mathematics

MDPI

Article

A Hybrid Particle Swarm Optimization Algorithm Enhanced with Nonlinear Inertial Weight and Gaussian Mutation for Job Shop Scheduling Problems

Hongli Yu [1], Yuelin Gao [2,*], Le Wang [1] and Jiangtao Meng [1]

[1] School of Computer Science and Engineering, North Minzu University, Yinchuan 750021, China; yuhongli217@yeah.net (H.Y.); sdnywll@163.com (L.W.); mjt_11@163.com (J.M.)

[2] Ningxia Province Key Laboratory of Intelligent Information and Data Processing, North Minzu University, Yinchuan 750021, China

* Correspondence: 1993001@nun.edu.cn; Tel.: +86-139-9510-0900

Received: 26 July 2020; Accepted: 11 August 2020; Published: 13 August 2020

Abstract: Job shop scheduling problem (JSSP) has high theoretical and practical significance in academia and manufacturing respectively. Therefore, scholars in many different fields have been attracted to study this problem, and many meta-heuristic algorithms have been proposed to solve this problem. As a meta-heuristic algorithm, particle swarm optimization (PSO) has been used to optimize many practical problems in industrial manufacturing. This paper proposes a hybrid PSO enhanced with nonlinear inertia weight and and Gaussian mutation (NGPSO) to solve JSSP. Nonlinear inertia weight improves local search capabilities of PSO, while Gaussian mutation strategy improves the global search ability of NGPSO, which is beneficial to the population to maintain diversity and reduce probability of the algorithm falling into the local optimal solution. The proposed NGPSO algorithm is implemented to solve 62 benchmark instances of JSSP, and the experimental results are compared with other algorithms. The results obtained by analyzing the experimental data show that the algorithm is better than other comparison algorithms in solving JSSP.

Keywords: job shop scheduling problem; particle swarm optimization; Gaussian mutation; nonlinear inertial weight

MSC: 90C59; 54A05

1. Introduction

Job Shop Scheduling Problem (JSSP) is a simplified model of many practical scheduling problems, including aircraft carrier scheduling, airport dispatching, high-speed rail scheduling, automobile pipeline, etc. Therefor, JSSP has high theoretical significance and practical significance. JSSP studies how to process n jobs on m machines to optimize the processing performance. In order to optimize the processing performance, it is necessary to determine the processing start moment or completion moment or processing sequence of each job on each machine under the premise of meeting the process constraints. The final processing time for all jobs is called makespan [1].

Similar to most discrete optimization problems, JSSP is also a type of strong NP-hard problems, which have non-polynomial time complexity and complex solution space structure, and usually need to adopt appropriate strategies to reduce the search range and problem complexity. Traditional operations research methods solve problems by establishing mathematical models. The typical representatives of such methods include branch and bound algorithm (BAB) [2], Lagrangian relaxation method (LR) [3],

dynamic programming algorithm (DR) [4], etc. BAB is a kind of algorithm commonly used to solve integer programming problems, which belongs to enumeration method. In the process of solving problems, BAB uses a mechanism that gives the upper and lower search limits of the solution to exclude solutions that are not between the upper and lower limits, thereby reducing the dimensionality of the subset of solutions to be selected and finding the target value more quickly. Experimental results show that the BAB is effective in solving small-scale scheduling problems, but the calculation time is often too long in solving large-scale problems. In the LR, when solving problems, some constraints of problems are relaxed appropriately to simplify the original problem, and then the optimal solution or approximate solution can be obtained by updating the Lagrangian operator. The DP algorithm decomposes the problem to be solved into several sub-problems, it first solves the sub-problems, and then obtains the solution of the original problem from the solution of these sub-problems. Generally, the above methods are effective only when solving small-scale problems, and have strong dependence on the problem to be solved, and are not suitable for solving actual production problems. However, heuristic algorithm can get the approximate optimum or optimum in acceptable time, which mainly includes two categories: constructive heuristic algorithms and meta-heuristic algorithms. Although the constructive heuristic algorithms can find the solution of the problem in the solution space, it depends on local information of the scheduling problem, and the solution obtained is generally the local optimal solution; meta-heuristic algorithms is a type of optimization algorithm that is inspired by natural principles and formulaically describes natural principles through computer language. It can get the optimum or approximate optimum solution in an acceptable time, and it has become a kind of practical and feasible algorithm for solving various scheduling problems. Meta-heuristic algorithms mainly include: artificial bee colony algorithm (ABC) [5,6] clonal selection algorithms (CSA) [7,8], simulated annealing algorithm (SA) [9], genetic algorithm (GA) [10–13], bat algorithm (BA) [14], whale optimization algorithm (WOA) [15], biogeography based optimization algorithm (BBO) [16,17], particle swarm optimization algorithm (PSO) [18–20], differential evolution algorithm (DE) [21], ant colony optimization algorithm (ACO) [22,23], etc.

To obtain good performance, a meta-heuristic algorithm needs to find some strategies to balance the local search capability (exploitation) and global search capability (exploration) of the algorithm [24]. The advantage of meta-heuristic algorithms is that it can combine the both in an optimal way in theory. However, when solving practical problems, the population size cannot reach the infinite in theory, and the fitness function cannot fully reflect the real adaptability of individuals, and the behavior of individuals can not perfectly reproduce the intelligence of individuals in nature, the above factors limit the performance of algorithms. Therefore, it has become a hot issue in the field of meta-heuristic algorithm research to more effectively balance exploration and exploitation by mixing different strategies in the algorithm. In recent years, many hybrid meta-heuristic algorithms have been used to solve JSSP. Wang and Duan [25] introduced chaos theory into the biogeographical optimization algorithm (BBO) to improve stability and accelerate convergence rate of the algorithm. Lu and Jiang [26] divided the bat optimization algorithm (BA) into two subpopulation, and added parallel search mechanism, communication strategy, and improved population discrete update method in the algorithm to solve the low-carbon JSSP. Rohainejad et al. [27] combined firefly algorithm (FA) and tabu search alogorithm (TS), which effectively reduced the overtime cost in JSSP, and made the algorithm more robust. Babukartik et al. [28] added the search strategy of the cuckoo algorithm (COA) into the ACO to improve the exploitation the efficiency of the algorithm for solving JSSP. Yu et al. [29] added a disturbance mechanism on the basis of DE and chaos theory to reduce the possibility of premature convergence of the teaching-learning-based optimization algorithm (TLB). Lu et al. [30] improved the social hierarchy and genetic factors of the multi-target gray wolf optimization algorithm (GWO) to enhance exploration and exploitation, thereby improving the efficiency in solving the dynamic JSSP. Min Liu et al. [31] added the Levy flight and DE into the WOA for solving JSSP. The Levy flight strategy enhanced the global search capability and convergence rate. The DE to enhance the local optimization capability of the algorithm and to keep the diversity of the scheduling scheme to

prevent the algorithm from premature convergence. Nirmala Sharma et al. [32] improved the iterative formula for onlooker bees in ABC to solve JSSP. Xueni Qiu et al. [33] combined artificial immune system (AIS) theory and PSO to solve the JSSP. The algorithm adopts clone selection and immune network theories to simulate the basic processes of immunity, cloning, hypermutation, antibody selection, etc., while using the PSO to accelerate the search process. Atiya Masood et al. [34] proposed a hybrid genetic algorithm to solve the multi-objective JSSP. Adel Dabah et al. [35] added a parallel strategy to the tabu search algorithm (TS) to improve the capability of the TS to solve the problem of blocking JSSP. Hong Lu and Yang Jing [36] proposed a new PSO based on the clonal selection theory to avoid premature convergence. R.F. Abdel-Kade [37] introduced two neighborhood-based operators in the PSO to improve the diversity of the population for solving JSSP. The diversity operator improves population diversity by repositioning neighboring particles, and the local search operator performs local optimization on neighboring regions. T.-K.Dao et al. [38] proposed a parallel BA with mixed communication strategy for solving JSSP. Each subpopulation of the algorithm is associated with each other through communication strategy and shares the computing load on the processor. The algorithm has excellent accuracy and convergence rate. Tianhua Jiang et al. [39] proposed an hybrid WOA for solving the energy-efficient JSSP.

