systems

MDPI

Article

Modeling and IAHA Solution for Task Scheduling Problem of Processing Crowdsourcing in the Context of Social Manufacturing

Gaohong Zhu [1,2] and Dianting Liu [1,2,*]

[1] Key Laboratory of Advanced Manufacturing and Automation Technology, Education Department of Guangxi Zhuang Autonomous Region, Guilin University of Technology, Guilin 541006, China; zhu_gaohong@foxmail.com
[2] College of Mechanical and Control Engineering, Guilin University of Technology, Guilin 541006, China
* Correspondence: 2000004@glut.edu.cn

Abstract: The paper addresses the discrete characteristics of the processing crowdsourcing task scheduling problem in the context of social manufacturing, divides it into two subproblems of social manufacturing unit selecting and subtask sorting, establishes its mixed-integer programming with the objective of minimizing the maximum completion time, and proposes an improved artificial hummingbird algorithm (IAHA) for solving it. The IAHA uses initialization rules of global selection, local selection, and random selection to improve the quality of the initial population, the Levy flight to improve guided foraging and territorial foraging, the simplex search strategy to improve migration foraging to enhance the merit-seeking ability, and the greedy decoding method to improve the quality of the solution and reduce solution time. For the IAHA, orthogonal tests are designed to obtain the optimal combination of parameters, and comparative tests are made with variants of the AHA and other algorithms on the benchmark case and a simulated crowdsourcing case. The experimental results show that the IAHA can obtain superior solutions in many cases with economy and effectiveness.

Keywords: social manufacturing; crowdsourced task scheduling; flexible job shop scheduling; artificial hummingbird algorithm

Citation: Zhu, G.; Liu, D. Modeling and IAHA Solution for Task Scheduling Problem of Processing Crowdsourcing in the Context of Social Manufacturing. *Systems* **2023**, *11*, 383. https://doi.org/10.3390/systems11080383

Academic Editor: Ed Pohl

Received: 12 June 2023
Revised: 14 July 2023
Accepted: 25 July 2023
Published: 27 July 2023

1. Introduction

Social manufacturing [1–3] is based on the information-physical-social interconnection of social manufacturing resources (SMR), as well as the self-organization of social resources and collaborative interaction and information sharing. It is a future-oriented service-oriented manufacturing model that covers the deep participation of distributed producers and consumers in the whole life cycle of products. In the social manufacturing model, decentralized social manufacturing resources collaboratively perform multiple manufacturing tasks based on multiple interest coordination and social mechanisms; in other words, their processing tasks are distributed through crowdsourcing and performed by discrete social manufacturing resources. As a new SOM model, social manufacturing has the characteristics of small and microenterprises, social resources, interconnection and interaction, product personalization, organization decentralization, and distribution. Among them, decentralization is an important feature of social manufacturing that is different from other manufacturing modes. With the continuous enhancement of the flat self-organized dynamic manufacturing community, the core enterprise no longer has the dominant power in the manufacturing network, and the traditional seller's market has gradually transferred to the buyer's market [1,4]. The task scheduling of processing crowdsourcing is an important link in the social manufacturing mode. The core of the problem to be solved lies in the reasonable allocation of effective resources and the necessary optimization means. The

purpose is to arrange the crowdsourcing tasks into SMRs in a reasonable and balanced way and to make a reasonable plan for the processing sequence of tasks on SMRs. Because task scheduling is a process with a complex environment and many uncertain factors, the quality of scheduling directly affects the competitiveness of social manufacturing resources involved in product manufacturing in the market. Therefore, an algorithm is needed for task scheduling and planning of SMR for collaborative manufacturing crowdsourcing tasks. In this context, the problem of processing crowdsourcing task scheduling in socialized manufacturing is proposed.

In general, except for manufacturing units that are socially dispersed, the processing crowdsourcing task scheduling problem in communalized manufacturing can usually be classified as the flexible job shop scheduling problem (FJSP) [5], and the FJSP problem has been proven to be an NP-hard problem [6]. For this difficult problem, experts and scholars have conducted decades of research and proposed many methods to solve it. Among them, the classical methods include the Lagrangian relaxation algorithm [7], dynamic programming [8], and branch-and-bound algorithm [9]; however, these classical solution methods have more or less some restrictions, such as requiring the function to be derivable, or convex, or the solution size cannot be too large, which restrict their applications. To overcome these shortcomings, scholars have developed heuristic algorithms, such as genetic algorithms [10,11], differential evolutionary algorithms [12], ant colony algorithms [13], particle swarm algorithms [14–16], artificial bee colony algorithms [17], and other intelligent optimization algorithms; however, traditional heuristic algorithms often have the disadvantages of slow running speed and easy to fall into local optimality in the late iteration. To address these drawbacks, experts and scholars at home and abroad have proposed many improvements in using them to solve FJSP, e.g., Wang et al. [18] designed three heuristic strategies and proposed a new artificial bee colony algorithm HMABC to achieve scheduling of practical production activities; Yuan et al. [19] developed a new transformation mechanism to enable differential evolutionary algorithms on continuous domains to explore discrete FJP problems; Sun et al. [20] proposed a method for combining crossover and variational operators considering machine load balancing and designed an improved hybrid genetic algorithm HGA-VNS for local search of critical processes on critical paths.

Currently, the research for FJSP is mainly in considering novel conditions such as uncertainties and green low-carbon requirements and constructing novel solution algorithms such as multiobjective intelligent optimization and deep learning. For example, Xu et al. [21] proposed a load-balancing-based step-size adaptive discrete particle swarm algorithm (LS-DPSO) to solve the G-FSP model. Wang et al. [22] considered the maximum completion time, total delay time, and total energy consumption of the job and used the hybrid adaptive differential evolution algorithm (HADE) to solve it. Lou et al. [23] combined adaptive local search and a multiobjective evolutionary algorithm and proposed a multiobjective modal algorithm based on learning decomposition (MOMA-LD) to solve the model. Yan et al. [24] studied the FJP problem with the objectives of maximum completion time, total workload, critical machine workload, and minimum advance/delay penalties and proposed an improved ant colony algorithm to solve the problem. Tian et al. [25] established an algorithm with the objectives of work duration, total machine load, machine on/off times, and tool change times as the objectives of a multiobjective optimization model and solved the model using NSGA-II.

Although more research has been conducted for FJSP, it has mainly focused on scheduling production lines within a single plant. Due to the similarity of work conditions, it has gradually been applied in recent years in areas such as aerospace materials manufacturing [18] and distributed factory production scheduling [26], while further research is needed in processing crowdsourcing task scheduling. The objective of this study is to develop a mixed-integer programming model for scheduling processing crowdsourcing tasks in community-based manufacturing and to design algorithms to solve it in order to improve the productivity of processing crowdsourcing tasks. In this task scheduling

optimization problem, one or more SMRs with the ability to provide specialized manufacturing services externally are regarded as social manufacturing units (SMU), which are characterized by socialization, decentralization, and specialization. SMRs may come from traditional large- and medium-sized enterprises that decompose and reorganize their internal manufacturing resources into several small and microenterprises with independent manufacturing service capabilities or from emerging small and microenterprises in market segments, start-up companies, or individuals with professional skills [3]. Each individual crowdsourced task will be coordinated by multiple decentralized SMUs to produce a complex product. For this purpose, the manufacturing task of the product is decomposed, and the manufacturing of a collection of parts or components in the product is defined as tasks, and the collection of manufacturing operations of parts or components is defined as subtasks. The manufacturing of a product can be decomposed into multiple tasks, and each task can be further decomposed into more subtasks; each subtask can be processed by multiple SMUs, and all subtasks of the same task have a serial order constraint relationship with each other. Then, all subtasks are matched with SMUs according to the optimization objective to get the optimal scheduling scheme. Figure 1 shows a conceptual model for processing crowdsourced task scheduling in social manufacturing.

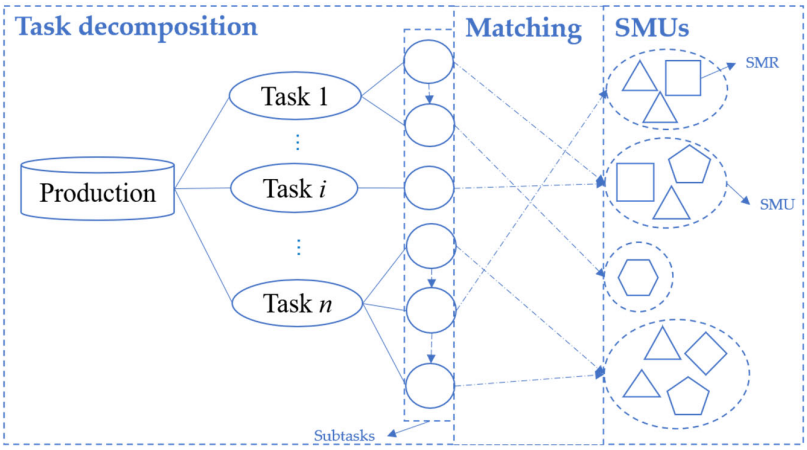

Figure 1. Conceptual model for processing crowdsourced task scheduling in social manufacturing.

