applied sciences

MDPI

Article

Controllability of Fractional-Order Particle Swarm Optimizer and Its Application in the Classification of Heart Disease

Fu-I Chou [1], Tian-Hsiang Huang [2], Po-Yuan Yang [3], Chin-Hsuan Lin [1], Tzu-Chao Lin [4], Wen-Hsien Ho [5,6,7,*] and Jyh-Horng Chou [1,5,8,*]

[1] Department of Electrical Engineering, National Kaohsiung University of Science and Technology, Kaohsiung 807, Taiwan; ryan.chou.0110.1111@nkust.edu.tw (F.-I.C.); migratorwing@gmail.com (C.-H.L.)
[2] Department of Computer Science and Information Engineering, National Penghu University of Science and Technology, Penghu 880, Taiwan; huangtx@gmail.com
[3] Department of Intelligent Robotics, National Pingtung University, Pingtung 900, Taiwan; pyyang@mail.nptu.edu.tw
[4] Department of Neurology, Kaohsiung Medical University Hospital, Kaohsiung 807, Taiwan; prochristy@gmail.com
[5] Department of Healthcare Administration and Medical Informatics, Kaohsiung Medical University, Kaohsiung 807, Taiwan
[6] Department of Medical Research, Kaohsiung Medical University Hospital, Kaohsiung 807, Taiwan
[7] Department of Mechanical Engineering, National Pingtung University of Science and Technology, Pingtung 912, Taiwan
[8] Department of Mechanical and Computer-Aided Engineering, Feng-Chia University, Taichung 407, Taiwan
* Correspondence: whho@kmu.edu.tw (W.-H.H.); choujh@nkust.edu.tw (J.-H.C.)

Citation: Chou, F.-I.; Huang, T.-H.; Yang, P.-Y.; Lin, C.-H.; Lin, T.-C.; Ho, W.-H.; Chou, J.-H. Controllability of Fractional-Order Particle Swarm Optimizer and Its Application in the Classification of Heart Disease. *Appl. Sci.* **2021**, *11*, 11517. https://doi.org/10.3390/app112311517

Academic Editor: Minvydas Ragulskis

Received: 14 November 2021
Accepted: 3 December 2021
Published: 5 December 2021

Abstract: This study proposes a method to improve fractional-order particle swarm optimizer to overcome the shortcomings of traditional swarm algorithms, such as low search accuracy in a high-dimensional space, falling into local minimums, and nonrobust results. In natural phenomena, our controllable fractional-order particle swarm optimizer can explore search spaces in detail to obtain high resolutions. Moreover, the proposed algorithm is memorable, i.e., position updates focus on the particle position of previous and last generations, rendering it conservative when updating the position, and obtained results are robust. For verifying the algorithm's effectiveness, 11 test functions compare the average value, overall best value, and standard deviation of the controllable fractional-order particle swarm optimizer and controllable particle swarm optimizer; experimental results show that the stability of the former is better than the latter. Furthermore, the solution position found by the controllable fractional-order particle swarm optimizer is more reliable. Therefore, the improved method proposed herein is effective. Moreover, this research describes how a heart disease prediction application uses the optimizer we proposed to optimize XGBoost hyperparameters with custom target values. The final verification of the obtained prediction model is effective and reliable, which shows the controllability of our proposed fractional-order particle swarm optimizer.

Keywords: optimization method; fractional derivative; fractional-order particle swarm optimizer; XGBoost

1. Introduction

Optimization methods involve automatically finding the best solution in a problem's solution space set. When converting a real problem into a mathematical model, simulating the actual physical characteristics of the problem requires a detailed description of the conversion process. Moreover, the mathematical model of the problem becomes more complex. Currently, three methods exist for solving optimization problems: numerical method, enumerative, and random search.

Numerical methods use the derivative as a technique to find the best value in the space. For example, the traditional neural network series of algorithms is based on this method's gradient descent to find the best parameters [1]. However, the numerical method has two shortcomings. First, it searches for the best solution from a local point of view, so there is no guarantee that the solution found is globally optimal. Second, numerical methods are not applicable for search spaces that are not smooth or continuous [2,3]. However, because usually many regional optimal solutions exist in a search space that is not smooth or continuous, it is easy to converge early in the search process and find the optimal local solution.