More and more hybrid meta-heuristic algorithms and are used to solveJSSP [31]. However, the above algorithms mainly overcome the shortcomings of algorithms by increasing the diversity of the population, preventing algorithms from falling into the local optimal solution, and increasing the stability of algorithms. Few scholars analyze the performance of the algorithm from the perspective of balancing the global search capability and local search capability of the algorithm. In fact, the performance of meta-heuristic algorithm largely depends on whether the global search capability and local search capability of the algorithm can be effectively balanced. Furthermore, the two strategies we integrated, including Gaussian mutation and nonlinear inertia weighting strategy, are common analysis strategies when viewed separately. However, when Gaussian mutation and nonlinear programming are integrated together, it shows the superiority of the integration strategy in expanding the scale of individual variation and ensuring the stability of individual scale. From the perspective of the stability of individual evolution, the advantages of integration strategies are more obvious in the later stages of individual evolution. The spatial dissipation of individual populations gradually slows down, the internal energy entropy gradually decreases, the trend of population convergence is more stable, and the evolution of individuals gradually stabilizes. Therefore, in this research, it is important to balance the global search and local search capabilities of the algorithm through different strategies to overcome the shortcomings of PSO. Similar to other meta-heuristic population intelligence algorithms (PIA), PSO is an innovative global optimization algorithm first proposed by Dr. Kennedy and Dr. Eberhart in 1995 [40]. Compared with other PIA, PSO has a simple concept, few parameters and fast convergence rate. Therefore, the particle swarm algorithm has become a very successful algorithm and has been used to optimize various practical problems. The unique information exchange and learning methods of PSO makes the movement of the whole population in the same direction, which endows the PSO with strong search performance and high adaptability to optimization problems. Meanwhile, the PSO algorithm has a strong exploitation but its exploration is very poor [41–45]. Exploitation and exploration are local optimization capability and global optimization capability of algorithms respectively. If a meta-heuristic algorithm is to have good search performance, it should be able to effectively balance exploitation and exploration in the iterative process of the algorithm. Similar to other random algorithms, PSO also has shortcomings: when the initial solution is far away from the global optimal solution, PSO often appears premature convergence and local stagnation. In order to overcome the above shortcomings of PSO, it is necessary to introduce some strategies to effectively balance exploitation and exploration. This paper proposes a hybrid PSO enhanced by nonlinear inertia weight and Gaussian mutation (NGPSO) to solve JSSP. In the exploration stage, PSO mixes Gaussian mutation to improve the global search capability of the algorithm. In the

exploration stage, the nonlinear inertia weight is used to improve the change rules of population to improve the local optimization performance.

The rest of this paper is structured as follows: Section 2 introduces the basic PSO algorithm. Section 3 describes the proposed NGPSO algorithm, and analyzes the reason why the nonlinear inertial weight operator and Gaussian mutation strategy are merged into the PSO. Section 4 introduces the application of NGPSO in the JSSP. Section 5 analyzes the experimental results of NGPSO in solving JSSP benchmark instances. Finally, Section 6 gives a brief summary and outlook for future work.

2. An Overview of the PSO

For a long time, population intelligence and evolutionary algorithms have attracted scholars in various research fields. Researchers can derive inspiration from the behavior of biological populations in nature or the laws of biological evolution to design algorithms [46]. As one of many swarm intelligence algorithms, PSO has been proven effective in various optimization fields [47–52].

PSO is an innovative global optimization algorithm first proposed by Dr. Kennedy and Dr. Eberhart in 1995 [39,53]. It conducts a collaborative global search by simulating the foraging behavior of biological populations such as birds and fish [54]. The PSO algorithm can enhance the information exchange between in the population individuals by exchanging learning information, and then promote the evolution direction of the population to be consistent. This mode of information exchange resulting from population individuals gives the PSO algorithm a strong search capability and a higher degree of adaptability to optimization problems. In a d-dimensional solution space [55], a particle i includes two vectors: one is a speed vector $V_i = (v_{i1}, v_{i2}, v_{i3}, \cdots, v_{id})$, and the other is a position vector $X_i = (x_{i1}, x_{i2}, x_{i3}, \cdots, x_{id})$ [56]. Each individual in the population will iterate through two formulas. The particle speed and position update formula is as follows:

$$v_{id}^{t+1} = \omega v_{id}^{t} + c_1 r_{1d}(pbest_{id}^{t} - x_{id}^{t}) + c_2 r_{2d}(gbest_{id}^{t} - x_{id}^{t}) \quad (1)$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1} \quad (2)$$

where, ω is the inertia weight, which is an important parameter affecting the search performance of the algorithm [57], its value indicates the amount of particles inheriting the current individual speed. c_1 and c_2 are called acceleration factors, c_1 is called cognitive coefficient, which represents the self cognitive experience of particles, c_2 is called social coefficient, which represents the capability of particles to learn from the current global optimal solution; r_1 and r_2 are two independent random numbers with sizes between [0, 1] [37]; $pbest_{id}$ is the extreme value of particle i in the d-th dimension, $gbest_{id}$ is the global extremum of all particles in the d-th dimension.

3. NGPSO Algorithm

The performance of the hybrid meta-heuristic optimization algorithm relies on the complementary advantages of optimization strategies, that is, some strategies have better local search capabilities, and the other strategies have better global search capabilities. As a verification index describing the performance of individual iterations in swarm intelligence algorithms, how to improve both the local optimization and global optimization capabilities of the algorithm has become the key to improving the performance of the algorithm [58–61]. From the perspective of the iterative process of the algorithm, the global search capability is essentially the full-area search level determined by the iterative individual's breadth-first principle, which shows that the algorithm can search a larger area in the solution space to obtain a better solution [62]. On the other hand, the local search capability is essentially the sub-area search level determined by the iterative individual's depth-first principle, which can use the searched area to improve the problem solution, prevent the algorithm from falling into a stagnation state, and provide the possibility for the algorithm to continue to iterate in the search area and finally obtain a high-precision optimal solution. Therefore, balancing the full-area search

level and the sub-area search level becomes the key to optimize the search capability and alleviate the premature convergence of the algorithm.

3.1. Nonlinear Inertia Weight Improves the Local Search Capability of PSO (PSO-NIW)

PSO is a very successful algorithm, often applied for solving various practical problems. Nevertheless, similar to other optimization algorithms, PSO also has shortcomings [63], we need to adopt appropriate strategies to improve its shortcomings. The inertial weight ω, as a learning level and distribution proportion for balancing particle behavior, indicates the degree of influence of the pre-particle speed on the current particle speed, and determines the search ability of the algorithm. For this reason, we can choose a suitable inertia weight as the improvement strategy of the algorithm, which can enhance the convergence rate and solution accuracy of PSO. At present, the linear decreasing strategy of inertia weight is commonly used, also called linear adjustment ω strategy [64]. In an actual iteration process, ω will show a linear decreasing trend as the number of individual iterations increases. When considering the actual optimization problem, the global search is always used first to quickly converge the population iteration space to a certain area, then to obtain high-precision solutions through local search. The ω update formula is as follows:

$$\omega = \omega_{max} - \frac{iter \times (\omega_{max} - \omega_{min})}{iter_{max}} \tag{3}$$

where ω_{max} and ω_{min} are the maximum and minimum inertia weights respectively; $iter$ and $iter_{max}$ are the current and maximum times of iterations respectively. Generally, the value of ω is initialized to 0.9 and then linearly decreased to 0.4, which will get better results. However, the search process for the solution of the problem in practical problems is often non-linear. Therefore, the linearly decreasing strategy of inertia weight is not suitable for solving practical problems. Therefore, in this paper, a nonlinear inertia weight adjustment curve is used to adjust ω, as shown in Equation (4):

$$\omega = \omega_{max} - \frac{iter \times (\omega_{max} - \omega_{min})}{iter_{max}} \times \sin(\frac{iter \cdot \pi}{2 \cdot iter_{max}}) \tag{4}$$

3.2. Gauss Mutation Operation Improves the Global Search Capability of PSO (PSO-GM)

Gaussian distribution, also called normal distribution, is usually expressed as $N(\mu, \sigma^2)$, where μ and σ^2 is mean value and standard deviation respectively, and the Gaussian distribution has $3-\sigma$ rules, that is, The random numbers generated according to Gaussian distribution [65] have 68%, 95% and 99.7% probability fall into the intervals of $[\mu-\sigma, \mu+\sigma]$, $[\mu-2\sigma, \mu+2\sigma]$ and $[\mu-3\sigma, \mu+3\sigma]$, respectively. From the point of view of population energy, Gaussian variation can be considered as a kind of energy dissipation [66]. Because the diversity of population increases, the entropy of population becomes higher, which reduces the energy of population and makes the population have higher stability and adaptability. The Gaussian $3-\sigma$ distribution rule provides a good mathematical theoretical support for optimizing the distribution of individuals in the solution space. If the variable x follows Gaussian distribution, the probability density function of the random variable x is shown in Equation (5):

