Hindawi
Scientific Programming
Volume 2021, Article ID 4731012, 6 pages
https://doi.org/10.1155/2021/4731012

Research Article

Solving No-Wait Flow Shop Scheduling Problem Based on Discrete Wolf Pack Algorithm

Rongshen Lai,[1,2] **Bo Gao**,[1] **and Wenguang Lin**[1,2]

[1]*School of Mechanical and Automotive Engineering, Xiamen University of Technology, Xiamen 361024, China*
[2]*Xiamen Key Laboratory of Intelligent Manufacturing Equipment, Xiamen 361024, China*

Correspondence should be addressed to Rongshen Lai; 122774584@qq.com

Received 5 November 2021; Accepted 6 December 2021; Published 21 December 2021

Academic Editor: Yi-Zhang Jiang

Aiming at the no-wait flow shop scheduling problem with the goal of minimizing the maximum makespan, a discrete wolf pack algorithm has been proposed. First, the methods for solving the no-wait flow shop scheduling problem and the application research of the wolf pack algorithm were summarized, and it was pointed out that there was lack of research on the application of the wolf pack algorithm to solve the no-wait flow shop scheduling problem. According to the analysis of characteristics of the no-wait flow shop scheduling problem, the individual wolf was coded by a decimal integer; wolf searching behavior was realized through the exchange of different code bits in the individual wolf, and the continuous code segment of the head wolf was randomly selected to replace the corresponding code of the fierce wolf, by which the behaviors of wolves raiding and sieging were realized, and the population was updated according to the rule of "survival of the strong." In particular, to fully explore the potential optimal solution in the solution space, loop operations were added to the wandering, summoning, and siege processes. Finally, based on a comparison with the leapfrog algorithm and the genetic algorithm, the effectiveness of the algorithm was verified.

1. Introduction

Production scheduling is a key link to ensure the efficient and orderly development of the manufacturing process. It is an important way to quickly respond to customer needs, improve corporate economic efficiency, and maintain market competitiveness. Research on scheduling issues has important theoretical and practical significance in the current intelligent manufacturing context. The no-wait flow shop scheduling problem (NWFSP) is a very important type of scheduling problem, which widely exists in food processing, chemical, metallurgy, and pharmaceutical industries, and is also a typical NP-hard problem [1]. Based on the inspiration of the marvelous group phenomenon in nature, researchers have now proposed many effective swarm intelligence optimization algorithms to solve this problem, such as genetic algorithm [2], particle swarm algorithm [3], ant colony algorithm [4], etc. The development of swarm intelligence optimization algorithms is in the ascendant, providing many options for solving complex optimization problems. The wolf pack algorithm (WPA) is a group intelligence optimization algorithm obtained by simulating the hunting activities of the wolf pack. It has the advantages of strong global search ability, good generalization ability, and easy operation. It has significant effect on processing multi-peak and high-dimensional complex functions, especially, WPA is suitable for solving various complex combinatorial optimization problems, such as TSP and knapsack problem. [5]. However, currently, there are few research reports on the application of the wolf pack algorithm to the NWFSP. To this end, this paper combines the implementation process of the wolf pack algorithm and the feature analysis of NWFSP and proposes an improved discrete wolf pack algorithm and proves its effectiveness through a practical example and comparison with the leapfrog algorithm (LFA) and the genetic algorithm (GA).

The remainder of this paper is organized as follows. In Section 2, a state-of-the-art about method for solving the no-wait flow shop scheduling problem and wolf pack algorithm is provided. In Section 3, the mathematical model of the no-wait flow shop scheduling problem is described. In Section 4, the proposed improved discrete wolf pack algorithm and the corresponding algorithm process are illustrated in detail. In Section 5, a case study and comparison with particle swarm algorithm and genetic algorithm are carried out to verify the effectiveness of the proposed algorithm. Finally, the conclusion and future research directions are pointed out in Section 6.

2. Research Status

Focusing on the research of discrete wolf pack algorithm for solving no-wait flow shop scheduling problem, this section focuses on the literature review from two aspects: the method for solving the no-wait flow shop scheduling problem and the application of wolf pack algorithm.

