mathematics

MDPI

Article

A Decentralized Optimization Algorithm for Multi-Agent Job Shop Scheduling with Private Information

Xinmin Zhou [1,2], Wenhao Rao [1,2], Yaqiong Liu [1,2] and Shudong Sun [1,2,*]

[1] School of Mechanical Engineering, Northwestern Polytechnical University, Xi'an 710072, China; siem@nwpu.edu.cn (X.Z.); raowh@mail.nwpu.edu.cn (W.R.); yaqiongliu@mail.nwpu.edu.cn (Y.L.)
[2] Key Laboratory of Industrial Engineering and Intelligent Manufacturing, Ministry of Industry and Information Technology, Xi'an 710072, China
* Correspondence: sdsun@nwpu.edu.cn

Abstract: The optimization of job shop scheduling is pivotal for improving overall production efficiency within a workshop. In demand-driven personalized production modes, achieving a balance between workshop resources and the diverse demands of customers presents a challenge in scheduling. Additionally, considering the self-interested behaviors of agents, this study focuses on tackling the problem of multi-agent job shop scheduling with private information. Multiple consumer agents and one job shop agent are considered, all of which are self-interested and have private information. To address this problem, a two-stage decentralized algorithm rooted in the genetic algorithm is developed to achieve a consensus schedule. The algorithm allows agents to evolve independently and concurrently, aiming to satisfy individual requirements. To prevent becoming trapped in a local optimum, the search space is broadened through crossover between agents and agent-based block insertion. Non-dominated sorting and grey relational analysis are applied to generate the final solution with high social welfare. The proposed algorithm is compared using a centralized approach and two state-of-the-art decentralized approaches in computational experiments involving 734 problem instances. The results validate that the proposed algorithm generates non-dominated solutions with strong convergence and uniformity. Moreover, the final solution produced by the developed algorithm outperforms those of the decentralized approaches. These advantages are more pronounced in larger-scale problem instances with more agents.

Keywords: multi-agent scheduling; decentralized decision making; genetic algorithm; negotiation optimization; social welfare

MSC: 90-10

Citation: Zhou, X.; Rao, W.; Liu, Y.; Sun, S. A Decentralized Optimization Algorithm for Multi-Agent Job Shop Scheduling with Private Information. *Mathematics* **2024**, *12*, 971. https://doi.org/10.3390/math12070971

Academic Editors: Andreas C. Georgiou, Fajun Yang and Chunjiang Zhang

Received: 14 December 2023
Revised: 27 February 2024
Accepted: 22 March 2024
Published: 25 March 2024

1. Introduction

In manufacturing systems, job shop scheduling plays a pivotal role and has a profound impact on the overall production efficiency of the shop [1]. Traditional job shop scheduling problems generally focus on the holistic optimization of resources on the shop floor. However, with the rapid development of customer-driven production models, job shop scheduling is facing new challenges. As multiple consumers compete for limited machine resources while pursuing individual objectives, the related scheduling goal is not to satisfy the job shop's preferences but to accommodate its objective and the unique demands of various consumers [2]. In practical production environments, numerous real-world scenarios exemplify the complexities outlined. For example, Xi'an Aero-Engine Group, a subsidiary of the Aviation Industry Corporation of China (AVIC), concurrently produces aircraft engine blades for both Pratt & Whitney Canada (PWC) and Safran Aircraft Engines (formerly known as SNECMA). While Xi'an Aero-Engine Group has its own objectives as a provider of manufacturing resources, Pratt & Whitney Canada (PWC) and Safran Aircraft Engines, as consumers, have their respective goals. During the production scheduling

process, decision-makers need to consider the objectives of all stakeholders. Simultaneously, with the promotion of globalized manufacturing, the original equipment manufacturers (OEM) production model has emerged. Renowned contract manufacturers such as Foxconn and Quanta undertake the production of various product models for different enterprises, ensuring the timely and demand-driven market entry of products. Additionally, since all stakeholders are self-interested and rational, they may not be willing to disclose their private information [3]. Therefore, in a production model driven by personalized consumer demands, manufacturing enterprises aiming to survive in a competitive environment must strive to meet the individual needs of self-interested users as much as possible. Balancing the contradictions between product diversification, resource coordination, and cost while satisfying the personalized demands of multiple users with private information becomes the new core of job shop scheduling. Within this context, this paper addresses the multi-agent job shop scheduling problem with private information (MJSSP-PI).

Meta-heuristic algorithms are typically employed for multi-agent scheduling problems by constructing multi-objective models and generating the Pareto solution sets [4–6]. Such algorithms have the capability to handle intricate scheduling constraints and converge swiftly. A single central authority possessing complete information on the entire scheduling system is required in these algorithms to make decisions. However, in practical scenarios, agents often have self-interest and may be reluctant to disclose their private information [7], rendering the centralized meta-heuristic algorithms unfeasible to address the targeted MJSSP-PI. In contrast, a decentralized scheduling architecture that relies on multiple local autonomous decision-makers with partial information is an available alternative [8]. Therefore, a decentralized optimization approach inspired by meta-heuristic algorithms has emerged as a research focus to solve the resource allocation problem with information asymmetry.

Under the decentralized architecture, various approaches such as the contract net protocol [9,10], auction theory [11,12], and meta-heuristic algorithms [13] have been commonly employed to solve multi-agent machine scheduling problems. In light of the strengths exhibited by meta-heuristic algorithms in tackling multi-objective optimization problems, the decentralized frameworks based on them have gained significant attention. For example, two decentralized frameworks inspired by evolutionary search and ant colony optimization were initially introduced for solving the multi-project scheduling problem in [14] and [15], respectively. Afterward, Lang et al. [16] constructed two decentralized negotiation mechanisms based on the genetic algorithm (GA) and simulated annealing (SA), which are more generic compared to [14] and [15]. The SA-based framework has been extended to address various real-world problems, including diagnostic services scheduling [17], the resource investment problem [18], and the community ride-sharing matching problem [19]. These methods have demonstrated high efficiency, but their reliance on multiple local optimizations of a single proposed solution restricts the extensive search of the Pareto frontier, potentially leading to sub-optimal results in terms of social welfare. In these meta-heuristic algorithms, the distinctive encoding approach of GA offers an advantageous feature for concealing the essential proprietary information of self-interested agents. Furthermore, GA exhibits the capacity to yield high-quality solutions within a constrained temporal framework through the evolutionary progression of multiple populations [20–22]. Therefore, this study proposes a decentralized negotiation optimization approach based on GA for addressing the MJSSP-PI.

Nevertheless, the GA-based approaches face challenges in escaping local optima and searching for superior solutions, particularly in scenarios with numerous objectives and intense inter-agent competition. Consequently, the scale of the problems that can be effectively solved is limited. Extensive research has been conducted to explore various strategies aimed at preventing meta-heuristic algorithms from getting trapped in local optima. Epitropakis et al. [23] and Lin et al. [24] found that the achievement of global optimality strongly depends on the algorithm's ability to effectively explore the solution space. Based on this finding, certain studies have been devoted to expanding the solution

space for solving job shop scheduling problems. For instance, refs. [25–27] enhanced the algorithm's search capability by introducing the variable neighborhood search, while [28] employed an extinction mechanism to solve local tramp. Furthermore, Wen et al. [29] embedded elite and mutation strategies in the evolution phase to enhance the diversity of solutions within the population. However, these methods are applied under the assumption of complete information and are inapplicable to our problem.

This study focuses on the multi-agent job shop scheduling problem with private information, which involves multiple consumer agents and a job shop agent, all driven by self-interest and individual objectives. To overcome the challenges above, a two-stage GA-based decentralized optimization algorithm (GDOA) is developed to generate a consensus solution with high social welfare. The main contributions are summarized as follows:

- An autonomous and parallel evolution of agents is developed, wherein agents individually construct private fitness functions that align with their specific objectives and subsequently evolve independently and concurrently. This process protects private information and ensures the fulfillment of each consumer agent's requirement.
- A novel evolution method, primarily including crossover between agents, mutation within agents, and agent-based block insertion, is proposed to prevent the algorithm from becoming trapped in local optima while safeguarding the privacy of agents.
- A negotiation decision-making approach comprising non-dominated sorting and grey relational analysis (GRA) is established to generate a final schedule that maximizes social welfare.

The remaining sections of this paper are structured as follows: In Section 2, a mathematical model for MJSSP-PI is constructed. The proposed GDOA algorithm is described in Section 3. Section 4 presents the computational experiments and the corresponding results. Finally, Section 5 focuses on conclusions and future work.