This study addresses the scheduling problem of processing crowdsourcing tasks in socialized manufacturing, establishes a mathematical model of scheduling with the objective of minimizing the maximum completion time, and proposes an improved artificial hummingbird algorithm (IAHA) to solve it. For its discrete characteristics, we divide the problem into two subproblems—social manufacturing unit selection and subtask sequence—and encode them separately according to the subproblem characteristics, match different decoding for the two encodings, and use greedy decoding methods to improve the quality of the solution. To improve the quality of the initial population of the IAHA, we used the initialization rules of global selection, local selection, and random selection (GLR). To enhance the merit-seeking ability of the IAHA, we used the Levy flight to improve the guided foraging and territorial foraging of the IAHA and the simplex search strategy (SSS) to improve the migration foraging of the IAHA. We designed four experiments, and the experiment results proved the superiority and effectiveness of the IAHA.

The rest of the paper is organized as follows: Section 2 defines the processing crowdsourcing task scheduling problem in social manufacturing and develops a mathematical model of the problem. Section 3 illustrates the IAHA in detail with a problem case. Section 4 conducts four experiments and discusses the results of the IAHA's experiments on the benchmark and crowdsourcing case. Section 5 summarizes the main contents of the paper and points out future research directions.

2. Problem Modeling

2.1. Problem Definition

The processing crowdsourcing task scheduling problem in social manufacturing can usually be described as follows: n tasks are processed in m SMUs, each involving multiple subtasks; each subtask can be processed on multiple SMUs, and the processing time varies from SMU to SMU. In this study, the scheduling problem is divided into two subproblems: determining the processing SMUs for each subtask and determining the processing sequence on each SMU. Each subtask is assigned to the appropriate SMU, and the order in which subtasks are assigned to each SMU is determined.

2.2. Constraints

Before building the mixed integer programming model, some constraints need to be satisfied: (1) only one subtask can be processed by the same SMU at the same moment; (2) the same subtask of the same task can be processed by only one SMU at the same moment; (3) each subtask of each task cannot be interrupted once it starts processing; (4) different tasks have the same priority among themselves; (5) the subtasks of different tasks have no sequential constraints between the different tasks and sequential constraints between subtasks of the same workpiece; (6) all tasks can be machined at zero moments; (7) each subtask has the same processing effect on all SMUs that satisfy its processing conditions; (8) the transport time when moving subtasks between SMUs is ignored; (9) the processing time of each subtask and its optional set of processing SMUs are known.

2.3. Mathematical Models

Some symbols need to be defined before the mathematical model can be built, and Table 1 shows some of the symbols that appear in FJSP and their corresponding meanings.

Table 1. Mathematical symbols and definitions.

Symbols	Definitions
n	Total number of tasks
m	Total number of SMU
M	SMU set
i, e	SMU number
j, k	Task number
h_j	Number of subtasks of task j
l	Task number, $l = 1, 2, \ldots, h_j$
O_{jh}	h-th subtask of task j
m_{jh}	Number of available processing SMUs for subtask O_{jh}
P_{ijh}	Processing time of subtask O_{jh} on SMU i
S_{jh}	Processing start time of subtask O_{jh}
C_{jh}	Processing end time of subtask O_{jh}
L	A large enough positive number
C_j	Completion time of task j
C_{max}	Completion time of the last task
T_0	Total number of all subtasks
X_{ijh}	1 if subtask O_{jh} is processed on SMU i, else 0
y_{ijhkl}	1 if subtask O_{jh} is processed on SMU i before subtask O_{kl}, else 0

In solving the scheduling problem of crowdsourcing tasks in community-based manufacturing, the maximum completion time is chosen as the evaluation index to measure the merits of the scheduling scheme. The completion time is the time when the last process of each workpiece is completed, the latest of which is the maximum completion time, which is the most fundamental indicator of the scheduling scheme, mainly reflecting the

productivity of the shop floor, and is the most widely used evaluation indicator in the study. It can be expressed as follows:

$$f = min(max(C_j)), 1 \leq j \leq n \tag{1}$$

The problem is subject to the following constraints:

$$S_{jh} + x_{xjh} \times p_{ijh} \leqslant C_{jh},\ i = 1, 2, \cdots, m;\ j = 1, 2, \cdots, n;\ h = 1, 2, \cdots, h_j \tag{2}$$

$$C_{jh} \leqslant S_{j(h+1)}, j = 1, 2, \cdots, n;\ h = 1, 2, \cdots, h_j - 1 \tag{3}$$

$$C_{jh_j} \leqslant C_{\max}, j = 1, 2, \cdots, n \tag{4}$$

$$\begin{gathered} S_{jh} + p_{ijh} \leqslant S_{kl} + L\left(1 - y_{ijhkl}\right),\ j = 0, 1, 2, 3, \cdots, n; \\ k = 1, 2, \cdots, n; h = 1, 2, \cdots, h_j; \\ l = 1, 2, \cdots, h_k; i = 1, 2, \cdots, m \end{gathered} \tag{5}$$

$$\begin{gathered} C_{jh} \leqslant S_{j(h+1)} + L\left(1 - y_{iklj(h+1)}\right), j = 1, 2, \cdots, n; \\ k = 0, 1, 2, \cdots, n; h = 1, 2, \cdots, h_j - 1; \\ l = 1, 2, \cdots, h_k; i = 1, 2, \cdots, m \end{gathered} \tag{6}$$

$$\sum_{i=1}^{m_{jh}} x_{ijh} = 1; h = 1, 2, \ldots, h_j; j = 1, 2, \ldots, n \tag{7}$$

$$S_{jh} \geqslant 0,\ C_{jh} \geqslant 0,\ j = 0, 1, \cdots, n; h = 1, 2, \cdots, h_i \tag{8}$$

(2) and (3) denote the sequential order constraint of the subtasks of each task; (4) denotes that the completion time of a single subtask is less than the total completion time; (5) and (6) denote that the same SMUs cannot process multiple subtasks at a certain time; (7) denotes the SMU constraint that there cannot be more than one SMU processing the same process at a certain time; (8) denotes that each parameter variable is positive.

3. Improved Artificial Hummingbird Algorithm for Model Solving

The artificial hummingbird algorithm (AHA) is a novel metaheuristic algorithm proposed by ZHAO et al. [27] in 2021. The AHA is inspired by the special flight skills and intelligent foraging strategies of hummingbirds. The AHA has been applied in the distributed power optimization configuration [28], coordinated scenery-thermal power generation scheduling [29], and arrhythmia classification [30]. We try to apply it to solve the processing crowdsourcing task scheduling model in social manufacturing and combine the characteristics of the crowdsourced task scheduling model to modify the AHA and refer to the improved artificial hummingbird algorithm as the IAHA, which is described in detail in this section.

3.1. Discrete Transformation

The AHA, as a new type of swarm intelligence optimization algorithm, has the advantages of few parameters, strong search capability, and easy implementation [27], but it also has the common problems of swarm intelligence optimization algorithms, such as slow convergence speed and easy to fall into local optimum [31]. In response to these drawbacks and the discrete characteristics of the processing crowdsourcing task scheduling problem, an improved artificial hummingbird algorithm (IAHA) is proposed in the study. To describe the problem more clearly, we introduce an example to illustrate it, and the processing information for each task is shown in Table 2.

Table 2. Processing information.

Tasks	Subtasks	SMU Processing Time			
		M_1	M_2	M_3	M_4
J_1	O_{11}	2	4	-	3
	O_{12}	4	-	5	2
	O_{13}	6	-	4	8
J_2	O_{21}	-	3	2	6
	O_{22}	5	7	-	-
J_3	O_{31}	2	3	5	-
	O_{32}	4	5	-	8
	O_{33}	6	-	-	4
	O_{34}	5	4	-	-

3.1.1. Question Encoding

The crowdsourcing task scheduling problem contains 2 components: social manufacturing unit selection (MS) and subtask sequencing (SS). We use a two-segment encoding based on SMU and subtasks, respectively, and the length of both encodings is the total number of subtasks T_0. The range of values for each value on the encoding is $[-\delta, \delta]$. If δ is obtained too small, the relative size of the updated individual subtask sequence value changes a lot, leading to the destruction of individuals; if δ is obtained too large, it will lead to an insignificant Levy flight effect, which is generally set to a number between 2 and 5. A legal encoding is shown in Figure 2.

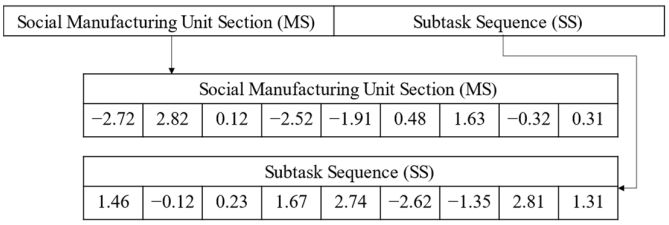

Figure 2. Legal code.

3.1.2. Code Transformation

The solution space of the AHA is a continuous space, and the processing crowdsourcing scheduling problem is a discrete combinatorial optimization problem, which requires mapping the continuous solution space to a discrete solution space. Different transformation mechanisms are required due to the different constraints on SMU selection and subtask sequence.