2.1. Research on Methods for Solving No-Wait Flow Shop Scheduling Problem. The no-wait flow shop scheduling model is mainly aimed at the production processes that cannot be interrupted, such as steel rolling, food production, and so on. This model has a wide range of applications and is difficult to solve. It has attracted the attention and in-depth research of many experts and scholars, such as Song et al. [6] proposed a neighborhood iterative search algorithm for NWFSP, which reduced the time complexity of the solution process and enhanced the ability to find the global optimal solution. Zhang and Yu [7] aimed at the NWFSP with makespan minimization and proposed a discrete fruit fly optimization algorithm based on the dominant population. Orhan and Abdullah [8] aimed at the non-wait flow shop scheduling problem with makespan minimization as the criterion, and proposed a new hybrid ant colony algorithm based on crossover and mutation mechanism, and the performance of the algorithm was compared with adaptive learning methods and genetic heuristic algorithms; Zhao et al. [9] used hybrid biogeographic optimization algorithm and variable neighborhood search algorithm comprehensively to solve NWFSP. Allahverdi [10] carried out a systematic review of no-wait flow scheduling problems.

2.2. Research on Wolf Pack Algorithm. Wolves are a highly social species with a strict hierarchy and strong domain awareness. Wolves are usually led by the head wolves with absolute superiority. They kill the prey through a clear division of labor and cooperation among members and distribute food according to the "survival of the strong" rule. Liu et al. [11] simulated the intelligent hunting behavior of wolves, abstracted the three behaviors of searching for prey, besieging prey, and updating wolves, and proposed the wolf colony algorithm (WCA) in 2011 to solve the optimization problem. The main processes included assigning artificial wolves from wolves to search prey within the range of prey activities. Once the prey was found, other artificial wolves will be notified of the position of the prey by howling, and other artificial wolves will approach the prey to encircle. The WCA mainly included five steps: initialization, selection of wolves to detect prey, treating the optimal position of some artificial wolf as the position of the prey, updating the wolf pack according to the "survival of the strong" rule, and judging whether the termination condition was met.

Based on the analysis of the characteristics of wolf pack cooperative hunting and prey distribution, Wu et al. [5] abstracted 3 kinds of artificial wolves (head wolf, searching wolf, fierce wolf), 3 kinds of intelligent behaviors (wandering, summoning, siege), and 2 kinds of intelligent rules (wolf generation rule "winner is king" and wolf pack update mechanism "survival of the strong"), and proposed the wolf pack algorithm (WPA) with different optimization strategy compared with WCA in 2013. The convergence of the algorithm was proved based on Markov chain theory, and the comparison with other algorithms verified that the algorithm had better global convergence and computational robustness. Based on WPA, Hui et al. [12] proposed an improved wolf pack algorithm in 2017 by the introduction of the concept of siege radius and optimization of step length and design the position update formula of fierce wolves. Based on the research of grey wolf hunting behavior [13], Mirjalili et al. [14] proposed a new meta-inspiration algorithm–grey wolf algorithm (GWA) in 2014, which simulated the leadership hierarchy and the hunting mechanism of grey wolf groups in nature, abstracted 4 grey wolf levels and 3 main steps of hunting, namely searching, encircling, and attacking prey, and finally compared with particle swarm optimization and some meta-heuristic algorithms such as gravity search, differential evolution, evolution planning and evolution strategy to verify the effectiveness of the algorithm.

The wolf pack algorithm has good performance in global search and local development capabilities. Since its proposal, it has continuously attracted the attention of scholars and has been quickly applied to engineering practice. Yi et al. [15] proposed a hierarchic wolf pack algorithm to solve the problem of optimal placement of sensors; Wu et al. used the wolf pack algorithm to solve the binary knapsack problem [16], the traveling salesman problem [17], and unconstrained global optimization problem [18]. Fang and Tang [19] used an improved wolf pack algorithm to solve the three-dimensional routing optimization problem for AVE/RS composite operation. Liu et al. [20] used the wolf pack algorithm to plan the UAV track with a known starting point and endpoint. Wang and Jiao [21] proposed an improved wolf pack algorithm to solve the optimal scheduling problem of hydropower stations and reservoirs. Xie and Zhang [22] proposed a discrete wolf pack algorithm according to the characteristics of the permutation flow shop scheduling problem.

It was found from the research review that compared with some other algorithms such as particle swarm algorithm, dynamic programming algorithm, et al., the wolf pack algorithm had shown stronger optimization ability and faster convergence speed in the process of solving a combinatorial optimization problem. However, aiming at the typical combinatorial optimization problem, the no-wait flow shop scheduling problem, the application research of the wolf pack algorithm was relatively lacking. To this end, an improved discrete wolf pack algorithm was proposed in this paper to solve the no-wait flow shop scheduling problem.