2. Problem Statement

2.1. Problem Description

The considered MJSSP-PI, as depicted in Figure 1, involves a scheduling system comprising a job shop and multiple consumers. The job shop is regarded as the job shop Agent (JSA), while multiple consumers are considered as consumer agents (CAs). Specifically, a set $J = \{J_1, J_2, \ldots, J_i, J_n\}$ of n jobs are processed on m machines $M = \{M_1, M_2, \ldots, M_j, \ldots, M_m\}$ that are owned by the job shop agent JSA. Job set J belongs to N consumer agents $CA = \{CA_1, CA_2, \ldots, CA_k, \ldots, CA_N\}$, and each CA_k has a job set J_{set}^k, where $J = J_{set}^1 \cup J_{set}^2 \cup \ldots \cup J_{set}^k \cup \ldots \cup J_{set}^N$ and $J_{set}^1 \cap J_{set}^k = \varnothing$, $k = 1, 2, \ldots, N$. Each job J_i consists of a sequence of m operations, in which $O_{i,j}$ denotes the jth operation of the job J_i, and $p_{i,j}$ represents the processing time of $O_{i,j}$. A three-element tuple $J_i = \{p_{i,j}, d_i, \omega_i\}$ characterizes the job J_i, where d_i is the due date and ω_i denotes the weight of J_i. The machine M_j is characterized by three types of energy consumption: starting energy consumption SE_j, unit processing energy consumption PE_j, and unit idle energy consumption IE_j. The processing time of jobs is public information known to the job shop agent and all consumer agents, while other information is private.

The assumptions and constraints considered in this paper are summarized as follows:

1. All jobs and machines are available at time 0;
2. Each operation can only be processed on one machine at a time;
3. Each machine can only process one operation at any time;
4. The preemption of any operation is not allowed;
5. The operations of a job should be processed sequentially in a predetermined order.

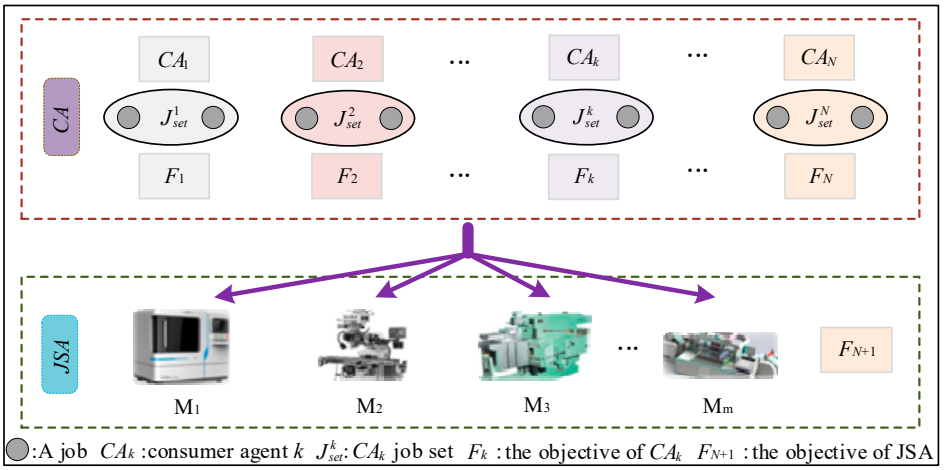

Figure 1. Schematic diagram of MJSSP-PI.

2.2. Mathematical Model

Given the consideration of the various individual objectives of agents in the MJSSP-PI, we introduced the special objectives of CAs and JSA before presenting the mathematical model. Meanwhile, to enhance the clarity of the mathematical model, the indexes of CAs and JSA are redefined in this section. Let l denotes different agents, where l = 1, 2, …, k, …, N, N + 1. The N consumer agents are indexed from 1 to N, and the job shop agent is indexed by N + 1.

The objectives of CAs and JSA are individual and private. An objective function set of consumer agents is constructed as (1), from which each consumer agent selects its preference.

$$minF(l) = \begin{cases} C^l_{max} = \max\limits_{i \in J^k_{set}} C_i \\ WC^l = \sum\limits_{i \in J^k_{set}} \omega_i C_i \\ WT^l = \sum\limits_{i \in J^k_{set}} \omega_i \times \max(0, C_i - d_i) \\ WE^l = \sum\limits_{i \in J^k_{set}} \omega_i \times \max(0, d_i - C_i) \end{cases}, l = 1, 2, \ldots, k, \ldots, N \quad (1)$$

where C^l_{max} denotes the makespan, WC^l is the total weighted completion time, WT^l means the total weighted tardiness, and WE^l represents the total weighted earliness.

The objective of JSA is to minimize the total energy consumption, which is formulated by Equation (2).

$$\begin{aligned} \min F(l) &= \sum_{j=1}^{m} \left(SE_j + PE_j \times \sum_{i=1}^{n} p_{i,j} + IE_j \times IT_j \right), l = N + 1 \\ IT_j &= \sum_{h=1}^{n} \left(st_{S^h_j, j} - ct_{S^{h-1}_j, j} \right), j = 1, 2, \ldots, m \end{aligned} \quad (2)$$

The total energy consumption $F(l)$ is the sum of the starting energy consumption, processing energy consumption, and idle energy consumption of all machines. The idle energy consumption of each machine is the product of the total idle time for all operations of the machine and the unit idle energy consumption.

Since the MJSSP-PI is solved in a distributed manner, the total social welfare [30] is applied to measure the schedule. To calculate the social welfare, it is necessary to normalize the objective of each agent into dimensionless quality, which is presented by the utility function. This utility function serves two purposes: first, it normalizes the objectives of individual agents, thereby eliminating the influence of varying objective values; second, it

ensures the preservation of privacy by concealing the actual objective values. The utility function is shown in Equation (3).

$$U\left(\pi^l\right) = \frac{\max\left(F\left(Par^l\right)\right) - F\left(\pi^l\right)}{\max\left(F\left(Par^l\right)\right) - \min\left(F\left(Par^l\right)\right)}, l = 1, 2, \ldots, k, \ldots, N, N+1 \tag{3}$$

where $F\left(\pi^l\right)$ is the objective function value corresponding to the agent l in the schedule π. Analogously, $max\left(F\left(Par^l\right)\right)$ and $min\left(F\left(Par^l\right)\right)$ denote the maximum and minimum objective values in the Pareto set P_{par}, respectively. $U\left(\pi^l\right)$ represents the transformed utility function value. Consequently, the total social welfare is given in Equation (4):

$$SW(\pi) = \sum_{l=1}^{N+1} U\left(\pi^l\right) \tag{4}$$

Based on the assumptions and the defined objectives, the mathematical model of MJSSP-PI is constructed as follows:

$$F = maxSW \tag{5}$$

$$st_{S_j^h,j} \geq ct_{S_j^{h-1},j}, j = 1, 2, \ldots, m, h = 2, 3, \ldots, n \tag{6}$$

$$st_{i,j} \geq ct_{i,j-1}, i = 1, 2, \ldots, n, j = 2, 3, \ldots, m \tag{7}$$

$$ct_{i,j} \geq p_{i,j}, i = 1, 2, \ldots, n, j = 1, 2, \ldots, m \tag{8}$$

$$ct_{i,j} \leq C_i, i = 1, 2, \ldots, n, j = 1, 2, \ldots, m \tag{9}$$

$$st_{S_j^h,j}, ct_{S_j^h,j}, st_{i,j}, ct_{i,j} \geq 0, i = 1, 2, \ldots, n, j = 1, 2, \ldots, m, h = 1, 2, 3, \ldots, n \tag{10}$$

Equation (5) represents that the optimization objective is to maximize the total social welfare. Constraint (6) ensures that each machine can only process one operation at any given time. Constraint (7) indicates that the completion time of any operation must be greater than the starting time of its immediate successor operation. Constraint (8) and Constraint (9) determine the completion time of an operation. Constraint (10) defines decision variables.

3. The Proposed GDOA Algorithm

Given the attributes of MJSSP-PI, the confidentiality of private information among agents must be maintained throughout the scheduling process. Consequently, in algorithmic design, the objectives encompass accommodating agent preferences, preserving the confidentiality of proprietary goal information, optimizing the quality of the Pareto optimization solution set, and attaining the solution with the highest social welfare value. Considering these objectives, this section presents a novel two-stage GA-based distributed optimization algorithm, including the parallel genetic evolution of consumer agents and negotiated decision-making. First, the main framework of the algorithm is introduced, followed by a detailed description of each stage.

This section may be divided by subheadings. It should provide a concise and precise description of the experimental results, their interpretation, as well as the experimental conclusions that can be drawn.

3.1. A Two-Stage GDOA Framework

The main framework of the proposed two-stage GDOA is illustrated in Figure 2, where G is the maximum number of iterations. The framework consists of a parallel genetic evolution stage and a negotiated decision-making stage.

