should be processed after its job predecessor is completed, and every job can only be processed at time 0 or later. Constraints (8) and (9) ensure that the different operations on the same machine follow a correct sequence by scheduling. Constraint (10) calculates the makespan.

B. PROBLEM ILLUSTRATION

To aid comprehension of IPPS, we provide a more complex example by network graphs as illustrated in Fig. 2. Table 2 gives the processing times of each operation on each machine. There are three node types: starting node, intermediate node and ending node. The starting node and ending node are dummy ones, respectively, indicate the start and the completion of the manufacturing process of a job. An intermediate node is operation node (marked with number), which contains the alternative machines that can perform the operation and processing time required for the operation according to the machine. For example, the operation 1 is assigned to machine M_2 or M_3 with processing times 9 or 12. The arrows connecting the nodes imply the precedence between them. OR relationships are used to describe processing flexibility that the same manufacturing feature can be completed by different process procedures. If the links following a node are connected by an OR symbol, the only one of OR link paths can be traversed. OR link path is an operation path that begins at an OR symbol and ends as it merges with the other paths. Of course, an OR link path can contain the other OR link paths. For the links that are not connected by OR symbol, all of them must be visited. Ho and Moodie [39] proposed more detailed discussions of the network representation on IPPS.

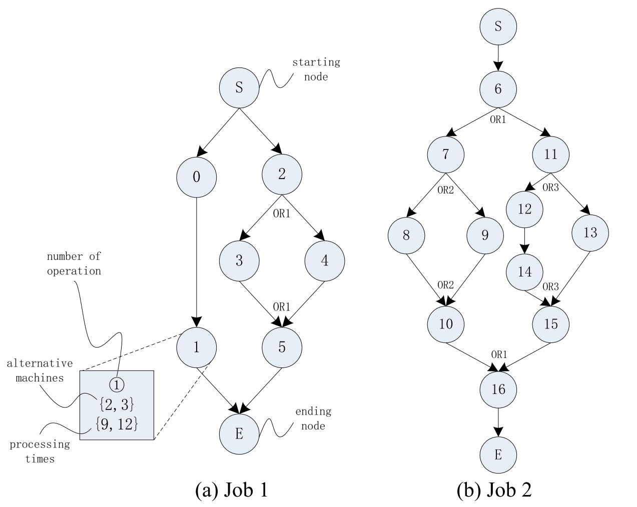


FIGURE 2. A sample IPPS instance networks.

IV. PROPOSED ALGORITHM AMSABC

A. SEARCH STRATEGIES

One of the key issues when designing optimization algorithms lies in the efficiency of search strategy. The search strategies also bear the great influence on the speed and quality of the solution in ABC algorithm.

In this section, different search strategies with different characteristics are introduced in details. On one hand, more randomly selected individuals can enhance exploration; on the other hand, it is beneficial to exploitation by introducing excellent individuals. The search process of individuals in the population is an adaptive process. The selection probability of strategies is dynamically adjusted to balance exploration and exploitation according to success rate.

1) DE/rand-TO-best/1

Literature [40] proposed that the two-difference-vector-based strategies are able to provide better perturbation than the one-difference-vector-based strategies. The equation adopts certain randomness in the first item, the movement towards the best-so-far solution to improve the exploitation in the second item, and some information about direction to increase the diversity in the third item, as shown in (11).

$$X'_{i,j} = X_{r1,j} + \phi_{i,j} \times \left(X_{best,j} - X_{r1,j}\right) + \varphi_{i,j} \times \left(X_{r2,j} - X_{r3,j}\right) \quad (11)$$

where $X_{best,j}$ represents the jth element of the elite individual in the current generation. $r1, r2, r3 \in \{1 \cdots FN\}$, and $r1 \neq r2 \neq r3 \neq best$. $\phi_{i,j}$ is a random number generated uniformly between [0, 1.0] and $\varphi_{i,j}$ is also a random number generated uniformly in the range [−1.0, 1.0].

2) CABC_ELITE

The above search strategy improves the exploitation ability to a certain extent in employed bee phase. However,

TABLE 2. The processing times of each operation on each machine.

Job 1	Operations	$O_{1,0}$	$O_{1,1}$	$O_{1,2}$	$O_{1,3}$	$O_{1,4}$	$O_{1,5}$					
	Machines	1,2	2,3	1,4	2,3	1,2,3	1,4					
	Times	11,10	9,12	15,17	9,13	9,12,14	10,15					
Job 2	Operations	$O_{2,6}$	$O_{2,7}$	$O_{2,8}$	$O_{2,9}$	$O_{2,10}$	$O_{2,11}$	$O_{2,12}$	$O_{2,13}$	$O_{2,14}$	$O_{2,15}$	$O_{2,16}$
	Machines	1,2,3	1,2	1,3	2,4	3	1,2	2,4	2,4	3,4	1,4	3,4
	Times	2,8,10	14,12	13,15	13,16	14	13,15	12,17	16,12	14,12	11,14	6,13

the exploitation ability needs to be further enhanced because of the poor exploitation ability of ABC. To utilize the properties of ergodicity, irregularity and randomness, chaotic mapping was used to replace the random numbers of the search equation in ABC algorithm, and Chaotic ABC algorithm (CABC) was proposed in [41]. In onlooker bee phase, we adopt two newly designed search equations to generate candidate solutions, namely CABC_Elite as shown in (12) and (13). CABC_Elite limits the search range of the random solution in CABC to the elite solutions of the population, and provides enhanced exploitability. So the two formulas can also be regarded as the variants of CABC.

$$X'_{i,j} = X_{pbest,j} + \varphi_{i,j} \times \left(X_{pbest,j} - X_{r1,j}\right) \tag{12}$$

$$X'_{i,j} = X_{pbest,j} + \varphi_{i,j} \times \left(X_{r1,j} - X_{r2,j}\right) \tag{13}$$

where $X_{pbest,j}$ represents the elite individual in the $top20\%$ population, and $r1 \neq r2 \neq pbest$. $\varphi_{i,j}$ is a uniform random number on $[-1.0, 1.0]$.

In employed bee phase, the search efforts should be focused exploration. To find better solutions, we use "DE/rand-to-best/1" that the individuals of population can search more areas. In onlooker bee phase, CABC_Elite can improve convergence speed efficiently by utilizing the information of high-quality solutions. In the later stage of evolution, the population falls into the local optimum. The selection of two formulas of CABC_Elite strategy provides different perturbation, which increases the possibility of escaping from localization trap.

Literature [42] proposed the more detailed definition of exploration and exploitation. Any search strategy has the ability to explore and exploit. The exploration process increases searching ability far from the current nearest neighborhood, while the exploitation process increases searching ability around with the current nearest neighborhood. In AMSABC, how to coordinate the proportion of exploration and exploitation is very important to take full advantage of the different search strategies. We introduce an adaptive mechanism to combine each search equation in the next section.

B. ADAPTIVE MECHANISM

To achieve the most satisfactory optimization performance by applying the above search strategies, it is common to perform a trial-and-error method for the most appropriate search strategy. Therefore, it may expend massive correlation calculation. Experience shows that the different search strategy can be more effective than single strategy in the different stages of evolution. Motivated by these observations, the adaptive mechanism is described as follows, in which the search strategies can be gradually self-adapted according to their previous experiences of generating promising solutions.

In our proposed method, a strategy parameter $pf \in (0, 1)$ is used to control the selection of the strategy.

$$pf_G = (1 - c) \times pf_G + c \times mean_A\left(p_{1,G}\right) \tag{14}$$

where c is a positive constant in $[0, 1]$, and $mean_A(\cdot)$ is the usual arithmetic mean operation. If $c = 0$, no adaptive strategy takes place. If $c = 1$, only the mean value of $p_{1,G}$ is active. Therefore, the value $0 < c < 1$, both the previous pf_G and the mean value of $p_{1,G}$ are active, as detailed state in Section V.B. Obviously, the previous strategy parameter and the current successful strategy values affect the strategy selection in the next generation.

The probability of applying the *kth* strategy at the generation G is $p_{k,G}$, $k = 1, 2, \cdots K$, where K is the total number of strategies in the strategy pool. To ensure that the probabilities of choosing strategies are always summed to 1, we further divide $S_{k,G}$ by $\sum_{k=1}^{K} S_{k,G}$ to calculate $p_{k,G}$. Obviously, the larger the success rate for the *kth* strategy, the higher the probability to be selected to generate solution. $p_{k,G}$ is updated in the following manner.

$$p_{k,G} = \frac{S_{k,G}}{\sum_{k=1}^{K} S_{k,G}} \tag{15}$$

$$S_{k,G} = \frac{ns_k}{ns_k + nf_k} + \varepsilon \tag{16}$$

where $S_{k,G}$ represents the success rate of candidate solutions generated by the *kth* strategy; ns_k and nf_k are the respective numbers of the offspring vectors generated by the *kth* strategy that survive or fail in the selection operation; ε is a small constant value to avoid the possible null success rates, the value of ε is 0.01.