(1) MS section

The continuous codes are converted into discrete codes according to Equation (9) and then converted into specific SMU numbers according to the subtask optional SMUs. When only one SMU of subtask O_{jh} is processable, the discrete encoding value is taken as 1. In the MS, the SMU corresponding to each subtask of each task is shown from left to right.

$$u(i) = \begin{cases} 1 & if\ m_{jh} = 1 \\ round\left(\frac{(e(i)+\delta)(m_{jh}-1)}{2\delta} + 1\right) & else \end{cases}, i = 1, 2, \cdots, \frac{d}{2} \quad (9)$$

where is $e(i)$ the i-th value of the MS continuous code; m_{jh} is the number of processing SMUs available for subtask O_{jh}; $u(i)$ is the i-th value of the transformed MS discrete code.

Figure 3 shows an MS transformation, where "-" indicates that the SMU is not available, the value of the intermediate code "i" indicates that the i-th SMU is selected for processing among the available SMUs of the subtask, and the value of the discrete code indicates the SMU selected for the subtask. For example, the intermediate code "3" of O_{12} means that the third SMU (M_4) is selected from the set of available SMUs {M_1, M_3, M_4} of O_{12}, and the

discrete code of O_{12} is 4. For the SMU selection problem, the above code and its conversion method do not generate illegal solutions.

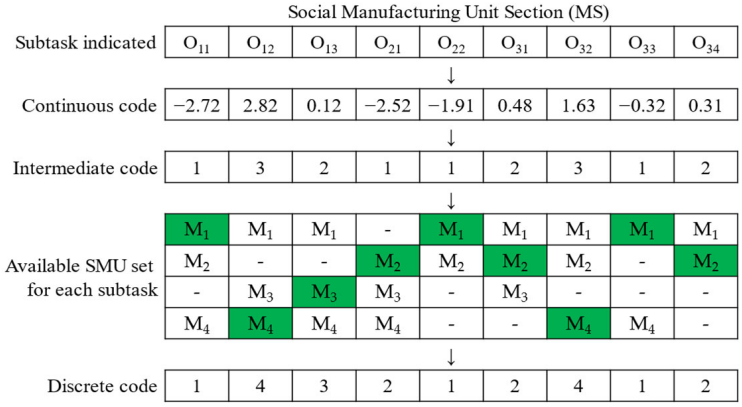

Figure 3. Discrete transformation of MS.

(2) SS section

The largest position value (LPV) rule [19] is used to transform the solution space of the subtask sequence. The values on the SS are assigned a fixed ID starting from 1. Then, they are sorted in order from largest to smallest. Finally, a new code is obtained based on the fixed ID corresponding to each value, which is converted into a sequence that can be used for decoding according to the number of subtasks of each task.

Figure 4 shows a SS transformation where the 9 in fixed ID maps both continuous code 1.31 and task J_3, and after LPV sort, the 9 moves to the 5th bit of the code. The value in the discrete code indicates the task number, and the number of occurrences of "*i*" from left to right indicates the *i*-th subtask of the task. At this time, the 5th bit of the discrete code is mapped to 3, indicating task J_3, and 3 is the 2nd occurrence, and O_{32} is in the 5th position of the processing. Figure 4 indicates the processing order of $O_{31} \rightarrow O_{21} \rightarrow O_{22} \rightarrow O_{11} \rightarrow O_{32} \rightarrow O_{12} \rightarrow O_{13} \rightarrow O_{33} \rightarrow O_{34}$.

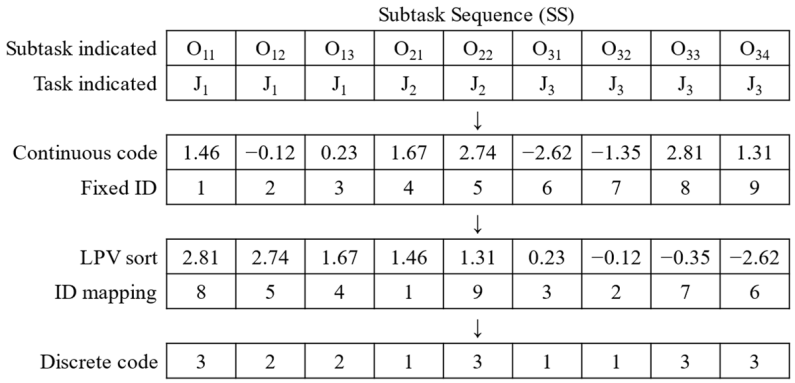

Figure 4. Discrete transformation of SS.

3.1.3. Problem Decoding

The encoding of the crowdsourced task scheduling problem consists of two parts, MS and SS, which need to be decoded separately. The decoding steps are as follows:

(1) MS section

The MS is read from left to right and converted into SMU order matrix J_M and time order matrix T, where $J_M(j, h)$ denotes the SMU number of subtask O_{jh}; $T(j, h)$ denotes the

processing time of subtask O_{jh}. Take the code of Figure 3 as an example, combined with processing information in Table 2, and MS is converted to J_M and T as follows:

$$J_M = \begin{bmatrix} 1 & 4 & 3 & - \\ 2 & 1 & - & - \\ 2 & 4 & 1 & 2 \end{bmatrix} \tag{10}$$

$$T = \begin{bmatrix} 2 & 2 & 4 & - \\ 3 & 5 & - & - \\ 3 & 8 & 6 & 4 \end{bmatrix} \tag{11}$$

(2) SS section

The SS is read sequentially from left to right, and the greedy decoding method (GDM) is used to find the free time of each SMU and decide whether to insert subtasks into the free time according to Equations (12) and (13). In this way, until all subtasks on the sequence are scheduled in their optimal positions, obtaining a higher quality scheduling scheme.

$$MS_{jh} = \max\left(C_{j(h-1)}, ITS_k\right) \tag{12}$$

$$S_{jh} = \begin{cases} MS_{jh} & if \ \left[MS_{jh}, MS_{jh} + P_{ijh}\right] \subseteq [ITS_k, ITE_k] \\ \max\left(C_{j(h-1)}, LM\right) & else \end{cases} \tag{13}$$

where MS_{jh} denotes the start processing time of subtask O_{jh} satisfying the insertion condition; $C_{j(h-1)}$ denotes the processing completion time of subtask $O_{j(h-1)}$; ITS_k, ITE_k denotes the start time and end time of the k-th idle time period, respectively; LM denotes the end time of the last current subtask of the SMU selected by the subtask; S_{jh} denotes the start processing time of subtask O_{jh}. If the insertion condition $[MS_{jh}, MS_{jh} + P_{ijh}] \subseteq [ITS_k, ITE_k]$ is satisfied, then $S_{jh} = MS_{jh}$.

Figure 5 shows a GDM operation where O_{22} was originally machined in M_1 after O_{32}. There is a long idle time between O_{11} and O_{32}. At this time, the possible start time of O_{22} $MS_{22} = max\ (5, 5) = 5$, the processing time $P_{122} = 4$, the 1st idle time period $[ITS_1, ITE_1] = [5, 11]$, and the insertion condition $[5, 9] \subseteq [5, 11]$ are satisfied, so that the completion time can be shortened.

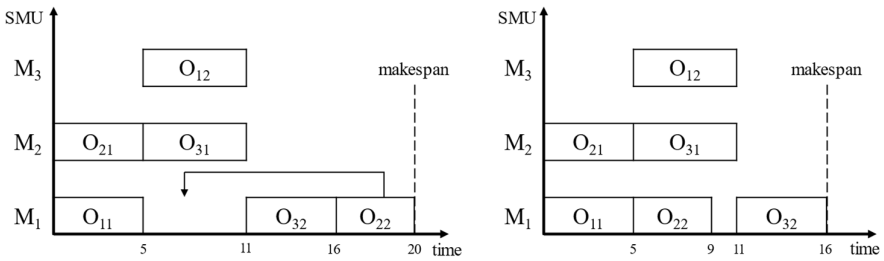

Figure 5. GDM works.

3.2. Population Initialization

Population initialization is a key problem in evolutionary algorithms, and the quality of the initial solution has a great impact on the speed and quality of the algorithm's solution. The AHA performs random initialization of the initial population, which cannot guarantee the quality of the initial solution, so we use the GLR method for population initialization, including global selection, local selection, and random selection [32]. GS considers the global load balance of all SMUs. LS only considers the global SMU load balance during each task processing. GS and LS make the workload of each selected SMU as balanced as possible to fully improve the utilization of SMUs. RS mainly considers distributing the initial population as decentralized as possible throughout the solution space to improve

the diversity of solutions. The three methods account for a population ratio of GS:LS:RS = 0.4:0.4:0.2, and the quality of the initial solution is improved by the combination of the three methods. The initialized population encoding is discrete and needs to be inverse transformed to continuous encoding according to Section 3.1.2 in order to use the IAHA for searching.

3.3. Levy Flight

The Levy flight is a random search path that obeys the Levy distribution [33]. Since the Levy distribution is a heavy-tailed distribution, the Levy flight is a search method with short and occasionally long-distance intervals. Currently, the Levy flight is widely used in intelligent optimization; for example, Liu et al. [34] used the levy flight combined with a differential evolution algorithm to solve the JSP problem. We introduced the Levy flight in the guided foraging and territorial foraging of the IAHA to enhance its search capability and ability to jump out of the local optimum.