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} exp[-\frac{(x-\mu)^2}{2\sigma^2}] \tag{5}$$

The Gaussian distribution function of the random variable x is as follows [67]:

$$\phi(x) = \frac{1}{\sigma\sqrt{2\pi}} exp[\int_{-\infty}^{x} -\frac{(t-\mu)^2}{2\sigma^2} dt] \tag{6}$$

Gaussian mutation is a mutation strategy composed of random disturbances generated by Gaussian distribution based on the original individual. In this paper, the d-dimensional

Gaussian mutation operator is $N_i = (n_{i1}, n_{i2}, n_{i3}, \cdots, n_{id})$, and $N_i = (n_{i1}, n_{i2}, n_{i3}, \cdots, n_{id})$ follows the standard Gaussian distribution $N(\mu, \sigma^2)$. Gaussian mutation of d-dimensional individual $P_i = (p_{i1}, p_{i2}, p_{i3}, \cdots, p_{id})$ can be expressed as follows:

$$P_i' = P_i + P_i \cdot N_i \tag{7}$$

where $P_i \cdot N_i = (p_{i1} \times n_{i1}, p_{i2} \times n_{i2}, p_{i3} \times n_{i3}, \cdots, p_{id} \times n_{id})$, P_i' is the individual after Gaussian mutation. Then we use the greedy selection strategy to prevent the algorithm population from degenerating, the specific operations are as follows:

$$P_i^{t+1} = \begin{cases} P_i', If\ fit(P_i') > fit(P_i); \\ P_i, otherwise. \end{cases} \tag{8}$$

3.3. The Main Process of NGPSO

The main process of the proposed hybrid NGPSO is given below:

Step 1: Initialize the population and related parameters, including the population size and algorithm termination conditions;

Step 2: Initialize the speed and position of the particles, and record the *pbest* and *gbest*;

Step 3: Update the position and speed of all individuals according to the individual speed update Equation (1) and position update Equation (2) in the PSO;

Step 4: In sequence to prevent the algorithm from premature convergence, the Gaussian mutation is performed on individual individuals to generate new populations;

Step 5: Re-evaluate the adaptability of individuals in the new population. If the fitness of the new individual is better than that of the previous generation, replace the previous generation with the new individual, otherwise the original individual will not be changed;

Step 6: Update *pbest* and *gbest*;

Step 7: Judge whether the algorithm reaches the termination condition, if the individual iteration accuracy reaches the termination condition, then end the algorithm, otherwise execute **Step 3**.

4. NGPSO for JSSP

4.1. The JSSP Model

JSSP is a simplified model of various practical scheduling problems. It is currently the most widely studied type of scheduling problem. JSSP description: There are m machines in a processing system, which are required to process n jobs, and each job contains k processes. the processing time of each process has been determined, and each job must be processed in the order of the process, the task of scheduling is to arrange the processing scheduling sequence of all jobs, so that the performance indicators are optimized under the premise of satisfying the constraints [68]. A common mathematical description of the n/m/Cmax (where Cmax is the maximum completion time, called makespan [69]) scheduling problem is as follows [70]:

$$\min \max_{1 \le k \le m} \{ \max_{1 \le i \le n} c_{ik} \} \tag{9}$$

$$\begin{aligned} s.t.\ & c_{ik} - p_{ik} + M(1 - a_{ihk}) \ge c_{ih}, i = 1, 2, \cdots, n; h, k = 1, 2, \cdots, m; \\ & c_{jk} - c_{ik} + M(1 - x_{ijk}) \ge p_{ij}, i = 1, 2, \cdots, n; k = 1, 2, \cdots, m; \\ & c_{ik} \ge 0, i = 1, 2, \cdots, n; k = 1, 2, \cdots, m; \\ & x_{ijk} = 0 \text{ or } 1; i, j = 1, 2, \cdots, n; k = 1, 2, \cdots, m. \end{aligned} \tag{10}$$

where Equation (9) is the objective function; Equation (10) are the operation sequence of the jobs and the processing machine determined by the constraints. Where c_{ik} and p_{ik} are the completion time

and processing time of the jobs i on the machine k; M is a sufficiently large real number; a_{ihk} and x_{ijk} indicator factor and indicator variables, and their meaning is as follows [71,72]:

$$a_{ihk} = \begin{cases} 1, \text{If machine } h \text{ processes job } i \text{ before machine } k, \\ 0, \text{otherwise.} \end{cases} \tag{11}$$

$$x_{ijk} = \begin{cases} 1, \text{If job } i \text{ is processed on machine } k \text{ before job } j, \\ 0, \text{otherwise.} \end{cases} \tag{12}$$

In addition, the disjunctive graph is also an important tool for describing JSSP. For the JSSP of $n \times m$ (N operations), the corresponding disjunctive graph $G = (V, A, E)$ is shown in Figure 1. Where V is the vertex set formed by all operations, including two virtual operations of 0 and $N + 1$ (0 and $N + 1$ are the two states of the beginning and the end for processing jobs); A is an edge set consisting of n sub-edges, the sub-edge formed by the solid line part is the processing path of a job on all machines from the beginning to the end according to the constraints; E is the arc set composed of m sub-edges, the sub-arc formed by the dotted line indicates the connection of the processing operations on the same machine. If the maximum completion time is considered as an indicator, the solution of JSSP will be transformed into a set of sequences (that is, trends) for each operation on the arc (that is, the machine) as a priority decision. When several operations conflict on the same machine, the above-mentioned jobs processing sequence will determine the sequence of operations. In the end we will get a conflict-free directed acyclic graph about machine operation, the critical path length in machine operation is makespan.

(1) The coding of the scheduling solution is complex and diverse, and the search method of the algorithm is also diverse;

(2) The solution space is more complicated. The JSSP problem of n workpieces and m machines contains $(n!)^m$ kinds of arrangements;

(3) There are limitations of technical constraints, and the feasibility of the solution needs to be considered;

(4) The calculation of scheduling indicators is longer than the search time of the algorithm;

(5) The optimization hypersurface lacks structural information. The solution space of the JSSP problem is usually complicated, and there are multiple minimum values with irregular distribution.

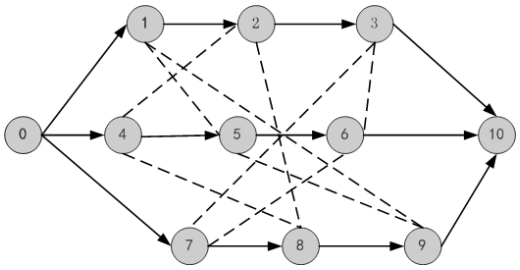

Figure 1. Disjunctive graph of 3 jobs and 3 machines JSSP.

4.2. Analysis of the Main Process of the NGPSO in Solving JSSP

When the NGPSO algorithm described in this section solves the JSSP problem, the analysis of its solution space is essentially a combinatorial optimization problem(COP). How to achieve the exact solution of the algorithm in the search space is the key to solving this problem. In sequence to make the NGPSO algorithm to better solve the COP, the evaluation of each solution in the solution space needs to be determined by the sequence of the NGPSO. As the individual discretization strategy, the heuristic rule of smallest position value (SPV) can transform continuous optimization problems into discrete optimization problems. At this time, the NGPSO algorithm can be reasonably used to solve COP [73]. Furthermore, the processing sequence of the jobs are determined by the discrete

solution, so that obtain the makespan value. The position vector of the particles is not the scheduling sequence, but due to the sequence relationship of the components in the position vector, and each particle corresponds to a scheduling sequence(namely, a scheduling scheme), we can use SPV rules to transform the continuous position vector into discrete scheduling sequence. Specifically, the SPV rule transform a particle's position vector $X_i = [x_{i1}, x_{i2}, \cdots, X_{in}]$ into a job scheduling sequence $J = [J1, J2, \cdots, Jn]$. In the SPV rule, the n components of the individual position vector are first sorted in the forward direction, the level of the smallest component is set to 1. Then, the remaining components are sequentially set to $2, \cdots, n$, so that the particle position vector is transformed into a job scheduling sequence. To illustrate the SPV rules more specifically, Table 1 gives an instance of transformations from continuous position vectors to discrete scheduling sequences. In this example, the position vector is $X_i = [0.36, 0.01, 0.67, 0.69, 1.19, 1.02]$. In X_i, x_{i2} is the smallest value of the components, and its level is set to 1; similarly, the x_{i1} level is set to 2, and so on to deduce the level of the remaining position components. In the following steps, we will explain the specific operation process of SPV rules in detail.

