Hindawi
Discrete Dynamics in Nature and Society
Volume 2022, Article ID 8545739, 10 pages
https://doi.org/10.1155/2022/8545739

*Research Article*

# Parameters Optimization of Multipass Milling Process by an Effective Modified Particle Swarm Optimization Algorithm

**Cuiyu Wang, Wenwen Wang, Yiping Gao, and Xinyu Li**

*School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, China*

Correspondence should be addressed to Yiping Gao; gaoyiping@hust.edu.cn

Received 28 June 2022; Revised 31 July 2022; Accepted 8 August 2022; Published 19 September 2022

Academic Editor: Xueyi Wang

The selection of the optimal metal milling parameters greatly impacts final product quality and production efficiency in modern manufacturing systems. The profit rate is also sensitive to the selected parameters. This research focuses on determining the optimal parameters of a multipass milling process using an improved particle swarm optimization (PSO) method. The objective is to minimize the production time. The proper number of passes, the optimal cut speed, and feed rate are considered as the parameters (the decision variables in the model) which are needed to be optimized. Furthermore, the permissive arbor strength, arbor deflection, and motor power are the constraints of the model. The penalty function method is used as the constraints handling technique to address the constraints efficiently in the proposed method. A case is adopted and solved to evaluate the performance of the proposed method. The experimental part is analyzed and compared with advanced methods. Experimental results show that the proposed method is very effective for parameters optimization of a multipass milling process and outperforms other methods.

## 1. Introduction

In recent years, optimal machining parameters for milling process play an important role to ensure the quality of final products, productivity, and the competition of the manufacturers [1]. As a result, determining the optimal milling parameters becomes a very hot research topic. Metal milling is one of the most important and widely used manufacturing processes. But it is difficult to choose the optimal milling parameters which are subject to multiconstraints of motor power, cutting force, and feasible range of machining parameters [2]. The single-pass milling process was originally preferred to find the minimum production time for economic reasons. However, with the development of the economy, the single-pass process cannot meet the needs of product quality [3]. Hence more and more researchers tend to focus on the multipass machining operation. Traditionally, the operators usually depend on the handbooks, and if the machines and processing types do not exist, the operators can only rely on their experiences to choose the parameters. However, handbooks and experiences cannot guarantee the selected parameters are optimal and this may lead to the waste of time and resources [4].

Several researchers used mathematical methods to find the authentic optimal machining parameters. The traditional strategies such as dynamic programming and geometric programming (GP) have been applied to the optimization of the single-pass cutting process [5]. Tolouei-Rad and Bidhendi [6] used a feasible direction method to determine the optimal parameters in milling operation. Petropoulos [7] used a GP method for practical optimization problems. Although the application of traditional methods mentioned above obtained better results than those gained by just following the handbooks, the results were far from robust when compared with the results obtained by heuristic approaches.

However, these traditional methods cannot be the optimal strategies to determine the optimal machining parameters, especially for the multipass process. This is because milling parameters optimization is a complicated multiconstraints nonlinear programming problem. Traditional methods rely on gradient information and are far from robust. So many researchers begin to apply heuristic methods to this problem. Vijayakumar et al. [8] used ant colony optimization (ACO) to find the satisfactory parameters in multipass turning operation. Palanisamy et al. [9] and Gakhe et al. [10] used a genetic algorithm (GA) to obtain the optimal machining parameters and minimize the total production time. Wang et al. [11] hybridized GA and SA and got the new strategy GSA, to select the optimal parameters in milling operation. The artificial bee colony (ABC) [12] was applied to the milling parameters optimization problem by Rao et al. [5]. Baykasoglu [13] proposed a weighted superposition attraction-repulsion algorithm for cutting conditions.

Particle swarm optimization (PSO) is a soft computation technology based on swarm intelligence [14, 15]. It can be used in the area of system design, multiobjective optimization, pattern recognition, signal processing, games, decision making, simulation, and identification [16, 17]. PSO had also been used to identify human tremors by Eberhart et al. [18], and the computational process was fast and the result was accurate. Han et al. [19] introduced PSO into the flexible manufacturing system to find the optimal scheduling and increase the use ratio of machines and productivity. Tang et al. [20] developed a parallel random matrix PSO for scheduling with budget constraints. Yang et al. [21] proposed an improved multiobjective PSO for scheduling with panel block construction. Sun and Ren [22] introduced graph density into PSO. PSO also had been used in the fields of CNC machining [23] and pulsed laser micromachining [24], and both gained good results. Fang et al. [25] proposed an improved adaptive PSO for cutting parameter optimization.

PSO is simple in concept, convenient in computation, fast in implementation, and brief in programming [26]. Using PSO to figure out nonlinear programming problems with multiple constraints is efficient. That is why PSO is considered to be one of the most popular methods among nature-inspired methods to solve complex optimization problems. However, the balance of the exploration (diversification) and exploitation (intensification) is one of the limitations, which impedes the application of PSO in the multipass milling process greatly. Hence, an effective modified particle swarm optimization method is proposed for the multipass milling process. The motivation of the proposed method is to balance the exploration (diversification) and exploitation (intensification) of PSO, which aims to improve the optimization results. With this motivation, three aspects of improvement, including the searchability, avoiding the local optima, and restart parameter, have been implemented, and the constraints handling strategy is introduced. Moreover, the penalty function method is combined with PSO to optimize the parameters optimization of the multipass milling process. The main contribution of this paper has several points. Firstly, in order to improve the PSO and avoid the local optima, the proposed method introduces three aspects of modification to balance the diversification and intensification. Secondly, a constraints handling strategy is introduced to construct the penalty function for the feasible particle. Finally, the proposed method is developed into a multipass milling process optimization problem, which has indicated the effectiveness of the proposed method.

The rest of this paper is organized as follows: the problem is formulated in Section 2. Section 3 introduces the proposed improved PSO. In Section 4, an application example has been selected to testify to the performance of the proposed PSO and the results are compared with the ones gained by other strategies. Some conclusions are given in the last section.