After combining Levy flights, guided foraging and territorial foraging locations were updated as follows.

$$x_i(t+1) = \begin{cases} x_i(t) + \alpha \oplus Levy(\lambda) & f[x_i(t)] \le f[v_i(t+1)] \\ v_i(t+1) & f[x_i(t)] \ge f[v_i(t+1)] \end{cases} \tag{14}$$

where x_i (t) is the t-th generation position of x_i; $\oplus$ is the point multiplication; α is the step control factor; v_i $(t+1)$ is the location of the candidate food source obtained by the AHA through guided foraging osr territorial foraging [27]; $Levy(\lambda)$ is the random search path, calculated as follows:

$$Levy(\lambda) = s^{-\lambda},\ 1 < \lambda \le 3 \tag{15}$$

where the step size s is calculated as follows:

$$s = \frac{\mu}{|v|^{1/\beta}} \tag{16}$$

where β is taken as a constant 1.5, and μ and v follow normal distribution defined as follows:

$$\mu \sim N\left(0,\ \sigma_\mu^2\right),\ v \sim N(0,\ 1) \tag{17}$$

where

$$\sigma_\mu = \left[\frac{\Gamma(1+\beta)\sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right)\beta \times 2^{\frac{\beta-1}{2}}}\right]^{\frac{1}{\beta}} \tag{18}$$

3.4. Simplex Search Strategy

The simplex search strategy (SSS) uses iterative local search [35] to improve the migration foraging of hummingbird flocks in the IAHA. The centroid location of the hummingbird $x^c(t)$ at the t-th iteration is calculated as follows.

$$x^c(t) = \frac{1}{N}\sum_{i=1}^{N} x_i(t) \tag{19}$$

The position of the worst hummingbird is mapped around centroid $x^c(t)$ to obtain the mapped position $x^r(t)$, which is calculated as follows:

$$x^r(t) = 2x^c(t) - x^{wor}(t), \tag{20}$$

where wor is the index of the worst-performing hummingbird.

The fitness of the mapped position $x^r(t)$ guides the search of the search space. The new position of the worst-performing hummingbird $x_{wor}(t+1)$ is obtained by comparing

the fitness of the mapped position $x^r(t)$, the optimal position $x^{best}(t)$ and the second worst position $x^{swor}(t)$, calculated as follows:

$$x_{wor}(t+1) = \begin{cases} (1+\gamma)x^c(t) - \gamma x^{wor}(t) & f(x^r) < f\left(x^{best}\right) \\ x^r(t) & f\left(x^{best}\right) < f(x^r) < f(x^{swor}) \\ (1-\beta)x^c(t) + \beta x^{wor}(t) & f(x^r) > f(x^{swor}) \end{cases} \quad (21)$$

where γ is the expansion factor; β is the contraction factor.

The IAHA can be obtained by combining the above-mentioned improved methods for the AHA, and its flow chart is shown in Figure 6.

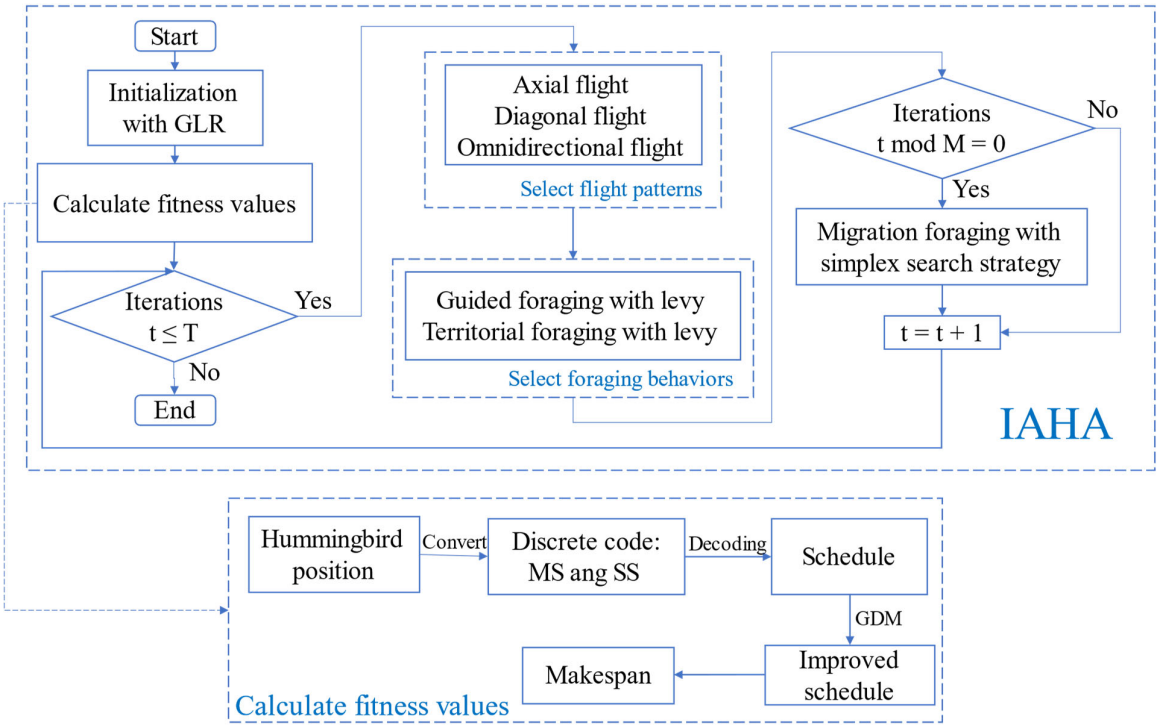

Figure 6. IAHA flow chart.

4. Experimental Results and Discussion

Four sets of experiments are conducted in the section: (1) designing orthogonal experiments to configure the optimal combination of parameters for the IAHA; (2) discussing the performance improvement of the improved strategy of the IAHA over the AHA and its variants; (3) comparing the performance with algorithms proposed by other scholars in a standard algorithm; (4) simulating a real processing crowdsourcing environment for scheduling. And the first three sets of experiments are checked for significant differences in the experimental results by the Friedman test [36]. At present, there is no public crowdsourcing task data set in the research of social manufacturing, so this paper adopts Kacem [37] and Brandimarte [38] standard cases, which are commonly used in FJSP, in the first three experiments. Kacem and Brandimarte belong to total FJSP(T-FJSP) and partial FJSP(P-FJSP), respectively. The expression of T-FJSP and P-FJSP in this paper is that subtasks can be processed on any SMU, and subtasks can only be processed on some SMU, respectively. T-FJSP belongs to the P-FJSP category.

The hardware environment for the simulation experiments in the study is Intel(R) Core (TM) i5-10400 CPU @ 2.90 GHz; RAM 16 GB; software environment is Windows 10, written in Python, simulation platform PyCharm 2020.1.

4.1. Comparison of Orthogonal Experiments of Parameters

The size of the parameters has a large impact on the performance of the IAHA. Therefore, we need a reasonable configuration. The main parameters of the IAHA are shown in Table 3.

Table 3. Main parameters of IAHA.

Parameters	Description
δ	Search space range
r_1, r_2, r_3	Probability of selecting axial, diagonal, and omnidirectional flight skills, totaling 1
f_1, f_2	Probability of selecting guided, territorial foraging, totaling 1
m	Migration foraging every m iteration
γ	Expansion factor
β	Contraction factor

In the IAHA, r_1, r_2, and r_3 determine the choice of flight skill each time foraging is performed, and the sum of the three parameters is one, which can actually be considered as two parameters. f_1 and f_2 determine the foraging mode chosen each time, and the sum of f_1 and f_2 is 1, which can be considered as one parameter. The IAHA only performs migration foraging once every m iteration, so the influence of γ and β in SSS on the IAHA's foraging performance is low, and in order to balance the convergence speed of SSS with the foraging performance, $\gamma = 2$ and $\beta = 0.5$ are often taken. Finally, there are five important parameters that have a great influence on the performance of the IAHA, which are δ, r_1, r_2, f_1, and m.

We designed orthogonal experiments to configure the best parameter combinations, and the above five parameters are set as five factors with four levels for each factor. According to the principle of orthogonal experimental design, only $16(4^2 = 16)$ sets of parameter combinations need to be tested to find the best parameter configuration. The orthogonal test table $L_{16}(4^5)$ is shown in Table 4.

Table 4. Orthogonal test table $L_{16}(4^5)$.

Test	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
δ	2	2	2	2	3	3	3	3	4	4	4	4	5	5	5	5
r_1	0.2	0.27	0.34	0.4	0.2	0.27	0.34	0.4	0.2	0.27	0.34	0.4	0.2	0.27	0.34	0.4
r_2	0.4	0.37	0.33	0.3	0.37	0.4	0.3	0.33	0.33	0.3	0.4	0.37	0.3	0.33	0.37	0.4
f_1	0.45	0.5	0.55	0.6	0.55	0.6	0.45	0.5	0.6	0.55	0.5	0.45	0.5	0.45	0.6	0.55
m	10	20	40	80	80	40	20	10	20	10	80	40	40	80	10	20

In the orthogonal experiments, the number of iterations of the algorithm is 200, and the population size is 200. Five cases of the standard test case Kacem [37] are used for the experiments to adjust the parameters, and 10 experiments are conducted for each case to take the average value as the experimental results; Table 5 shows the experimental results of the orthogonal experiments.