3. Mathematical Model of NWFSP

The no-wait flow shop scheduling problem can be described as follows [3].

Given: (1) m machines and n workpieces. (2) The processing sequence of the workpieces on the machines is the same. (3) The processing time of each workpiece on each machine. (4) All workpieces can be processed at zero time.

Constraints: (1) A workpiece can only be processed on one machine at a certain time. (2) A machine can only process one workpiece at a certain time. (3) The transportation time of the workpiece and the start-up time of the machine is included in processing time. (4) All processes of the same workpiece must be processed continuously, that is, once each workpiece starts to be processed, each process must be performed continuously, and there is no waiting time between two adjacent processes.

Goal: To determine a scheduling plan that minimizes the maximum makespan.

Based on the literature [23], assuming that the processing time of the workpiece i on the machine k is $p_{i,k}$, according to the continuous production requirement, the difference between the start time of two adjacent workpieces $i-1$ and i (start processing time interval) $d_{i-1,i}$ shall meet the following requirement, as shown in(1):

$$d_{i-1,i} = \max\left\{\max_{2\le k\le m}\left\{\sum_{q=1}^{k} p_{i-1,q} - \sum_{q=1}^{k-1} p_{i,q}\right\},\ p_{i-1,1}\right\}. \quad (1)$$

The maximum makespan is calculated as:

$$T_{\max} = \sum_{j=2}^{n} d_{j-1,j} + \sum_{k=1}^{m} p_{n,k}. \quad (2)$$

According to the analysis, the calculation complexity of the maximum makespan is $O(mn^2)$.

4. Improved Discrete Wolf Pack Algorithm

The wolf pack algorithm realized the whole process simulation of the searching of prey and environmental information detection by individual wolves, the sharing and interaction of information between artificial wolves, and the whole process of artificial wolf capturing prey based on individual behavior decisions of their own responsibilities. Wolf pack algorithm [5] consists of three intelligent behaviors of wandering, summoning, and besieging, and the "winner is king" rule of wolf competition, and the "survival of the strong" wolf pack update mechanism. Based on the discrete wolf pack algorithm for solving the TSP problem proposed by Wu et al. [17], in view of the characteristics of NWFSP, this section introduces the population initialization, intelligent behaviors, and rule description of the improved discrete wolf pack algorithm in detail.

4.1. Coding Rules and Population Initialization. NWFSP is a typical discrete combinatorial optimization problem. According to the characteristics of the problem, the decimal encoding method is adopted, that is, each workpiece is represented by a decimal integer, and all the workpieces are processed on the machine according to the predetermined process. The processing sequence of all the workpieces constitutes a decimal sequence $X_i = (x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{in})$, which is used to represent the position of the ith artificial wolf in the wolf pack algorithm, where n represents the total number of workpieces to be processed, and x_{ij} represents the coding number of the jth processed workpiece in the sequence X_i. Taking the scheduling problem of 7 workpieces (coded from 1 to 7) as an example, the decimal sequence $X_i = \{4, 6, 3, 2, 5, 1, 7\}$ indicates that the processing order of the 7 workpieces is workpiece 4 $\longrightarrow$ workpiece 6 $\longrightarrow$ workpiece 3 $\longrightarrow$ workpiece 2 $\longrightarrow$ workpiece 5 $\longrightarrow$ workpiece 1 $\longrightarrow$ workpiece 7.

Determining the population size to be N, and using the random generation method to get the initial population $X = \{X_1, X_2, \ldots, X_i, \ldots, X_N\}$, where $1 \leqq i \leqq N$.

4.2. Intelligent Behavior and Rules. Based on the analysis of the characteristics of wolf pack cooperative hunting activities, the wolf pack algorithm abstracts the 3 intelligent behaviors of wolf pack wandering, summoning, and besieging, combined with the solution goal of NWFSP (that is, minimizing the maximum makespan), and elaborates the corresponding behavior rules in detail.

4.2.1. Selection of Head Wolf. The artificial wolf with the optimal objective function value (that is, the shortest makespan) in the initial population is selected as the head wolf; in the iteration process, after each iteration, the objective function value of the current head wolf is compared with the objective function values of other artificial wolves, If there is an artificial wolf whose objective function value is better than that of the head wolf, that artificial wolf is used to replace the head wolf; if the optimal objective function value corresponds to multiple artificial wolves, one of them is randomly selected as the head wolf.