## 2. Problem Formulation

This paper uses the mathematical model proposed by [27]. The milling parameters optimization model is made up of three essential parts, which are objective function, constraints condition, and decision variables. In this model, the total production time is the objective function. The number of passes, the depth of cut for the rough pass and final pass, cutting speed, and feed per teeth are the parameters to be optimized. The constraints of this problem include the arbor strength, arbor deflection, and permissible motor power.

### 2.1. Objective Function

*2.1. Objective Function.* In this multipass milling model, a component's milling process contains some rough processes and one finishing process. The rough processes fulfil the removal of a large number of materials while the finishing process ensures the permissible surface roughness [5].

The total production time for a component is $T_{pr}$; it is composed of the following items:

$$T_{pr} = T_p + T_L + T_\alpha + T_m + T_c. \quad (1)$$

$T_{\mathrm{p}}$ is the machine set-up time. $T_p = T_s/N_b$, where $T_s$ is the set-up time for a new batch and $N_b$ means the total number of components in a produced batch.

$T_L$ is the loading-unloading time for one part.

$T_\alpha$ is the process of adjusting and quick return time.

$T_{\mathrm{m}}$ is the actual machining time. $T_m = L/f$, where $L$ (mm) is the cutting length and $f$ (mm/min) means the feed rate. $f = f_z \cdot Z \cdot N$, where $f_z$ stands for the feed per tooth (mm/tooth), $Z$ refers to the total number of teeth on the cutter, N means spindle speed (rpm), and $N$ can be given as $N = 1000 \cdot \mathrm{V}/\pi\mathrm{D}$, where $V$ is cutting speed (m/min) and $D$ means cut diameter (mm).

$T_{\mathrm{c}}$ is the time per component to change machining tool. $T_c = T_d T_m/T$, where $T_d$ means actual time for changing a dull cutting edge, and $T = (C_V^{1/m} D^{b_v/m} / V^{1/m} \alpha^{e_v/m} f_z^{u_v/m} \alpha_w^{r_v/m} Z^{n_v/m} \lambda_S^{q_v/m})(\mathrm{B_m B_h B_p B_t})^{1/m}$ represent tool life. For a multipass milling process, the total machining time should take every milling pass into consideration that $T_{pr}$ becomes

$$T_{pr} = \frac{T_S}{N_b} + T_L + N_P T_a + \sum_{i=1}^{N_P} \left( \frac{\pi DL}{1000 f_{zi} Z V_i} + \frac{T_d \pi L V_i^{1/m-1} a_i^{e_v/m} f_{zi}^{u_v/m-1} a_w^{r_v/m} Z^{n_v/m-1} \lambda_s^{q_v/m}}{1000 C_V^{1/m} D^{b_v/m-1} \left(B_m B_h B_p B_t\right)^{1/m}} \right). \quad (2)$$

$N_p$ is the number of passes to remove the total depth of cut.

$i$ is a subscript that refers to the $i$-th pass of a process.

$T_1 = T_s/N_b + T_L$ and $T_2 = \sum (T_{\alpha i} + T_{mi} + T_d T_{mi}/T)$.

*2.2. Constraint Conditions.* Feasible results are subject to some technological constraints that are related to the machining tools, cutting tools, and materials. Design variables are bounded by the following constraints [27].

*2.2.1. Power Constraint.* Tool power of machine ought to be less than effective power, which is shown as

$$P_c \le P_m \eta. \quad (3)$$

And for plain milling case,

$$C_{zp} \cdot \alpha_w \cdot Z \cdot D^{b_z} \alpha^{e_z} f_z^{u_z} V/6120 \le P_m \eta. \quad (4)$$

$P_c$ is the cutting power

$P_m$ is the nominal motor power

$\eta$ is the overall efficiency of the cutting machine

$F_c$ is the cutting force which can be calculated according to $C_{zp} \cdot \alpha_w \cdot Z \cdot D^{b_z} \alpha^{e_z} f_z^{u_z}$

*2.2.2. Arbor Strength.* Cutting force $F_c$ should not exceed the permissible strength $F_s$, and all the coefficients are following [27]:

$$F_c - F_s \le 0,$$

$$F_c - \left( \frac{0.1 k_b d_\alpha^3}{0.08 L_\alpha + 0.65 \sqrt{(0.25 L_\alpha)^2 + (0.5 D k_b / 1.3 k_t)}} \right) \le 0. \quad (5)$$

$F_c$ is the cutting force

$F_s$ is the permissible force for the arbor strength

$k_b$ is the permissible bending strength of the arbor

$d_\alpha$ is the diameter of the arbor

$L_\alpha$ is the arbor length between supports

$k_t$ is the permissible torsional stress of the arbor material

*2.2.3. Arbor Deflection.* Arbor deflection must be checked when choosing the satisfying feed rate:

$$F_c - F_d \le 0,$$

$$F_d = 4 \frac{Eed_a^4}{L_a^3}. \quad (6)$$

$F_d$ is the permissible strength of the arbor deflection

$E$ is the modulus of elasticity of the arbor material

$e$ is the permissible value of the arbor deflection, and $e = 0.2$ in roughing pass meanwhile $e = 0.2$ in finishing pass

*2.3. Decision Variables.* The number of passes, the cut depth of each pass, the feed rate, and the feed per tooth are design variables. The last two parameters can be gained from the range of spindle speed and spindle feed rate. The first two parameters are determined by the permissible boundary of cutting depth and the total milling depth. The feasible range of depth of cut is specified as [5]

$$\alpha_{\min} \le \alpha \le \alpha_{\max}, \quad (7)$$

where $\alpha_{\min}$ and $\alpha_{\max}$ determine the boundary of the choice of the depth of cut in every pass.

Yet the maximum spindle speed $N_{\max}$ and the minimum spindle speed $N_{\min}$ determine the range of milling speed:

$$V_{\min} \le V \le V_{\max}, \quad (8)$$

where $V_{\min} = \pi D N_{\min}/1000$ and $V_{\max} = \pi D N_{\max}/1000$.