Table 5. Experimental results of orthogonal experiments.

Test	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Kac1	11.9	12	11.9	12.8	13.5	12.1	11.5	12.5	12.2	11.3	14.1	13.2	11.3	13.9	12.2	11.5
Kac2	15.3	15.1	15.1	15.4	15.8	14.2	15.5	14.4	14.8	14.8	16.4	14.8	15.1	16.2	15.4	15.3
Kac3	12.7	11.8	11.4	12.6	13.2	12.4	12.8	11.4	11.8	11.4	13	13.5	13	13.9	11.4	12.2
Kac4	8.5	8.0	7.5	9.2	8.9	7.5	8.4	8.0	8.5	7.7	9.2	8.7	8.7	9.4	7.2	8.5
Kac5	11.2	12.7	11.8	12.5	13.1	12.5	13.3	11.9	12.9	11.4	13.1	12.5	13.1	13.6	11.6	12.2

Friedman's ranking of the experimental results was performed, and the sum of the result ranking of each experimental scheme in each case was calculated to obtain the final

ranking of the schemes. The ranking and final ranking of each experimental scheme are shown in Table 6, which shows that the final ranking of experiment 10 is better than the other experimental schemes.

Table 6. Friedman ranking results of orthogonal experiments.

Test	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Kac1	5.5	7	5.5	12	14	8	3.5	11	9.5	1.5	16	13	1.5	15	9.5	3.5
Kac2	9.5	7	7	11.5	14	1	13	2	4	4	16	4	7	15	11.5	9.5
Kac3	10	5.5	2.5	9	14	8	11	2.5	5.5	2.5	12.5	15	12.5	16	2.5	7
Kac4	9	5.5	2.5	14.5	13	2.5	7	5.5	9	4	14.5	11.5	11.5	16	1	9
Kac5	1	10	4	8	13	8	15	5	11	2	13	8	13	16	3	6
Total	35	35	21.5	55	68	27.5	49.5	26	39	14	72	51.5	45.5	78	27.5	35
Final	7	7	2	13	14	5	11	3	9	1	15	12	10	16	4	7

Significance tests were performed on the ranking results of this orthogonal test scheme. The original hypothesis H_0: there are no significant differences in the results of the orthogonal experiment. The alternative hypothesis H_1: there are significant differences in the results of the orthogonal experiment.

The results of the Friedman statistics of the program are shown in Table 7. From Table 7, the Friedman statistic is $\chi^2 = 46.17$, and the degree of freedom is k-1 = 15. When the significance level α is 0.05, querying the chi-square distribution table yields $\chi^2_{0.05(15)} = 25$. Since $\chi^2 > \chi^2_{0.05(15)}$, H_0 is rejected, and H_1 is accepted: there are significant differences in the results of the orthogonal experiment. When the significance level α is 0.01, the chi-square distribution table $\chi^2_{0.01(15)} = 30.58$. Since $\chi^2 > \chi^2_{0.01(15)}$, H_0 is rejected, and H_1 is accepted: there are significant differences in the results of the orthogonal experiment.

Table 7. Results of Friedman statistics of orthogonal experiments.

k	α	χ^2	$\chi^2_{\alpha(k-1)}$
16	0.05	46.17	25.00
16	0.01	46.17	30.58

In summary, the results of the Friedman test with two significance levels of 0.05 and 0.01, in turn, showed that there are significant differences in the results obtained from this orthogonal test scheme, and the best performance was achieved with the parameter combination of test 10. Therefore, the parameter combination in test 10 was used as the algorithm parameter values for the IAHA in the later experiments with $\delta = 4$, $r_1 = 0.27$, $r_2 = 0.30$, $f_1 = 0.55$, and $m = 10$.

4.2. Comparison of IAHA and Other Variants of AHA

To verify the effects of the improvements made on the performance of the IAHA, variants of the AHA were constructed for comparison experiments, as shown in Table 8. Four improvement strategies were used to improve the AHA in three aspects where the GDM is an improvement of the decoding method, GLR is the initialization of the population using GLR, and the Levy flight and SSS are improvements of the foraging method for hummingbirds.

Table 8. Results of Friedman statistics of orthogonal experiments.

Algorithm	GDM	GLR	Levy	SSS
IAHA	√	√	√	√
AHA	×	×	×	×
AHA-GDM	√	×	×	×
AHA-GLR	×	√	×	×
AHA-Levy-SSS	×	×	√	√

In the comparison experiments, the number of iterations of all algorithms is 500, and the population size is 300. Five cases of the standard case Kacem are used for experiments to compare the effect of the improved method on the performance of the algorithm. Ten experiments are conducted for each case, and the mean square error (*MSE*) is calculated by Equation (22) and used as the experimental results along with the mean (*Mean*), as shown in Table 9.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(r_i - r_{\min})^2 \quad (22)$$

where n = 10; r_i is the maximum completion time obtained for each experiment; r_{min} is the optimal value of the maximum completion time known to be studied for each case, The r_{min} of Kacem 1–5 [37] are 11, 14, 11, 7, and 11, respectively [38].

Table 9. Results of Friedman statistics of orthogonal experiments.

Algorithm	Kac1		Kac2		Kac3		Kac4		Kac5	
	Mean	*MSE*	*Mean*	*MSE*	*Mean*	*MSE*	*Mean*	*MSE*	*Mean*	*MSE*
IAHA	11.0	0.000	14.1	0.100	11.0	0.000	7.2	0.200	11.1	0.100
AHA	11.9	0.900	15.0	1.200	12.1	1.300	7.9	0.900	11.8	0.800
AHA-GDM	11.5	0.500	14.8	0.800	11.8	0.800	7.4	0.400	11.6	0.600
AHA-GLR	11.6	0.600	14.7	0.700	11.6	0.600	7.6	0.600	11.6	0.600
AHA-Levy-SSS	11.3	0.300	14.6	0.600	11.2	0.200	7.2	0.200	11.4	0.400

According to the experimental results, the AHA deviates from other algorithms in *Mean* and has difficulty in searching r_{min} due to its relatively simple foraging method, which leads to its poor performance on *MSE*. The AHA-Levy-SSS improves the foraging method, the global and local search ability of the algorithm is enhanced, and, finally, outperforms the AHA on *MSE*. The AHA-GDM and AHA-GLR optimized the decoding method and initial population; the performance was slightly better than the AHA but still inferior to the AHA-Levy-SSS due to its relatively poor foraging ability. The IAHA achieved the best performance on both *Mean* and *MSE*.

Friedman's ranking was performed based on the *Mean* of the experimental results, and the sum of the result ranking of each algorithm in each case was calculated to derive the final ranking of the scheme. The ranking and final ranking of each algorithm are shown in Table 10.

Significance tests were performed on the ranking results of five algorithms. The original hypothesis H_0: there are no significant differences in the performance of each algorithm. The alternative hypothesis H_1: there are significant differences in the performance of each algorithm.

Table 10. Variant comparison experiments Friedman ranking results.

Algorithm	IAHA	AHA	AHA-GDM	AHA-GLR	AHA-Levy-SSS
Kac1	1	5	3	4	2
Kac2	1	5	4	3	2
Kac3	1	5	4	3	2
Kac4	1.5	5	3	4	1.5
Kac5	1	5	3.5	3.5	2
Total	5.5	25	17.5	17.5	9.5
Final	1	5	3.5	3.5	2

The results of Friedman's statistics for the comparison experiments are shown in Table 11. From Table 11, the Friedman statistic is $\chi^2 = 18.64$, and the degree of freedom is $k - 1 = 4$. When the significance level α is 0.05, querying the chi-square distribution table yields $\chi^2_{0.05(4)} = 9.49$. Since $\chi^2 > \chi^2_{0.05(4)}$, H_0 is rejected, and H_1 is accepted: there are significant differences in the performance of each algorithm. When the significance level α is 0.01, the chi-square distribution table $\chi^2_{0.01(4)} = 13.28$. Since $\chi^2 > \chi^2_{0.01(4)}$, H_0 is rejected, and H_1 is accepted: there are significant differences in the performance of each algorithm.

Table 11. Results of Friedman statistics for variant comparison experiments.

k	α	χ^2	$\chi^2_{\alpha(k-1)}$
5	0.05	18.64	9.49
5	0.01	18.64	13.28

To observe the specific performance of the algorithms during iterations, Figure 7 plots the convergence curves of the maximum completion time (makespan) with the number of iterations for the five algorithms in one iteration of Kacem case 5. The populations diverge at the first generation of individuals. The IAHA and AHA-GLR with GLR showed excellent makespan, demonstrating that GLR significantly improved the quality of the initial populations. As iterations increase, the AHA-Levy-SSS, with its excellent search capability, quickly closes the gap with the AHA and AHA-GDM and surpasses the AHA-GLR after 50 iterations, proving that Levy and SSS enhance the search capability of the AHA and accelerate the convergence speed. Although the AHA-GDM is inferior to the AHA-GLR and AHA-Levy-SSS, it is always ahead of the AHA throughout the iterations, and the AHA-Levy-SSS, which lacks the GDM, always lags behind the IAHA at iteration number 100–200 due to the lack of GDM's ability to optimize the scheduling results, proving that the GDM has the ability to optimize the scheduling results.