The enumeration method, such as grid search, uses the objective function to test all solutions in the search space at that level when the segmentation level is selected. This method has a better chance of obtaining the best solution, but it requires considerable computation time. Therefore, when the search space is ample, the enumeration method is inefficient. The random search method is currently a commonly used optimization method, which finds the best solution space by imitating natural or biological behavior, and particle swarm optimization (PSO) [4] is one of the optimization methods that imitate biological behavior.

PSO is an intelligent swarm algorithm that observes the behavior of swarm creatures. This method is used to find the best position in the current space. After PSO was published, many scholars proposed different methods to improve the algorithm. These methods have been applied in many fields [5–7].

For example, Shi and Eberhart [8] proposed a constant inertia weight to improve the moving direction of particles. Different inertial weights make it possible to find a balance between local and global searches. Thus, the algorithm considers the best solution in the whole domain. Shi and Eberhart [9] again proposed linearly decreasing inertia weights. In the same year, Suganthan [10] proposed a linear decrease with dynamic characteristics applied to individual learning parameters c_1 and group learning parameters c_2, which effectively improved global search. Clerc [11,12] put forward the concept of the shrinkage coefficient, and its idea is to change the moving direction of particles to increase local search. Shi and Eberhart [13] proposed the maximum speed method to improve search. Ratnaweera et al. [14] improved the method proposed by Suganthan by changing the swarm learning parameter from linearly decreasing to linearly increasing. Chatterjee and Siarry [15] proposed to change inertia weights in a nonlinearly decreasing way. Ko et al. [16] extended the concept of nonlinear change to individual and group learning parameters. They changed the individual learning parameters to nonlinear decreasing individual learning parameters and the group learning parameters to nonlinear increasing group learning parameters.

With many scholars proposing methods to improve the original algorithm, the particle swarm algorithm search has been dramatically improved. Since 2002, Clerc and Kennedy [12] have used the dynamic system notation in control theory to explore the internal operation of the particle swarm algorithm. Many scholars have also proposed the stability of particle swarm algorithms under different conditions based on a dynamic system representation [17–20]. In particular, in 2014, Lin [21] proposed the PSO algorithm with controllability. The method explores the particle swarm algorithm from the viewpoint of state controllability in dynamic systems. When the controllable conditions are met, the particle swarm algorithm's position and velocity vectors are controlled by its own best solution position vector and the global best solution position vector; this makes the convergence better than in the original particle swarm algorithm.

However, the searchability of the integer-order particle swarm algorithm proposed by Lin [21] is poor and is unstable in high-dimensional or complex spaces. Therefore, this study suggests combining fractional-order particle swarm optimizer and PSO algorithm with controllability, which is called "the controllable fractional-order particle swarm algorithm". We can use the proposed algorithm to improve the PSO algorithm with controllability such that it performs better in high-dimensional or complex spaces.

We need to optimize hyperparameters efficiently and systematically for machine learning algorithms. Therefore, this study applies the "controllable fractional-order particle swarm algorithm" to optimize machine learning hyperparameters. We demonstrated how to efficiently and systematically find hyperparameters in extreme gradient boosting (XGBoost) [22] machine learning algorithms using our recommended method. Further, we used the heart disease data set downloaded from the UCI website. The six best hyperparameters found using our recommended method and the hyperparameters officially recommended by XGBoost [23] were trained and tested, and data sets were compared. Experimental results showed that the recommended method had better performance, and no false-negative results were produced. This method can help physicians quickly determine whether a patient has heart disease using the learned model.