Also, the feasible feed rate must be in the range determined by the maximum and minimum feed rate of the machine.

$$f_{z\min} \le f_z \le f_{z\max}, \quad (9)$$

where $f_{z\min} = f_{\min}/ZN_{\max}$ and $f_{z\max} = f_{\max}/ZN_{\min}$ and $f_{\max}$ are the minimum and maximum spindle feed rate. The PSO will be introduced in the following section based on the given multipass milling model.

## 3. Proposed PSO for Parameter Optimization

*3.1. Conventional PSO for Parameter Optimization.* PSO is a swarm-intelligence-based algorithm and was first proposed as the simulation of a bird flock's social behavior. PSO randomly generates a set of birds with random velocities and random positions. After that, an initial group of swarms is produced. Individuals in the population are particles. The number of particles in the swarm is the population size. Every particle searches for the best position to get the minimum value of the objective function just like the bird searching for food. The search space of the optimization problem is bounded by the feasible range of each decision variable. The position of each particle represents a potential solution because every dimension of the search space stands for the corresponding design variable. Each particle keeps track of its coordinates in the problem space which is associated with the best solution it has achieved so far [19]; it is $p_{\text{best}}$. Another important value that is tracked is the overall best value. The overall best location is called $g_{\text{best}}$. Fitness function is the criterion to judge the positions of particles. And fitness function is composed of the objective function and the constraints. $p_{\text{best}}$ and $g_{\text{best}}$ are continually updated by the better locations and better fitness in the searching process. This searching process is exactly like the self-learning and social-learning behavior of birds. In a word, the PSO concept consists of each time stop. Each particle changing velocity toward $P_{\text{best}}$ and $g_{\text{best}}$ randomly generates the acceleration for each best position. Every particle modifies its position and velocity after one search time by the equations given as follows:

$$v_i^{t+1} = \omega \cdot v_i^t + c_1 r_1 \left(P_{\text{best}} - P_i^t\right) + c_2 r_2 \left(G_{\text{best}_i} - p_i^t\right) \qquad i = 1, \ldots, d, \tag{10}$$

$$p_i^{t+1} = p_i^t + v_i^{t+1}, \quad i = 1, \ldots, d. \tag{11}$$

$d$ is the dimension of the optimization problem (the number of design variables)

$v_i = (v_1, v_2, \ldots, v_d)$ is the velocity of particles in the d-dimension search space

$p_i = (p_1, p_2, \ldots, p_d)$ is the i-th particle's d-dimension position

$p_{\text{best}_i} = (p_{\text{best}_1}, p_{\text{best}_2}, \ldots, p_{\text{best}_d})$ is the personal best position

$g_{\text{best}_i} = (g_{\text{best}_1}, g_{\text{best}_2}, \ldots, g_{\text{best}_d})$ is the global best position

$\omega$ means the inertia weight controls the particle's exploration and exploitation ability

$c_1$, $c_2$ are the acceleration constants that represent the weighting that pulls each particle toward $p_{\text{best}}$ and $g_{\text{best}}$ position

$r_1, r_2$ are two uniform random numbers between [0, 1]

The process for implementing PSO is as follows:

(i) Initialize the population of particles with random positions and random velocities within the search ranges

(ii) Evaluate each particle with both the objective function and constraints

(iii) Compare each particle's objective value with its $p_{\text{best}}$, take their constraints satisfaction into consideration, and if the current position is better, take the place of the previous fitness value and $p_{\text{best}}$

(iv) Compare each particle fitness evaluation and constraints satisfaction state with the overall previous best $g_{\text{best}}$, updating $g_{\text{best}}$ if it exits better particle and keeping in track of the best fitness value

(v) Change the velocity and position of each particle by (10) and (11)

(vi) Loop to (ii) if the maximum number of iterations is not reached or the given precision is met

*3.2. Constraints Handling Technique.* In PSO, a traditional method to deal with the constraints is usually adopted by Yang et al. [28]. In this method, the more constraints a particle violates, the worse the particle is. The degree of violation of constraints is neglected. So, it is not so easy for this method to judge the particles. The strategy is shown in Table 1.

The constraints handling technique is operated in the encoding work rather than be put into the objective function. But it becomes hard to ensure the results are feasible when the number of constraints increases. The final optimal particle may just violate fewer constraints but is still not feasible.

Thus, the penalty function method is adopted in this study. The overall constraints are contained in the penalty function. The fitness function is composed of the penalty function and the objective function. The results are usually feasible and subject to all the constraints when setting the punishment factor to a big enough number. The details to build the fitness function with the penalty function are introduced next.

For a general optimization problem, the mathematical expressions can be described as follows [29].

Assume that $x = (x_1, x_2, \ldots, x_n)$ is a point in n-dimension space $R^n$ and $f(x)$, $h_i(x)$, and $g_i(x)$ are given functions with $n$ variables. Consider the constraints as follows:

$$\begin{aligned} &h_i(x)\,(i = 1, \ldots, k) = 0, \\ &g_i(x)\,(i = 1, \ldots, p) \le 0. \end{aligned} \tag{12}$$

The optimization target is to obtain the minimum $f(x)$:

$f(x)$ is the objective function

$h_i(x)$ is $k$ equality constraints

$g_i(x)$ is $p$ inequality constraints

$x$ are the design variables

The fitness function is given as follows when considering $c$:

$$F(x) = f(x) + \delta P(x). \tag{13}$$

$\delta$ is the punishment factor

$P(x)$ is the punishment term

Without the $\varepsilon$ (a positive tolerance value for equality constraints) in [29], the penalty function can be written as

$$P(x) = m_1 \sum_{i=1}^{k} \left|h_i(x)\right|^2 + m_2 \sum_{i=1}^{p} \{\max(0, g_i(x))\}^2. \tag{14}$$

$m_1$ is the weight of equality constraints

$m_2$ is the weight of inequality constraints

Table 1: The commonly used updating strategy for $p_{\text{best}}$.

| Updating strategy for the personal best position (compare current fitness value and $p_{\text{best}}$) |
|---|
| (i) If these two solutions are both feasible, select the one with better fitness value; if both two fitness values are equal, randomly select one of them. |
| (ii) If one is feasible and the other is infeasible, select the feasible one. |
| (iii) If these two solutions are both infeasible, select the one with the lower number of constraints violations; if they violate the same number of constraints, randomly select one of them. |

The fitness function is built based on (13) and (14) with the constraints and objective function. The process of building this structure is simple, fast, and time-saving. The results are very stable while using this penalty function method.