4.2.2. Wandering Behavior. All artificial wolves in the wolf pack except the head wolf are regarded as detecting wolves to search for prey in the solution space. Assume that Y_i^0 and Y_{lead} respectively represent the prey odour concentration perceived by the wolf i and the head wolf in the initial population. The maximum number of wanderings are set as K. During the first wandering process, let the wolf i take one step forward in h directions respectively (the step length at this time is called the wandering step length $step_a$), and record the position in the pth direction and the perceived prey odour concentration Y_i^p $(1 \le p \le h)$, then return to the original position; if $Y_i^p \ge Y_{\text{lead}}$, the detecting wolf i will replace the head wolf; if $Y_i^0 < Y_i^p < Y_{\text{lead}}$, use the coordinates of one step forward in the pth direction to replace the coordinates of the detecting wolf i before wandering;

if $Y_i^p < Y_i^0 < Y_{\text{lead}}$, the coordinates of detecting wolf i remain unchanged. Specifically, the detecting wolf i takes one step forward in the p-th direction ($p = 1, 2, \ldots, h$), that is, randomly interchange two selected workpieces x_{ij} and x_{ik} in the code $Xi = \{x_{i1}, x_{i2}, \ldots, x_{im}\}$ of the detecting wolf i. Or randomly select a workpiece from the code $Xi = \{x_{i1}, x_{i2}, \ldots, x_{im}\}$ of the detecting wolf i, and then insert the workpiece after other workpieces from left to right. Repeat the above-mentioned wandering behavior until the maximum number of wanderings is reached, and then proceed to the summoning phase of the head wolf.

4.2.3. Summoning Behavior. All artificial wolves in the pack except the head wolf are regarded as fierce wolves. When the head wolf howls to summon the fierce wolves, the fierce wolves quickly rush towards the current position of the head wolf with a large step length. Referring to related literature [11], the summoning behavior is designed as follows.

Randomly select a piece of continuous code $x_{ls}, x_{l(s+1)}, \ldots, x_{le}$ with the starting point x_{ls} and the ending point x_{le} in the head wolf, respectively, where the number of code digits is the raiding step length $step_b$, replace a segment of continuous code $x_{is}, x_{i(s+1)}, \ldots, x_{ie}$ in the corresponding position of the fierce wolf i. This operation reflects the leadership of the head wolf (the optimal individual) to the wolf pack. If the elements of these two continuous codes are the same, the codes in other positions in the fierce wolf i will not be changed, which reflects the differentiation of the wolf pack individual. The element in $x_{is}, x_{i(s+1)}, \ldots, x_{ie}$ that does not belong to $x_{ls}, x_{l(s+1)}, \ldots, x_{le}$ are randomly placed in the rest of the fierce wolf i. The specific operation is as follows: Suppose the code of head wolf is $X_{\text{lead}} = \{5, 3, 4, 2, 8, 1, 7, 6\}$, and the code of the fierce wolf i is $X_i = \{4, 6, 3, 2, 8, 5, 7, 1\}$, randomly select a piece of continuous code from the head wolf as $\{4, 2, 8, 1\}$, and the corresponding code in the fierce wolf i is $\{3, 2, 8, 5\}$; use the code segment $\{4, 2, 8, 1\}$to replace the code segment $\{3, 2, 8, 5\}$, then the code $\{4, 6, 4, 2, 8, 1, 7, 1\}$ is obtained. In this code, the elements 4 and 1 appear repeatedly, and the elements 3 and 5 are missing. The elements 6 and 7 in the fierce wolf i does not belong to $\{4, 2, 8, 1\}$, so keep the position unchanged; randomly arrange 3 and 5 and replace the repeated elements 4 and 1, and finally get the updated code $\{3, 6, 4, 2, 8, 1, 7, 5\}$ or $\{5, 6, 4, 2, 8, 1, 7, 3\}$ of the fierce wolf i.

After updating the code of the fierce wolf i, recalculate its fitness value. If $Y_i > Y_{\text{lead}}$, the fierce wolf i will replace the head wolf. After reaching the set number of raids, the process enters the siege behavior.