2. Materials and Methods

The particle swarm algorithm was initially inspired by Kennedy and Eberhart [4] by observing the foraging behavior of birds. It is a kind of mimic optimization algorithm with the concept of swarm intelligence. Suppose there is a flock of birds randomly scattered in a space where food exists, and there are many food piles of different sizes in the space. Then, the largest food pile is the best position (P^g) in this space. First, each bird starts to search for food piles at a random location, searches for routes with its own experience, and records the largest food pile (P_i^l) that it has ever searched. When a particular bird finds a better food pile than all the current bird flocks, it will notify the other bird flocks to move toward the best food pile. Therefore, the following search route of each bird will be affected by three factors: the direction in its own experience (the direction of its own speed), the current direction of finding the best food pile position by itself (the direction of its own best solution), and best food pile position direction found among all the flocks (the best solution direction in the whole domain).

2.1. Particle Swarm Algorithm

In the theory of particle swarm algorithm, each bird in the space is regarded as a particle in the "solution space". The current position of each particle is considered to be a solution to the optimization problem of the solution space, and each solution corresponds to an answer, which is called the objective function value or fitness value of the solution space. Each particle has its own speed (V_i) and uses its own speed direction, the best solution (*pbest*) currently found by itself, and the best solution (*gbest*) found by the current group to generate a new particle speed. After determining the update speed and direction of the particles, the conditions are used to generate a new update position. Subsequently, the value of the objective function is brought into the position of each particle to judge the pros and cons of the current position. If it is better than the previously searched solution, replace it. Otherwise, keep the original best solution. We used this mechanism to iteratively search in the solution space to find the best solution in the space and used the following Equations (1) and (2) to express the search mechanism for finding the best solution:

$$V_i(k+1) = V_i(k) + c_1 r_1 \left(P_i^l(k) - p_i(k)\right) + c_2 r_2 (P^g(k) - p_i(k)) \tag{1}$$

$$p_i(k+1) = p_i(k) + V_i(k+1) \tag{2}$$

where $i = 1, 2, \ldots, m$ (m denotes the number of particles); k represents the iteration index; $V_i(k)$ denotes the velocity vector of the i-th particle in the θ dimension; $p_i(k)$ represents the position of the i-th particle in the θ dimension vector; $P_i^l(k)$ denotes the position vector of the best solution θ dimension of each iteration; $P^g(k)$ represents the position vector of the best solution θ dimension of the group; c_1 denotes the individual learning parameter; c_2 represents the group learning parameter; r_1 and r_2 denote random numbers between 0 and 1; θ represents the dimensionality of the search space.

The individual learning parameter c_1 and the group learning parameter c_2 represent the acceleration weights of the best solution and the best solution of the group that advances the particle to each iteration, respectively. When the value of c is small, the particle is allowed to perform multiple searches near the target area before reaching the best solution of its own or the best solution of the group in each iteration. This increases the probability of finding the best solution in the entire domain but at the cost of more computation power and time. When the value of c is large, the particles are allowed to reach their own best solution or the best solution of the group at a faster speed in each iteration. This will save some unnecessary calculations and time and improve the convergence speed. Moreover, when the value c_1 or c_2 is 0, the particle swarm algorithm will have different characteristics [24].

The first part of Equation (1) is the particle's previous inertia, i.e., the velocity of its previous experience. The second part is the "cognition" part, which represents the thinking of the particle itself. Finally, the third part is the "social" part, which implies that the information among particles is shared such that the particles can cooperate. Therefore, the core of the particle swarm algorithm is to use these three parts to update the particle speed and position in a linear combination and to calculate the fitness value to complete the problem optimization.

2.2. Fractional-Order Particle Swarm Algorithm

Fractional calculus is derived from traditional calculus [25]. For example, Equation (3) represents fractional differentiation based on the Grünwald–Letnikov definition, where D stands for the differential operator, λ denotes a fractional-order power, and Γ is the Euler function. According to Solteiro Pires et al. [26], if h is expressed in discrete terms, it can be approximated as Equation (4).

$$D^{\lambda}[x(t)] = \lim_{h\to 0}\left[\frac{1}{h^{\lambda}}\sum_{k=0}^{+\infty}\frac{(-1)^{k}\Gamma(\lambda+1)x(t-kh)}{\Gamma(k+1)\Gamma(\lambda-k+1)}\right] \tag{3}$$

$$D^{\lambda}[x(t)] = \frac{1}{T^{\lambda}}\sum_{k=0}^{r}\frac{(-1)^{k}\Gamma(\lambda+1)x(t-kT)}{\Gamma(k+1)\Gamma(\lambda-k+1)} \tag{4}$$

where T denotes the sampling period, and r represents the truncation order.