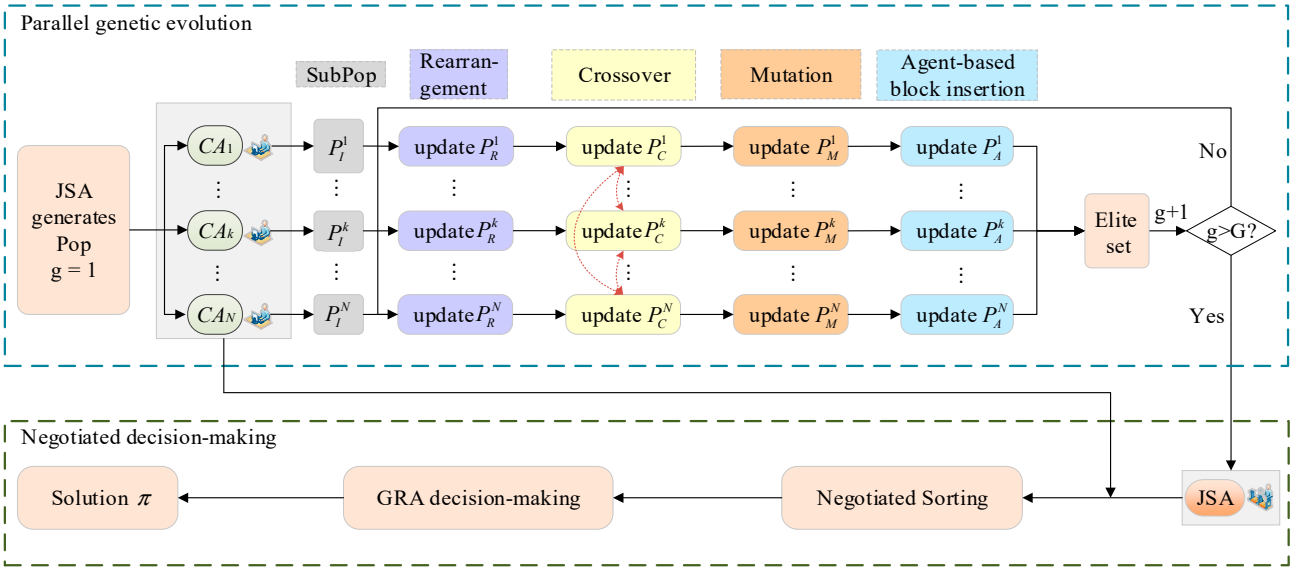

Figure 2. The main structure of the GDOA approach.

In the parallel genetic evolution stage, the consumer agents evolve autonomously and concurrently, driven by their objectives. This independent evolution ensures the protection of each consumer agent's private information. The evolution process involves five essential steps: subpopulation initialization, rearrangement, crossover between agents, mutation within agents, and agent-based block insertion. By following these steps, the consumer agents can create multiple subpopulations that satisfy their objectives, effectively safeguarding the self-interest of the consumer agents. Eventually, these subpopulations are merged into an elite set.

Moving to the negotiated decision-making stage, the consumer agents collaborate with the job shop agent to perform a negotiated sorting on the generated elite set. Subsequently, the job shop agent utilizes a grey relational analysis (GRA) [31] to create a collaborative schedule. This stage ensures that the objectives of each agent are adequately represented, leading to a consensus schedule with high social welfare.

3.2. Parallel Genetic Evolution

Before detailing the five steps of genetic evolution, the encoding method of the MJSSP-PI is presented. A straightforward operation-based encoding method is developed. Take a problem instance with $N = 3$, $m = 3$ as an example, where the $J_{set}^1 = \{1,4\}$, $J_{set}^2 = \{3\}$, and $J_{set}^3 = \{2,5,6\}$. A chromosome is shown in Figure 3. The sequence in which the number of a job appears signifies the order of operations. For example, the three consecutive occurrences of "2" represent the operations $O_{2,1}$, $O_{2,2}$, and $O_{2,3}$, respectively. As each operation is assigned to a fixed processing machine, a chromosome can be decoded into a feasible solution. Meanwhile, the chromosome does not contain any private information of the agents. By employing this encoding approach, the feasibility of the solutions and the security of the private information are maintained throughout the parallel genetic evolution of consumer agents.

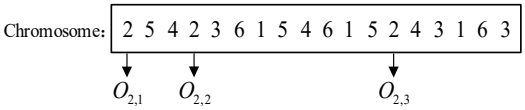

Figure 3. An example of a chromosome.

3.2.1. Subpopulation Initialization

The subpopulation initialization of consumer agents consists of two parts: population initialization and subpopulation selection. In the first step, the JSA randomly generates the

initial population *Pop* with size *P*. In the second step, each consumer agent CA_k chooses its initial subpopulation P_I^k with high fitness values from *Pop*. The fitness value is computed based on the private fitness functions, as shown in Equation (11). The smaller the objective value of an agent, the larger its fitness function value. When the objective value of an agent is 0, the maximum fitness value is 1.

$$f(\pi^l) = \frac{1}{F(\pi^l) + 1} \tag{11}$$

where $f\left(\pi^l\right)$ is the fitness value of agent l and $F\left(\pi^l\right)$ is the objective function value corresponding to the agent l in the schedule π.

The whole process of subpopulation selection is outlined in Algorithm 1.

Algorithm 1: Subpopulation selection

Input: J, $p_{i,j}$
Output: initial subpopulation P_I^k (k = 1, 2, . . ., N)
1: JSA randomly generates the initial population *Pop*
2: **for** CA_k in CA **do**
3: Construct fitness objective $f(\pi)$
4: **for** *pop* in *Pop* **do**
5: Calculate fitness values f(*pop*)
6: **end for**
7: Sort f(*pop*) in ascending order
8: Obtain P_I^k containing P/N chromosomes with top f(*pop*)
9: **end for**
10: Return P_I^k

Through the utilization of the algorithm, the privacy of agents' personal information is effectively safeguarded. Moreover, they produce subpopulations that align with the respective objectives of consumer agents, thereby laying a solid groundwork for further evolution toward the objectives.

3.2.2. Rearrangement

After the subpopulation initialization, the consumer agents execute a rearrangement operation on the existing P_I^k to generate an updated subpopulation P_R^k. In Algorithm 2, the pseudo-code of the rearrangement operation is shown. Specifically, each CA_k submits its P_I^k to JSA, resulting in a fused collection *FC* (lines 1–6). Then, the JSA modifies each chromosome in *FC* based on the job set J_{set}^k (lines 7–16). Following this modification, each CA_k performs subpopulation selection, producing P_R^k (lines 17–25). This process aims to keep the subpopulation evolving toward the individual target while expanding the search space. The flowchart of the rearrangement operation is shown in Figure 4.

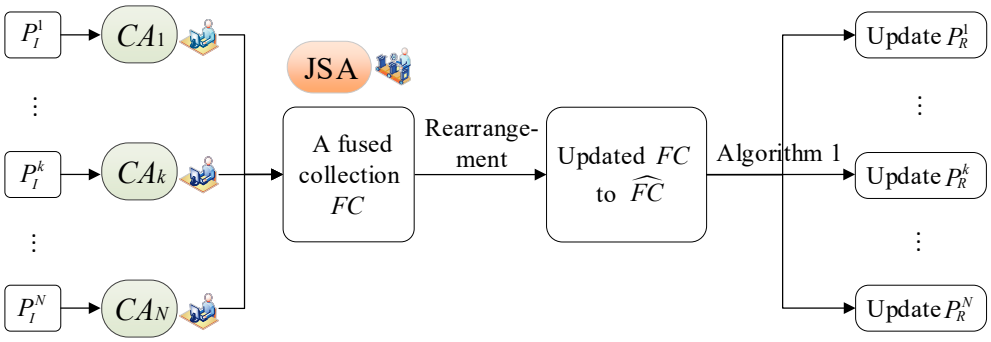

Figure 4. The process of rearrangement operation.

Algorithm 2: Rearrangement operation

Input: J, P_I^k ($k = 1, 2, \ldots, N$)
Output: P_R^k ($k = 1, 2, \ldots, N$)
1: Initialize $FC = \varnothing$
2: **for** CA_k in CA **do**
3: submit P_I^k to JSA
4: Append P_I^k to FC
5: **end for**
6: Return FC
7: // JSA modifies FC
8: $\hat{FC} = \varnothing$
9: **for** fc in FC **do**
10: $\hat{fc} = \varnothing$
11: **for** J_{set}^k in J **do**
12: Swapping genes on fc to generate a new fc^k (as shown in Figure 5)
13: Append fc^k to $\hat{fc}$ ($\left|\hat{fc}\right| = N$)
14: **end for**
15: Append $\hat{fc}$ to $\hat{FC}$ ($\left|\hat{FC}\right| = N \times |FC|$)
16: Return $\hat{FC}$
17: // CA_k updates subpopulation P_R^k
18: $P_R^k = \varnothing$
19: **for** CA_k in CA **do**
20: **for** fc' in $\hat{FC}$ **do**
21: Calculate fitness values $f(fc')$
22: **end for**
23: **Sort** $f(fc')$ in ascending order
24: Generate P_R^k consisting of P/N chromosomes with top $f(fc')$
25: **end for**
26: Return P_R^k

We take the chromosome fc in section subpopulation initialization as an example to illustrate line 12 of Algorithm 2. As shown in Figure 5, JSA modifies fc according to J_{set}^k. For instance, when considering $J_{set}^1 = \{1, 4\}$, the genes with "1" and "4" on fc are swapped with the genes containing the first $\left|J_{set}^1\right|$ jobs numbers on fc. Here, the specific values being swapped are "2" and "5". This process generates a new chromosome fc^1. Continuing this procedure based on J_{set}^2 and J_{set}^3, two chromosomes fc^2 and fc^3 are produced, respectively. Consequently, one chromosome is augmented into N chromosomes, effectively enlarging the search space available for exploration.