*3.3. The Proposed PSO.* Based on the standard PSO, a new improved PSO is applied to this study. To obtain better computational results and avoid quick convergence, three improvements are applied to standard PSO. A new fitness function based on the problem is constructed.

*3.3.1. Modification of PSO*

(1) Particles no longer learn from one's own $p_{\text{best}}$. Parts of the swarm obtain $p_{\text{best}}$ of other members. The different $p_{\text{best}}$ guide the particle flying in another direction. The searchability of the swarm is enhanced. In this study, ten percent of particles were randomly chosen to implement this modification and change $p_{\text{best}}$ used in (10).

(2) To avoid the particles getting trapped into local optima, after an iteration of the proposed method, the particles do not update their position and velocity by (12) and (13). Instead, ten percent of particles implement their initialization and randomly get their positions and velocities. In this way, that the majority of the experience has been held back and delivered to the next generation. At the same time, the diversity of the swarm is kept.

(3) The last modification of the standard PSO is a restart parameter to stop its quick convergence. When all the particles go to the same direction guided by $g_{best}$, the swarm always have a fast convergence rate before finding the optimal point. To get thorough flying in the search space, the particles have some probability to reset and restart the method so the particles can jump out of local optima.

When $g_{best}$ does not update, a counter-repeat times (*RepeatTime*) increase by 1. When the repeat times reach the max repeat time (*MaxRepeatTime*) that means the particles may trap in the local minimum. We set a parameter *rs* that combines with the present iteration times and the max iteration times (*rs* is *present iteration times*/max *iteration times*). Another parameter *rand* is a randomly generated number in $[0, 1]$. On the condition that the max repeat times are reached, compare *rs* with *rand*. If *rs* is smaller than *rand*, all particles have an initialized value and all parameters are reset. But in each restart, the best global fitness is saved. It can be seen that the restart probability decreased with the iteration time rising because *rs* is increased by each iteration.

*3.3.2. Fitness Function.* In this optimization problem, there are only three inequality constraints. Based on the standard PSO and recommended constraints handling technique, the constraints in (13) and (14) can be substituted with the milling model. The fitness function is shown as follows:

$$\text{Fit} = T_{pr} + m_2\left[\max\left(0, \left(\frac{F_c V}{6120} - P_m\eta\right)\right)^2 + \max\left(0, \left(F_c - F_s\right)\right)^2 + \max\left(0, \left(F_c - F_d\right)\right)^2\right]. \tag{15}$$

According to (2), the fitness function above is a quaternion function (about $N_p$, $\alpha_i$, $V_i$, $f_{zi}$). So, this is a four-dimension optimization problem. Nevertheless, the optimization results of $N_{\text{p}}$ and $\alpha_{\text{i}}$ cannot always be implemented in practice, because the removal of each pass $(\alpha_{\text{i}})$ depends on the machining accuracy of the machine. And the number of passes of the cutting process must be an integer. For the reasons above, the two parameters, $N_{\text{p}}$ and $\alpha_{\text{i}}$, are optimized by another strategy.

There are some frequently used machining options of $N_{\text{p}}$ and $\alpha_{\text{i}}$. $\alpha_{\text{i}}$ is defined in the range of $(\alpha_{\min}, \alpha_{\max})$. Then the $N_p$ is fixed by the total cut depth $(\alpha)$ and $\alpha_i$, $N_{\text{p}} = \alpha/\alpha_{\text{i}}$. Usually, four or five strategies are implemented to obtain the final best results. The set of $N_p$ and $\alpha_i$ that obtain the minimum machining time is considered as the optimal parameters.

The representation of this milling problem by particles is shown in Figure 1.

*3.3.3. Computational Process.* In this study, the programming language is Microsoft Visual *C*++ and the flow chart of the proposed PSO is shown in Figure 2.

*3.3.4. Complexity of the Proposed Method.* Assuming the population size is *K*, the maximum iteration times is *L* (*maximum iteration times*), the maximum repetition times is *R* (*maximum repeat times*), and the problem size is *N* (*N parameters need to be optimized*), and the computational complexity is analyzed as follows: at each iteration, problem modeling, fitness calculation, optimal state selection of

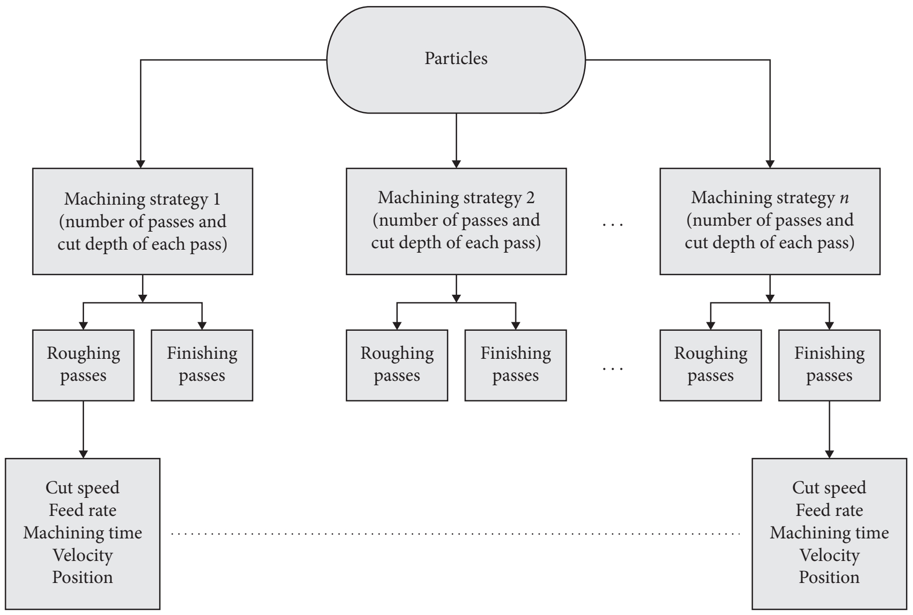
Particles
Machining strategy 1 (number of passes and cut depth of each pass)
Machining strategy 2 (number of passes and cut depth of each pass)
. . .
Machining strategy *n* (number of passes and cut depth of each pass)
Roughing passes
Finishing passes
Cut speed
Feed rate
Machining time
Velocity
Position

FIGURE 1: The representation of a solution to the milling parameters optimization problem.