In contrast to the integer-order derivative as a finite series, the fractional-order derivative requires an infinite number of terms. This implies that the information obtained using the fractional order is more global than the integer-order differentiation. Therefore, the solution space that can be explored for the fraction order is more refined than the integer order, and it is expected to obtain better solution space accuracy. Further, the fractional differentiation allows the particle swarm algorithm to memorize the position. Therefore, the velocity vector is affected by the positions of the previous and last generations. This makes the fractional-order particle swarm algorithm more conservative in the search process, making the solution space results more similar and stable every time.

If r = 4 is used as an example, the speed and position vectors are updated with the following Equations (5) and (6).

$$\begin{gathered} V_i(k+1) = V_i(k) + c_1{\cdot}r_1\Big[P_i^l(k) - \lambda{\cdot}p_i(k) - \tfrac{\lambda}{2}(1-\lambda)p_i(k-1) \\ -\tfrac{1}{6}(1-\lambda)(2-\lambda)p_i(k-2) - \tfrac{1}{24}(1-\lambda)(2-\lambda)(3-\lambda)p_i(k-3)\Big] + c_2{\cdot}r_2 \\ {\cdot}(P^g(k) - p_i(k)) \end{gathered} \tag{5}$$

$$p_i(k+1) = p_i(k) + V_i(k+1) \tag{6}$$

To improve the point of the direction from which the particles are searched, Shi and Eberhart [8] added an inertia weight term ω to the velocity $V_i(k)$ to facilitate the contribution of the particle itself to the update velocity. When ω is large, the direction of the update speed will depend on the direction of the previous generation speed. At this time, the search direction is more stable, which improves the global search ability of the particles in space. However, when ω is considerably large, overcorrection occurs. Consequently, the particle correction speed is excessively large and deviates from the better solution, resulting in "flying" trajectories.

When ω is small, the update speed direction is dominated by the optimal local solution and the direction of the global optimal solution. At this time, a local search capability is provided. However, because the solution searched by the particle is not explored globally, the obtained solution may not achieve the global best solution due to its locality. Therefore, Shi and Eberhart [9] once again proposed a solution, changing the "constant inertia weight" to a "linearly decreasing inertia weight." When ω is set to a larger value in the initial stage, the particle swarm has a better ability to expand the search to find the best solution area in the whole domain. After the number of iterations increases, the value of ω is gradually reduced. The particle swarm will switch from an extended search to a local search to find a better solution in the best found so far. The formula for changing the "constant term inertia weight ω" to "time-varying linear inertia weight $\omega(k)$" is shown in Equation (7).

$$\omega(k) = \omega_{min} + \frac{iter_{max} - k}{iter_{max}} \times (\omega_{max} - \omega_{min}) \tag{7}$$

where k denotes the number of iterations; ω_{max} represents the maximum value of the inertia weight; ω_{min} denotes the minimum value of the inertia weight; $iter_{max}$ represents the maximum number of iterations.

To avoid excessive velocity exceeding the search space during the particle update, Shi and Eberhart [13] used the maximum velocity method (V_{max} Method) to limit particle velocity and improve particle search capability. Among them, the value of V_{max} cannot be set too large because the particles can have a high speed, which may cause the particles to fly out of the search range. The value of V_{max} cannot be set too small either because the particle swarm will search the space too slowly and thus will not be able to search the global space and is limited to the best solution in a local range. The formula of the maximum speed method is as follows:

$$V_i(k+1) = \begin{cases} V_{max}, & V_i(k+1) > V_{max} \\ -V_{max}, & V_i(k+1) < -V_{max} \end{cases} \tag{8}$$

Among them, this study set V_{max} to be 0.2 times of the maximum search range, i.e., $V_{max} = 0.2 \times X_{max}$.