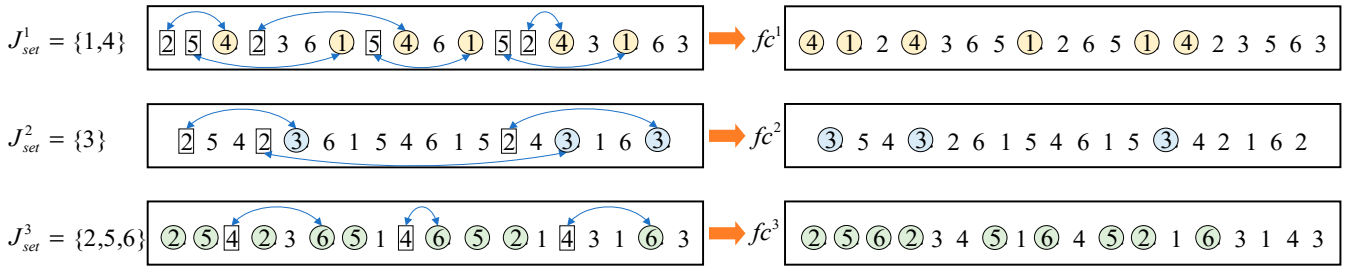

Figure 5. Chromosome rearrangement.

3.2.3. Crossover between Agents

The consumer agents evolve aligning with their objectives. To mitigate the risk of being trapped in local optima, a crossover between agents is proposed. Algorithm 3 illustrates the crossover method. A sequential pairwise exchange is carried out between each agent CA_k and the other $N - 1$ consumer agents in Q_1 rounds. The crossover between

consumer agents leads to an expansion of the size of the crossed chromosomes of CA_k by a factor of $Q_1 \cdot (N-1)$, effectively increasing the overall solution space by $NQ_1 \cdot (N-1)$, thus preventing the solution from converging to a local optimum prematurely. The precedence operation crossover (POX) [32] is utilized for each pair of chromosomes to cross. Finally, CA_k updates the subpopulation P_R^k to P_C^k. The flowchart of the crossover between agents is shown in Figure 6.

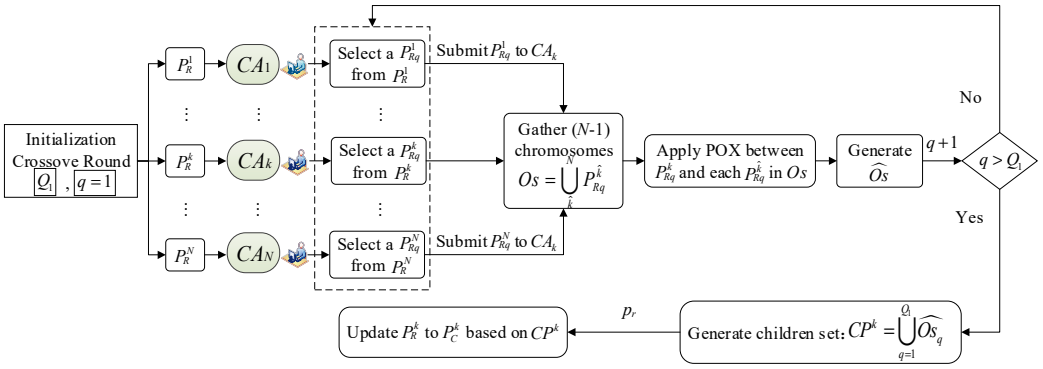

Figure 6. The process of crossover between agents.

Algorithm 3: Crossover between agents.

Input: P_R^k ($k = 1, 2, \ldots, N$), crossover round Q_1, reception probability p_r
Output: P_C^k ($k = 1, 2, \ldots, N$)
for CA_k in CA **do**
  Initialize $P_C^k = \varnothing$, $CP^k = \varnothing$, $q = 1$
  while $q \leq Q_1$ **do**
    Randomly select P_{Rq}^k from P_R^k
    $CA_{\hat{k}}\left(\hat{k} \neq k\right)$ randomly selects $P_{Rq}^{\hat{k}}$ from $P_R^{\hat{k}}$
    $CA_{\hat{k}}\left(\hat{k} \neq k\right)$ submits $P_{Rq}^{\hat{k}}$ to CA_k
    Generate $Os = \bigcup_{\hat{k}}^{N} P_{Rq}^{\hat{k}}$ ($|Os| = N-1$)
    $\hat{Os} = \varnothing$
    for os in Os **do**
      Crossover os and P_{Rq}^k to create os' (POX)
      Append os' to $\hat{Os}$
    end for
    $q = q + 1$
    Append $\hat{Os}$ to CP^k ($\left|CP^k\right| = Q_1 \times (N-1)$)
  end while
  for cp in CP^k **do**
    if $\left|P_C^k\right| \leq \left|P_R^k\right|$ **then**
      if random (0,1) < p_r then
        Append cp to P_C^k
      end if
    end if
end for
Return P_C^k

Through the crossover, the search space is further expanded, resulting in an increasing diversity of chromosomes and a significant reduction in the probability of being confined to local optima. Furthermore, a reception probability is imposed to reduce the increased cost of solution space expansion. It is crucial to emphasize that during the crossover process, the exchange of information between agents is exclusively limited to chromosomes, thereby ensuring the preservation of privacy.

3.2.4. Mutation within Agent

Subsequent to the crossover, the consumer agents execute a mutation operation on their subpopulations P_C^k separately, as described in Algorithm 4. Initially, each consumer agent employs the roulette wheel selection to choose two parent individuals from the subpopulation. Then, the parents undergo the POX crossover and a single swap in sequence, resulting in new offspring. Finally, the offspring are used to update the parent individuals. The whole process repeats Q_2 rounds. This mutation process, performed within the consumer agents, ensures that the consumer agents evolve for self-benefit while simultaneously not diminishing the search space. The flowchart of the mutation within agent is shown in Figure 7.

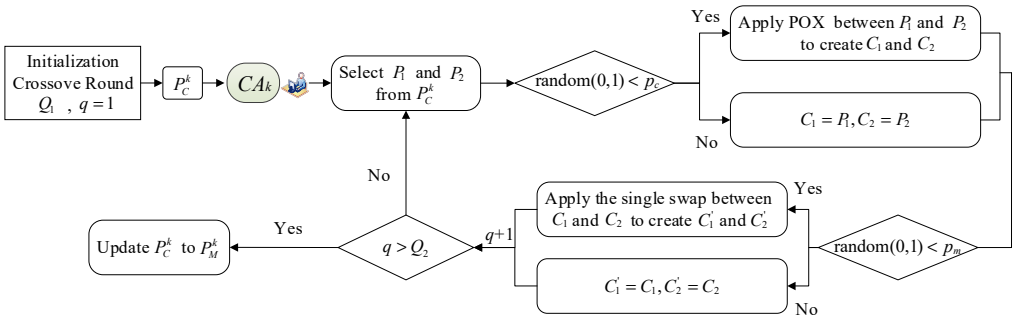

Figure 7. The process of mutation within agent.

Algorithm 4: Mutation within agent

Input: P_C^k (k = 1, 2, …, N), mutation round Q_2, crossover probability p_c, mutation probability p_m
Output: P_M^k (k = 1, 2, …, N)

```
for CA_k in CA do
Initialize P_M^k = ∅, q = 1
while q ≤ Q_2 do
    Select P_1 and P_2 from P_C^k based on the roulette wheel selection
    if random (0, 1) < p_c then
        Crossover P_1 and P_2 to create C_1 and C_2 (POX)
    else
        C_1 = P_1, C_2 = P_2
    if random (0, 1) < p_m then
        Apply the single swap between C_1 and C_2 to create C_1' and C_2'
    else
        C_1' = C_1, C_2' = C_2
    q = q + 1
    Append (C_1', C_2') to P_M^k
  end while
end for
Return P_M^k
```

3.2.5. Agent-Based Block Insertion

Considering the distinctive attributes of multiple agents in the addressed problem, a novel agent-based block insertion is developed to increase the diversity of the subpopulation. Algorithm 5 outlines the core pseudo-code of this method. First, each consumer agent selects a chromosome from P_M^k. Subsequently, an agent-based block of the agent CA_k is defined as a combination of $\left|J_{set}^k\right|$ or more consecutive genes belonging to the job set J_{set}^k on this chromosome. The chromosome is then updated by exchanging the agent-based block with its preceding gene. Finally, CA_k executes these operations on each chromosome in P_M^k, resulting in the update of P_M^k to P_A^k. The insertion of the agent-based block achieves a further expansion of the search space without deteriorating the solutions.