In summary, the results of the Friedman test with two significance levels of 0.05 and 0.01 show that the results obtained by the five algorithms are significantly different, and the IAHA achieves the best performance. The four strategies play major roles in different stages of the IAHA. In the early IAHA, the prepopulation quality is optimized mainly by GLR so that the IAHA can find relatively good scheduling results without too many iterations; in the middle IAHA iteration, the optimization-seeking ability of the algorithm is enhanced mainly by Levy and SSS to make it converge quickly; in the late IAHA iteration, the quality of scheduling results is improved mainly by the GDM.

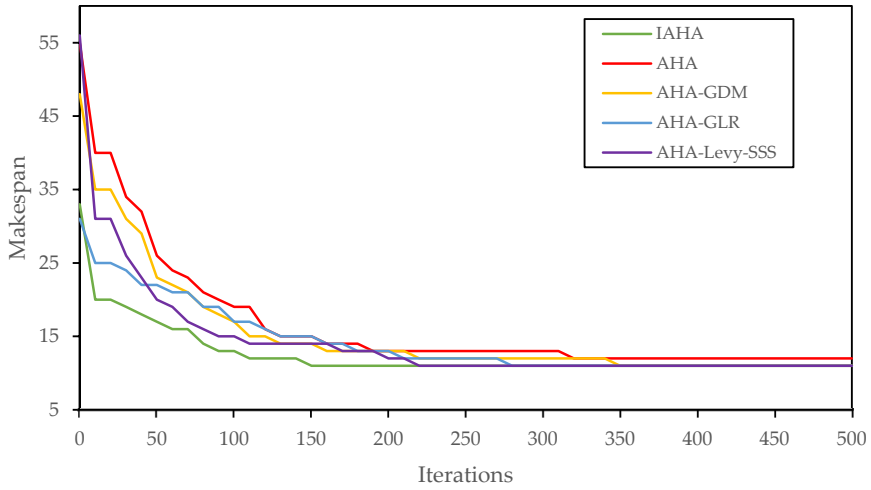

Figure 7. Convergence curves of 5 algorithms on Kacem 5.

4.3. Comparison of IAHA and Other Algorithms

To verify the effectiveness and superiority of the IAHA in solving crowdsourced task scheduling problems, the first 10 cases from the standard Brandimarte case [38] and 5 cases from Kacem [37], which are more commonly studied, are selected for experiments. The IAHA adjusts the population size and the number of iterations appropriately according to the case size, and the adjustment results are shown in Table 12, where n is the number of tasks and m is the SMU number.

Table 12. Population size and number of iterations of IAHA in test cases.

Test	$n \times m$	Population	Iteration
Mk01	10 × 6	300	500
Mk02	10 × 6	300	500
Mk03	15 × 8	300	500
Mk04	15 × 8	300	500
Mk05	15 × 4	300	500
Mk06	10 × 15	500	500
Mk07	20 × 5	300	500
Mk08	20 × 10	500	700
Mk09	20 × 10	500	700
Mk10	20 × 15	500	700
Kac1	4 × 5	200	500
Kac2	8 × 8	300	500
Kac3	10 × 7	300	500
Kac4	10 × 10	300	500
Kac5	15 × 10	500	500

The IAHA was run five times independently on each case to take the optimal solution, and the experimental results were compared with other algorithms, as in Table 13, where C^* indicates the optimal value of makespan currently searched for that case in the literature and the optimal solution searched for by each of the Cmax algorithms, and "-" indicates that the algorithm did not provide data for that arithmetic case. The comparison algorithms are HA by Ziaee [39], PHMH by Bozejko [40], NIMASS by Xiong [41], ADCSO by Jiang [42], HLO-PSO by Ding [43], SLGA by Chen [44], and IAHA by the paper, where the experimental results of the comparison algorithms are from the original data in the literature.

Table 13. Optimal solutions obtained for the 7 algorithms in the test cases.

Test	C^*	C_{max}						
		HA	PHMH	NIMASS	ADCSO	HLO-PSO	SLGA	IAHA
Mk01	36	42	41	40	40	40	40	40
Mk02	24	28	30	28	27	28	27	26
Mk03	204	204	204	204	204	204	204	204
Mk04	48	75	65	65	64	63	60	64
Mk05	168	179	174	177	173	175	172	173
Mk06	33	69	71	67	66	71	69	57
Mk07	133	149	148	144	144	144	144	144
Mk08	523	555	551	523	523	523	523	523
Mk09	299	342	410	312	311	326	320	311
Mk10	165	242	267	229	226	238	254	204
Kac1	11	11	-	-	11	-	11	11
Kac2	14	15	-	14	14	-	14	14
Kac3	11	13	-	-	11	-	11	11
Kac4	7	7	-	7	7	-	-	7
Kac5	11	12	-	11	11	-	-	11

Figure 8 shows the Gantt chart of the scheduling results obtained by the IAHA on the Mk10 case, respectively. In Figure 8, the first number on the block represents the task number, and the second number represents the subtask number

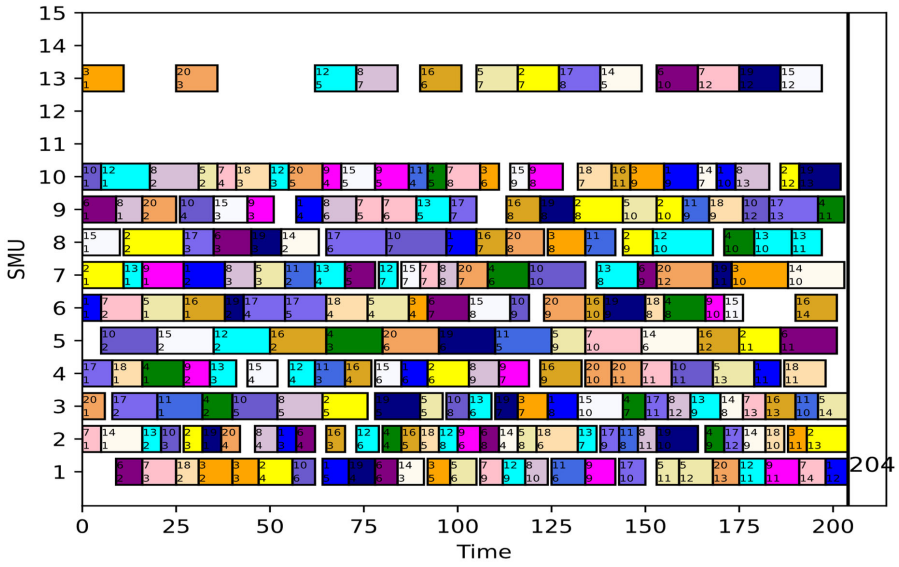

Figure 8. Gantt chart obtained by IAHA at Mk10.

Since the IAHA searches for C^* in all five Kacem cases and many algorithms do not provide results, the comparison between the IAHA and other algorithms will not be discussed later.

To facilitate a visual comparison of the performance of the IAHA with other algorithms on the Brandimarte cases, the experimental results are converted into relative percentage deviation (*RPD*). *RPD* is the percentage deviation of the algorithm result with respect to C^*, which denotes the closeness of C_{max} to C^*, and is calculated as follows:

$$RPD = \frac{C_{max} - C^*}{C^*} \times 100\% \tag{23}$$

Table 14 shows the *RPD* results of the seven algorithms on the test cases. The average *RPD* of the IAHA achieves the smallest value, which indicates that the IAHA performs the

best among the seven algorithms, especially on the Mk6 and Mk10 cases. The performance of the IAHA pulls away from the other algorithms by a large gap.

Table 14. *RPD* of the 7 algorithms on the test cases.

Test	*RPD*						
	HA	**PHMH**	**NIMASS**	**ADCSO**	**HLO-PSO**	**SLGA**	**IAHA**
Mk01	16.67	13.89	11.11	11.11	11.11	11.11	11.11
Mk02	16.67	25.00	16.67	12.50	16.67	12.50	8.33
Mk03	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Mk04	56.25	35.42	35.42	33.33	31.25	25.00	33.33
Mk05	6.55	3.57	5.36	2.98	4.17	2.38	2.98
Mk06	109.09	115.15	103.03	100.00	115.15	109.09	72.73
Mk07	12.03	11.28	8.27	8.27	8.27	8.27	8.27
Mk08	6.12	5.35	0.00	0.00	0.00	0.00	0.00
Mk09	14.38	37.12	4.35	4.01	9.03	7.02	4.01
Mk10	46.67	61.82	38.79	36.97	44.24	53.94	23.64
Mean	28.44	30.86	22.30	20.92	23.99	22.93	19.17

Friedman's ranking was performed on the *RPD*, and the sum of the resultant rankings of each algorithm was calculated for each case to obtain the final ranking of the solutions. The ranking and final ranking of each experimental scheme are shown in Table 15, which shows that the final Friedman ranking of the IAHA is the best among the seven algorithms.

Table 15. Results of Friedman ranking for 7 algorithms.