2.3. Robust and Controllable

Combine the "linearly decreasing inertia weight" of Equation (7) with Equation (5) and expand Equation (6) into Equations (9) and (10).

$$\begin{gathered} V_i(k+1) = \omega(k) \cdot V_i(k) - (c_1 \cdot r_1 \cdot \lambda + c_2 \cdot r_2) p_i(k) - c_1 \cdot r_1 \cdot \tfrac{\lambda}{2}(1-\lambda) p_i(k-1) \\ -c_1 \cdot r_1 \cdot \tfrac{\lambda}{6}(1-\lambda)(2-\lambda) p_i(k-2) - c_1 \cdot r_1 \cdot \tfrac{\lambda}{24}(1-\lambda)(2-\lambda)(3-\lambda) p_i(k-3) \\ +c_1 \cdot r_1 \cdot P_i^l(k) + c_2 \cdot r_2 \cdot P^g(k) \end{gathered} \tag{9}$$

$$\begin{gathered} p_i(k+1) = \omega(k) \cdot V_i(k) + (1 - c_1 \cdot r_1 \cdot \lambda - c_2 \cdot r_2) p_i(k) \\ -c_1 \cdot r_1 \cdot \tfrac{\lambda}{2}(1-\lambda) p_i(k-1) - c_1 \cdot r_1 \cdot \tfrac{\lambda}{6}(1-\lambda)(2-\lambda) p_i(k-2) \\ -c_1 \cdot r_1 \cdot \tfrac{\lambda}{24}(1-\lambda)(2-\lambda)(3-\lambda) p_i(k-3) + c_1 \cdot r_1 \cdot P_i^l(k) + c_2 \cdot r_2 \cdot P^g(k) \end{gathered} \tag{10}$$

Subsequently, rewrite these equations into the state Equation (11):

$$x(k+1) = A_I \cdot x(k) + B_I \cdot u(k) \tag{11}$$

where $x(k) = \left[p_i^T(k), p_i^T(k-1), p_i^T(k-2), p_i^T(k-3),\ V_i^T(k)\right]^T \in R^n$ denotes the system state vector; $u(k) = \left[p_i^{lT}(k), p^{gT}(k)\right]^T \in R^n$ represents the control input vector; $i = 1, 2, \ldots, m$ (m denotes the number of particles) and $n = 5\theta$, while θ represents the dimension for the search space, and k is the iteration index. A_I denotes the system matrix, and B_I represents the input matrix, as shown in the following Equation (12):

$$A_I = \begin{bmatrix} A_0 & A_1 & A_2 & A_3 & A_4 \\ I & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \\ A_5 & A_6 & A_7 & A_8 & A_9 \end{bmatrix} \quad B_I = \begin{bmatrix} c_1 \cdot r_1 \cdot I & c_2 \cdot r_2 \cdot I \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ c_1 \cdot r_1 \cdot I & c_2 \cdot r_2 \cdot I \end{bmatrix} \tag{12}$$

where

$$A_0 = (1 - c_1 \cdot r_1 \cdot \lambda - c_2 \cdot r_2) \cdot I$$
$$A_1 = -c_1 \cdot r_1 \cdot \frac{\lambda}{2}(1-\lambda) \cdot I$$
$$A_2 = -c_1 \cdot r_1 \cdot \frac{\lambda}{6}(1-\lambda)(2-\lambda) \cdot I$$
$$A_3 = -c_1 \cdot r_1 \cdot \frac{\lambda}{24}(1-\lambda)(2-\lambda)(3-\lambda) \cdot I$$
$$A_4 = \omega(k) \cdot I$$
$$A_5 = (-c_1 \cdot r_1 \cdot \lambda - c_2 \cdot r_2) \cdot I$$
$$A_6 = -c_1 \cdot r_1 \cdot \frac{\lambda}{2}(1-\lambda) \cdot I$$
$$A_7 = -c_1 \cdot r_1 \cdot \frac{\lambda}{6}(1-\lambda)(2-\lambda) \cdot I$$
$$A_8 = -c_1 \cdot r_1 \cdot \frac{\lambda}{24}(1-\lambda)(2-\lambda)(3-\lambda) \cdot I$$
$$A_9 = \omega(k) \cdot I$$

and I denotes the $\theta \times \theta$ unit matrix. r_1 and r_2 denote random numbers between 0 and 1. Equations (9) and (10) are equivalent to Equation (11) based on Equation (12) and matrix multiplication rules.