C. ENCODING AND DECODING

Literature [43] proposed a feasible solution representation for the IPPS which uses three discrete vectors $X = \{X^{ors}, X^{os}, X^{ms}$. The first vector is named OR sequence, which is relevant to the process plan to decide which OR link path is chosen, i.e., $X^{ors} = \{0\,1\,0\,1\}$. Based on Fig. 3, $X^{ors}(4) = 1$ corresponding to the operation 13 is selected. These three OR link paths, from job 2, correspond to the values $6 - 11 - 13 - 15 - 16$ are selected in the operation sequence vector X^{os}. It's different from the operation representation form in literature [43]. The operation sequence vector contains non-repeated integer values, and

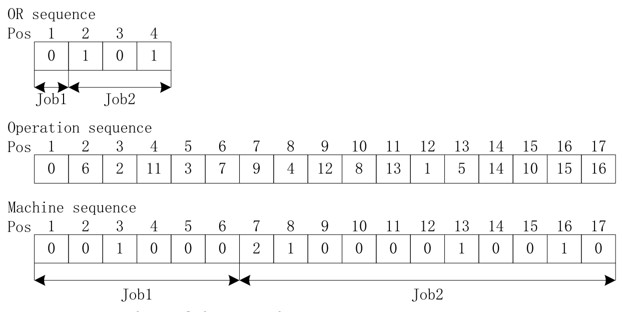


FIGURE 3. Encoding of the sample IPPS instance.

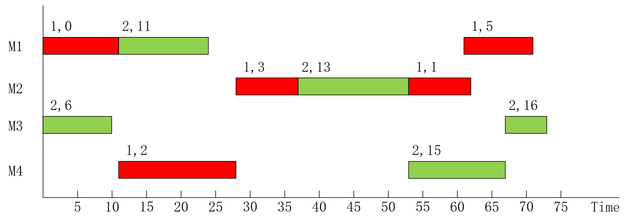


FIGURE 4. The Gantt chart of the IPPS instance. ($C_m = 73, W_m = 34, W_t = 115$)

its length equals the total number of operations, i.e., $X^{os} =$ {0 6 2 11 3 7 9 4 12 8 13 1 5 1 4 10 15 16. For example, the first element in the operation sequence represents the first operation of the job 1 (see Fig. 2). Then, the following element expresses the first operation of job 2, while the last element is the last operation of job 2. The third vector is the machine sequence, each element of which represents the corresponding selected machine for the operation, i.e., $X^{ms} =$ {0 0 1 0 0 0 2 1 0 0 0 0 1 0 0 1 0. Based on Fig. 3, X^{ms} (6) $= 0$ considers using machine 1, for the thirteenth value in X^{os}, i.e., X^{os} (13) $= 5$. The value 5 in X^{os} (13) represents the sixth operation of job 1.

Because each element in X^{os} can be located in any position, as any permutation-based representation, the operation sequence doesn't must be feasible, the encoding is not directly decoded. Meanwhile, the operations, together with the corresponding machines, that belong to unselected OR link paths, are first deleted. And then the order appearing in the operation sequence is converted into unique active scheduling. This procedure is an improved Giffler and Thompson algorithm which was originally proposed to generate active schedules for the JSP [44].

The solution representation mentioned above, depicted in Fig. 4. Based on Table 2, Numbers (in the form of [job, operation]) outside the blocks are the operation associated with the job.

D. POPULATION INITIALIZATION

In order to generate a feasible operation sequence and load balance phenomenon between the machines in a process plan. The initialization procedure by effective heuristics is as follows:

Step 1: OR connection of each job, which represents OR link paths, is initiated by randomly generating an integer for each component of the job.

Algorithm 1 Improved Giffler and Thompson Algorithm

```
(1) The operations, together with the corresponding machines, that belong to unselected OR link paths, are first deleted from each chromosome.
(2) Set a set of the first operations of every jobs $Q(t) = \{O_{ij} | i = 1, \ldots, n; j = 1, \ldots, m\}$.
(3) Set $t = 1$, and perform the following steps until $t = OperationNumber$.
(4) Set $O^*$ is an operation based on $c(O^*) = min\{c(O_{ij}|O_{ij} \in S(t))\}$, $m^*$ is the machine that performs the operation $O^*$, and making sure $O_{im^*}$ is an operation of set $\{O_{im^*} \in S(t)$ according to the precedence between operations of each job in $X^{os}$; $r(O_{im^*}) < c(O^*)\}$
(5) Each individual decides the next operation $Q(t+1) = Q(t) \backslash \{O_{im^*}\}$ based on selection from the network graph. Through removing the operation $o_{im^*}$ and appending the next operation of job $i$, can get a new set $S(t+1)$.
(6) If $Q(t+1)$ is no empty, then let $t = t + 1$, and go to STEP 4. Otherwise discard it.
```

Step 2: Set an initial available set of operations, which includes the first operations of each jobs.

Step 3: Perform the following steps until the available set is empty.

Step 4: Select an operation from the available set at random, and randomly assign a machine in the set of machines.

Step 5: Update the available set by removing the selected operation and appending the all immediate successor of the operation if the immediate predecessors of the successor are already selected. Go to Step 3.

E. NEIGHBORHOOD STRUCTURE

Considering the problem characteristics and the balance of exploration and exploitation, we perform the neighborhood structure to OR sequence, operation sequence and machine sequence in simultaneously. The neighborhood structure is given as follows.

$$x'_{i,j,l} = x_{a,j,l} + \varphi_{i,j,l} \times \left(x_{b,j,l} - x_{c,j,l}\right) = x_{a,j,l} \oplus \delta_{i,j,l}$$

$$= \begin{cases} mod\left(round\left(X^{ors}_{a,l} + \delta^{ors}_{i,l} + no\right), no\right) \\ OR\ sequence \\ mod\left(round\left(x^{os}_{a,j} + \delta^{os}_{i,j} + n\right), n\right) \\ Operation\ sequence \\ mod\left(round\left(X^{ms}_{a,j} + \delta^{ms}_{i,j} + mac[j]\right), mac[j]\right) \\ Machine\ sequence \end{cases} \tag{17}$$

$$\delta_{i,j,l} \Leftrightarrow \begin{cases} \delta^{ors}_{i,l} = \varphi_{i,j} \times \left(X^{ors}_{b,l} - X^{ors}_{c,l}\right) \ OR\ sequence \\ \delta^{os}_{i,j} = \varphi_{i,j} \times \left(x^{os}_{b,j} - x^{os}_{c,j}\right) \ Operation\ sequence \\ \delta^{ms}_{i,j} = \varphi_{i,j} \times \left(X^{ms}_{b,j} - X^{ms}_{c,j}\right) \ Machine\ sequence \end{cases} \tag{18}$$

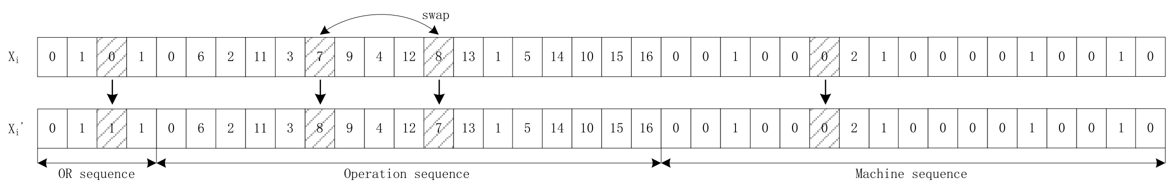


FIGURE 5. Illustration of the neighborhood structure.

where *mod* is the remainder operator; *round* is the rounding operator; *mac*[*j*] is the optional machine number of operation *j*. And then, by using mutation operation, data exchange between $x_{i,j}$ and $x'_{i,j}$ is implemented in three sequences.