4.2.4. Siege Behavior. After the raid process, the distance between the wolves and the prey is relatively close, and the head wolf will command the fierce wolves to besiege the prey. The design of the siege behavior is similar to the summoning behavior. In order to ensure that the wolves perform a fine search near the prey, the siege step length $step_c$ should not be greater than the raiding step length $step_b$ at this time, and the siege behavior ends when the set number of siege is reached.

4.2.5. Population Update. In order to prevent the population from entering the local optimum, the population is updated after each iteration. The specific operation is as follows: arrange the population from large to small according to the fitness value, remove the last R artificial wolves in the population, and then randomly generate R artificial wolves to join the population for the next iteration.

4.3. Algorithm Flow. The specific process of the improved discrete wolf pack algorithm is shown in Figure 1 and described in detail as follows.

Step 1: Parameter initialization. Set the population size of the wolf pack N, the maximum number of wandering times $T_{\max}$, the maximum number of iterations of the algorithm $k_{\max}$, the range of the search direction h, and the number of updates of the wolf pack R.

Step 2: Initialize the spatial position of the wolf pack, calculate its objective function value, and select the artificial wolf with the optimal fitness value as the head wolf X_{lead}.

Step 3: Wolf pack detecting. If the function value of the detecting wolf is greater than the head wolf, it will replace the head wolf and initiate a summoning behavior. Otherwise, the detecting wolves will continue to wander until the maximum number of wanderings is reached, the head wolf will summon the other wolves.

Step 4: Wolf pack raiding. The fierce wolves rush towards the prey. If the prey odour concentration of some fierce wolf is greater than that of the head wolf, this fierce wolf will replace the head wolf, and the fierce wolves will rush to the range close to the prey, and proceed to the next step.

Step 5: Wolf pack siege. If the prey odour concentration of some fierce wolf is greater than that of the head wolf, this fierce wolf will replace the head wolf.

Step 6: Population renewal. Update the position of the head wolf according to the "winner is king" rule, update the wolf pack according to the "survival of the strong" mechanism, and then enter the next iteration process.

Step 7: Determine whether the accuracy requirement is met or the maximum number of iterations is reached. If the algorithm termination condition is met, the optimal solution is output, that is, the position of the head wolf or the position of the prey; otherwise, go to Step 3 to continue.

5. Performance Verification

In order to verify the feasibility of the improved discrete wolf pack algorithm (IDWPA) designed in this paper in solving the NWFSP problem, this paper uses 5 sets of calculation examples to compare the algorithm with LFA and GA [24]. The termination condition of the algorithm is reaching the number of iterations. The initialization parameters of IDWPA are set as follows: the population size of the wolf