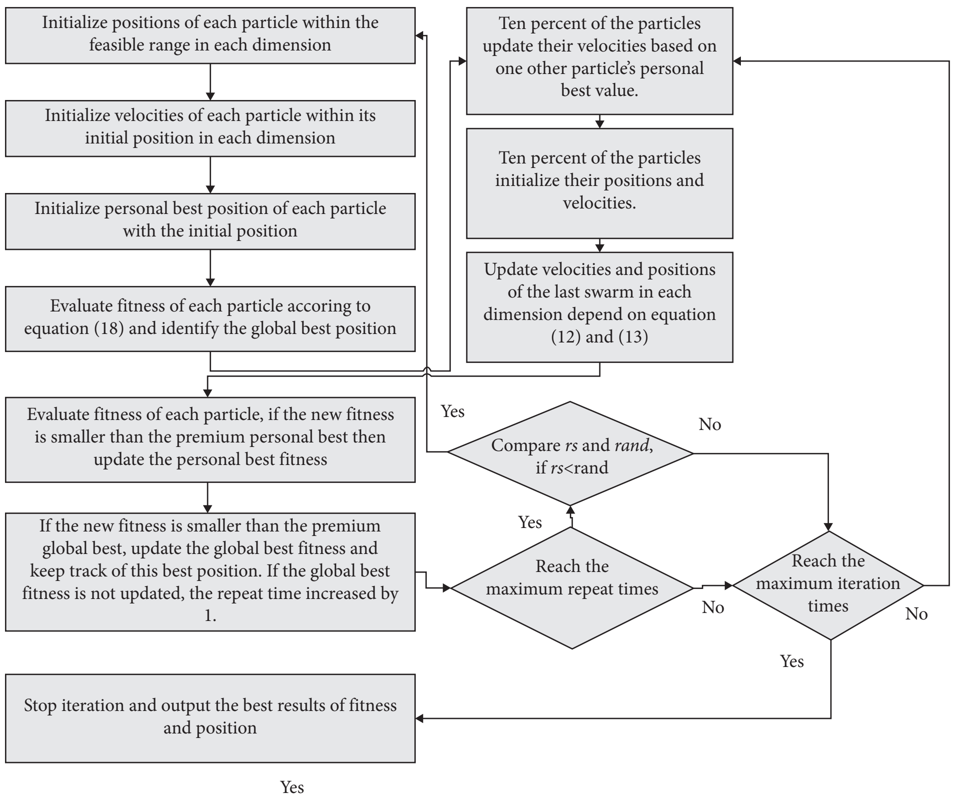
Initialize positions of each particle within the feasible range in each dimension
Initialize velocities of each particle within its initial position in each dimension
Initialize personal best position of each particle with the initial position
Evaluate fitness of each particle accoring to equation (18) and identify the global best position
Evaluate fitness of each particle, if the new fitness is smaller than the premium personal best then update the personal best fitness
If the new fitness is smaller than the premium global best, update the global best fitness and keep track of this best position. If the global best fitness is not updated, the repeat time increased by 1.
Stop iteration and output the best results of fitness and position
Ten percent of the particles update their velocities based on one other particle's personal best value.
Ten percent of the particles initialize their positions and velocities.
Update velocities and positions of the last swarm in each dimension depend on equation (12) and (13)
Compare *rs* and *rand*, if *rs*<rand
Reach the maximum repeat times
Reach the maximum iteration times
Yes
No
Yes
No
No
Yes
Yes

FIGURE 2: The flowchart of the proposed PSO.

individual and population, speed calculation, and state update are required. Therefore, the time complexity of the proposed method is

$$O(L,K,R,N) = O\left(K * \left(KN + (R-1) * (L-1) * 0.1 * N^2\right)\right) \approx O\left(K^2\right) + O\left(KRLN^2\right). \quad (16)$$

When the size of the problem is much larger than the population size, the time complexity of this method can be simplified as $K * R * L * O(N^2)$; that is, the calculation amount is proportional to the square of the size of the research problem and is proportional to the population size ($K$), the maximum number of repeats ($R$), and the maximum number of iterations ($L$).

## 4. Experiment Study

An application example is adopted to evaluate the performance of the proposed PSO. The parameters, the constants, and exponents' values are specified as

Type of milling: plain milling.

Motor power: $P_w = 5.5\text{KW}$, efficiency $\eta = 0.7$.

Arbor diameter $d_a = 27\text{mm}$, arbor length between supports $L_\alpha = 210mm$.

Permissible bending and torsional stress of arbor: $k_b = 140MPa = 14.27\text{kg/mm}^2$, $k_t = 120\text{MPa} = 12.23\text{kg/mm}^2$.

Modulus of elasticity of arbor material $E = 200GPa = 20387\text{kg/mm}^2$.

Spindle speed range: $(31.5 \sim 2000)$ rpm, feed rate range: $(14 \sim 900)$ mm/min.

Tool material: HSS; tool diameter: $D = 63\text{mm}$; number of teeth $Z = 8$.

Workpiece material: structural carbon steel ($C \leq 0.6\%$).

Tensile strength: 750 MPa; Brinell hardness number: 150.

Length, width, depth of cut: $L = 160\text{mm}$, $\alpha_w = 50\text{mm}$, $\alpha = 5\text{mm}$.

Loading and unloading time of one work piece $T_L = 1.5\text{min}$.

Set up time of fixtures and machine tool $T_s = 10\text{min}$; tool change time $T_c = 5\text{min}$.