Similarly,

$$x'_{i,j,l} = x_{a,j,l} + \phi_{i,j,l} \times \left(x_{b,j,l} - x_{a,j,l}\right) + \varphi_{i,j,l} \times \left(x_{c,j,l} - x_{d,j,l}\right) = X_{a,j,l} \oplus \delta\,(1)_{i,j,l} \oplus \delta\,(2)_{i,j,l}$$
$$= \begin{cases} mod\left(round\left(X^{ors}_{a,l} + \delta\,(1)^{ors}_{i,l} + \delta\,(2)^{ors}_{i,l} + 2 \times no\right),\ no\right)\ OR\ sequence \\ mod\left(round\left(x^{os}_{a,j} + \delta\,(1)^{os}_{i,j} + \delta\,(2)^{os}_{i,j} + 2 \times n\right),\ n\right)\ Operation\ sequence \\ mod\left(round\left(X^{ms}_{a,j} + \delta\,(1)^{ms}_{i,j} + \delta\,(2)^{ms}_{i,j} + 2 \times mac[j]\right),\ mac[j]\right)\ Machine\ sequence \end{cases} \tag{19}$$

According to the formula described above, set $j = 6, l = 3$, $\phi_i = 0.5$, $\varphi_i = -0.3$, give individuals X_i, X_a, X_b, X_c and X_d from Table 2, i.e.,

$$X_i = \{0\ 1\ 0\ 1, 0\ 6\ 2\ 11\ 3\ 7\ 4\ 12\ 8\ 13\ 15\ 14\ 10\ 15\ 16,\ 0\ 0\ 1\ 0\ 0\ 0\ 2\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\},$$
$$X_a = \{1\ 0\ 0\ 1, 0\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16,\ 1\ 0\ 1\ 0\ 2\ 1\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 1\ 0\ 1\},$$
$$X_b = \{0\ 1\ 1\ 1, 6\ 7\ 8\ 9\ 10\ 11\ 12\ 13\ 14\ 15\ 16\ 0\ 1\ 2\ 3\ 4\ 5,\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 0\ 1\ 0\},$$
$$X_c = \{0\ 0\ 0\ 1, 0\ 6\ 1\ 7\ 2\ 14\ 3\ 9\ 4\ 10\ 5\ 12\ 11\ 8\ 13\ 15\ 16,\ 1\ 0\ 1\ 0\ 1\ 1\ 2\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\},$$
$$X_d = \{1\ 1\ 1\ 0, 11\ 0\ 2\ 12\ 1\ 8\ 12\ 3\ 15\ 16\ 4\ 5\ 6\ 7\ 14\ 9\ 1\ 0,\ 1\ 0\ 1\ 0\ 1\ 0\ 2\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\}.$$

The procedure of the mutation operator is as follows.

$$x'_{i,6,3} = x_{a,6,3} + \phi_i \times \left(x_{b,6,3} - x_{a,6,3}\right) + \varphi_i \times \left(x_{c,6,3} - x_{d,6,3}\right)$$
$$\delta\,(1)^{ors}_{i,3} = \phi_i \times \left(X^{ors}_{b,3} - X^{ors}_{a,3}\right) = 0.5 \times (1 - 0) = 0.5$$
$$\delta\,(2)^{ors}_{i,3} = \varphi_i \times r\left(X^{os}_{c,3} - X^{ors}_{d,3}\right) = -0.3 \times (0 - 1) = 0.3$$
$$x^{ors'}_{i,3} = mod\left(round\left(x^{ors}_{a,3} + \delta\,(1)^{ors}_{i,3} + \delta\,(2)^{ors}_{i,3} + 2 \times 4\right),\ 4\right) = mod\left(round\left(0 + 0.5 + 0.3 + 2 \times 8\right),\ 4\right) = 1$$
$$\delta\,(1)^{os}_{i,6} = \phi_i \times \left(X^{os}_{b,6} - X^{os}_{a,6}\right) = 0.5 \times (11 - 5) = 3$$
$$\delta\,(2)^{os}_{i,6} = \varphi_i \times \left(X^{os}_{c,6} - X^{os}_{d,6}\right) = -0.3 \times (14 - 8) = -0.18$$
$$x^{os'}_{i,6} = mod\left(round\left(x^{os}_{a,6} + \delta\,(1)^{os}_{i,6} + \delta\,(2)^{os}_{i,6} + 2 \times 17\right),\ 17\right) = mod\left(round\left(5 + 3 - 0.18 + 2 \times 17\right),\ 17\right) = 8$$
$$\delta\,(1)^{ma}_{i,6} = \phi_i \times \left(X^{ma}_{b,6} - X^{ma}_{a,6}\right) = 0.5 \times (0 - 1) = -0.5$$
$$\delta\,(2)^{ma}_{i,6} = \varphi_i \times \left(X^{ma}_{c,6} - X^{ma}_{d,6}\right) = -0.3 \times (1 - 0) = -0.3$$
$$x^{ma'}_{i,6} = mod\left(round\left(x^{ma}_{a,6} + \delta\,(1)^{ma}_{i,6} + \delta\,(2)^{os}_{i,6} + 2 \times 17\right),\ 17\right) = mod\left(round\left(1 - 0.5 - 0.3 + 2 \times 2\right),\ 2\right) = 0$$

The swap operation is implemented in operation sequence. Fig. 5 gives an example of the neighborhood structure.

F. PROBLEM-SPECIFIC MULTI-OBJECTIVE LOCAL SEARCH

To further improve the searching ability of the proposed algorithm, we develop a problem-specific multi-objective local search method for the non-dominated solutions found in the current iteration. It contains both the N$_5$ neighborhood structure [45] and the intensification search. The first one is applied to optimize the makespan criterion. The second one is then applied to improve the critical machine workload and the total workload. The Local search tries to perform a slight perturbation to the individual of optimal solution set. The two methods are randomly executed one at a time when the local search called for it.

The N$_5$ neighborhood structure is defined by the interchange of two successive tasks, where either task is an operation in a block that belongs to a critical path. More precisely we swap the first two (and last two) operations in every block, each of which contains at least two operations. In the first block only the last two operations are swapped, and via symmetry in the last block, only the first two are swapped. The illustration is given at the bottom of Fig. 6(a).

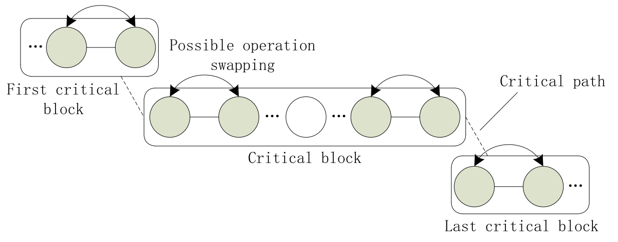
The intensification search tries to randomly remove an operation to the processed by the machine whose workload is largest to other available machines. As illustrated in Fig. 6(b), the machine of the operation is changed from the alternative ones at random. During the optimization process, the critical machine workload and the total workload are perturbation at the same time.

G. POPULATION SELECTION PROCEDURE

The selection process is hierarchical, following the prescriptions captured in NSGA-II [46]. The individuals have to go through two rounds of selection. In the first stage, the combined population R_t ($R_t = P_t \cup Q_t$), which P_t is the parent population, and Q_t is the child population, is sorted in ascending order according to non-domination sorting algorithm. The new population is formed by adding solution from the best non-dominated set (F_1) to subsequent non-dominated sets in

Algorithm 2 Algorithm AMSABC

```
Step 1. Initialization phase
    Initialize Parameters: $limit$, $trail\ (i)$, $pf$, FoodNumber $(FN) = NP/2$, MaximumCycleNumber($MCN$), $ns_k$, $nf_k$.
    Initial the population (see Section IV.D).
    Evaluate the objective values of each solution. Record the non-dominated solutions as the best solutions.
    If the stopping criterion is satisfied, output the best solutions so far. Otherwise, perform following steps.
for $cn = 1$ to $MCN$ do
Step 2. Employed bee phase
    for $i = 1$ to $FN$ do
        Randomly choose $j$ from $\{1, 2, \cdots, NP\}$ and $l$ from $\{1, 2, \cdots, NO\}$.
        Randomly choose $X_{best}$ from the non-dominated set.
        Randomly choose $r1$, $r2$, $r3$ from $\{1, 2, \cdots, FN\}$, $r1 \neq r2 \neq r3 \neq best$.
$X'_{i,j} = X_{r1,j} + \phi_{i,j} \times \left(X_{best,j} - X_{r1,j}\right) + \varphi_{i,j} \times \left(X_{r2,j} - X_{r3,j}\right)$
                $//\phi_{i,j} \in [0, 1.0]$, $\varphi_{i,j} \in [-1.0, 1.0]$
        if $f\left(X'_i\right) < f\left(X_i\right)$ do
          $X_i = X'_i$; trail (i) = 0;
        else
          $trail\ (i) = trail\ (i) + 1$;
        end if
      end for
Update the best solution achieved so far.
Step 3. Onlooker bee phase
for $i = 1$ to $FN$ do
        Randomly choose $j$ from $\{1, 2, \cdots, NP\}$ and $l$ from $\{1, 2, \cdots, NO\}$.
        Randomly choose $X_i$ from $top20\%$.
        Randomly choose $X_{pbest}$ as one of the $top20\%$ best solutions.
        if $rand\ (0, 1) < pf$ do
          $flag = 0$
          Randomly choose $r1$ from $\{1, 2, \cdots, FN\}$, $r1 \neq pbest$.
            $X'_{i,j} = X_{pbest,j} + \varphi_{i,j} \times \left(X_{pbest,j} - X_{r1,j}\right) //\varphi_{i,j} \in [-1.0, 1.0]$
        else
          $flag = 1$
          Randomly choose $r1, r2$ from $\{1, 2, \cdots, FN\}$, $r1 \neq r2 \neq pbest$.
          $X'_{i,j} = X_{pbest,j} + \varphi_{i,j} \times \left(X_{r1,j} - X_{r2,j}\right) //\varphi_{i,j} \in [-1.0, 1.0]$
        end if
        if $f\left(X'_i\right) < f\left(X_i\right)$
          $X_i = X'_i$; trail (i) = 0;
          if $flag = 0$, $ns_1$ ++; otherwise, $ns_2$ ++.
        else
          $trail\ (i) = trail\ (i) + 1$;
          if $flag = 0$, $nf_1$ ++; otherwise, $nf_2$ ++.
        end if
      end for
Update $pf$ with the strategy success rate (see Section IV.B).
Select solutions of the next generation (see Section IV.G).
Update the best solution achieved so far.
Step 4. Scout bee phase
        Randomly choose $trail(i)$ from the maximums of $trail(\cdot)$.
        if $trail\ (i) > limit$ do
          Replace $X_i$ with a new randomly produced solution.
          $trail\ (i) = 0$
        end if
Step 5. Local search phase
    For each non-dominated solutions found in the current iteration, perform the local search process (see Section IV.F).
    Replace the worst solution with the best one of the newly generated solutions randomly.
end for
```