Test	HA	PHMH	NIMASS	ADCSO	HLO-PSO	SLGA	IAHA
Mk01	7	6	3	3	3	3	3
Mk02	5	7	5	2.5	5	2.5	1
Mk03	4	4	4	4	4	4	4
Mk04	7	5.5	5.5	3.5	2	1	3.5
Mk05	7	4	6	2.5	5	1	2.5
Mk06	4.5	6.5	3	2	6.5	4.5	1
Mk07	7	6	3	3	3	3	3
Mk08	7	6	3	3	3	3	3
Mk09	6	7	3	1.5	5	4	1.5
Mk10	5	7	3	2	4	6	1
Total	59.5	59	38.5	27.5	40.5	32.5	23.5
Final	7	6	4	2	5	3	1

Significance tests were performed on the ranking results of seven algorithms. The original hypothesis H_0: there are no significant differences in the performance of each algorithm. The alternative hypothesis H_1: there are significant differences in the performance of each algorithm.

The results of Friedman's statistics for the comparison experiments are shown in Table 16. From Table 16, the Friedman statistic is $\chi^2 = 26.76$, and the degree of freedom is $k - 1 = 6$. When the significance level α is 0.05, querying the chi-square distribution table yields $\chi^2_{0.05(6)} = 12.59$. Since $\chi^2 > \chi^2_{0.05(6)}$, H_0 is rejected, and H_1 is accepted: there are significant differences in the performance of each algorithm. When the significance level α is 0.01, the chi-square distribution table $\chi^2_{0.01(6)} = 16.81$. Since $\chi^2 > \chi^2_{0.01(6)}$, H_0 is rejected, and H_1 is accepted: there are significant differences in the performance of each algorithm. In summary, the results of Friedman's test with two significance levels of 0.05 and 0.01, respectively, show that there are significant differences in the performance of HA, PHMH, NIMASS, ADCSO, HLO-PSO, SLGA, and IAHA, and the IAHA has the best performance.

Table 16. Experimental Friedman statistics of IAHA compared with other algorithms.

k	α	χ^2	$\chi^2_{\alpha(k-1)}$
7	0.05	26.76	12.59
7	0.01	26.76	16.81

4.4. Application to Processing Crowdsourcing Task Scheduling

In order to further verify the effectiveness of the model and algorithm in this paper, we used 30 local small- and medium-sized manufacturing enterprises as SMUs, set the processing time of subtasks according to the enterprise scale, and simulated 15 processing crowdsourcing task data as cases. The processing crowdsourcing task case size is 15 × 30. The AHA and IAHA were used to simulate and solve them, respectively, and the parameter settings follow the results of the 4.1 orthogonal test; the number of populations is 300, and the number of iterations is 500.

The convergence curves can be obtained using the IAHA and AHA for the above case, as shown in Figure 9. It can be seen that the makespan of the IAHA and AHA converge at 105 and 116, respectively. The IAHA obtains a significantly better solution than the AHA and has a better performance in this processing crowdsourcing case. The IAHA, due to its unique initialization method, is already significantly in the lead in the early iterations of the algorithm and plays a key role in guiding the evolutionary direction of the population. In the late iteration of the algorithm, the IAHA is more likely to obtain better solutions due to its superior foraging method improvement strategy. Figure 10 shows the Gantt chart of the scheduling solution obtained by the IAHA.

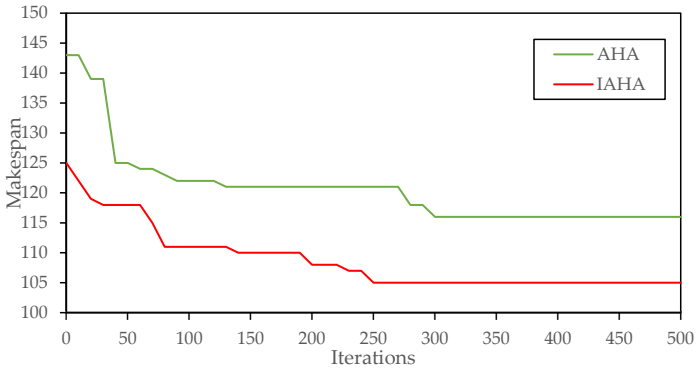

Figure 9. Convergence curves of IAHA and AHA on crowdsourcing case.

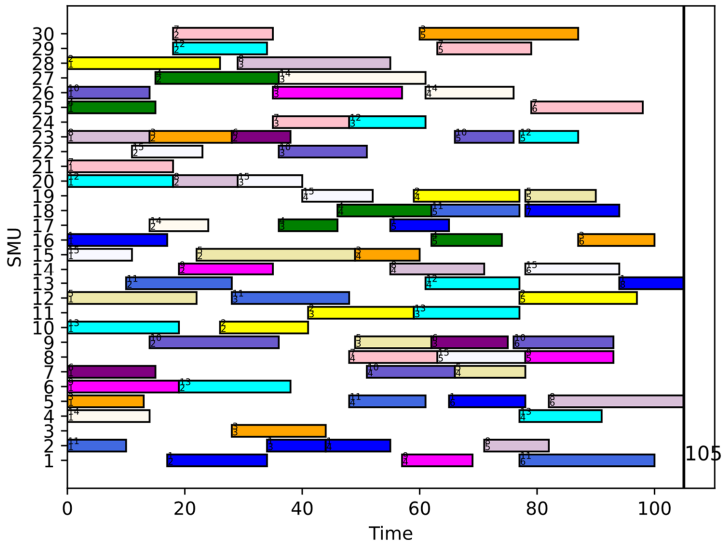

Figure 10. Gantt chart obtained by IAHA in crowdsourcing case.

5. Conclusions

In the social manufacturing model, dispersed manufacturing resources accomplish multiple manufacturing tasks collaboratively under multiple interest coordination and social mechanisms. In the study, a mixed-integer programming model with the optimization objective of minimizing the maximum completion time is developed for the mathematical formulation of the processing crowdsourcing task scheduling problem in social manufacturing. To solve the problem, the study proposes an improved artificial hummingbird algorithm (IAHA) as a solution algorithm for the processing crowdsourcing task scheduling problem based on a novel bionic optimization algorithm artificial hummingbird algorithm (AHA). For the discrete characteristics of the problem, the problem is divided into two subproblems: social manufacturing unit selection and subtask sequence. Suitable encodings are designed for the two subproblems, different decoding methods are matched for the two encodings, and greedy decoding methods are used to improve the quality of understanding. To improve the quality of the initial population and speed up the convergence, the IAHA used the initialization rules of global selection, local selection, and random selection. To enhance the merit-seeking ability of the IAHA, the Levy flight was used to improve the guided foraging and territorial foraging of the IAHA, and the simplex search strategy was used to improve the migration foraging of the IAHA.

Four different experiments are designed in this study: (1) the optimal combination of parameters for the IAHA in the scheduling problem is obtained through parameter orthogonality experiments; (2) the effectiveness of various improvement strategies of the IAHA is verified through comparison experiments among the IAHA and other variants of the AHA; (3) the superiority of the IAHA is demonstrated through comparison experiments between the IAHA and algorithms proposed by other scholars on standard cases; (4) the performance of the IAHA on a simulated processing crowdsourcing task case demonstrates that the IAHA can obtain a more efficient and economical scheduling solution.

The IAHA still has some limitations, such as the following: (1) The IAHA has good performance when facing relatively small-scale manufacturing environments, but the complexity of the IAHA increases when facing very large-scale environments; the solution time explodes, so the use of deep reinforcement learning and other methods should be considered to reduce the solution time. (2) The demand for low carbon in today's society is increasing, so the multiobjective optimization problem of adding some green performance and other indicators should be considered.

Author Contributions: Methodology, software, validation, formal analysis, and writing—original draft preparation, G.Z. conceptualization, supervision, project administration, funding acquisition, and writing—review and editing, D.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 71961005, and the Natural Science Foundation of Guangxi Zhuang Autonomous Region, grant number 2020GXNSFAA297024.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jiang, P.Y.; Leng, J.W.; Ding, K. Analyzing and delimiting overlapping concept boundaries of social manufacturing. *Comput. Integr. Manuf. Syst.* **2018**, *24*, 829–837.
2. Jiang, P.Y.; Ding, K.; Leng, J.W. Social Manufacturing: Drivers, Research Status, and Trends. *Ind. Eng. J.* **2016**, *19*, 1–9.
3. Jiang, P.Y.; Shi, L.H.; Yang, M.L.; Guo, W.; Makanda, I.L.D.; Dong, W.H. Research and development of social manufacturing model and 3D printing testbed for industrial internet. *Sci. Sin. Technol.* **2022**, *52*, 88–103. [CrossRef]
4. Jiang, P.Y.; Ding, K.; Leng, J.W. Towards a cyber-physical-social-connected and service-oriented manufacturing paradigm: Social manufacturing. *Manuf. Lett.* **2016**, *7*, 15–21.
5. Brucker, P.; Schlie, R. Job-shop scheduling with multi-purpose machines. *Computing* **1990**, *45*, 369–375. [CrossRef]