Process adjusting and quick return time $T_\alpha = 0.1\text{min/part}$.

Lot size $N_b = 100$; cutting inclination: $30^\circ$.

The constants and exponents used in the example are listed as follows:

$C_v = 35.4$, $m = 0.33$, $b_v = 0.45$, $e_v = 0.3$, $B_m = 1.0$, $B_h = 1$, $B_p = 0.8$, $B_t = 0.8$, $u_v = 0.4$, $r_v = 0.1$, $n_v = 0.1$, $q_v = 0$, $C_{zp} = 68.2$, $b_z = -0.86$, $e_z = 0.86$ and $u_z = 0.72$

$\alpha_{\min}$ and $\alpha_{\max}$ are 0.5 mm, 4 mm for the milling operation, so $0.5 \leq \alpha_i \leq 4$

Calculate the boundary of $f_z$ (feed per teeth) and $V$ (feed speed) based on (13) and (14). The following equations are obtained:

$$0.00875\left(\frac{\text{mm}}{\text{tooth}}\right) \leq f_z \leq 3.571\left(\frac{\text{mm}}{\text{tooth}}\right),$$
$$6.234\left(\frac{m}{\text{min}}\right) \leq V \leq 395.84\left(\frac{m}{\text{min}}\right). \tag{17}$$

The maximum iteration number is set as 40, the population of particles is 20, inertia weight ($\omega$) factor is 0.9, acceleration coefficients $c_1$, $c_2$ are 1.05, and the punishment factor $\delta$ is 50. All the hyperparameters, such as the $c_1$ and $c_2$, are selected based on preliminary experiments, and the experiment results are not sensitive with a feasible selection range. Moreover, the preliminary experiment also suggests that the hyperparameter in this experiment will provide an acceptable result as well. But if it wants to exploit the best performance of the proposed method, a fined search for the hyperparameter is more feasible.

For this simplified two-dimension problem, the width of each dimension is (3.571–0.000875) and (395.84–6.234). It can be seen that the two ranges differ so much. So, the initial speeds are assigned with the initial position coordinates of particles rather than be set at the 10~20% of the dynamic range traditionally. The searchability of particles is balanced in two dissimilar dimensions by the recommended assignment.

Four typical machining strategies with different $N_p$ and $\alpha_i$ are chosen. And the different strategies to accomplish the total 5 mm removal are tabulated in Table 2.

Some other methods are implemented to have a comparison with the proposed PSO. Geometric programming (GP) is one of the typical traditional dynamic programming methods. Artificial bee colony (ABC) and simulated annealing (SA) are kinds of soft computing methods that are researched mostly in recent years. All the comparison methods are the advanced methods of multipass milling process, and for a fair comparison, the proposed improved PSO (IPSO) is compared with these methods directly without any extra design or strategies. The purpose of the comparison is to evaluate if the proposed method is effective for multipass milling process.

Table 3 shows the four groups of computational results of IPSO and ABC. IPSO is superior to ABC in all four strategies. The production time of one part is shorted by 0.038 minutes, especially in machining strategy 1. The total machining time of a batch is shortened by $0.038 \times 100 = 38$ minutes.

The computational results obtained by GP are tabulated in Table 4. Also, Table 4 gives the results of SA in the commonly used strategy 2. The final production time demonstrates the superiority of IPSO to GP and SA. IPSO cut the production time of one part by 0.031 minutes contrasted to SA. And for GP, production time decreases by almost one minute.

IPSO has a self-adapting feature that the particles follow the local best and global best position by adjusting their velocities. The two best positions are updated by fitness estimation and affect the direction of swarms in the next iteration. Particles continually gather to the near-optimal area by conducting two ongoing comparisons. IPSO has efficient search ability due to its active inner learning structure.

ABC gains results approximate to IPSO in Table 3 but ABC algorithm is so complicated in the concept of foragers, and unemployed foragers consist of scout bee and recruit, employed foragers, and experienced foragers [12, 30]. SA is a kind of probabilistic hill-climbing computing algorithm and is suitable for combinatorial optimization problems or nonlinear response functions. According to Table 4, the results obtained by SA are inferior to the results gained by IPSO. Moreover, SA is time-consuming. With about 100 iterations, SA can obtain the final best results [5]. Unlike IPSO, SA uses Metropolis strategy to update the current best solution and only when the annealing time is long enough, SA can provide good results. When compared to GP which is

Table 2: Four machining strategy options.

| Machining strategy | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| **Roughing operation** | $\alpha_{\text{rough}} = 2$<br>$\alpha_{\text{rough}} = 2$ | $\alpha_{\text{rough}} = 1.5$<br>$\alpha_{\text{rough}} = 1.5$<br>$\alpha_{\text{rough}} = 1.5$ | $\alpha_{\text{rough}} = 2$<br>$\alpha_{\text{rough}} = 1$<br>$\alpha_{\text{rough}} = 1$ | $\alpha_{\text{rough}} = 1$<br>$\alpha_{\text{rough}} = 1$<br>$\alpha_{\text{rough}} = 1$<br>$\alpha_{\text{rough}} = 1$ |
| **Finishing operation** | $\alpha_{\text{finish}} = 1$ | $\alpha_{\text{finish}} = 0.5$ | $\alpha_{\text{finish}} = 1$ | $\alpha_{\text{finish}} = 1$ |

Table 3: Computational results of PSO and ABC.