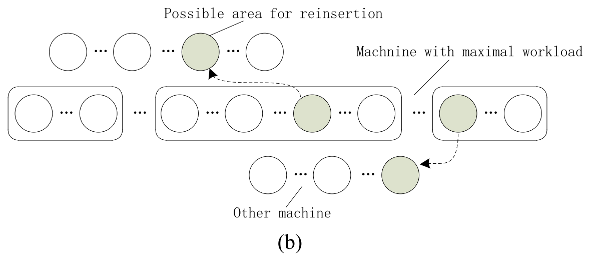


FIGURE 6. Local search methods. (a) N_5 (b) intensification search.

rank order ($F_2, F_3, \ldots$). The procedure is continued until the size exceeds NP. Thereafter, the solutions of the last non-dominated set that reach the second stage are sorted according to the crowding distance value. In order to use crowding distance value sorting to choose the NP population members, it is necessary to fill all population slots in descending order of neighboring distance.

H. FRAMEWORK OF PROPOSED ALGORITHM

The corresponding pseudocode of the proposed AMSABC is given as follows.

I. COMPLEXITY ANALYSIS OF AMSABC ALGORITHM

Compared with the original ABC, AMSABC has additional computation burden on the selection process and the problem-specific local search. First, the individuals in the population are sorted and shared according to NSGA-II. The computational complexity of this process is $O(m \cdot NP^2)$, where m is the number of objectives. Secondly, the problem-specific local search contains both N_5 and the intensification search. The two methods are randomly executed every time. The computational complexity of local search is $O(D \cdot NP^2)$, where D is the dimension of feasible solutions pace. Since the complexity of the original ABC is $O(D \cdot NP)$, the overall complexity of AMSABC is $O(D \cdot NP^2)$.

It should be noted that the additional computational cost by computing the pairwise distance between individual is negligible for the IPPS with costly evaluation [47]. Therefore, the additional overhead caused by the introduced operations is relatively small when the IPPS evaluation is costly.

V. EXPERIMENTAL RESULTS AND ANALYSIS

To verify the superiority of the proposed AMSABC for the IPPS problem, numerical experiments and discussion are conducted here. The model and algorithm are coded in C#. All the experiments are run on a computer with Intel(R) Core^{TM}i5-6500 CPU @3.20GHz with 8GB RAM.

TABLE 3. Characteristic of jobs.

Job	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
n	8	12	19	13	12	16	14	12	17	9	9	16	11	11	13	13	15	13
TO	8	14	19	16	18	20	21	20	20	11	9	18	18	13	15	21	22	17
PF	1	2	1	4	3	3	9	10	8	2	1	2	5	4	4	6	12	8
SF	0.14	0.61	0.76	0.38	0.51	0.72	0.33	0.41	0.72	0.39	0.67	0.74	0.33	0.58	0.73	0.38	0.69	0.72
OF	1.62	2.42	4.10	1.81	4.05	2.60	2.47	1.80	3.80	2.27	3.55	1.83	3.61	2.46	1.87	3.67	1.95	2.29
Flexibility	LLL	LMM	LHH	MLL	MMH	MHM	HLM	HML	HHH	LLM	LMH	LHL	MLH	MMM	MHL	HLH	HML	HHM

TABLE 4. The proportion influence on the strategy parameter *pf* from the value of *c*.

c	evolution generations									
	5	10	15	20	25	30	35	40	45	50
0.05	77.38%	59.87%	**46.33%**	35.85%	27.74%	**21.46%**	16.61%	12.85%	**9.94%**	7.69%
0.1	59.05%	34.87%	**20.59%**	12.16%	7.18%	**4.24%**	2.50%	1.48%	**0.87%**	0.52%
0.15	44.37%	19.69%	8.74%	3.88%	1.72%	0.76%	0.34%	0.15%	0.07%	0.03%
0.2	32.77%	10.74%	3.52%	1.15%	0.38%	0.12%	0.04%	0.01%	0.00%	0.00%
0.25	23.73%	5.63%	1.34%	0.32%	0.08%	0.02%	0.00%	0.00%	0.00%	0.00%
0.3	16.81%	2.82%	0.47%	0.08%	0.01%	0.00%	0.00%	0.00%	0.00%	0.00%

All experimental instances, experimental settings, performance metrics, results, and discussions are presented in the following subsections.

A. EXPERIMENTAL SETTINGS

The experimental instances are adopted from Kim [8], which is the most famous IPPS benchmark. Many researchers have worked with it for several years, and the single-objective (makespan) results have been pushed to the limit almost [43]. In this experiment, the 24 test-bed instances are constructed with 18 jobs. In each instance, there are a total of 15 machines to process the jobs. The number of operations of these instances varies from 79 to 300. The characteristic of the jobs is listed in Table 3. In this table, the total number of operations (TO) is all possible operations of each job. *HLM* means the job has high PF, low SF, and medium OF, and other description of flexibility likes it.

B. EXPERIMENTAL PARAMETERS

Based on the existing literature [8] and our experimental observation, the population size *NP* is 100, *limt* is 50, and the food number *FN* is 50. Each implemented algorithm will be terminated when the predefined maximum *FEs* of examined solutions is exhausted. The *FEs* is set to 5.0E+04, 7.0E+04, 1.3E+0.5, 2.0E+05 and 2.5E+05 for the test-bed instances 1-9, 10-15, 16-21, 22-23 and 24, respectively. Each instance is optimized over 30 independent runs. Moreover, we use the same set of initial random populations to evaluate different algorithms.

The value of c on the strategy parameter pf is used to control the proportion influence of historical experience. Obviously, a smaller value of c concentrates more on historical experience, while a larger value more on recent experience. It can be seen from Table 4, after the evolution of 15, 30 and 45 generations, when $c = 0.05$, the proportion influences of pf are 46.33%, 21.46% and 9.94% respectively; when $c = 0.1$, the influences are only 20.59%, 4.24% and 0.87%, respectively. In order to minimize the impact of random assignment on evolutionary process, a smaller value of c can be accepted in this article, that is $c = 0.05$.

C. PERFORMANCE METRICS

Multi-objective optimization problem performance measures are more complex than those of single-objective optimization problems. We are interested in the quality of the obtained Pareto fronts in terms of convergence to the Pareto optimal set and maintenance of diversity in solutions on the obtained Pareto front. The following performance metrics for multi-objective optimization are used in this article to evaluate.

1) IVERTED GENERATIONAL DISTANCE (IGD)

Let P^* be a set of uniformly distributed points along the true Pareto front (PF). Let A be the approximation PF obtained by the compared algorithm. The metric IGD is defined as

$$IGD\left(A, P^*\right) = \frac{1}{|P^*|} \sum\nolimits_{x \in P^*} \min\nolimits_{y \in A} d\left(x, y\right) \tag{20}$$

where $d(x, y)$ is the Euclidean distance (measured in objective space) between the points x and y. The IGD can well reflect the diversity and convergence of A [48]. To have a smaller value of IGD, A must be enough close to P^*.

Because the true Pareto front is unknown in the simulation here, we adopt a reference Pareto front PF_{ref} in performance evaluation formulas. In this article, PF_{ref} includes the non-dominated solutions extracted from all results found by the experimental methods in all dependent runs. In addition, all objective vectors of the Pareto solutions are normalized, which could be obtained by

$$\tilde{f}_i(x) = \frac{\left(f_i(x) - f_i^{min}\right)}{\left(f_i^{max} - f_i^{min}\right)} \tag{21}$$

where f_i^{max} and f_i^{min} are the maximal and minimal values of $f_i(x)$ among all non-dominated solutions obtained by gathering all experimental methods.

TABLE 5. Performance evaluation of the effect of local search using average IGD and *C* values.