An example of the agent-based block insertion is shown in Figure 8. In case 1, when scanning the gene of an individual P_1, a contiguous gene combination {4,1,1,4} is identified based on the job set $J_{set}^1 = \{1, 4\}$. Since this contiguous combination comprises four genes, exceeding the number of operations $m = 3$, it is designated as the representative block B_1 = {4,1,1,4} corresponding to individual P_1 for CA_1. In case 2, upon scanning the genome of the individual P_1, no contiguous gene combination greater than or equal to m is observed. Consequently, the agent-based block B_1 corresponding to individual P_1 for CA_1 remains empty, denoted as $B_1 = \varnothing$. After obtaining the agent-based block of an individual, it is determined whether the block is empty. If it is empty, the positions of the individual's genes remain unchanged. If it is not empty, the block is inserted before its preceding gene.

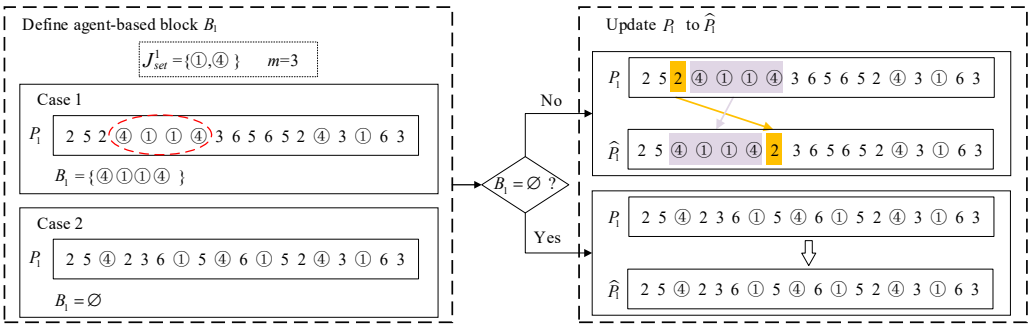

Figure 8. An example of the agent-based block insertion.

Algorithm 5: Agent-based block insertion

Input: J, P_M^k (k = 1, 2, . . ., N)
Output: P_A^k (k = 1, 2, . . ., N)
for CA_k in CA **do**
  Initialize $P_A^k = \varnothing$
  for p in P_M^k **do**
    Define agent-based block B_p
      if A = a combination of $\left|J_{set}^k\right|$ or more consecutive genes with J_{set}^k on p exits **then**
        $B_p = A$
        Modify p to $\hat{p}$ by exchanging B_p with its preceding gene
      else
        $B_p = \varnothing$
        $\hat{p} = p$
      end if
    Append $\hat{p}$ to P_A^k
  end for
end for
Return P_A^k

3.2.6. Parallel Genetic Evolution Algorithm

Upon the completion of the five evolutionary steps, each CA_k submits its P_A^k to JSA. The JSA then selects the top chromosomes that best align with its objective from the collection $\bigcup_{k=1}^{N} P_A^k$, forming a provisional elite set. The purpose of it is to further satisfy the goals of the job shop agent while ensuring that the objectives of the consumer agents have already been met. Additionally, it is imperative to re-emphasize that the chromosomes do not incorporate any private information, ensuring the robust preservation of privacy for all agents.

The aforementioned steps are iterated multiple times to construct the final parallel genetic evolution algorithm. The specific pseudo-code for this algorithm is presented in Algorithm 6.

Algorithm 6: Parallel genetic evolution algorithm

Input: J, maximum round G
Output: Final elite set ES
Initialize $g = 1$
while $g \leq G$ **do**
// CA evolves
  for CA_k in CA **do**
    Apply **Algorithm 1**
    Apply **Algorithm 2**
    Apply **Algorithm 3**
    Apply **Algorithm 4**
    Apply **Algorithm 5**
    Submit P_A^k to JSA forming collection $C_g = \bigcup_{k=1}^{N} P_A^k$
  end for
// JSA generates ES
  Initialize $\hat{ES} = \varnothing$
  Construct fitness objective following Equation (11)
  for c in C_g **do**
    Calculate fitness values $f(c)$
  end for
  Sort $f(c)$ in ascending order
  Obtain ES_g containing $\left|P_A^k\right|$ chromosomes with top $f(c)$
  $g = g + 1$
  Append ES_g to $\hat{ES}$
end while
Deleting duplicate chromosomes in $\hat{ES}$ forming ES
Return ES

3.3. Negotiated Decision-Making

In this section, a negotiated decision-making approach is designed to obtain a final solution that both exhibits consensuses among stakeholders and maximizes social welfare from the elite set. The proposed approach consists of two parts: negotiated sorting and GRA decision-making.

3.3.1. Negotiated Sorting

To ensure the confidentiality of agents' objective information, this study employs a ranking-based method to represent the preferences of agents. Specifically, for each individual in the final elite set ES, all agents sort them according to their respective objectives, thereby generating a negotiated sorting matrix, as shown in Equation (12).

$$\mathrm{SM} = \begin{array}{c} \\ CA_1 \\ \vdots \\ CA_l \\ \vdots \\ CA_N \\ \mathrm{JSA} \end{array} \begin{array}{c} \begin{array}{ccccccc} ES^1 & \cdots & ES^p & \cdots & ES^{|ES|} \end{array} \\ \begin{bmatrix} r_{1,1} & \cdots & r_{1,p} & \cdots & r_{1,|ES|} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ r_{l,1} & \cdots & r_{l,p} & \cdots & r_{l,|ES|} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ r_{N,1} & \cdots & r_{N,p} & \cdots & r_{N,|ES|} \\ r_{N+1,1} & \cdots & r_{N+1,p} & \cdots & r_{N+1,|ES|} \end{bmatrix} \end{array} \tag{12}$$

where ES^p represents the pth solution in ES and $r_{l,p}$ is the ranking of ES^p by agent l, with smaller values indicating the agent l. A smaller $r_{l,p}$ signifies a smaller $F(ES^p)$ and a higher $U(ES^p)$, where $F(ES^p)$ is the objective function value corresponding to the agent l in the solution ES^p and $U(ES^p)$ represents the transformed utility function value (as described in Section 2).

Given the sorting matrix SM, JSA conducts non-dominated sorting [33] to obtain a matrix NSM, as shown in Equation (13), that represents the non-dominated solutions and the corresponding ranking.

$$\text{NSM} = \begin{array}{c} \\ CA_1 \\ \vdots \\ CA_l \\ \vdots \\ CA_N \\ \text{JSA} \end{array} \begin{array}{c} \begin{array}{ccccccc} NS^1 & \cdots & NS^p & \cdots & NS^{|NS|} \end{array} \\ \begin{bmatrix} r_{1,1} & \cdots & r_{1,p} & \cdots & r_{1,|NS|} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ r_{l,1} & \cdots & r_{l,p} & \cdots & r_{l,|NS|} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ r_{N,1} & \cdots & r_{N,p} & \cdots & r_{N,|NS|} \\ r_{N+1,1} & \cdots & r_{N+1,p} & \cdots & r_{N+1,|NS|} \end{bmatrix} \end{array} \tag{13}$$

where NS^p denotes the pth solution in the non-dominated set NS.

3.3.2. GRA Decision Making

In this section, a GRA decision-making method is developed. The proposed method enables the generation of a solution with high social welfare from the non-dominated set, without revealing private information. The process is summarized as follows:

Step 1: Generate correlation coefficients: the correlation coefficient $\eta_{l,p}$ of each $r_{l,p}$ in NSM is calculated by Equation (14). A higher value of $\eta_{l,p}$ indicates that the agent l processes a greater utility value on the solution NS^p.

$$\eta_{l,p} = \frac{\min\left(r_{l,1},\ldots,r_{l,p},\ldots,r_{l,|NS|}\right)+\rho*\max\left(r_{l,1},\ldots,r_{l,p},\ldots,r_{l,|NS|}\right)}{r_{l,p}+\rho*\max\left(r_{l,1},\ldots,r_{l,p},\ldots,r_{l,|NS|}\right)} \quad (l = 1,2,\ldots,N,N+1) \tag{14}$$

where $\eta_{l,p}$ represents the degree of correlation between the ranking value of the pth solution in the lth agent-sorting matrix and the ideal value, and $\min\left(r_{l,1},\ldots,r_{l,p},\ldots,r_{l,|NS|}\right)$ and $\max\left(r_{l,1},\ldots,r_{l,p},\ldots,r_{l,|NS|}\right)$ denote the minimum and maximum ranking values in the lth agent-sorting matrix, respectively. ρ is the discrimination coefficient and $\rho = 0.5$.

Step 2: Calculate the objective weights: the formula for computing the objective weight λ_l for the agent l is given in Equation (15).

$$\lambda_l = \frac{\frac{1}{|NS|}\sum_{p=1}^{|NS|}\eta_{l,p}}{\sum_{l=1}^{N+1}(\frac{1}{|NS|}\sum_{p=1}^{|NS|}\eta_{l,p})} = \frac{1}{N+1},\ (l = 1,2,\ldots,N,N+1) \tag{15}$$

where λ_l represents the weight of the lth agent objective matrix calculated based on the correlation coefficients. It can be observed that the weight of each agent is $1/(N+1)$, which is consistent with the importance level of each agent.