Table 1. A simple examples of the SPV rule.

Dimension	**1**	**2**	**3**	**4**	**5**	**6**
position value	0.36	0.01	0.67	0.69	1.19	1.02
smallest position value	2	1	3	4	6	5

First phase: Algorithm initialization.

Set the initial parameters of the algorithm, including the inertia weight, the maximum number of iterations, and the population size. In order to solve the discrete JSSP problem, an individual position vector is generated by a random function, and the randomized individual position vector is mapped into a discrete scheduling sequence by SPV rules. The specific operation: the position vector is converted into a one-to-one mapping sequence, then the position vector with the constraint mapping sequence is decoded within the necessary processing time to generate an initial scheduling scheme. Obviously, this decoding method can generate a suitable scheduling sequence. In the description of job processing time and job scheduling sequence in the 3×3 JSSP listed in Table 2, one dimension is a independent operation of a job. A schedule list is $n \times m$ independent operations, so the size of a scheduling list is $n \times m$. After individual initialization, the fitness of the individuals in the population is calculated, and the makespan is used as the target value. Figure 2 shows the mapping process between individual position variables and scheduling sequences. A job contains three operations, so a job appears exactly 3 times in the scheduling list. Figure 3 shows a feasible scheduling scheme generated from the scheduling list of Figure 2.

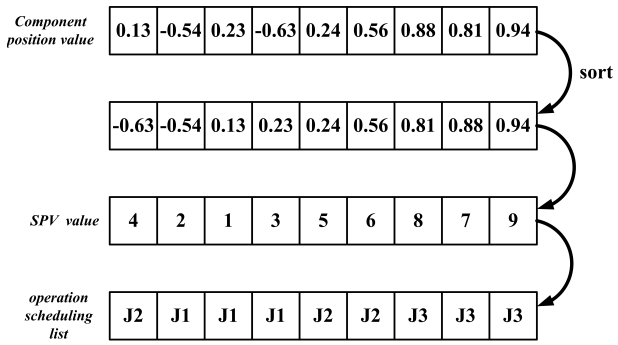

Figure 2. The mapping between individual position vectors and scheduling sequences in the 3×3 JSSP problem.

Table 2. Processing data of a 3 × 3 JSSP.

Projects	Jobs	Operations Number		
		1	**2**	**3**
operation time	J1	3	3	2
	J2	1	5	3
	J3	3	2	3
machine sequence	J1	M1	M2	M3
	J2	M1	M3	M2
	J3	M2	M1	M3

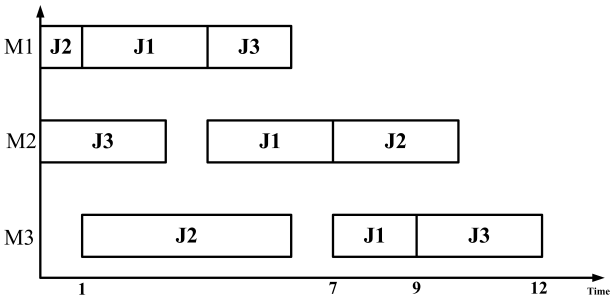

Figure 3. Scheduling scheme generated from the operation list in Figure 2.

Second phase: Update individual scheduling sequence

In this phase, as shown in Figure 4, each individual in the population updates the position through Equations (1) and (2). The position coordinate of the current particles is formed by the *pbest* and *gbest* in the population of the previous generation, which is a continuous vector [31]. Then a new scheduling list is generated through SPV rules, as a new discrete vector, which can solve the JSSP. In the current solution process, individuals are mainly searching randomly during the searching optimal solution, and the nonlinear inertia weight can better reflect the nonlinear search process of the PSO, and can improve the exploitation of the PSO. If the scheduling sequence corresponding to the current particle individual is more adaptable than before, the local individual optimal value *pbest* of the original individual will be replaced by the current individual position. If the fitness of the global extreme point in the current population is higher than the global extreme point of the previous generation, the current extreme point is used to replace the global extreme point *gbest* of the previous generation.

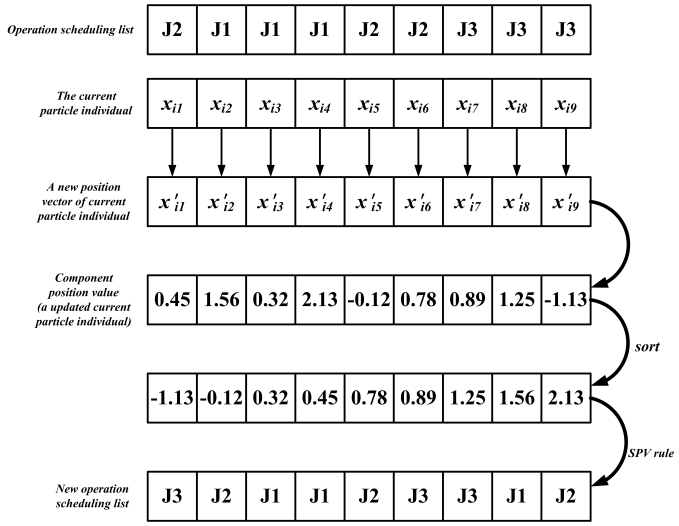

Figure 4. Update process of individual position vector.

Third phase: Increase individual diversity

After the iteration of the previous stage, all iterative individuals have a tendency to move in the same direction. Although this phenomenon contributes to the rapid convergence of the NGPSO, it will reduce the population diversity. All particles will perform mutation operations through Equations (6) and (7) to increase the diversity of the population to avoid the NGPSO from falling into the local optimal solution prematurely and losing the global search capability. Select individuals in the next generation population through greedy selection strategies. If the fitness of the new individual after mutation is higher than that of the previous generation, the original particle will be replaced by the new particle, otherwise the original particle will remain unchanged [74].

5. Experimental Results and Analysis

In this study, nonlinear inertia weights (NIW) and Gaussian mutation (GM) strategies are used to improve the basic PSO. We will verify the effectiveness of the two strategies in solving the JSSP problem by comparing the four algorithms PSO-NF, PSO-GM, and PSO. In total, 62 JSSP benchmark instances from Dr. Ling Wang's book: *JSSP and Genetic Algorithms* [75]. Including five ABZ benchmarks, this type of problem was given by Adams in 1988, including two different scales of five classic problems, where the scale of ABZ5 and ABZ6 is 10×10, and the scale of ABZ7, ABZ8 and ABZ9 is 20×15; three FT benchmarks, this type of problem was put forward by Fisher and Thompson in 1963, including three typical problems FT06, FT10 and FT20, with scales of 6×6, 10×10 and 20×5 respectively; 40 LA benchmarks, this type of question was given by Lawrence in 1984, named LA1 $\sim$ LA40, and divided into eight different scale questions, namely 10×5, 15×5, 20×5, 10×10, 15×10, 20×10, 30×10, 15×15; 10 ORB benchmarks, this type of problem was given by Applegate in 1991, named $ORB1 \sim ORB10$, with a scale of 10×10; and four YN benchmarks, This type of problem was put forward by Yamada et al. in 1992, including four typical problems, called YN1 $\sim$ YN4, with a scale of 20×20. Table 3 shows the results of 33 JSSP benchmark instances running 30 times on a computer equipped with Windows 7 (64) system, 8 GB RAM, Intel Core i7, CPU 3.4 GHZ, and MATALAB 2017a. OVCK in Table 3 is the optimal value currently known, the best is the optimal value in 30 runs, the optimal solution is marked in bold, and Avg is the average value in 30 runs. The following conclusions can be drawn from the experimental results shown in Table 3: The NGPSO, PSO-NIW, PSO-GM, and PSO algorithms obtained 26, 17, 21, and 14 optimal solutions from 33 benchmark instances. In solving simple problems, the performance of the four algorithms is similar, but for complex problems, the performance of the NGPSO algorithm is significantly better than the other three algorithms. When comparing the average test results of the four algorithms for JSSP, the number of optimal values obtained by the NGPSO algorithm is the largest number of 33 times in total, which exceeds the number of optimal values obtained by other algorithms. In addition, in the numerical experiment test, the population number is set to 100, the maximum number of iterations is set to $10,000$.