Instance	Jobs	Operations	Characteristic	IGD		AMSABC(A) vs AMSABC$_{EL}$(B)	
				AMSABC	AMSABC$_{EL}$	$C(A,B)$	$C(B,A)$
1	6	79	Low PF	0.1032	0.2500	0.0000	0.0000
2	6	100	Medium PF	0.1703	0.1333	0.0250	0.0000
3	6	121	High PF	0.0360	0.0518	**0.4756**	0.2121
4	6	95	Low SF	0.0254	0.0298	0.0000	0.0000
5	6	96	Medium SF	**0.0298**	0.0667	0.1667	0.0345
6	6	109	High SF	0.0441	0.0537	**0.1333**	0.0333
7	6	99	Low OF	0.0473	0.0218	0.0000	0.0000
8	6	96	Medium OF	0.0304	0.0537	0.2585	0.3333
9	6	105	High OF	0.0233	0.0285	**0.3012**	0.1724
10	9	132	Low or Medium PF	0.1737	0.1256	**0.3730**	0.0669
11	9	168	Medium or High PF	**0.0404**	0.0789	**0.5218**	0.0000
12	9	146	Low or Medium SF	**0.0733**	0.2687	0.2326	0.0065
13	9	154	Medium or High SF	0.0651	0.0400	0.3500	**0.4908**
14	9	151	Low or Medium OF	0.0398	0.0240	0.3451	0.4000
15	9	149	Medium or High OF	0.0278	**0.0114**	0.0264	0.0333
16	12	179	Low or Medium PF	0.0402	0.0541	0.4055	0.0386
17	12	221	Medium or High PF	0.0142	0.0521	0.1326	0.0160
18	12	191	Low or Medium SF	0.0186	0.0250	0.2366	0.0333
19	12	205	Medium or High SF	0.0473	0.0893	0.2505	0.1333
20	12	195	Low or Medium OF	0.0254	0.0304	0.2685	0.0800
21	12	201	Medium or High OF	**0.0333**	0.0765	0.5218	0.0358
22	15	256	—	**0.0405**	0.1333	**0.7050**	0.2348
23	15	256	—	**0.0113**	0.1450	**0.8130**	0.2630
24	18	300	—	**0.0385**	0.2349	**0.5533**	0.1584
Wilcoxon test					73 228		
$R - R + p\_value$					0.0268		

2) C-METRIC (C)

Let A and B be approximates PF that are obtained by two different algorithms. This indicator represents the percentage of solutions in B that are dominated by at least one solution in A. The value metric is to the interval [0, 1].

$$C\left(A,B\right)=\frac{\left|\{b\in B|\exists a\in A:a\succ b\}\right|}{|B|} \tag{22}$$

$C\left(A,B\right)=1$ means that all solutions in B are dominated by or equal to solutions in$A$. The opposite, $C\left(A,B\right)=0$, represents the situation when none of the solutions in B is covered by the set A. Note that $C\left(B,A\right)$ is not necessarily equal to $1-C\left(A,B\right)$. If $C\left(A,B\right)$ is larger than $C\left(B,A\right)$, then A is better than B in a sense.

D. EFFECTIVENESS OF LOCAL SEARCH STRATEGY

To investigate the impact of the problem-specific multi-objective local search on the proposed AMSABC, the performance comparison between AMSABC and AMSABC$_{EL}$ is carried out in this section. Where AMSABC$_{EL}$ is a simplified algorithm designed by eliminating local search from AMSABC.

In Table 5, average IGD and C values over 30 independent runs on all 24 test-bed instances are presented. The characteristics of the instances are also provided. The second column and third column show the number of jobs and the number of operations for each instance, respectively; the fourth column lists the level of the three types of flexibility. The Wilcoxon signed-rank test [49] is further carried out on the sample data of the two metrics obtained by the compared algorithms with the significance of 0.05. For each instance, the result that is significantly better than the others is marked in bold. (with smaller IGD, and with greater C).

As can be seen from Table 5, AMSABC can obtain better IGD results for 18 problems (the improvement is significant for 7 problems). In contrast, AMSABC$_{EL}$ only achieves a significantly better value on problem 16. In addition to the refreshed solutions, the advantage of AMSABC is mainly concentrated after problem 16, indicating that the problem-specific local search method has obvious competence for solving large-scale (problems 16-24) problems. As for the metric C, AMSABC is significantly better than these 8 problems, whereas AMSABC$_{EL}$ obtained only one optimal value. The main reason for this effect is that by applying the local search heuristic, the proposed AMSABC enhances the objective-specific exploitation ability.

To further verify the effectiveness of AMSABC, non-parametric test was performed by using SPSS based on the experimental results in Table 5. It can be seen from Wilcoxon's test results that the p_value is less than 0.05, indicating that it is significantly different form AMASABC.

TABLE 6. Comparison of the different search strategies in onlooker bee using average IGD and C values.

Instance	IGD			AMSABC(A) vs CABC_Elite1(B)		AMSABC(A) vs CABC_Elite2(C)	
	AMSABC	CABC_Elite1	CABC_Elite2	$C(A,B)$	$C(B,A)$	$C(A,C)$	$C(C,A)$
1	0.1032	0.1850	0.1100	0.0000	0.0000	0.0000	0.0000
2	0.1703	0.2743	0.1747	1.0000	0.0000	1.0000	0.0000
3	0.0360	**0.0256**	0.0333	0.0414	0.0000	1.0000	0.0000
4	**0.0254**	0.0443	0.0356	**1.0000**	0.0000	1.0000	0.0000
5	0.0298	0.1004	0.0674	**1.0000**	0.0000	0.3133	0.0000
6	0.0441	0.0583	0.0626	**1.0000**	0.0000	**1.0000**	0.0000
7	0.0473	0.0704	0.0598	**1.0000**	0.0000	**1.0000**	0.0000
8	0.0304	0.0234	0.0333	**1.0000**	0.0000	**1.0000**	0.0000
9	0.0233	0.2264	0.1280	**1.0000**	0.0000	**1.0000**	0.0000
10	0.1737	0.2373	0.1521	**1.0000**	0.0000	**1.0000**	0.0000
11	0.0404	0.0333	**0.0250**	**1.0000**	0.0000	**1.0000**	0.0000
12	0.0733	0.1112	0.1151	**1.0000**	0.0000	**1.0000**	0.0000
13	0.0651	0.0733	0.0532	**0.9169**	0.0000	**1.0000**	0.0000
14	**0.0398**	0.0857	0.0734	**1.0000**	0.0000	**0.9333**	0.0000
15	0.0402	0.0504	0.1785	**1.0000**	0.0000	**1.0000**	0.0000
16	0.0278	0.0420	0.0521	**1.0000**	0.0000	**1.0000**	0.0000
17	**0.0142**	0.0653	0.0323	**1.0000**	0.0000	**1.0000**	0.0000
18	**0.0186**	0.0333	0.0225	**1.0000**	0.0000	**1.0000**	0.0000
19	**0.0473**	0.0869	0.0722	**1.0000**	0.0000	**1.0000**	0.0000
20	**0.0254**	0.0755	0.1350	**1.0000**	0.0000	**1.0000**	0.0000
21	**0.0333**	0.0462	0.0253	**0.8757**	0.0000	**1.0000**	0.0000
22	**0.0405**	0.1703	0.2500	**1.0000**	0.0000	**1.0000**	0.0000
23	**0.0113**	0.3333	0.1699	**1.0000**	0.0000	**1.0000**	0.0000
24	**0.0385**	0.2765	0.1141	**1.0000**	0.0000	**1.0000**	0.0000
Wilcoxon test		8 292	38 262				
$R - R + p\_value$		0.0005	0.0014				

E. EFFECTIVENESS OF MULTI-STRATEGY

To show the positive effect of the adoption of multi-strategy collaboration, we replaced the CABC_Elite with a single search equation (denoted as CABC_Elite1 and CABC_Elite2). CABC_Elite1 is different from CABC_Elite2 only in that it uses the Eq. (12) in onlooker bee instead of the Eq. (13).

It can be observed from Table 6 that the multi-strategy collaboration in onlooker bee further improves the performance of the proposed AMSABC. Specifically, for the metric IGD, AMSABC is significantly better than CABC_Elite1 and CABC_Elite2 on 10 problems, and are significantly outperformed by CABC_Elite1 and CABC_Elite2 on only problem 3 and problem 11, respectively. Meanwhile, most of the C values of AMSABC take value 1 or the value very close to 1, indicating that AMSABC effectively prevents the algorithm from falling into local optimum by using two adaptive strategies, and makes the calculation more efficient. The p_value is close to zero; hence, there are significant differences between the compared methods. It can be concluded that the adoption of multi-strategy improves the exploration abilities.

In order to further prove the validity of the proposed adaptive mechanism, we select some instances to further test. Fig. 7 is the varying situations of pf on instances 1, 15, 23 and 24. Data acquisition is performed every 10 generations (the corresponding generation number of the instance 24 is 20) in the figure. From the transformation curves in the graphs, the variation trends of pf are significantly different for different instances.

As can be seen from the above figure, when $pf > 0.5$, it means that CABC_Elite1 has more opportunities to be used than CABC_Elite2, conversely, it means that CABC_Elite1 has less opportunities to be chosen. For different instances, the change trend of pf is different. For example, we take instance 23 as an example to analyze how the two strategies work. At first, pf decline is likely to fluctuate around 0.35, indicating that CABC_Elite2 had 65% chances to be selected. This is because the population is concentrated on exploration. Then, as the population evolution, pf fluctuated up and down with 0.5. Finally, the pf tends to be near 0.6, which indicates that AMSABC accelerates its convergence in the late evolutionary stage.