If each pair (A_I, B_I) is controllable, then the state Equation (11) is said to be robust and controllable [27]. Suppose that a fixed fractional value λ, an inertial weight constant reference value ω_0, a volume learning parameter constant reference value c_{10}, and a group learning parameter constant reference value c_{20} are selected as the nominal values of the fractional-order particle swarm algorithm. In that case, the equation of state (11) can be rewritten as an uncertain linear system, i.e., the system will be transformed into a nominal fractional-order particle swarm optimization (FPSO) combined with uncertain matrices:

$$x(k+1) = (A_{I0} + \Delta A) \cdot x(k) + (B_{I0} + \Delta B) \cdot u(k) \tag{13}$$

where

$$A_{I0} = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 \\ I & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \\ a_5 & a_6 & a_7 & a_8 & a_9 \end{bmatrix} \quad B_{I0} = \begin{bmatrix} c_{10} \cdot I & c_{20} \cdot I \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ c_{10} \cdot I & c_{20} \cdot I \end{bmatrix} \tag{14}$$

$$a_0 = (1 - c_{10} \cdot \lambda - c_{20}) \cdot I$$
$$a_1 = -c_{10} \cdot \frac{\lambda}{2}(1-\lambda) \cdot I$$
$$a_2 = -c_{10} \cdot \frac{\lambda}{6}(1-\lambda)(2-\lambda) \cdot I$$

$$a_3 = -c_{10}\cdot\frac{\lambda}{24}(1-\lambda)(2-\lambda)(3-\lambda)\cdot I$$
$$a_4 = \omega(k)\cdot I$$
$$a_5 = (-c_{10}\cdot\lambda - c_{20})\cdot I$$
$$a_6 = -c_{10}\cdot\frac{\lambda}{2}(1-\lambda)\cdot I$$
$$a_7 = -c_{10}\cdot\frac{\lambda}{6}(1-\lambda)(2-\lambda)\cdot I$$
$$a_8 = -c_{10}\cdot\frac{\lambda}{24}(1-\lambda)(2-\lambda)(3-\lambda)\cdot I$$
$$a_9 = \omega(k)\cdot I$$

and ΔA and ΔB denote the uncertainty matrices of the (system matrix) A_I and (input matrix) B_I, respectively, as shown in the following equation:

$$\Delta A = \sum_{j=1}^{n} \varepsilon_j A_j \text{ and } \Delta B = \sum_{j=1}^{n} \varepsilon_j B_j \tag{15}$$

In this study, a sufficient condition is proposed to explain that the linear system with unstructured parametric uncertainties is robust and controllable: assume that the linear interval system of Equation (11), $x(k+1) = A_I\cdot x(k) + B_I\cdot u(k)$, is controllable. If the following conditions are true, Equations (13) and (15) are robust and controllable.

$$\sum_{j=1}^{n} \varepsilon_j \varphi_j < 1 \tag{16}$$

where

$$\varphi_j = \left\{ \begin{array}{l} \mu\left(-S^{-1}U^H E_j V\begin{bmatrix} I_{n^2} & 0_{n^2\times n(m-1)} \end{bmatrix}^T\right), \varepsilon_j \geq 0 \\ -\mu\left(S^{-1}U^H E_j V\begin{bmatrix} I_{n^2} & 0_{n^2\times n(m-1)} \end{bmatrix}^T\right), \varepsilon_j < 0 \end{array} \right\} \tag{17}$$