To further verify the effectiveness of the proposed NGPSO, the NGPSO is compared with PSO1 [36], PSO2 [76], CSA, GA, DE, and ABC, where PSO1 algorithm is based on tabu search algorithm (TS) and SA improved PSO. The TS makes the algorithm jump out of the local optimal value by retrieving the tabu search table. The SA prevents the algorithm from falling into a local optimal value with a certain probability. Combining local search and global search, PSO can achieve higher search efficiency. PSO2 algorithm uses GA and SA to overcome the shortcomings of PSO. Where crossover operation and mutation operation of GA can update the algorithm search area, SA can improve the local search capability. Tables 4 and 5 describes the performance comparison between different algorithms. OVCK in Tables 4 and 5 is the optimal value currently known, the best is the optimal value in 30 runs, the optimal solution is marked in bold, The parameter settings of the four comparison algorithms are as follows: the population size is 150, and the maximum number of iterations is 1000. The following results can be obtained from Tables 4 and 5: (1) For the best value, the NGPSO algorithm obtained 12 known optimal solutions in 29 benchmark instances, PSO1, PSO2, CSA, GA, DE, and ABC, obtained eight, five, three, five, nine, and six known optimal solutions, respectively. (2) For the average

value (Avg), the average value obtained by the NGPSO algorithm is smaller than other algorithms in most instances.

Table 3. Comparison of NGPSO, PSO-NIW, PSO-GM, PSO.

Instance	OVCK	NGPSO			PSO-NIW			PSO-GM			PSO		
		Best	**Worst**	**Avg**	**Best**	**Worst**	**Avg**	**Best**	**Worst**	**Avg**	**Best**	**Worst**	**Avg**
FT06	**55**	**55**	55	55	**55**	55	55	**55**	55	55	**55**	55	55
FT10	**930**	**930**	1056	955	**930**	1118	967	**930**	1073	959	932	1149	977
FT20	**1165**	1210	1311	1247	1178	1253	1249	1167	1313	1256	1180	1274	1261
LA1	**666**	**666**	666	666	**666**	666	666	**666**	666	666	**666**	666	666
LA2	**655**	**655**	655	655	**655**	655	655	**655**	655	655	**655**	655	655
LA3	**597**	**597**	676	635	*597*	680	539	609	679	646	624	690	653
LA4	**590**	**590**	622	609	613	646	627	604	635	618	627	678	634
LA5	**593**	**593**	593	593	**593**	593	593	**593**	593	593	**593**	593	593
LA6	**926**	**926**	926	926	**926**	926	926	**926**	926	926	**926**	926	926
LA7	**890**	**890**	890	890	**890**	890	890	**890**	890	890	**890**	890	890
LA8	**863**	**863**	863	863	**863**	863	863	**863**	863	863	**863**	863	863
LA9	**951**	**951**	951	951	**951**	951	951	**951**	951	951	**951**	951	951
LA10	**958**	**958**	997	969	963	1053	998	**958**	1022	988	**958**	1069	999
LA11	**1222**	**1222**	1222	1222	**1222**	1222	1222	**1222**	1222	1222	**1222**	1222	1222
LA12	**1039**	**1039**	1039	1039	**1039**	1039	1039	**1039**	1039	1039	**1039**	1039	1039
LA13	**1150**	**1150**	1150	1150	**1150**	1150	1150	**1150**	1150	1150	**1150**	1150	1150
LA14	**1292**	**1292**	1292	1292	**1292**	1292	1292	**1292**	1292	1292	**1292**	1292	1292
LA15	**1207**	**1207**	1207	1207	**1207**	1207	1207	**1207**	1207	1207	**1207**	1207	1207
LA16	**945**	**945**	945	945	**945**	992	978	**945**	972	956	962	994	980
LA17	**784**	794	803	798	811	878	849	**784**	866	833	822	879	836
LA18	**848**	**848**	999	913	**848**	1033	924	**848**	1054	933	857	1088	957
LA19	**842**	**842**	1032	905	889	1087	923	878	1073	933	891	1131	946
LA20	**902**	908	1228	965	1113	1342	1127	1009	1304	1130	1152	1385	1223
LA21	**1046**	1183	1271	1208	1190	1318	1224	**1046**	1324	1230	1201	1398	1244
LA22	**927**	**927**	989	966	936	1014	979	**927**	1112	982	957	1194	992
LA23	**1032**	**1032**	1245	1123	1109	1299	1203	1097	1283	1196	1100	1307	1208
LA24	**935**	968	1153	983	998	1184	1076	**935**	1166	1047	1003	1182	1089
LA25	**977**	**977**	1089	994	987	1145	1018	980	1129	1007	991	1211	1014
LA26	**1218**	**1218**	1443	1311	1233	1489	1383	1226	1442	1362	1287	1459	1399
LA27	**1235**	1394	1476	1412	1423	1499	1445	1403	1478	1431	1396	1503	1477
LA28	**1216**	**1216**	1440	1381	1230	1457	1390	1219	1444	1387	1290	1445	1379
LA29	**1152**	1280	1397	1304	1310	1429	1339	1344	1450	1410	1339	1557	1412
LA30	**1355**	**1335**	1567	1417	1404	1620	1503	1396	1592	1500	1428	1701	1511

In this table, the bold text in the OVCK column means is the optimal value currently known.The bold text in the Best column indicates that the algorithm found OVKC. For the convenience of comparison, it is marked in bold here.

In order to evaluate the performance of NPSO more intuitively, the convergence curve of the algorithms used for comparison in solving the ABZ6 instance is shown in Figure 5, where the vertical axis is makespan, and the horizontal axis is the number of iterations. It can be observed from the convergence images of Figure 5a,b that the convergence rate of NGPSO is slower than other comparison algorithms in the early stage of algorithm iteration, this shows that the algorithm has stronger global search capability than other comparison algorithms in the early stage of iteration, and the algorithm is not easy to fall into the local optimal solution in the early stage. In addition the NGPSO algorithm still maintains a strong search capability in the later period of the algorithm iteration. From the perspective of the entire iterative process, the global search capabilities and local search capabilities of NGPSO algorithm have been effectively balanced, which shows that the introduction of NIW and GM into PSO can effectively improve PSO, and is more suitable for solving JSSP than other comparison algorithms.

Table 4. Comparison of NGPSO, PSO1, PSO2, CSA.

Instance	OVCK	NGPSO			PSO1			PSO2			CSA		
		Best	Worst	Avg	Best	Worst	Avg	Best	Worst	Avg	Best	Worst	Avg
ABZ5	**1234**	**1234**	1457	1311	**1234**	1466	1323	1287	1487	1344	**1234**	1583	1447
ABZ6	**943**	**943**	1219	1093	**943**	1223	1137	954	1235	1120	1004	1284	1121
ABZ7	**656**	713	997	862	689	1087	901	664	1176	925	985	1329	1150
ABZ8	**665**	729	1298	1031	819	1470	1285	927	1483	1232	1199	1589	1337
ABZ9	**679**	930	1015	976	985	1024	990	958	1016	982	1008	1079	1034
ORB1	**1059**	1174	1297	1226	1214	1317	1276	1177	1282	1235	1232	1374	1353
ORB2	**888**	913	1038	957	969	1074	989	943	1010	970	1022	1114	1070
ORB3	**1005**	1104	1243	1172	1144	1265	1181	1166	1270	1192	1228	1295	1267
ORB4	**1005**	**1005**	1163	1140	**1005**	1182	1066	1016	1167	1078	1046	1211	1132
ORB5	**887**	**887**	1002	987	**887**	1028	994	913	1013	988	**877**	1092	997
ORB6	**1010**	1124	1203	1170	1187	1245	1221	1171	1247	1213	1191	1265	1222
ORB7	**397**	**397**	464	440	435	468	447	**397**	468	441	458	482	460
ORB8	**899**	1020	1106	1054	1018	1163	1056	**899**	1155	1080	1073	1173	1085
ORB9	**934**	980	1128	1032	1012	1139	1043	1021	1154	1042	1011	1150	1032
ORB10	**944**	1027	1157	1048	1040	1143	1067	1036	1154	1063	**944**	1204	1097
LA31	**1784**	**1784**	2143	1972	**1784**	2149	1986	1849	2156	1961	1872	2212	2057
LA32	**1850**	**1850**	2202	1987	**1850**	2237	1996	1944	2248	2002	1982	2397	2123
LA33	**1719**	**1719**	2001	1894	**1719**	2035	1907	**1719**	2022	1897	1829	2155	1956
LA34	**1721**	**1721**	2060	1939	1878	2147	1963	**1721**	2126	1955	2060	2304	2187
LA35	**1888**	**1888**	2212	1986	**1888**	2271	2010	1930	2308	2039	1918	2431	2181
LA36	**1268**	1408	1665	1523	1396	1691	1545	1415	1703	1562	1511	1775	1660
LA37	**1397**	1515	1693	1560	1524	1792	1623	1551	1787	1635	1613	1805	1743
LA38	**1196**	**1196**	1596	1388	1332	1781	1569	1198	1610	1454	1483	1708	1624
LA39	**1233**	1662	1799	1701	1712	1725	1718	1711	1825	1748	1731	1833	1768
LA40	**1222**	**1222**	1537	1413	1289	1583	1425	1244	1615	1423	1453	1661	1528
YN1	**888**	1248	1346	1291	1303	1411	1346	1259	1395	1289	1291	1426	1318
YN2	**909**	911	1208	1102	927	1198	1109	932	1226	1117	1042	1318	1207
YN3	**893**	**893**	1376	1189	903	1344	1201	**893**	1457	1255	1003	1487	1311
YN4	**968**	984	1299	1106	979	1376	1137	1008	1410	1124	1153	1677	1329