F. COMPARISON WITH OTHER STATE-OF-THE-ART ALGORITHMS

In this section, the proposed AMSABC is compared with the existing state-of-the-art methods, which are called modified genetic algorithm (MGA) [43], multi-objective memetic algorithm (MOMA) [10], CABC [41], hybrid genetic algorithm and variable neighborhood search (GAVNS) [50], and enhanced ACO (EACO) [51], respectively. To make a fair comparison, CABC uses same parameter settings as AMSABC, and other algorithms are the same as original

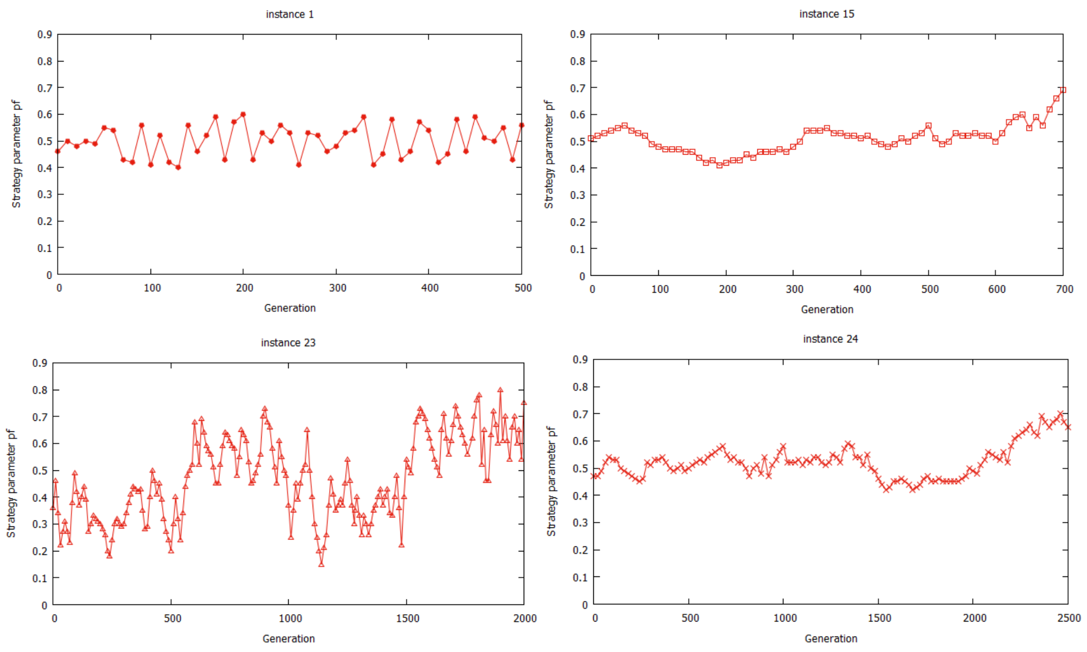


FIGURE 7. Variation trends of parameter pf on four instances.

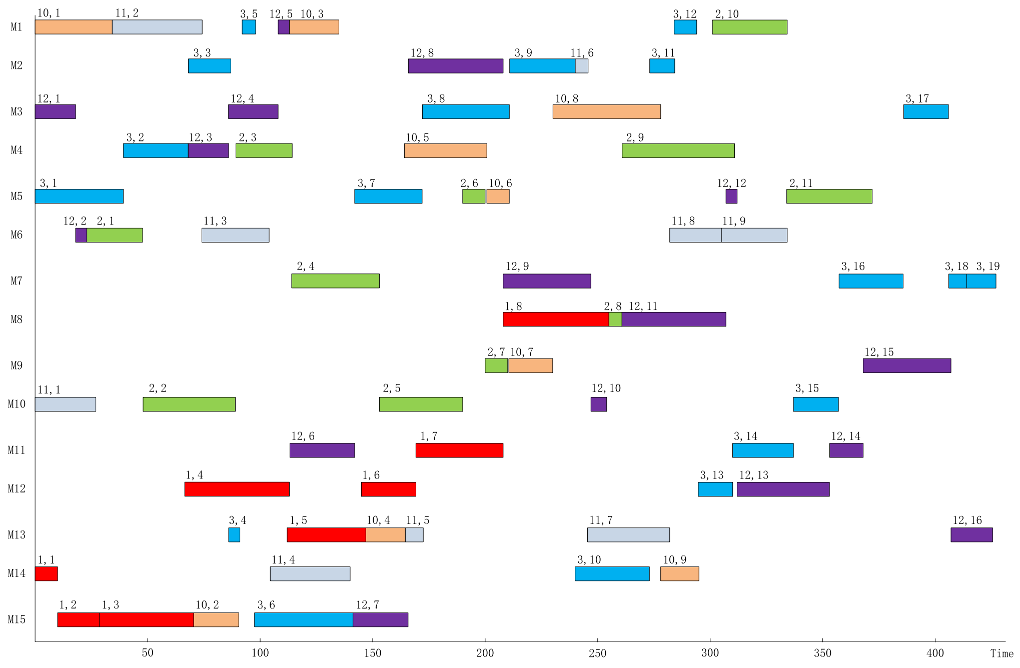


FIGURE 8. The Gantt chart for the solution of IPPS instance 1. ($C_m = 427, W_m = 150, W_t = 1822$).

literature. It should be pointed out that the two metrics are computed after predefined runs rather than each run.

As for the metric IGD, it can be seen from Table 7 that AMSABC yields the best results on 19 out of 24 instances, and the second-best IGD values on 3 instances. CABC performs best on 8 instances, MOMA performs best on 13 instances, MGA performs best on 17 instances, GAVNS performs best on 9 instances, and EACO also performs best on 9 instances. Obviously, only from the number of instances that perform best, the performance of AMSABC is significantly better than other algorithms except for MGA slightly better and CABC worse. Therefore, based on the experimental results of Table 7, SPSS is adopted for non-parametric test, and the test results are listed in the last two rows of Table 7. From the results of Wilcoxon test, it can be seen that the *p_value* of algorithms are less than 0.05 except MGA, which indicates that there are a significant difference between the compared algorithms and AMSABC, and AMSABC is far superior to them.

TABLE 7. **Performance evaluation of the effect of AMSABC and other algorithms using IGD values.**

Instance	AMSABC	CABC	MOMA	MGA	GAVNS	EACO
1	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
2	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
3	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
4	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
5	**0.0025**	0.0648	0.0355	0.1414	0.1414	0.1414
6	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
7	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
8	**0.0000**	**0.0000**	0.0250	**0.0000**	**0.0000**	**0.0000**
9	**0.0000**	0.1382	**0.0000**	**0.0000**	**0.0000**	**0.0000**
10	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
11	0.0404	0.3820	0.3333	**0.0000**	0.0669	0.0902
12	0.0733	0.0838	**0.0643**	0.0651	0.1525	0.2133
13	**0.0000**	0.0440	**0.0000**	**0.0000**	0.0074	0.0000
14	**0.0240**	0.2532	0.1429	0.0669	0.1343	0.0890
15	**0.0020**	0.3333	**0.0020**	**0.0020**	0.0307	0.0234
16	0.0278	0.0127	**0.0059**	0.1911	0.1911	0.2199
17	**0.0142**	0.2631	0.1420	**0.0142**	0.0358	0.1597
18	**0.0186**	0.0900	0.0250	**0.0186**	0.0687	0.0714
19	0.0193	0.0865	0.0366	**0.0155**	0.0799	0.0601
20	**0.0000**	0.0025	0.0019	**0.0000**	0.0254	0.0322
21	**0.0026**	0.0037	0.0027	0.0107	0.1118	0.2021
22	**0.0405**	0.1277	0.0783	0.0986	0.3478	0.5625
23	**0.0113**	0.3487	**0.0113**	0.0167	0.0413	0.0763
24	0.0085	0.0841	0.0289	**0.0063**	0.0427	0.1939
Wilcoxon test		4 132	11 80	14 41	0 120	0 105
$R - R + p\_value$		0.0009	0.0159	0.1688	0.0007	0.0010

For each instance, the minimal IGD values obtained by the compared algorithms are marked in bold.