Step 3: Calculate the grey correlation: the grey correlation $\delta(NS^p)$ of the solution NS^p is calculated following Equation (16).

$$\delta(NS^p) = \sum_{l=1}^{N+1}\eta_{l,p}\cdot\lambda_l, (p = 1,2,\ldots,|NS|) \tag{16}$$

A positive correlation between $\delta(NS^p)$ and SW is evident, which proves that the greater the $\delta(NS^p)$, the higher the social welfare of the solution NS^p.

Step 4: Determine the final solution: select the solution NS^{p*} with $max(\delta(NS^p))$ as the final solution.

3.3.3. Negotiated Decision-Making Algorithm

Based on the description above, the process of the negotiated decision-making algorithm is shown in Algorithm 7.

Algorithm 7: Negotiated decision-making algorithm

Input: Final elite set ES
Output: Final solution π^*
for CA_k in CA **do**
  for ES^p in ES **do**
    Sort ES according to (12)
    Submit $r_{l,p}$ ($l = k$) to JSA
  end for
end for
JSA **do**
  Generate a matrix SM based on $r_{l,p}$
  Perform non-dominated sorting on SM to produce NSM (13) and NS
  Calculate $\eta_{l,p}$ with (14)
  Calculate λ_l with (15)
  for π in NS **do**
    Calculate $\delta(\pi)$ with (16)
  end for
  Select π^* with $max(\delta(\pi))$
  Return π^*

4. Computational Experiments

In this section, computational experiments are presented to evaluate the performance of the proposed GDOA in addressing the MJSSP-PI. Two types of experiments are carried out: the convergence and evenness of the non-dominated solutions generated by GDOA are analyzed, and the social welfare of the schedule generated by GDOA is verified. During the experiments, GDOA is compared against a well-known centralized algorithm NSGA-III [34] and two decentralized algorithms, namely, a generic negotiation mechanism (GNMS) [20] and a genetic decision-based two-stage negotiation algorithm (GTNA) [13]. All algorithms are implemented in Python 3.9, and the experiments are carried out on a PC with an Intel Core i7-7700 3.60 GHz CPU and 16 GB of RAM.

4.1. Experiment Setup

4.1.1. Problem Instances

A total of 734 problem instances of MJSSP-PI are constructed based on the datasets of the classic job shop scheduling problem in [35]. These instances involve a job shop agent and a varying number of consumer agents, ranging from 2 to 16. Therefore, the total number of agents ranges from 3 to 17. The number of jobs ranges from 10 to 50, and the number of machines includes $m \in \{5, 10, 15, 20\}$. The due date of the job J_i is given by $d_i = U[0.5, 9] \times \sum_{j=1}^{m} p_{ij}$, and the weight is randomly generated from $\omega_i \sim U[1, 5]$. The characteristics of the machine $\{SE_j, PE_j, IE_j\}$ are generated following $SE_j \sim U[100, 200]$, $PE_j \sim U[5, 8]$, and $IE_j \sim U[1, 3] \cdot PE_j$, where SE_j and PE_j are integers, and IE_j is a decimal number. The combinations of the parameters are shown in Table 1.

Table 1. Combinations of parameters for problem instances.

n	m	N Number of CA (JSA = 1)	Number of Instances
10	5, 10	2–3	60
15	5, 10, 15	2–5	32
20	5, 10, 15, 20	2–6	180
30	10, 15, 20	2–10	126
40	15, 20	2–13	96
50	10, 15,2 0	2–16	240

4.1.2. Performance Measurement

The quality of the non-dominated solution set generated by the proposed GDOA has a substantial impact on the final solution since the latter is selected from the former. Therefore, the evaluation must consider two dimensions: evaluate the generated non-dominated solution set and measure the final schedule.

First, generational distance (GD) and spacing are used to measure the closeness and evenness of the non-dominated solutions, respectively [36]. They are calculated as follows:

$$\mathrm{GD}(x) = \frac{\sqrt{\sum_{i=1}^{|NS_x|} \left(\min_{p_j \in P_{par}} \sum_{l=1}^{N+1} \left(F(p_j^l) - F(p_i^l) \right)^2 \right)}}{|NS_x|} \tag{17}$$

where NS_x denotes the non-dominated set produced by the algorithm x, p_i represents the ith solution in NS_x, p_j signifies the jth solution in the Pareto set P_{par}, and $F\left(p_i^l\right)$ and $F\left(p_j^l\right)$ correspond to the objective value of solutions p_i and p_j at the lth objective, respectively. A lower value of GD indicates superior performance.

$$\mathrm{Spacing}(x) = \sqrt{\frac{1}{|NS_x|} \sum_{i=1}^{|NS_x|} \left(d_i - \overline{d} \right)^2}, \ \overline{d} = \frac{1}{|NS_x|} \sum_{i=1}^{|NS_x|} d_i \tag{18}$$

where d_i is the Euclidean distance between the solution p_i and its nearest consecutive solution in NS_x. A smaller value of spacing indicates a more uniform distribution of the non-dominated solutions.

Since the true Pareto solution set is unknown, the Pareto set in this paper is obtained by combining the non-dominated solution sets generated by GDOA, NSGA-III, GNMS, and GTNA algorithms.

Second, the ratio of the social welfare of the schedule and the best solution in the Pareto set is computed to measure the final schedule. This metric is presented below:

$$\mathrm{RSW}(x) = \frac{SW(\pi_x^*)}{maxSW\left(P_{par}\right)} \tag{19}$$

where π_x^* is the schedule produced by the algorithm x, $SW(\pi_x^*)$ means the social welfare of π_x^*, and $maxSW\left(P_{par}\right)$ indicates the maximum social welfare in the Pareto set P_{par}.

4.1.3. Parameter Setting

The main parameters of each algorithm are shown in Table 2, in which G, P, P/N, p_c, and p_m represent the iteration round, initial population size, subpopulation size, crossover probability, and mutation probability, respectively. Note that parameters p_r, Q_1, and Q_2 are specific to GDOA, indicating the reception probability, crossover round, and mutation round, respectively. Each benchmark instance is independently run five times and the average value is taken as the final result.

Table 2. Parameter settings.

Parameter	GDOA	NSGA-III	GNMS	GTNA
G	50	200	50	50
P	100	100	100	$100 \times (N + 1)$
P/N	100	100	100	$100 \times (N + 1)$
p_c	0.5	0.5	0.5	0.5
p_m	0.1	0.1	0.1	0.1
p_r	0.6	--	--	--
Q_1	50	--	--	--
Q_2	$100 \times N$	--	--	--

4.2. Analysis of Results

4.2.1. Experiment 1: Non-Dominated Solution Set

The first experiment's purpose is to verify the convergence and evenness of the non-dominated solutions. Table 3 shows the comparative results between GDOA and the centralized approach, NSGA-III. In terms of metric GD, it is evident that GDOA consistently achieves higher GD values compared to NSGA-III for problem instances with a smaller number of agents ($l \leq 7$). This can be attributed to the inherent limitations of decentralized decision-making under incomplete information, which gives GDOA a disadvantage over centralized algorithms. However, for instances with more agents ($l > 8$), GDOA demonstrates better convergence by exhibiting lower GD values compared to NSGA-III. This highlights the superior convergence performance of GDOA, particularly in high-dimensional objective spaces where NSGA-III faces challenges. Additionally, it is noteworthy that the GD values of GDOA show a certain decreasing trend as the number of agents increases, indicating the continuous improvement in its convergence.

Table 3. Comparison of GDOA with NSGA-III.

l	GD		Spacing	
	GDOA	NSGA-III	GDOA	NSGA-III
3	0.286	0.181	0.069	0.028
4	0.329	0.187	0.069	0.028
5	0.235	0.139	0.035	0.026
6	0.181	0.091	0.030	0.032
7	0.153	0.075	0.026	0.031
8	0.078	0.097	0.022	0.033
9	0.120	0.153	0.024	0.033
10	0.076	0.152	0.027	0.038
11	0.089	0.158	0.048	0.029
12	0.043	0.123	0.047	0.035
13	0.023	0.152	0.034	0.039
14	0.052	0.133	0.028	0.048
15	0.022	0.156	0.031	0.058
16	0.026	0.142	0.036	0.059
17	0.025	0.157	0.032	0.056

By analyzing the metric of spacing, it can be seen that the spacing of GDOA is smaller than that of NSGA-III in 70.7% of problem instances. In 29.3% of specific cases involving l = 3, 4, 5, 11, and 12, NSGA-III outperforms GDOA. This can be attributed to its strategy of selecting optimal solutions based on reference points, resulting in a more evenly distributed set of dominated solutions. However, when considering the entirety of cases, GDOA consistently maintains a smaller spacing. These findings suggest that the non-dominated solutions generated by GDOA display a more uniform distribution compared to NSGA-III. The underlying cause for this observation lies in that each agent evolves toward its objective within the proposed GDOA, and the continuous expansion of the solution space leads to an increased number of non-dominated solutions.