| Method | | Results of IPSO | | | | Results of ABC | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| S. no. | Machining strategy | $f_z$ (mm/tooth) | $V$ (m/min) | $T_2$ (min) | **Production time** $T_1 + T_2$ | $f_z$ (mm/tooth) | $V$ (m/min) | $T_2$ (min) | **Production time** $T_1 + T_2$ | $T_1$ (min) |
| 1 | $\alpha_{\text{rough}} = 2$<br>$\alpha_{\text{rough}} = 2$<br>$\alpha_{\text{finish}} = 1$ | 0.242<br>0.242<br>0.190 | 53.53<br>53.53<br>72.61 | 0.456<br>0.456<br>0.428 | **3.24** | 0.231<br>0.231<br>0.189 | 48.117<br>48.117<br>74.090 | 0.475<br>0.475<br>0.428 | **3.278** | 1.9 |
| 2 | $\alpha_{\text{rough}} = 1.5$<br>$\alpha_{\text{rough}} = 1.5$<br>$\alpha_{\text{rough}} = 1.5$<br>$\alpha_{\text{finish}} = 0.5$ | 0.341<br>0.341<br>0.341<br>0.435 | 50.86<br>50.86<br>50.86<br>64.19 | 0.340<br>0.340<br>0.340<br>0.212 | **3.232** | 0.337<br>0.337<br>0.337<br>0.432 | 46.982<br>46.982<br>46.982<br>64.41 | 0.343<br>0.343<br>0.343<br>0.211 | **3.240** | 2.0 |
| 3 | $\alpha_{\text{rough}} = 2$<br>$\alpha_{\text{rough}} = 1$<br>$\alpha_{\text{rough}} = 1$<br>$\alpha_{\text{finish}} = 1$ | 0.242<br>0.554<br>0.554<br>0.190 | 53.53<br>47.33<br>47.33<br>72.61 | 0.456<br>0.225<br>0.225<br>0.428 | **3.334** | 0.231<br>0.552<br>0.552<br>0.189 | 48.117<br>47.519<br>47.519<br>74.090 | 0.475<br>0.226<br>0.226<br>0.428 | **3.355** | 2.0 |
| 4 | $\alpha_{\text{rough}} = 1$<br>$\alpha_{\text{rough}} = 1$<br>$\alpha_{\text{rough}} = 1$<br>$\alpha_{\text{rough}} = 1$<br>$\alpha_{\text{finish}} = 1$ | 0.554<br>0.554<br>0.554<br>0.554<br>0.190 | 47.33<br>47.33<br>47.33<br>47.33<br>72.61 | 0.225<br>0.225<br>0.225<br>0.225<br>0.428 | **3.428** | 0.552<br>0.552<br>0.552<br>0.552<br>0.189 | 47.519<br>47.519<br>47.519<br>47.519<br>74.090 | 0.226<br>0.226<br>0.226<br>0.226<br>0.428 | **3.432** | 2.1 |

Table 4: Computational results of IPSO, SA, and GP.

| Machining strategy | | $\mathbf{f_z}$ (mm/tooth) | **V** (m/min) | $\mathbf{T_{pr}} = \mathbf{T_1} + \mathbf{T_2}$ (min) |
|---|---|---|---|---|
| PSO | $\alpha_{\text{rough}} = 1.5$<br>$\alpha_{\text{rough}} = 1.5$<br>$\alpha_{\text{rough}} = 1.5$<br>$\alpha_{\text{finish}} = 0.5$ | 0.341<br>0.341<br>0.341<br>0.435 | 50.86<br>50.86<br>50.86<br>64.19 | **3.232** |
| SA | $\alpha_{\text{rough}} = 1.5$<br>$\alpha_{\text{rough}} = 1.5$<br>$\alpha_{\text{rough}} = 1.5$<br>$\alpha_{\text{finish}} = 0.5$ | 0.336<br>0.336<br>0.336<br>0.429 | 44.633<br>44.633<br>44.633<br>57.23 | **3.263** |
| GP | $\alpha_{\text{rough}} = 3$<br>$\alpha_{\text{finish}} = 2$ | 0.338<br>0.57 | 26.4<br>25.16 | **4.205** |

one type of traditional optimization strategy, IPSO is better than the results of GP. Traditional methods alone cannot give a robust result of the multiconstraints nonlinear programming optimization problem.

## 5. Conclusion and Future Work

Process planning of the multipass milling process is crucial for the determination of the cutting parameters. The selection of the passes, depth of cut, cut speed, and feed per teeth have a great impact on the production time greatly. The cost and final quality of products are also affected by the recommended parameters. In this study, the multipass machining process is modeled on basis of the proposed PSO which is one kind of swarm intelligence algorithm. Its search mechanism of going after the local best and global best positions ensures the convergence of particles.

The penalty function method is used as the constraints handling technique. The constraints of the arbor strength, the arbor deflection, and the cutting power are satisfied by the punishment function method. An experimental example is presented to test the proposed IPSO. The computational results of IPSO are superior to the results gained by ABC, SA, and GP. Meanwhile, IPSO has a relatively high convergence rate.

In this study, IPSO is successfully used in milling parameters optimization problems, but it has not been frequently tested in other metal machining parameters optimization problems such as drilling, grinding, and turning. In future work, the searching ability and convergence rate of IPSO will be tested by increasing the dimensions of the optimization problem. Furthermore, for a complicated machining process with different machining operations, this will largely decrease the total process time and increase production. Besides, hybridizing IPSO with other optimization algorithms to enhance its performance is another research direction.

## Data Availability

The data are from the related work, and it has been illustrated in the experimental parts.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This research was supported by the National Key Technology Support Program under Grant no. 2015BAF01B04 and Key R&D Program of Huber Province under Grant no. 2021AAB001.

## References

[1] Y. Gao, X. Li, X. V. Wang, L. Wang, and L. Gao, “A review on recent advances in vision-based defect recognition towards industrial intelligence,” *Journal of Manufacturing Systems*, vol. 62, pp. 753–766, 2022.

[2] L. Ma, N. Li, Y. Guo et al., “Learning to optimize: reference vector reinforcement learning adaption to constrained many-objective optimization of industrial copper burdening system,” *IEEE Transactions on Cybernetics*, pp. 1–14, 2021.

[3] Y. Gao, L. Gao, X. Li, and X. Yan, “A semi-supervised convolutional neural network-based method for steel surface defect recognition,” *Robotics and Computer-Integrated Manufacturing*, vol. 61, Article ID 101825, 2020.

[4] Y. Zhou, T. Xing, Y. Song et al., “Digital-twin-driven geometric optimization of centrifugal impeller with free-form blades for five-axis flank milling,” *Journal of Manufacturing Systems*, vol. 58, pp. 22–35, Jan. 2021.