6. Mati, Y.; Xie, X. The complexity of two-job shop problems with multi-purpose unrelated machines. *Eur. J. Oper. Res.* **2004**, *152*, 159–169. [CrossRef]
7. Kaskavelis, C.A.; Caramanis, M.C. Efficient Lagrangian relaxation algorithms for industry size job-shop scheduling problems. *IIE Trans.* **1998**, *30*, 1085–1097. [CrossRef]
8. Chen, H.; Chu, C.; Proth, J.M. An improvement of the Lagrangian relaxation approach for job shop scheduling: A dynamic programming method. *IEEE Trans. Robot. Autom.* **1998**, *14*, 786–795. [CrossRef]
9. Ríos-Mercado, R.Z.; Bard, J.F. Computational experience with a branch-and-cut algorithm for flowshop scheduling with setups. *Comput. Oper. Res.* **1998**, *25*, 351–366. [CrossRef]
10. Wang, L.; Luo, C.; Cai, J.A. Variable interval rescheduling strategy for dynamic flexible job shop scheduling problem by improved genetic algorithm. *J. Adv. Transport.* **2017**, *2017*, 1–12.
11. Huang, M.; Wang, M.; Xu, L. An improved genetic algorithm using opposition based learning for flexible job shop scheduling problem. In Proceedings of the 2nd International Conference on Cloud Computing and Internet of Things (CCIOT), Dalian, China, 22 October 2016.
12. Cao, Y.; Shi, H.B.; Han, Z.H. Multi-objective flexible job shop scheduling problem using differential evolution algorithm. In Proceedings of the 9th International Conference on Modelling, Identification and Control (ICMIC), Kunming, China, 10 July 2017.
13. Wang, L.; Cai, J.; Li, M.; Liu, Z. Flexible job shop scheduling problem using an improved ant colony optimization. *Sci. Program.* **2017**, *2017*, 9016303. [CrossRef]
14. Kato, E.R.; Diego, D.A.A.G.; Tsunaki, R.H. A new approach to solve the flexible job shop problem based on an hybrid particle swarm optimization and random-restart hill climbing. *Comput. Ind. Eng.* **2018**, *125*, 178–189. [CrossRef]
15. Zhang, G.; Shao, X.; Li, P.; Liang, G. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job shop scheduling problem. *Comput. Ind. Eng.* **2009**, *56*, 1309–1318. [CrossRef]
16. Tang, H.T.; Chen, R.; Li, Y.B.; Peng, Z.; Guo, S.S.; Du, Y.Z. Flexible job-shop scheduling with tolerated time interval and limited starting time interval based on hybrid discrete PSO-SA: An application from a casting workshop. *Appl. Soft Comput.* **2019**, *78*, 176–194.
17. Gao, K.Z.; Suganthan, P.N.; Pan, Q.K.; Chua, T.J.; Chong, C.S.; Cai, T.X. An improved artificial bee colony algorithm for flexible job shop scheduling problem with fuzzy processing time. *Expert Syst. Appl.* **2016**, *65*, 52–67. [CrossRef]
18. Wang, Y.F.; Ge, J.R.; Miao, S.; Jiang, T.H.; Shen, X.N. Application of hybrid artificial bee colony algorithm based on load balancing in aerospace composite material manufacturing. *Expert Syst. Appl.* **2023**, *215*, 119375. [CrossRef]
19. Yuan, Y.; Xu, H. Flexible job shop scheduling using hybrid differential evolution algorithms. *Comput. Ind. Eng.* **2013**, *65*, 246–260. [CrossRef]
20. Sun, K.; Zheng, D.; Song, H.; Cheng, Z.; Lang, X.; Yuan, W.; Wang, J. Hybrid genetic algorithm with variable neighborhood search for flexible job shop scheduling problem in a machining system. *Expert Syst. Appl.* **2023**, *215*, 119359. [CrossRef]
21. Xu, W.; Wu, W.; Wang, Y.; He, Y.; Lei, Z. Flexible job-shop scheduling method based on interval grey processing time. *Appl. Intell.* **2022**, *53*, 14876–14891. [CrossRef]
22. Wang, G.G.; Gao, D.; Pedrycz, W. Solving Multi-Objective Fuzzy Job-Shop Scheduling Problem by a Hybrid Adaptive Differential Evolution Algorithm. *IEEE Trans. Ind. Inform.* **2022**, *18*, 8519–8528. [CrossRef]
23. Lou, H.; Wang, X.; Dong, Z.; Yang, Y. Memetic algorithm based on learning and decomposition for multiobjective flexible job shop scheduling considering human factors. *Swarm Evol. Comput.* **2022**, *75*, 101204. [CrossRef]
24. Yan, S.; Zhang, G.; Sun, J.; Zhang, W. An improved ant colony optimization for solving the flexible job shop scheduling problem with multiple time constraints. *Math. Biosci. Eng.* **2023**, *20*, 7519–7547. [CrossRef] [PubMed]
25. Tian, Y.; Gao, Z.; Zhang, L.; Chen, Y.; Wang, T. A Multi-Objective Optimization Method for Flexible Job Shop Scheduling Considering Cutting-Tool Degradation with Energy-Saving Measures. *Mathematics* **2023**, *11*, 324. [CrossRef]
26. Wu, X.L.; Liu, X.J. Differential evolution algorithm for solving distributed flexible job shop scheduling problem. *Comput. Integr. Manuf. Syst.* **2019**, *25*, 2539–2558. [CrossRef]
27. Zhao, W.; Wang, L.; Mirjalili, S. Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications. *Comput. Methods Appl. Mech. Eng.* **2022**, *388*, 114194. [CrossRef]
28. Ramadan, A.; Ebeed, M.; Kamel, S.; Ahmed, E.M.; Tostado-Veliz, M. Optimal allocation of renewable DGs using artificial hummingbird algorithm under uncertainty conditions. *Ain Shams Eng. J.* **2023**, *14*, 101872. [CrossRef]
29. Kansal, V.; Dhillon, J.S.; Yan, J. Ameliorated artificial hummingbird algorithm for coordinated wind-solar-thermal generation scheduling problem in multi-objective framework. *Appl. Energy* **2022**, *326*, 120031. [CrossRef]
30. Kiymac, E.; Kaya, Y. A novel automated CNN arrhythmia classifier with memory-enhanced artificial hummingbird algorithm. *Expert Syst. Appl.* **2023**, *213*, 119162. [CrossRef]
31. Zou, Y.J.; Song, Y.C.; Wang, Y.; Wang, X.K. AGV and machine integrated scheduling method based on discrete whale optimization algorithm. *J. Chongqing Univ.* **2022**, *45*, 55–74.
32. Wang, X.J.; Gao, L.; Zhang, C.Y.; Li, X.Y. A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem. *Int. J. Adv. Manuf. Technol.* **2010**, *51*, 757–767. [CrossRef]
33. Shlesinger, M. Lévy Flights and Related Topics in Physics. *Lect. Notes Phys.* **1995**, *450*, 3–540.
34. Liu, M.; Yao, X.; Li, Y. Hybrid whale optimization algorithm enhanced with Levy flight and differential evolution for job shop scheduling problems. *Appl. Soft Comput.* **2020**, *87*, 105954. [CrossRef]

35. Lagarias, J.C.; Reeds, J.A.; Wright, M.H.; Wright, P.E. Convergence properties of the Nelder-Mead simplex method in low dimensions. *Siam J. Optimiz.* **1998**, *9*, 112–147. [CrossRef]
36. Derrac, J.; Garcia, S.; Molina, D.; Herrera, F. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm Evol. Comput.* **2011**, *1*, 3–18. [CrossRef]
37. Kacem, I.; Hammadi, S.; Borne, P. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. *IEEE Trans. Syst. Man Cybern. Part C* **2002**, *32*, 1–13. [CrossRef]
38. Brandimarte, P. Routing and scheduling in a flexible job shop by tabu search. *Ann. Oper. Res.* **1993**, *41*, 157–183. [CrossRef]
39. Ziaee, M. A heuristic algorithm for solving flexible job shop scheduling problem. *Int. J. Adv. Manuf. Technol.* **2014**, *71*, 519–528. [CrossRef]
40. Bozejko, W.; Uchronski, M.; Wodecki, M. Parallel hybrid metaheuristics for the flexible job shop problem. *Comput. Ind. Eng.* **2010**, *59*, 323–333. [CrossRef]
41. Xiong, W.; Fu, D. A new immune multi-agent system for the flexible job shop scheduling problem. *J. Intell. Manuf.* **2018**, *29*, 857–873. [CrossRef]
42. Jiang, T.H.; Zhang, C. Adaptive discrete cat swarm optimisation algorithm for the flexible job shop problem. *Int. J. Bio-Inspir. Comput.* **2019**, *13*, 199. [CrossRef]
43. Ding, H.; Gu, X. Hybrid of human learning optimization algorithm and particle swarm optimization algorithm with scheduling strategies for the flexible job-shop scheduling problem. *Neurocomputing* **2020**, *414*, 313–332. [CrossRef]
44. Chen, R.; Yang, B.; Li, S.; Wang, S. A Self-Learning Genetic Algorithm based on Reinforcement Learning for Flexible Job-shop Scheduling Problem. *Comput. Ind. Eng.* **2020**, *149*, 106778. [CrossRef]