In this table, the bold text in the OVCK column means is the optimal value currently known.The bold text in the Best column indicates that the algorithm found OVKC. For the convenience of comparison, it is marked in bold here.

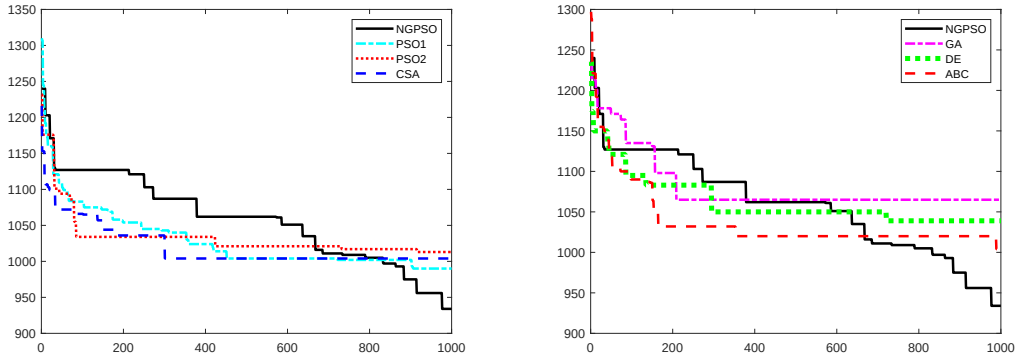

Figure 5. Objective functions convergence curves (Subgraphs a, b respectively represent the convergence curves of different comparison algorithms when solving ABZ6.) (**a**) Comparison of NGPSO, PSO1, PSO2, CSA in ABZ6; (**b**) Comparison of NGPSO, GA, DE, ABC in ABZ6.

Table 5. Comparison ofNGPSO, GA, DE, ABC.

Instance	OVCK	NGPSO			GA			DE			ABC		
		Best	**Worst**	**Avg**	**Best**	**Worst**	**Avg**	**Best**	**Worst**	**Avg**	**Best**	**Worst**	**Avg**
ABZ5	**1234**	**1234**	1457	1311	**1234**	1460	1336	**1234**	1465	1347	1236	1572	1443
ABZ6	**943**	**943**	1219	1093	954	1247	1142	948	1179	1103	992	1285	1138
ABZ7	**656**	713	997	862	729	1128	927	720	1166	918	973	1239	1106
ABZ8	**665**	729	1298	1031	805	1386	1242	822	1384	1165	748	1314	1137
ABZ9	**679**	930	1015	976	988	1124	1003	926	1023	988	944	1052	989
ORB1	**1059**	1174	1297	1226	1210	1348	1296	1191	1302	1244	1213	1320	1306
ORB2	**888**	913	1038	957	958	1064	997	932	1125	965	947	1096	988
ORB3	**1005**	1104	1243	1172	1136	1273	1201	1154	1281	1203	1228	1295	1267
ORB4	**1005**	**1005**	1163	1140	**1005**	1178	1057	**1005**	1183	1135	**1005**	1239	1142
ORB5	**887**	**887**	1002	987	913	1125	1006	**887**	1046	994	**877**	1092	1017
ORB6	**1010**	1124	1203	1170	1187	1349	1227	1171	1247	1213	1194	1277	1236
ORB7	**397**	**397**	464	440	**397**	479	456	**397**	468	441	**397**	471	455
ORB8	**899**	1020	1106	1054	1038	1188	1068	**899**	1149	1067	1043	1164	1076
ORB9	**934**	980	1128	1032	997	1158	1042	1003	1184	1055	1011	1150	1032
ORB10	**944**	1027	1157	1048	1024	1242	1107	1026	1163	1059	**944**	1164	1147
LA31	**1784**	**1784**	2143	1972	**1784**	2216	1978	**1784**	2177	1978	**1784**	2241	2078
LA32	**1850**	**1850**	2202	1987	1862	2249	2003	1935	2250	2013	1903	2344	2013
LA33	**1719**	**1719**	2001	1894	1786	2126	1923	**1719**	2103	1914	1749	2052	1948
LA34	**1721**	**1721**	2060	1939	1889	2155	1972	**1721**	2140	1953	1764	2054	1962
LA35	**1888**	**1888**	2212	1986	**1888**	2310	2015	1926	2319	2044	1902	2332	2135
LA36	**1268**	1408	1665	1523	1416	1631	1566	1423	1698	1557	1411	1765	1650
LA37	**1397**	1515	1693	1560	1549	1812	1647	1544	1767	1628	1524	1783	1646
LA38	**1196**	**1196**	1596	1388	1211	1682	1467	1214	1607	1450	**1196**	1609	1425
LA39	**1233**	1662	1799	1701	1677	1831	1782	1688	1845	1732	1681	1840	1746
LA40	**1222**	**1222**	1537	1413	1263	1602	1438	1255	1602	1431	1257	1610	1424
YN1	**888**	1248	1346	1291	1288	1430	1351	1260	1419	1292	1288	1419	1306
YN2	**909**	911	1208	1102	934	1201	1116	946	1240	1125	979	1253	1126
YN3	**893**	**893**	1376	1189	914	1355	1212	**893**	1387	1262	904	1422	1218
YN4	**968**	984	1299	1106	993	1384	1149	993	1426	1133	1082	1327	1129

In this table, the bold text in the OVCK column means is the optimal value currently known.The bold text in the Best column indicates that the algorithm found OVKC. For the convenience of comparison, it is marked in bold here.

6. Conclusions

This paper proposes the NGPSO algorithm to solve the JSSP problem. Nonlinear inertial weight (NIW) and Gaussian mutation (GM) strategies are introduced to basic PSO, where NIW is used to improve the local exploration of the NGPSO algorithm, the GM is used to improve the global exploration and maintain the population diversity of the NGPSO. In other words, NIW and GM have jointly improved the exploitation and exploration of PSO. In order to verify the effectiveness of the introduced strategies, firstly, the proposed NGPSO is used to solve 33 JSSP benchmark instances. It can be concluded that the proposed strategies can improve the performance of the PSO in solving JSSP problems. In addition, compared with PSO1, PSO2 in published papers and basic CSA, the proposed NGPSO algorithm can achieve better results than the comparison algorithm. On the whole, NGPSO can effectively solve the JSSP. In the future, the algorithm needs to be further applied to other scheduling problems, such as flexible job shop scheduling problem. In addition, the algorithm needs more rigorous formal proof, such as the convergence analysis of the algorithm.

Author Contributions: H.Y. solved the overall framework, algorithm framework, paper structure, and result analysis of the research ideas of the paper; Y.G. provided the research ideas of the paper; L.W. provides data analysis and corresponding analysis results, writing of the paper; J.M. provides the overall research ideas and main methods of the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Natural Science Foundation of China under Grant (11961001, 61561001), the Construction Project of first-class subjects in Ningxia higher Education (NXY LXK2017B09), and the major proprietary funded project of North Minzu University (ZDZX201901).