TABLE 8. **Performance evaluation of the effect of AMSABC and other algorithms using C values.**

Instance	AMSABC(A) vs CABC(B)		AMSABC(A) vs MOMA(C)		AMSABC(A) vs MGA(D)		AMSABC(A) vs GAVNS(E)		AMSABC(A) vs EACO(F)	
	$C(A,B)$	$C(B,A)$	$C(A,C)$	$C(C,A)$	$C(A,D)$	$C(D,A)$	$C(A,E)$	$C(E,A)$	$C(A,F)$	$C(F,A)$
1	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
2	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
3	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
4	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
5	**0.6667**	0.0000	**0.9091**	0.0000	0.1724	**0.8000**	**0.7500**	0.0000	**0.7143**	0.0000
6	**1.0000**	0.0000	**1.0000**	0.0000	**1.0000**	0.0000	**1.0000**	0.0000	**1.0000**	0.0000
7	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
8	**0.1000**	0.0000	0.4444	**0.4545**	**0.0000**	**0.0000**	0.1667	**0.2121**	**0.0000**	**0.0000**
9	**0.4000**	0.1034	**0.2500**	0.0000	**0.3000**	0.1818	**1.0000**	0.0000	**0.3012**	0.1333
10	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
11	0.0000	**0.2000**	**0.1429**	0.1421	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
12	0.1429	**0.2857**	**0.0000**	**0.0000**	**0.3730**	0.1724	**0.0345**	0.0000	**0.2500**	0.1181
13	**0.0264**	0.0000	0.0000	**0.1000**	0.0264	**0.5000**	**0.2412**	0.0870	0.0264	**0.3333**
14	0.0000	**0.5000**	**0.1429**	0.0000	0.0000	**0.0250**	**0.0000**	**0.0000**	**0.3487**	0.0000
15	**0.7410**	0.0347	0.1327	**0.0333**	**0.4545**	0.0000	**0.6350**	0.1875	**0.8571**	0.0000
16	**0.7817**	0.3333	**0.5328**	0.0000	**0.1111**	0.0000	**0.5833**	0.0000	**0.6667**	0.0000
17	**0.2500**	0.0000	**0.5877**	0.2414	**0.1818**	0.1429	**0.4000**	0.1034	0.0000	**0.1111**
18	**0.0438**	0.0125	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
19	**0.6667**	0.0000	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0250**	0.0000	**0.0000**	**0.0000**
20	**0.7625**	0.1243	0.1000	0.0000	**0.0345**	0.0000	**0.4215**	0.0015	**0.0813**	0.0000
21	**0.8284**	0.0121	0.0000	**0.1111**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.3478**	0.0000
22	**0.6250**	0.0000	**1.0000**	0.0000	**0.6000**	0.1747	**0.5714**	0.3215	**0.4444**	0.2314
23	**0.5612**	0.2651	**0.6667**	0.0438	0.1165	**0.6350**	**0.3650**	0.0328	**0.3289**	0.1165
24	**0.6000**	0.1143	**0.9091**	0.0653	0.3451	**0.4908**	**0.8817**	0.2347	**0.7255**	0.2028

For each instance, the greater C values obtained by the compared algorithms are marked in bold.

AMSABC is compared respectively with the other algorithms in Table 8 using C values. When comparing with CABC, AMSABC is worse than CBAC on problems 11, 12 and 14, but AMSABC is better than CABC on 15 instances. As for MOMA, it obtains the best C values on problems 8, 13, 15 and 21, which has the same mechanism to adaptive multi-method search. The refreshed solutions of the AMSABC and MGA are mainly concentrated after problem 16, indicating that two algorithms have obvious competence for solving large-scale problems. Compared with GAVNS and EACO, AMSABC is superior to them on major instances. We believe that the high performance of AMSABC stems from the adaptive multi-method search procedure and the problem-specific multi-objective local search method, because most of the non-dominated solutions obtained by other algorithms are dominated by the ones obtained by AMSABC.

Fig. 8 gives a Gantt chart for the best solution for instance 1, where each operation is represented by a rectangle labeled with the job number and the operation number. It can be observed from Fig. 8 that the solution obtained by the proposed algorithm is effective.

VI. CONCLUSION AND FUTURE WORK

In this article, we study the multi-objective IPPS problem with makespan, total workload, and critical workload criteria, which has a strong industrial background and is very close to the real manufacturing situation. An AMSABC algorithm is developed to solve the complex optimization problem. In this algorithm, each solution is represented by using three discrete vectors. Then, to decode each solution for feasible scheduling, we designed an improved Giffler and Thompson algorithm. Next, the corresponding neighborhood structure is adopted, which realizes the information exchange and exploit behavior of the individual in the process of optimization. In each evolution, two novel search strategies with different characteristics, and combines them through a new adaptive mechanism, are used by both the employed bee and onlooker bee procedures. It worth noting that the adaptive mechanism can dynamical adjusts exploration and exploitation by selecting the most suitable strategy to generate solutions at different stage of optimization. Finally, a problem-specific multi-objective local search has been embedded in the proposed algorithm to further improve the objective-specific exploitation performance. To validate the accuracy and performance of the proposed algorithm, 24 instances with different scales are used for simulation tests. Five efficient algorithms are selected for detailed comparisons. It can be obviously seen from the comparisons that the proposed approach is more effective at least equally.

In our future work, the self-adaptive selection of search strategies [26] and how dynamically adjust exploration and exploitation to the performance of ABC algorithm will need to be further investigated. Moreover, it's interesting to apply AMSABC to handle realistic applications, such as fuzzy production scheduling [52], [53], energy-efficient scheduling [54], [55], crowd evacuation simulation [25] and so on.

REFERENCES

[1] G. Chryssolouris, S. Chan, and N. P. Suh, "An integrated approach to process planning and scheduling," *CIRP Ann.*, vol. 34, no. 1, pp. 413–417, 1985.

[2] A. S. Jain and S. Meeran, "Deterministic job-shop scheduling: Past, present and future," *Eur. J. Oper. Res.*, vol. 113, no. 2, pp. 390–434, Mar. 1999.

[3] J. C. Beck and M. S. Fox, "Constraint-directed techniques for scheduling alternative activities," *Artif. Intell.*, vol. 121, nos. 1–2, pp. 211–250, Aug. 2000.

[4] Y. K. Kim, K. Park, and J. Ko, "A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling," *Comput. Oper. Res.*, vol. 30, no. 8, pp. 1151–1171, Jul. 2003.

[5] Q. Lihong and L. Shengping, "An improved genetic algorithm for integrated process planning and scheduling," *Int. J. Adv. Manuf. Technol.*, vol. 58, nos. 5–8, pp. 727–740, Jan. 2012.

[6] L. Zhang and T. N. Wong, "An object-coding genetic algorithm for integrated process planning and scheduling," *Eur. J. Oper. Res.*, vol. 244, no. 2, pp. 434–444, Jul. 2015.

[7] A. Seker, S. Erol, and R. Botsali, "A neuro-fuzzy model for a new hybrid integrated process planning and scheduling system," *Expert Syst. Appl.*, vol. 40, no. 13, pp. 5341–5351, Oct. 2013.

[8] Y. K. Kim. *A Set of Data for the Integration of Process Planning and Job Shop Scheduling*. [Online]. Available: http://syslab.chonnam.ac.kr/links/data-pp&s.doc

[9] Y. W. Guo, W. D. Li, A. R. Mileham, and G. W. Owen, "Optimisation of integrated process planning and scheduling using a particle swarm optimisation approach," *Int. J. Prod. Res.*, vol. 47, no. 14, pp. 3775–3796, Jul. 2009.

[10] L. Jin, C. Zhang, X. Shao, X. Yang, and G. Tian, "A multi-objective memetic algorithm for integrated process planning and scheduling," *Int. J. Adv. Manuf. Technol.*, vol. 85, no. 3, pp. 1–16, 2015.

[11] P. Mohapatra, A. Nayak, S. K. Kumar, and M. K. Tiwari, "Multi-objective process planning and scheduling using controlled elitist non-dominated sorting genetic algorithm," *Int. J. Prod. Res.*, vol. 53, no. 6, pp. 1712–1735, Mar. 2015.

[12] L. Zhang and T. N. Wong, "Solving integrated process planning and scheduling problem with constructive meta-heuristics," *Inf. Sci.*, vols. 340–341, pp. 1–16, May 2016.

[13] X. Li, L. Gao, and W. Li, "Application of game theory based hybrid algorithm for multi-objective integrated process planning and scheduling," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 288–297, Jan. 2012.

[14] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Erciyes Univ., Kayseri, Turkey, Tech. Rep. TR06, 2005.

[15] D. Lei, Y. Yuan, and J. Cai, "An improved artificial bee colony for multi-objective distributed unrelated parallel machine scheduling," *Int. J. Prod. Res.*, vol. 6, pp. 1–13, Jun. 2020.

[16] G. Gong, R. Chiong, Q. Deng, and X. Gong, "A hybrid artificial bee colony algorithm for flexible job shop scheduling with worker flexibility," *Int. J. Prod. Res.*, vol. 58, no. 14, pp. 1–15, 2019.

[17] B. Chen, H. Xu, L. Wang, and B. Liang, "Improved artificial bee colony algorithm for solving multi-objective flexible job-shop scheduling problem," *Inf. Control*, vol. 48, no. 1, pp. 119–126, 2019.

[18] 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.

[19] K. Z. Gao, Z. M. He, Y. Huang, P. Y. Duan, and P. N. Suganthan, "A survey on meta-heuristics for solving disassembly line balancing, planning and scheduling problems in remanufacturing," *Swarm Evol. Comput.*, vol. 57, Sep. 2020, Art. no. 100719, doi: 10.1016/j.swevo.2020.100719.