[5] R. Venkata Rao and P. J. Pawar, “Parameter optimization of a multi-pass milling process using non-traditional optimization algorithms,” *Applied Soft Computing*, vol. 10, no. 2, pp. 445–456, 2010.

[6] M. Tolouei-Rad and I. M. Bidhendi, “On the optimization of machining parameters for milling operations,” *International Journal of Machine Tools and Manufacture*, vol. 37, no. 1, pp. 1–16, 1997.

[7] P. G. Petropoulos, “Optimal selection of machining rate variables by geometric programming,” *International Journal of Production Research*, vol. 11, no. 4, pp. 305–314, 1973.

[8] K. Vijayakumar, G. Prabhaharan, P. Asokan, and R. Saravanan, “Optimization of multi-pass turning operations using ant colony system,” *International Journal of Machine Tools and Manufacture*, vol. 43, no. 15, pp. 1633–1639, 2003.

[9] P. Palanisamy, I. Rajendran, and S. Shanmugasundaram, “Optimization of machining parameters using genetic algorithm and experimental validation for end-milling operations,” *International Journal of Advanced Manufacturing Technology*, vol. 32, no. 7-8, pp. 644–655, 2007.

[10] V. Gaikhe, J. Sahu, and R. Pawade, “Optimization of cutting parameters for cutting force minimization in helical ball end milling of inconel 718 by using genetic algorithm,” *Procedia CIRP*, vol. 77, pp. 477–480, 2018.

[11] Z. G. Wang, Y. S. Wong, and M. Rahman, “Optimisation of multi-pass milling using genetic algorithm and genetic simulated annealing,” *International Journal of Advanced Manufacturing Technology*, vol. 24, no. 9-10, pp. 727–732, 2004.

[12] D. Karaboga and B. Basturk, “On the performance of artificial bee colony (ABC) algorithm,” *Applied Soft Computing*, vol. 8, no. 1, pp. 687–697, 2008.

[13] A. Baykasoğlu, “Optimising cutting conditions for minimising cutting time in multi-pass milling via weighted superposition attraction-repulsion (WSAR) algorithm,” *International Journal of Production Research*, vol. 59, no. 15, pp. 4633–4648, 2021.

[14] J. Li, Y. Sun, and S. Hou, “Particle swarm optimization algorithm with multiple phases for solving continuous optimization problems,” *Discrete Dynamics in Nature and Society*, vol. 2021, pp. 1–13, 2021.

[15] L. Ma, M. Huang, S. Yang, R. Wang, and X. Wang, “An adaptive localized decision variable analysis approach to large-scale multiobjective and many-objective optimization,” *IEEE Transactions on Cybernetics*, vol. 52, no. 7, pp. 6684–6696, 2022.

[16] B. Song, Z. Wang, and L. Zou, “An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve,” *Applied Soft Computing*, vol. 100, Article ID 106960, 2021.

[17] J. Zeng, B. Roy, D. Kumar et al., “Proposing several hybrid PSO-extreme learning machine techniques to predict TBM performance,” *Engineering with Computers*, 2021.

[18] Eberhart and Y. Shi, “Particle swarm optimization: developments, applications and resources,” in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, vol. 1, pp. 81–86, Seoul, Korea (South), May 2001.

[19] L. Han, K. Xing, X. Chen, and F. Xiong, “A Petri net-based particle swarm optimization approach for scheduling deadlock-prone flexible manufacturing systems,” *Journal of Intelligent Manufacturing*, vol. 29, no. 5, pp. 1083–1096, 2018.

[20] X. Tang, C. Shi, T. Deng, Z. Wu, and L. Yang, “Parallel random matrix particle swarm optimization scheduling algorithms with budget constraints on cloud computing systems,” *Applied Soft Computing*, vol. 113, Article ID 107914, 2021.

[21] Z. Yang, C. Liu, X. Wang, and W. Qian, “An improved multiobjective PSO for the scheduling problem of panel block construction,” *Discrete Dynamics in Nature and Society*, vol. 2016, Article ID 5413520, 13 pages, 2016.

[22] Y. Sun and H. Ren, “A GD-PSO algorithm for smart transportation supply chain ABS portfolio optimization,” *Discrete Dynamics in Nature and Society*, vol. 2021, Article ID e6653051, 9 pages, 2021.

[23] B. R. C. Karuppanan and M. Saravanan, “Optimized sequencing of CNC milling toolpath segments using metaheuristic algorithms,” *Journal of Mechanical Science and Technology*, vol. 33, no. 2, pp. 791–800, 2019.

[24] R. Naik and N. Sathisha, “Desirability function and GA-PSO based optimization of electrochemical discharge micro-machining performances during micro-channeling on silicon-wafer using mixed electrolyte,” *Silicon*, 2022.

[25] Y. Fang, L. Zhao, P. Lou, and J. Yan, “Cutting parameter optimization method in multi-pass milling based on improved adaptive PSO and SA,” *Journal of Physics*, vol. 1848, no. 1, Article ID 012116, 2021.

[26] Y. Fan, P. Wang, A. A. Heidari, H. Chen, T. Hamza, and M. Mafarja, “Random reselection particle swarm optimization for optimal design of solar photovoltaic modules,” *Energy*, vol. 239, Article ID 121865, 2022.

[27] A. İ. Sönmez, A. Baykasoğlu, T. Dereli, and İ. H. Fılız, “Dynamic optimization of multipass milling operations via geometric programming,” *International Journal of Machine Tools and Manufacture*, vol. 39, no. 2, pp. 297–320, 1999.

[28] W. a. Yang, Y. Guo, and W. Liao, “Multi-objective optimization of multi-pass face milling using particle swarm intelligence,” *International Journal of Advanced Manufacturing Technology*, vol. 56, no. 5-8, pp. 429–443, 2011.

[29] H. Liu, Z. Cai, and Y. Wang, “Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization,” *Applied Soft Computing*, vol. 10, no. 2, pp. 629–640, 2010.

[30] D. Karaboga and B. Basturk, “A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm,” *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.