Figure 9 presents the comparison of GD and spacing between GDOA and two state-of-the-art decentralized approaches, GNMS and GTNA. As can be seen from the different lines, showing the results compared to GNMS and GTNA, GDOA has significantly smaller values for both the GD and spacing metrics. This firmly establishes that the proposed GDOA outperforms existing decentralized methods in terms of convergence and distributivity for addressing the MJSSIP-PI. Furthermore, the decreasing orange line validates the effectiveness of the GDOA in handling scenarios with a larger number of agents.

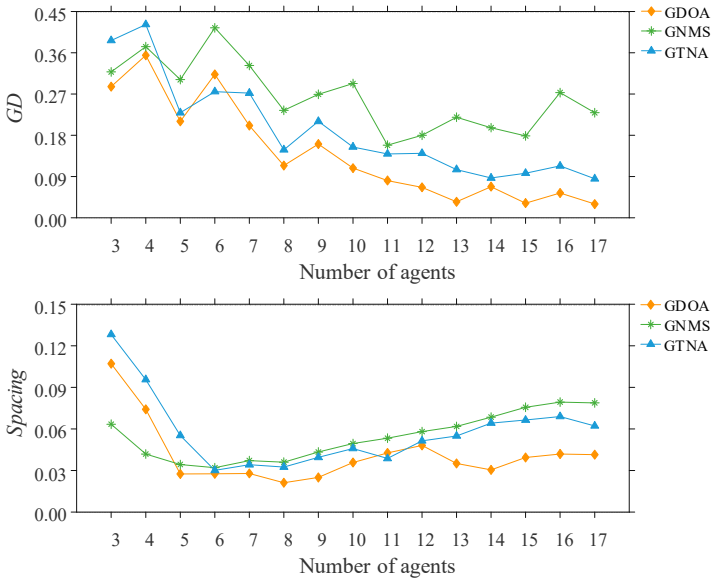

Figure 9. Comparison of GDOA with GNMS and GTNA in GD and spacing.

In summary, Experiment 1 supports the efficacy of the proposed decentralized approach GDOA in generating non-dominated solutions with favorable convergence and evenness. Consequently, selecting the final solution from these non-dominated solutions is reasonable.

4.2.2. Experiment 2: Final Solution

In the second experiment, the quality of the generated final solution is assessed using the RSW criterion. Due to the inability of NSGA-III to determine the final solutions, we only compare GDOA with the two decentralized approaches, GNMS and GTNA.

Figure 10 illustrates the social welfare variation in the schedule obtained through different approaches. It is evident that the RSW of GDOA is considerably higher than those of GNMS and GTNA, implying that GDOA can generate a higher-quality solution compared to the other two decentralized algorithms. Additionally, the RSW of GDOA shows an increasing trend with the growing consumer agents. The reason is that GDOA can generate a larger and better non-dominated solution set. As a result, the selected final solution demonstrates superior performance.

As the instances involve various combinations of jobs and machines ($n \times m$), an analysis of the performance of the proposed GDOA in different cases is needed. The maximum $n \times m$ in each combination of rows is chosen according to Table 1. Figure 11 illustrates the variation in RSW achieved by GDOA across instances of different $n \times m$. In general, the RSW of the obtained solution increases with the growing number of jobs and machines and tends to stabilize. With a constant $m = 20$, the RSW of GDOA rises slightly at an increasing number of jobs n, and the mean RSW remains consistently above 0.90. This indicates that the advantage of the GDOA algorithm in solving large-scale instances becomes more pronounced. Moreover, the RSW demonstrates more pronounced fluctuations in scenarios with smaller $n \times m$, whereas this fluctuation gradually diminishes as $n \times m$ increases. For instance, when $n \times m = 10 \times 10$, the RSW ranges from 0.55 to 0.92.

Similarly, for $n \times m = 20 \times 20$, the average RSW ranges from 0.80 to 1.00. Furthermore, for $n \times m = 50 \times 20$, the average RSW ranges between 0.90 and 1.00. This further highlights the stability of the proposed algorithm in large-scale scenarios.

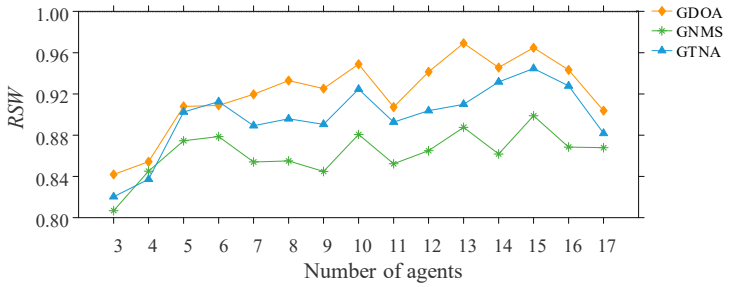

Figure 10. Comparison of GDOA with GNMS and GTNA in RSW.

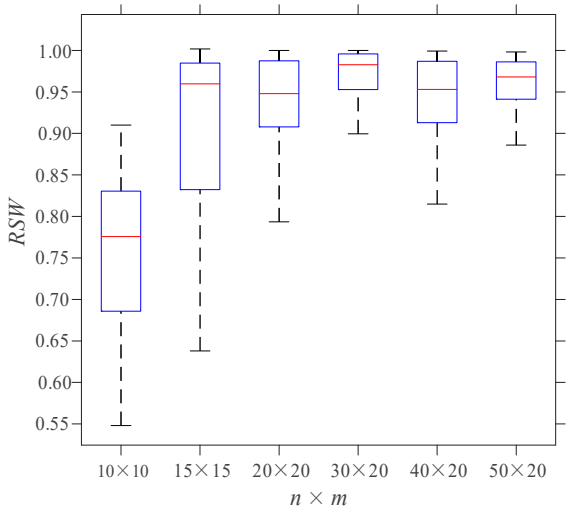

Figure 11. Changes in RSW of instances with different jobs and machines.

5. Conclusions

This paper presents a two-stage GA-based decentralized optimization algorithm to address MJSSP-PI, including the parallel genetic evolution algorithm and the negotiated decision-making algorithm. The first stage enables consumer agents to evolve independently and in parallel to achieve individual objectives. The method follows five steps: sub-population, rearrangement, crossover between agents, mutation within agent, and agent-based block insertion, where each agent generates solutions satisfying their objectives. The development of crossover between consumer agents and agent-based block insertion strongly expands the search space. Within the second stage, the final solution with high social welfare is produced by non-dominated sorting and GRA-decision making. Computational experiments on 734 problem instances are conducted. The results indicate that the proposed GDOA can effectively generate high-quality solutions, and it exhibits remarkable stability when dealing with large-scale problems that involve a large number of agents.

In addition to the aforementioned advantages, this study also identifies several limitations. Notably, when implementing the proposed algorithms in large-scale experiments, a notable increase in computation time is observed as the number of agents expands. This underscores the need for the further optimization of the algorithms to mitigate computational costs effectively. Moreover, the selection of parameters significantly influences experimental outcomes, underscoring the importance of meticulous parameter tuning in future research endeavors. Furthermore, owing to the involvement of intricate processes and constraints pertaining to temporal and financial resources, the algorithm has yet to be validated in real-world production scenarios.

In considering future research directions, several promising avenues warrant exploration. Our intentions encompass the development of efficient parallelization techniques or distributed computing frameworks aimed at mitigating algorithmic computation time. Furthermore, the integration of advanced machine learning methodologies for automated parameter selection and optimization holds potential to enhance algorithmic stability and reliability across diverse problem instances. Additionally, the incorporation of real-world production application scenarios within experimental designs stands as a pivotal endeavor to further corroborate the practical utility and applicability of the proposed algorithm. Moreover, future research could also explore the broader implications of decentralized optimization algorithms in various domains, including supply chain management, transportation, and energy distribution. By leveraging interdisciplinary collaborations and integrating insights from fields such as operations research, computer science, and economics, we can further advance the understanding and application of decentralized optimization techniques in complex real-world systems.

Author Contributions: Conceptualization, X.Z. and W.R.; methodology, W.R. and Y.L.; software, W.R.; writing—original draft preparation, X.Z.; writing—review and editing, W.R. and Y.L.; supervision, S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research work is supported by the National Natural Science Foundation of China, grant number 51975482.

Data Availability Statement: Data are contained within the article.