[20] K. Gao, Z. Cao, L. Zhang, Z. Chen, Y. Han, and Q. Pan, "A review on swarm intelligence and evolutionary algorithms for solving flexible job shop scheduling problems," *IEEE/CAA J. Autom. Sinica*, vol. 6, no. 4, pp. 904–916, Jul. 2019.

[21] L. Z. Cui, G. H. Li, Q. Z. Lin, Z. H. Du, W. F. Gao, J. Y. Chen, and N. Lu, "A novel artificial bee colony algorithm with depth-first search framework and elite-guided search equation," *Inf. Sci.*, vols. 367–368, pp. 1012–1044, Nov. 2016.

[22] G. Li, P. Niu, and X. Xiao, "Development and investigation of efficient artificial bee colony algorithm for numerical function optimization," *Appl. Soft Comput.*, vol. 12, no. 1, pp. 320–332, Jan. 2012.
[23] W.-L. Xiang, X.-L. Meng, Y.-Z. Li, R.-C. He, and M.-Q. An, "An improved artificial bee colony algorithm based on the gravity model," *Inf. Sci.*, vol. 429, pp. 49–71, Mar. 2018.
[24] M. S. Kiran, H. Hakli, M. Gunduz, and H. Uguz, "Artificial bee colony algorithm with variable search strategy for continuous optimization," *Inf. Sci.*, vol. 300, pp. 140–157, Apr. 2015.
[25] S. Wang, H. Liu, K. Gao, and J. Zhang, "A multi-species artificial bee colony algorithm and its application for crowd simulation," *IEEE Access*, vol. 7, pp. 2549–2558, 2019, doi: 10.1109/ACCESS.2018.2886629.
[26] J.-Q. Li, Q.-K. Pan, and M. F. Tasgetiren, "A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities," *Appl. Math. Model.*, vol. 38, no. 3, pp. 1111–1132, Feb. 2014.
[27] A. Jain, P. K. Jain, and I. P. Singh, "An integrated scheme for process planning and scheduling in FMS," *Int. J. Adv. Manuf. Technol.*, vol. 30, nos. 11–12, pp. 1111–1118, Oct. 2006.
[28] X. Li, L. Gao, X. Shao, C. Zhang, and C. Wang, "Mathematical modeling and evolutionary algorithm-based approach for integrated process planning and scheduling," *Comput. Oper. Res.*, vol. 37, no. 4, pp. 656–667, Apr. 2010.
[29] X. Shao, X. Li, L. Gao, and C. Zhang, "Integration of process planning and scheduling—A modified genetic algorithm-based approach," *Comput. Oper. Res.*, vol. 36, no. 6, pp. 2082–2096, Jun. 2009.
[30] W. D. Li and C. A. Mcmahon, "A simulated annealing-based optimization approach for integrated process planning and scheduling," *Int. J. Comput. Integr. Manuf.*, vol. 20, no. 1, pp. 80–95, Jan. 2007.
[31] R. J. Seethaler and I. Yellowley, "Process control and dynamic process planning," *Int. J. Mach. Tools Manuf.*, vol. 40, no. 2, pp. 239–257, Jan. 2000.
[32] L. Wang and W. Shen, "DPP: An agent-based approach for distributed process planning," *J. Intell. Manuf.*, vol. 14, no. 5, pp. 429–439, 2003.
[33] R. Shrestha, T. Takemoto, K. Ichinose, and N. Sugimura, "A study on integration of process planning and scheduling system for holonic manufacturing with modification of process plans," *Int. J. Manuf. Technol. Manage.*, vol. 14, nos. 3–4, pp. 359–378, 2008.
[34] F. Zhao, Y. Hong, D. Yu, Y. Yang, and Q. Zhang, "A hybrid particle swarm optimisation algorithm and fuzzy logic for process planning and production scheduling integration in holonic manufacturing systems," *Int. J. Comput. Integr. Manuf.*, vol. 23, no. 1, pp. 20–39, Jan. 2010.
[35] S. Zhang, Y. Xu, Z. Yu, W. Zhang, and D. Yu, "Combining extended imperialist competitive algorithm with a genetic algorithm to solve the distributed integration of process planning and scheduling problem," *Math. Problems Eng.*, vol. 2017, pp. 1–13, Jan. 2017.
[36] L. Wang, Y. Song, and W. Shen, "Development of a function block designer for collaborative process planning," in *Proc. Int. Conf. Comput. Supported Cooperat. Work Design*, vol. 3865, 2005, pp. 434–444.
[37] M. L. R. Varela, G. D. Putnik, V. K. Manupati, G. Rajyalakshmi, J. Trojanowska, and J. Machado, "Integrated process planning and scheduling in networked manufacturing systems for I4.0: A review and framework proposal," *Wireless Netw.*, vol. 27, no. 3, pp. 1587–1599, Apr. 2021.
[38] Y. W. Guo, W. D. Li, A. R. Mileham, and G. W. Owen, "Applications of particle swarm optimisation in integrated process planning and scheduling," *Robot. Comput.-Integr. Manuf.*, vol. 25, no. 2, pp. 280–288, 2007.
[39] Y. C. Ho and C. L. Moodie, "Solving cell formation problems in a manufacturing environment with flexible processing, and routing capabilities," *Int. J. Prod. Res.*, vol. 34, no. 10, pp. 2901–2923, 2010.
[40] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.
[41] B. Alatas, "Chaotic bee colony algorithms for global numerical optimization," *Expert Syst. Appl.*, vol. 37, no. 8, pp. 5682–5687, Aug. 2010.
[42] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Comput. Surv.*, vol. 45, no. 3, pp. 1–33, Jun. 2013.
[43] K. R. Baker and D. Trietsch, *Principles of Sequencing and Scheduling*. Hoboken, NJ, USA: Wiley, 2009.
[44] E. Nowicki and C. Smutnicki, "A fast taboo search algorithm for the job shop problem," *Manage. Sci.*, vol. 42, no. 6, pp. 797–813, Jun. 1996.
[45] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[46] Y. Cai and J. Wang, "Differential evolution with neighborhood and direction information for numerical optimization," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 2202–2215, Dec. 2013.
[47] Q. Liu, X. Li, L. Gao, and Y. Li, "A modified genetic algorithm with new encoding and decoding methods for integrated process planning and scheduling problem," *IEEE Trans. Cybern.*, early access, Oct. 15, 2020, doi: 10.1109/TCYB.2020.3026651.
[48] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.
[49] J. Demiar and D. Schuurmans, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, no. 1, pp. 1–30, 2006.
[50] X. Li, L. Gao, Q. Pan, L. Wan, and K.-M. Chao, "An effective hybrid genetic algorithm and variable neighborhood search for integrated process planning and scheduling in a packaging machine workshop," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 49, no. 10, pp. 1933–1945, Oct. 2019.
[51] S. Zhang and T. N. Wong, "Integrated process planning and scheduling: An enhanced ant colony optimization heuristic with parameter tuning," *J. Intell. Manuf.*, vol. 29, no. 3, pp. 585–601, Mar. 2018.
[52] K. Z. Gao, P. N. Suganthan, Q. K. Pan, M. F. Tasgetiren, and A. Sadollah, "Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion," *Knowl.-Based Syst.*, vol. 109, pp. 1–16, Oct. 2016, doi: 10.1016/j.knosys.2016.06.014.
[53] J. Q. Li, Z. M. Liu, C. D. Li, and Z. X. Zheng, "Improved artificial immune system algorithm for type-2 fuzzy flexible job shop scheduling problem," *IEEE Trans. Fuzzy Syst.*, early access, Aug. 12, 2020, doi: 10.1109/TFUZZ.2020.3016225.
[54] K. Z. Gao, Y. Huang, A. Sadollah, and L. Wang, "A review of energy-efficient scheduling in intelligent production systems," *Complex Intell. Syst.*, vol. 6, pp. 237–249, Sep. 2020, doi: 10.1007/s40747-019-00122-6.
[55] J.-Q. Li, Y.-Q. Han, P.-Y. Duan, Y.-Y. Han, B. Niu, C.-D. Li, Z.-X. Zheng, and Y.-P. Liu, "Meta-heuristic algorithm for solving vehicle routing problems with time windows and synchronized visit constraints in prefabricated systems," *J. Cleaner Prod.*, vol. 250, Mar. 2020, Art. no. 119464, doi: 10.1016/j.jclepro.2019.119464.

YANG CAO received the M.S. degree with Liaoning University, China, in 2005. He is currently pursuing the Ph.D. degree with Northeastern University. He is also an Associate Professor with the Faculty of Information and Control Engineering, Shenyang Jianzhu University, Shenyang, China. His main research interests include computational intelligent, complex optimization theory and application, and the engineering application research of production scheduling method.

HAIBO SHI received the Ph.D. degree in engineering from the Harbin Institute of Technology, China, in 2003. He is currently a Research Fellow and the Director of doctor student with the Shenyang Institute of Automation. His main research interests include production and operation management, modeling and simulation technology of manufacturing process, MES technology, digital equipment, and intelligent system technology.