Acknowledgments: The authors are grateful to the responsible editor and the anonymous references for their valuable comments and suggestions, which have greatly improved the earlier version of this paper.

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

References

1. Garey, M.R.; Johnson, D.S.; Sethi, R. The complexity of flowshop and jobshop scheduling. *Math. Oper. Res.* **1976**, *1*, 117–129. [CrossRef]
2. Carlier, J.; EPinson, E. An algorithm for solving the job-shop problem. *Manag. Sci.* **1989**, *35*, 164–176. [CrossRef]
3. Moosavi, E.; Gholamnejad, J.; Ataeepour, M.; Khorram, E. Improvement of lagrangian relaxation performance for open pit mines constrained long-term production scheduling problem. *J. Cent. South Univ.* **2014**, *21*, 2848–2856. [CrossRef]
4. Li, H.; Womer, N.K. Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming. *Eur. J. Oper. Res.* **2015**, *246*, 20–33. [CrossRef]
5. Asadzadeh, L. A parallel artificial bee colony algorithm for the job shop scheduling problem with a dynamic migration strategy. *Comput. Ind. Eng.* **2016**, *102*, 359–367. [CrossRef]
6. Yurtkuran, A.; Emel, E. A discrete artificial bee colony algorithm for single machine scheduling problems. *Int. J. Prod. Res.* **2016**, *54*, 6860–6878. [CrossRef]
7. Murugesan, R.; Balan, K.S.; Kumar, V.N. Clonal selection algorithm using improved initialization for solving JSSP. *Int. Conf. Commun. Control Comput. Technol.* **2010** , 470–475. [CrossRef]
8. Lu, H.; Yang, J. An improved clonal selection algorithm for job shop scheduling. *Ubiquitous Comput.* **2009** , 34–37. [CrossRef]
9. Hui, T.; Dongxiao, N. Combining simulate anneal algorithm with support vector regression to forecast wind speed. *Nternational Conf. Geosci. Remote Sens.* **2010**, *2*, 92–94.
10. Croce, F.D.; Tadei, R.; Volta, G. A genetic algorithm for the job shop problem. *Comput. Oper. Res.* **1995**, *22*, 15–24. [CrossRef]
11. Wang, L.; Zheng, D.Z. A modified genetic algorithm for job shop scheduling. *Int. J. Adv. Manuf. Technol.* **2002**, *20*, 72–76. [CrossRef]
12. Salido, M.A.; Escamilla, J.; Giret, A.; Barber, F. A genetic algorithm for energy-efficiency in job-shop scheduling. *Int. J. Adv. Manuf. Technol.* **2016**, *85*, 1303–1314. [CrossRef]
13. Zhang, R.; Chiong, R. Solving the energy-efficient job shop scheduling problem: A multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption. *J. Clean. Prod.* **2016**, *112*, 3361–3375. [CrossRef]
14. Zhonghua, H.; Boqiu, Z.; Hao, L.; Wei, G. Bat algorithm for flexible flow shop scheduling with variable processing time. *Int. Conf. Mechatron.* **2017**, *690*, 164–171.
15. Abdelbasset, M.; Manogaran, G.; Elshahat, D.; Mirjalili, S. A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem. *Future Gener. Comput. Syst.* **2018**, *85*, 129–145. [CrossRef]
16. Simon, D. Biogeography-based optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 702–713. [CrossRef]
17. Rahmati, S.H.A.; Zandieh, M. A new Biogeography-Based Optimization (BBO) algorithm for the flexible job shop scheduling problem. *Int. J. Adv. Manuf. Technol.* **2012**, *58*, 1115–1129. [CrossRef]
18. Xinyu, S.; Weiqi, L.; Qiong, L.; Chaoyong, Z. Hybrid discrete particle swarm optimization for multi-objective flexible job-shop scheduling problem. *Int. J. Adv. Manuf. Technol.* **2013**, *67*, 2885–2901.
19. Lian, Z.; Jiao, B.; Gu, X. A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan. *Appl. Math. Comput.* **2006**, *183*, 1008–1017.
20. Niu, Q.; Jiao, B.; Gu, X. Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time. *Appl. Math. Comput.* **2008**, *205*, 148–158.
21. Ponsich, A.; Coello Coello, C.A. A hybrid differential evolution—Tabu search algorithm for the solution of job-shop scheduling problems. *Appl. Soft Comput.* **2013**, *13*, 462–474. [CrossRef]
22. Huang, K.L.; Liao, C.J. Ant colony optimization combined with taboo search for the job shop scheduling problem. *Comput. Oper. Res.* **2008**, *35*, 1030–1046. [CrossRef]
23. Saidimehrabad, M.; Dehnaviarani, S.; Evazabadian, F.; Mahmoodian, V. An Ant Colony Algorithm (ACA) for solving the new integrated model of job shop scheduling and conflict-free routing of AGVs. *Comput. Ind. Eng.* **2015**, *86*, 2–13. [CrossRef]

24. Zhang, S.J.; Gu, X.S. An effective discrete artificial bee colony algorithm for flow shop scheduling problem with intermediate buffers. *J. Cent. South Univ.* **2015**, *22*, 3471–3484. [CrossRef]
25. Wang, X.; Duan, H. A hybrid biogeography-based optimization algorithm for job shop scheduling problem. *Comput. Ind. Eng.* **2014**, *73*, 96–114. [CrossRef]
26. Lu, Y.; Jiang, T. Bi-population based discrete bat algorithm for the low-carbon job shop scheduling problem. *IEEE Access* **2019**, *7*, 14513–14522. [CrossRef]
27. Rohaninejad, M.; Kheirkhah, A.S.; Vahedi Nouri, B.; Fattahi, P. Two hybrid tabu search–firefly algorithms for the capacitated job shop scheduling problem with sequence-dependent setup cost. *Int. J. Comput. Integr. Manuf.* **2015**, *28*, 470–487. [CrossRef]
28. Babukartik, R.G. Hybrid algorithm using the advantage of ACO and cuckoo search for job scheduling. *Int. J. Inf. Technol. Converg. Serv.* **2012**, *2*, 25–34. [CrossRef]
29. Yu, K.; Wang, X.; Wang, Z. An improved teaching-learning-based optimization algorithm for numerical and engineering optimization problems. *J. Intell. Manuf.* **2016**, *27*, 831–843. [CrossRef]
30. Lu, C.; Xiao, S.; Li, X.; Gao, L. An effective multi-objective discrete grey wolf optimizer for a real-world scheduling problem in welding production. *Adv. Eng. Softw.* **2016**, *99*, 161–176. [CrossRef]
31. Liu, M.; Yao, X.; Li Yi. Hybrid whale optimization algorithm enhanced with lévy flight and differential evolution for job shop scheduling problems. *Appl. Soft Comput. J.* **2020**, *87*, 16. [CrossRef]
32. Sharma, N.; Sharma, H.; Sharma, A. Beer froth artificial bee colony algorithm for job-shop scheduling problem. *Appl. Soft Comput.* **2018**, *68*, 507–524. [CrossRef]
33. Qiu, X.; Lau, H.Y. An AIS-based hybrid algorithm with PSO for job shop scheduling problem. *IFAC Proc. Vol.* **2010**, *43*, 350–355. [CrossRef]
34. Masood, A.; Chen, G.; Mei, Y.; Zhang, M. Reference point adaption method for genetic programming hyper-heuristic in many-objective job shop scheduling. *Eur. Conf. Evol. Comput. Comb. Optim.* **2018**, 116–131. [CrossRef]
35. Dabah, A.; Bendjoudi, A.; Aitzai, A.; Taboudjemat, N.N. Efficient parallel tabu search for the blocking job shop scheduling problem. *Soft Comput.* **2019**, *23*, 13283–13295. [CrossRef]
36. Zhang, X.; Koshimura, M.; Fujita, H.; Hasegawa, R. An efficient hybrid particle swarm optimization for the job shop scheduling problem. *IEEE Int. Conf. Fuzzy Syst.* **2011**, 622–626. [CrossRef]
37. Abdel-Kader, R.F.An improved PSO algorithm with genetic and neighborhood-based diversity operators for the job shop scheduling problem. *Appl. Artif. Intell.* **2018**, *32*, 433–462. [CrossRef]
38. Dao, T.; Pan, T.; Nguyen, T.; Pan, J.S. Parallel bat algorithm for optimizing makespan in job shop scheduling problems. *J. Intell. Manuf.* **2018**, *29*, 451–462. [CrossRef]
39. Jiang, T.; Zhang, C.; Zhu, H.; Gu, J.; Deng, G. Energy-efficient scheduling for a job shop using an improved whale optimization algorithm. *Mathematics* **2018**, *6*, 220. [CrossRef]
40. Eberhart, R.; Kennedy, J. A new optimizer using particle swarm theory. In Proceedings of the Mhs95 Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995; IEEE: Piscataway, NJ, USA, 2002.
41. Pan, F.; Chen, D.; Lu, L. Improved PSO based clustering fusion algorithm for multimedia image data projection. *Multimed. Tools Appl.* **2019**, *79*, 1–14. [CrossRef]
42. Şenel, F.A.; Gökçe, F.; Yüksel, A.S.; Yiğit, T. A novel hybrid PSO–GWO algorithm for optimization problems. *Eng. Comput.* **2018**, *20*, 1359–1373. [CrossRef]
43. Yang, X. Swarm intelligence based algorithms: A critical analysis. *Evol. Intell.* **2014**, *7*, 17–28. [CrossRef]
44. Ostadmohammadi Arani, B.; Mirzabeygi, P.; Shariat Panahi, M. An improved PSO algorithm with a territorial diversity-preserving scheme and enhanced exploration–exploitation balance. *Swarm Evol. Comput.* **2013**, *11*, 1–15. [CrossRef]
45. Rezaei, F.; Safavi, H.R. GuASPSO: A new approach to hold a better exploration–exploitation balance in PSO algorithm. *Soft Comput.* **2020**, *24*, 4855–4875. [CrossRef]
46. Deepa, O.; Senthilkumar, A. Swarm intelligence from natural to artificial systems: Ant colony optimization. *Int. J. Appl. Graph Theory Wirel. Ad Hoc Netw. Sens. Netw.* **2016**, *8*, 9–17.
47. Zuo, X.; Zhang, G.; Tan, W. Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid iaas cloud. *IEEE Trans. Autom. Ence Eng.* **2014**, *11*, 564–573. [CrossRef]