Acknowledgments: The authors would like to thank the anonymous reviewers and the editor for their positive comments.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

Indexes	
i	Index of jobs
j	Index of machines
i, j	Index of operations
h	Index of processing position of machine
Parameters	
n	Total number of jobs
N	Number of CA
m	Number of machines
$p_{i,j}$	Processing time of operation $O_{i,j}$
d_i	Due date of operation $O_{i,j}$
ω_i	Weight of job J_i
SE_j	Starting energy consumption of M_j
PE_j	Unit processing energy consumption of M_j
IE_j	Unit idle energy consumption of M_j
Variables	
C_i	Completion time of J_i
S_j^h	Job processed on the hth position of M_j
$O_{S_j^h,j}$	Operation of job S_j^h that processed on the hth position of M_j
$st_{S_j^h,j}$	Starting time of the operation $O_{S_j^h,j}$
$ct_{S_j^h,j}$	Completion time of the operation $O_{S_j^h,j}$
$st_{i,j}$	Starting time of the operation $O_{i,j}$
$ct_{i,j}$	Completion time of the operation $O_{i,j}$
IT_j	Total idle time of M_j

References

1. Çaliş, B.; Bulkan, S. A research survey: Review of AI solution strategies of job shop scheduling problem. *J. Intell. Manuf.* **2015**, *26*, 961–973. [CrossRef]
2. Perez-Gonzalez, P.; Framinan, J.M. A common framework and taxonomy for multicriteria scheduling problems with interfering and competing jobs: Multi-agent scheduling problems. *Eur. J. Oper. Res.* **2014**, *235*, 1–16. [CrossRef]
3. Klein, M.; Faratin, P.; Sayama, H.; Bar-Yam, Y. Protocols for negotiating complex contracts. *IEEE Intell. Syst.* **2003**, *18*, 32–38. [CrossRef]
4. Yazdani, M.; Khalili, S.M.; Jolai, F. A parallel machine scheduling problem with two-agent and tool change activities: An efficient hybrid metaheuristic algorithm. *Int. J. Comput. Integr. Manuf.* **2016**, *29*, 1075–1088. [CrossRef]
5. Goli, A.; Ala, A.; Hajiaghaei-Keshteli, M. Efficient multi-objective meta-heuristic algorithms for energy-aware non-permutation flow-shop scheduling problem. *Expert Syst. Appl.* **2023**, *213*, 119077. [CrossRef]
6. Wang, G.; Li, X.; Gao, L.; Li, P. Energy-efficient distributed heterogeneous welding flow shop scheduling problem using a modified MOEA/D. *Swarm Evol. Comput.* **2021**, *62*, 100858. [CrossRef]
7. Yan, R.; Pei, Z. Information asymmetry, pricing strategy and firm's performance in the retailer- multi-channel manufacturer supply chain. *J. Bus. Res.* **2011**, *64*, 377–384. [CrossRef]
8. Fink, A.; Homberger, J. Decentralized Multi-Project Scheduling. In *Handbook on Project Management and Scheduling*; Schwindt, C., Zimmermann, J., Eds.; Springer: Cham, Switzerland, 2015; Volume 2, pp. 685–706. [CrossRef]
9. Owliya, M.; Saadat, M.; Jules, G.G.; Goharian, M.; Anane, R. Agent-Based Interaction Protocols and Topologies for Manufacturing Task Allocation. *IEEE Trans. Syst. Man Cybern.-Syst.* **2013**, *43*, 38–52. [CrossRef]
10. Guizzi, G.; Revetria, R.; Vanacore, G.; Vespoli, S. On the open job-shop scheduling problem: A decentralized multi-agent approach for the manufacturing system performance optimization. *Procedia CIRP* **2019**, *79*, 192–197. [CrossRef]
11. Zeng, C.; Liu, Z.; Tang, J.; Fan, Z.P.; Yan, C. Auction-based approach to the job-shop problem with parallel batch processing and a machine availability constraint. *Eng. Optimiz.* **2023**, *55*, 71–88. [CrossRef]
12. Liu, Y.; Sun, S.; Wang, X.V.; Wang, L. An iterative combinatorial auction mechanism for multi-agent parallel machine scheduling. *Int. J. Prod. Res.* **2022**, *60*, 361–380. [CrossRef]
13. Sun, S.; Zhou, X.; Chang, S. Negotiation Scheduling Algorithm for Multi-agent Job Shop with Private Information. *Chin. J. Mech. Eng.* **2022**, *58*, 210–217. [CrossRef]
14. Homberger, J. A (μ, λ)-coordination mechanism for agent-based multi-project scheduling. *OR Spectrum* **2012**, *34*, 107–132. [CrossRef]
15. Fink, A.; Homberger, J. An ant-based coordination mechanism for resource-constrained project scheduling with multiple agents and cash flow objectives. *Flex. Serv. Manuf. J.* **2013**, *25*, 94–121. [CrossRef]
16. Lang, F.; Fink, A. Learning from the Metaheuristics: Protocols for Automated Negotiations. *Group Decis. Negot.* **2015**, *24*, 299–332. [CrossRef]
17. Gao, J.; Wong, T.; Wang, C. Coordinating patient preferences through automated negotiation: A multiagent systems model for diagnostic services scheduling. *Adv. Eng. Inform.* **2019**, *42*, 100934. [CrossRef]
18. Fink, A.; Gerhards, P. Negotiation mechanisms for the multi-agent multi-mode resource investment problem. *Eur. J. Oper. Res.* **2021**, *295*, 261–274. [CrossRef]
19. Gao, J.; Wong, T.; Selim, B.; Wang, C. VOMA: A Privacy-Preserving Matching Mechanism Design for Community Ride-Sharing. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 23963–23975. [CrossRef]
20. Homberger, J.; Fink, A. Generic negotiation mechanisms with side payments—Design, analysis and application for decentralized resource-constrained multi-project scheduling problems. *Eur. J. Oper. Res.* **2017**, *261*, 1001–1012. [CrossRef]
21. He, N.; Zhang, D.Z.; Yuce, B. Integrated multi-project planning and scheduling—A multiagent approach. *Eur. J. Oper. Res.* **2022**, *302*, 688–699. [CrossRef]
22. Xu, W.; Hu, Y.; Luo, W.; Wang, L.; Wu, R. A multi-objective scheduling method for distributed and flexible job shop based on hybrid genetic algorithm and tabu search considering operation outsourcing and carbon emission. *Comput. Ind. Eng.* **2021**, *157*, 107318. [CrossRef]
23. Epitropakis, M.G.; Plagianakos, V.P.; Vrahatis, M.N. Balancing the exploration and exploitation capabilities of the Differential Evolution Algorithm. In Proceedings of the IEEE Congress on Evolutionary Computation, Hong Kong, China, 1–6 June 2008; pp. 2686–2693. [CrossRef]
24. Lin, L.; Gen, M. Auto-tuning strategy for evolutionary algorithms: Balancing between exploration and exploitation. *Soft Comput.* **2009**, *13*, 157–168. [CrossRef]
25. Li, X.; Gao, L.; Pan, Q.; Wan, L.; Chao, K.M. 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.* **2018**, *49*, 1933–1945. [CrossRef]
26. 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]
27. Thevenin, S.; Zufferey, N. Learning variable neighborhood search for a scheduling problem with time windows and rejections. *Discret. Appl. Math.* **2019**, *261*, 344–353. [CrossRef]
28. Qi, X.; Zhang, D.; Lu, H.; Li, R. A GA-Based Scheduling Method for Civil Aircraft Distributed Production with Material Inventory Replenishment Consideration. *Mathematics* **2023**, *11*, 3135. [CrossRef]

29. Wen, X.; Song, Q.; Qian, Y.; Qiao, D.; Wang, H.; Zhang, Y.; Li, H. Effective Improved NSGA-II Algorithm for Multi-Objective Integrated Process Planning and Scheduling. *Mathematics* **2023**, *11*, 3523. [CrossRef]
30. Nguyen, N.T.; Nguyen, T.T.; Roos, M.; Rothe, J. Computational complexity and approximability of social welfare optimization in multiagent resource allocation. *Auton. Agents Multi-Agent Syst.* **2014**, *28*, 256–289. [CrossRef]
31. Javed, S.A.; Gunasekaran, A.; Mahmoudi, A. DGRA: Multi-sourcing and supplier classification through Dynamic Grey Relational Analysis method. *Comput. Ind. Eng.* **2022**, *173*, 108674. [CrossRef]
32. Zhang, C.; Rao, Y.; Li, P. An effective hybrid genetic algorithm for the job shop scheduling problem. *Int. J. Adv. Manuf. Technol.* **2008**, *39*, 965–974. [CrossRef]
33. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
34. Liu, Y.; You, K.; Jiang, Y.; Wu, Z.; Liu, Z.; Peng, G.; Zhou, C. Multi-objective optimal scheduling of automated construction equipment using non-dominated sorting genetic algorithm (NSGA-III). *Autom. Constr.* **2022**, *143*, 104587. [CrossRef]
35. Strassl, S.; Musliu, N. Instance space analysis and algorithm selection for the job shop scheduling problem. *Comput. Oper. Res.* **2022**, *141*, 105661. [CrossRef]
36. Yen, G.G.; He, Z. Performance Metric Ensemble for Multiobjective Evolutionary Algorithms. *IEEE Trans. Evol. Comput.* **2014**, *18*, 131–144. [CrossRef]