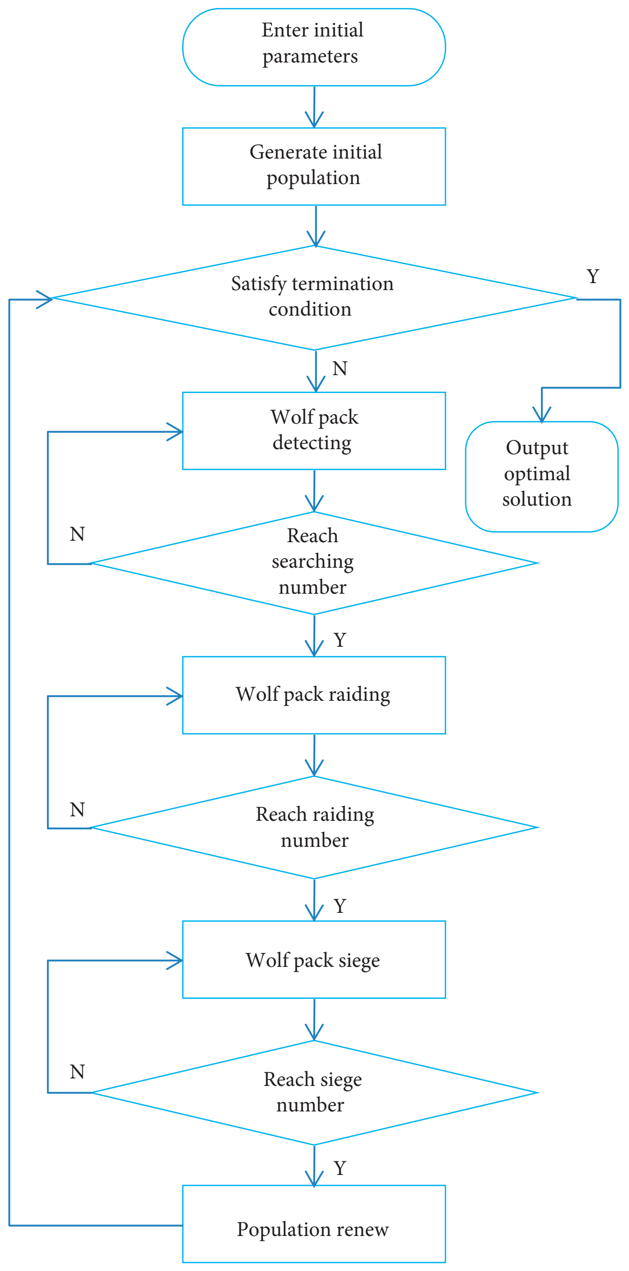

Figure 1: Flowchart of the improved discrete wolf pack algorithm.

pack $N = 20$, the maximum number of wandering times $T_{\max} = 10$, the maximum number of iterations of the algorithm $k_{\max} = 100$, the range of the search direction $h = 5$, and the number of updates of the wolf pack $R = 4$.

Table 1 shows the optimal solution and the average solution with three different algorithms. Compared with LFA and GA, IDWPA designed in this paper reduces the makespan of the NWFSP problem and improves the average equipment utilization rate.

It can be seen from the population evolution iterative process that the improved discrete wolf pack algorithm is easier to jump out of the local optimum and has a faster convergence rate when solving the NWFSP problem. Furthermore, the scheduling result is more accurate, and the utilization rate of the machine is improved to a certain extent, so the production efficiency of the enterprise can be improved to a certain extent.

Table 1: Scheduling solutions of three algorithms.

Case	Optimal solution			Average solution		
	IDWPA	LFA	GA	IDWPA	LFA	GA
10×10	1249	1249	1249	1262	1266	1285
15×15	1708	1719	1756	1742	1743	1808
20×10	1756	1756	1766	1776	1782	1877
20×20	1220	1217	1228	1287	1291	1298
50×10	3545	3509	3555	3580	3584	3654

6. Conclusions

Aiming at the no-wait flow shop scheduling problem, an improved discrete wolf pack algorithm was proposed. First of all, this paper summarized the method of solving the problem of no-waiting flow shop scheduling and the research on the application of the wolf pack algorithm and points out the research gap of using the wolf pack algorithm to solve NWFSP. Then, the decimal integer coding method was adopted; the wandering behavior of wolf detecting was realized through the exchange of single code bits, the raiding and siege behaviors were realized by replacing the continuous code segment. The population was updated according to the rule of "survival of the strong," and the summoning and siege links were added with cyclic operations. Finally, a comparison with the leapfrog algorithm and genetic algorithm was performed to verify the effectiveness of the proposed algorithm.

Data Availability

The data used to support the findings of this study are included in the article.

Conflicts of Interest

All the authors do not have any possible conflicts of interest.

Acknowledgments

This research was supported by the Xiamen Key Laboratory of Intelligent Manufacturing Equipment, Fujian Province Education and Research Project for Young and Middle-Aged Teacher (Grant nos. JAT200467 and JAT200472), Fujian Social Science Fund (FJ2021B128), the Fujian Supporting Project of STS Plan of Chinese Academy of Sciences (2021T3069), and the National Innovation Method Fund of China (No. 2019IM010300).

References

[1] N. G. Hall and C. Sriskandarajah, "A survey of machine scheduling problems with blocking and no-wait in process," *Operations Research*, vol. 44, no. 3, pp. 510–525, 1996.
[2] X. Zhu, X. Li, and Q. Wang, "Based on the total idle time increment no-waiting flow scheduling hybrid genetic

algorithm," *Computer Research and Development*, vol. 48, no. 3, pp. 455–463, 2011.

[3] Q. Pan, W. Wang, and J. Zhu, "Modified discrete particle swarm optimization algorithm for no-wait flow shop problem," *Computer Integrated Manufacturing Systems*, vol. 13, no. 6, pp. 1127–1130+1136, 2007.

[4] Q. Pan, B. Zhao, Y. Qu, and Y. Bi, "Ant-colony heuristic algorithm for no-wait flow shop problem with makespan criterion," *Computer Integrated Manufacturing Systems*, vol. 13, no. 9, pp. 1801–1804+1815, 2007.