48. Chen, S.M.; Chiou, C.H. Multiattribute decision making based on interval-valued intuitionistic fuzzy sets, PSO techniques, and evidential reasoning methodology. *IEEE Trans. Fuzzy Syst.* **2015**, *23*, 1905–1916. [CrossRef]
49. Chen, S.M.; Jian, W.S. Fuzzy forecasting based on two-fctors second-order fuzzy-tend logical relationship groups, similarity measures and PSO techniques. *Inf. Sci.* **2016**, *391*, 65–79.
50. Wang, X.W.; Yan, Y.X.; Gu, X.S. Welding robot path planning based on levy-PSO. *Kongzhi Yu Juece Control Decis.* **2017**, *32*, 373–377.
51. Godio, A.; Santilano, A. On the optimization of electromagnetic geophysical data: Application of the PSO algorithm. *J. Appl. Geophys.* **2018**, *148*, 163–174. [CrossRef]
52. Khan, I.; Pal, S.; Maiti, M.K. A hybrid PSO-GA algorithm for traveling salesman problems in different environments. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.* **2019**, *27*, 693–717. [CrossRef]
53. Lu, L.; Luo, Q.; Liu, J.; Wu, X. An improved particle swarm optimization algorithm. *Granul. Comput.* **2008**, *193* , 486–490.
54. Qin, Q.; Cheng, S.; Zhang, Q.; Li, L.; Shi, Y. Biomimicry of parasitic behavior in a coevolutionary particle swarm optimization algorithm for global optimization. *Appl. Soft Comput.* **2015**, *32*, 224–240. [CrossRef]
55. Hema, C.R.; Paulraj, M.P.; Yaacob, S.; Adom, A.H.; Nagarajan, R. Functional link PSO neural ntwork based classification of EEG mental task signals. *Int. Symp. Inf. Technol.* **2008**, *3*, 1–6.
56. Rao, A.R.; Sivasubramanian, K. Multi-objective optimal design of fuzzy logic controller using a self configurable dwarm intelligence algorithm. *Comput. Struct.* **2008**, *86*, 2141–2154.
57. Pan, Q.; Tasgetiren, M.F.; Liang, Y. A discrete particle swarm optimization algorithm for single machine total earliness and tardiness problem with a common due date. *IEEE Int. Conf. Evol. Comput.* **2006**, 3281–3288. [CrossRef]
58. Lin, L.; Gen, M. Auto-tuning strategy for evolutionary algorithms: Balancing btween exploration and exploitation. *Soft Comput.* **2009**, *13*, 157–168. [CrossRef]
59. Crepinsek, M.; Liu, S.H.; Mernik, M. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.* **2013**, *45*, 1–35. [CrossRef]
60. Zhao, Z.; Zhang, G.; Bing, Z. Job-shop scheduling optimization design based on an improved GA. *World Congr. Intell. Control Autom.* **2012**, 654–659. [CrossRef]
61. Zhu, L.J.; Yao, Y.; Postolache, M. Projection methods with linesearch technique for pseudomonotone equilibrium problems and fixed point problems. *U.P.B. Sci. Bull. Ser. A.* **2020**, in press.
62. Fan, S.S.; Chiu, Y. A decreasing inertia weight particle swarm optimizer. *Eng. Optim.* **2007**, *39*, 203–228. [CrossRef]
63. El Sehiemy, R.A.; Selim, F.; Bentouati, B.; Abido, M.A. A Novel multi-objective hybrid particle swarm and salp optimization algorithm for technical-economical-environmental operation in power systems. *Energy* **2019**, *193*, 116817.
64. Shi, Y.H.; Eberhart, R.C.; Shi, Y.; Eberhart, R.C. Empirical study of particle swarm optimization. *Congr. Evol. Comput.* **1999**, *3*, 101–106.
65. Krohling, R.A. Gaussian particle swarm with jumps. *Congr. Evol. Comput.* **2005**, *2*, 1226–1231.
66. Wiech, J.; Eremeyev, V.A.; Giorgio, I. Virtual spring damper method for nonholonomic robotic swarm self-organization and leader following. *Contin. Mech. Thermodyn.* **2018**, *30*, 1091–1102. [CrossRef]
67. Yao, X.; Liu, Y.; Lin, G. Evolutionary programming made faster. *IEEE Trans. Evol. Comput.* **1999**, *3*, 82–102.
68. Yu, S. Differential evolution quantum particle swarm optimization for solving job-shop scheduling problem. In *Chinese Control and Cecision Conference*; IEEE: Piscataway, NJ, USA,2019.
69. Zhu, Z.; Zhou, X. An efficient evolutionary grey wolf optimizer for multi-objective flexible job shop scheduling problem with hierarchical job precedence constraints. *Comput. Ind. Eng.* **2020**, *140*, 106280. [CrossRef]
70. Song, X.; Chang, C.; Zhang, F. A novel parallel hybrid algorithms for job shop problem. *Int. Conf. Nat. Comput.* **2008**, 452–456. [CrossRef]
71. Song, X.; Chang, C.; Cao, Y. New particle swarm algorithm for job shop scheduling problems. *World Congr. Intell. Control Autom.* **2008**, 3996–4001. [CrossRef]
72. Wang, L.; Tang, D. An improved adaptive genetic algorithm based on hormone modulation mechanism for job-shop scheduling problem. *Expert Syst. Appl.* **2011**, *38*, 7243–7250. [CrossRef]

73. Tasgetiren, M.F.; Sevkli, M.; Liang, Y.-C.; Gençyilmaz, G. Particle swarm optimization algorithm for single machine total weighted tardiness problem. *Evol. Comput.* **2004**, *2*, 1412–1419.
74. Rao, Y.; Meng, R.; Zha, J.; Xu, X. Bi-objective mathematical model and improved algorithm for optimisation of welding sop scheduling problem. *Int. J. Prod. Res.* **2019**, *58*, 2767–2783. [CrossRef]
75. Ling, W. *Jop Shop Scheduling Problems and Genetic Algorithm*; Tsinghua University Press: Beijing, China,2003; pp. 59–63. (In Chinese)
76. Meng, Q.; Zhang, L.; Fan, Y. A hybrid particle swarm optimization algorithm for solving job shop scheduling problems. In *Theory, Methodology, Tools and Applications for Modeling and Simulation of Complex Systems*; Springer: Singapore, 2016; pp. 71–78.