[5] H. Wu, F. Zhang, and L. Wu, "New swarm intelligence algorithm-wolf pack algorithm," *Systems Engineering and Electronics*, vol. 35, no. 11, pp. 2430–2438, 2013.

[6] C. Song, X. Liu, and W. Wang, "An iterative neighbourhood search algorithm for large scale no-wait flow-shop," *Control and Decision*, vol. 26, no. 4, pp. 535–539, 2011.

[7] Q. Zhang and Z. Yu, "Discrete fruit fly optimization algorithm based on dominant population for solving no-wait flow shop scheduling problem," *Computer Integrated Manufacturing Systems*, vol. 23, no. 3, pp. 609–615, 2017.

[8] E. Orhan and G. Abdullah, "A new hybrid ant colony optimization algorithm for solving the no-wait flow shop scheduling problems," *Applied Soft Computing*, vol. 72, pp. 166–176, 2018.

[9] F. Zhao, S. Qin, Y. Zhang, W. Ma, C. Zhang, and H. Song, "A hybrid biogeography-based optimization with variable neighborhood search mechanism for no-wait flow shop scheduling problem," *Expert Systems with Applications*, vol. 126, no. JUL, pp. 321–339, 2019.

[10] A. Allahverdi, "A survey of scheduling problems with no-wait in process," *European Journal of Operational Research*, vol. 255, no. 3, pp. 665–686, 2016.

[11] C. Liu, X. Yan, and H. Wu, "The wolf colony algorithm and its application," *Chinese Journal of Electronics*, vol. 20, no. 2, pp. 212–216, 2011.

[12] X. Hui, Q. Guo, P. Wu et al., "An improved wolf pack algorithm," *Control and Decision*, vol. 32, no. 7, pp. 1163–1172, 2017.

[13] C. Muro, R. Escobedo, L. Spector, and R. P. Coppinger, "Wolf-pack (*Canis lupus*) hunting strategies emerge from simple rules in computational simulations," *Behavioural Processes*, vol. 88, no. 3, pp. 192–197, 2011.

[14] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[15] T. Yi, C. Wang, and H. Li, "Hierarchic wolf algorithm for optimal triaxial sensor placement," *Journal of Building Structures*, vol. 35, no. 4, pp. 223–229, 2014.

[16] H. Wu, F. Zhang, R. Zhan, and S. Wang, "A binary wolf pack algorithm for solving 0-1 knapsack problem," *Systems Engineering and Electronics*, vol. 36, no. 8, pp. 1660–1667, 2014.

[17] H. Wu, F. Zhang, H. Li et al., "Discrete wolf pack algorithm for traveling salesman problem," *Control and Decision*, vol. 30, no. 10, pp. 1861–1867, 2015.

[18] H.-S. Wu and F.-M. Zhang, "Wolf pack algorithm for unconstrained global optimization," *Mathematical Problems in Engineering*, vol. 2014, no. 1, pp. 1–17, Article ID 46508, 2014.

[19] Y. Fang and M. Tang, "Three-dimensional routing optimization for AVS/RS's composite operation," *Computer Integrated Manufacturing Systems*, vol. 21, no. 3, pp. 702–708, 2015.

[20] Y. Liu, W. Li, H. Wu et al., "Track planning for unmanned aerial vehicles based on wolf pack algorithm," *Journal of System Simulation*, vol. 27, no. 8, pp. 1838–1843, 2015.

[21] J. Wang and Y. Jiao, "Improvement of wolf pack algorithm and its application to optimal operation of reservoirs," *Engineering Journal of Wuhan University*, vol. 50, no. 2, pp. 161–167+173, 2017.

[22] R. Xie and H. Zhang, "Discrete wolf pack algorithm for permutation flow shop scheduling problem," *Control Engineering China*, vol. 27, no. 2, pp. 288–296, 2020.

[23] J. Grabowski and J. Pempera, "Some local search algorithms for no-wait flow-shop problem with makespan criterion," *Computers & Operations Research*, vol. 32, no. 8, pp. 2197–2212, 2005.

[24] Y. Wu, Y. Wang, H. Zhang et al., "Improved frog leaping algorithm for solving no-waiting flow shop scheduling problem," *Modular Machine Tool & Automatic Manufacturing Technique*, vol. 7, pp. 81–84, 2020.