I_{n^2} denotes the identity matrix of $n^2 \times n^2$, and $\overline{n}$ represents the number of uncertain matrices. The matrices S, E_j are defined as follows:

$$E_j = \begin{bmatrix} I_n & 0 & \bullet & \bullet & \bullet & 0 & 0 & \bullet & \bullet & \bullet & 0 & B_j \\ -A_j & I_n & \bullet & \bullet & \bullet & 0 & 0 & \bullet & \bullet & \bullet & B_j & 0 \\ \bullet & -A_j & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ 0 & 0 & \bullet & \bullet & \bullet & I_n & 0 & B_j & \bullet & \bullet & 0 & 0 \\ 0 & 0 & \bullet & \bullet & \bullet & -A_j & B_j & \bullet & \bullet & \bullet & 0 & 0 \end{bmatrix} \tag{18}$$

and $E_I = E_0 + \sum_{j=1}^{n} \varepsilon_j E_j$ allows singular value decomposition to become

$$E_I = U\begin{bmatrix} S & 0_{n^2\times n(m-1)} \end{bmatrix} V^H \tag{19}$$

where $U \in R^{n^2\times n^2}$ and $V \in R^{n(2n-1)\times n(2n-1)}$ are unitary matrices, $S = diag[\sigma_1, \sigma_2, \cdots, \sigma_{n^2}]$. The singular values of E_I are $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_{n^2} \geq 0$. The proof of this sufficient condition is shown in Appendix A.

2.4. Uncertain Parameter Range Corresponds to a Random Number Range

This section uses the sufficient condition proved in Appendix A to analyze the uncertain linear system of the fractional-order particle swarm algorithm for finding the range of uncertain parameters corresponding to the range of random numbers. Herein, we selected the maximum inertia weight ω_{max} of the fractional-order particle swarm algorithm to be 0.9, the minimum inertia weight ω_{min} to be 0.4, the individual learning parameter c_1 to be 2, and the group learning parameter c_2 also to be 2. Moreover, we set the individual learning parameter constant value c_{10} to 1 and the group learning parameter constant value c_{20} to 1. Therefore, the following Equation (20) can be obtained:

$$x(k+1) = \left(A_{I0} + \sum_{j=1}^{2} \varepsilon_j A_j\right) \cdot x(k) + \left(B_{I0} + \sum_{j=1}^{2} \varepsilon_j B_j\right) \cdot u(k) \tag{20}$$

where

$$A_{I0} = \begin{bmatrix} -\lambda I & -(\lambda/2)(1-\lambda)I & -(\lambda/6)(1-\lambda)(2-\lambda)I & -(\lambda/24)(1-\lambda)(2-\lambda)(3-\lambda) & \omega(k)I \\ I & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \\ (-\lambda-1)I & -(\lambda/2)(1-\lambda)I & -(\lambda/6)(1-\lambda)(2-\lambda)I & -(\lambda/24)(1-\lambda)(2-\lambda)(3-\lambda) & \omega(k)I \end{bmatrix}$$

$$A_1 = \begin{bmatrix} \lambda I & (\lambda/2)(1-\lambda)I & (\lambda/6)(1-\lambda)(2-\lambda)I & (\lambda/24)(1-\lambda)(2-\lambda)(3-\lambda)I & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \lambda I & (\lambda/2)(1-\lambda)I & (\lambda/6)(1-\lambda)(2-\lambda)I & (\lambda/24)(1-\lambda)(2-\lambda)(3-\lambda)I & 0 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ I & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B_1 = \begin{bmatrix} I & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ I & 0 \end{bmatrix} B_2 = \begin{bmatrix} 0 & I \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & I \end{bmatrix}$$

$$B_{I0} = \begin{bmatrix} I & I \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ I & I \end{bmatrix}$$

$$\varepsilon_1 = \begin{bmatrix} -1 & 1 \end{bmatrix}$$

$$\varepsilon_2 = \begin{bmatrix} -1 & 1 \end{bmatrix}$$

and $\omega(k) = \omega_{min} + ((iter_{max} - k)/iter_{max}) \times (\omega_{max} - \omega_{min})$. In this system, the inertia weight parameter of each generation is a constant value, which does not affect the robustness and controllability of Equation (20).

As the fractional-order particle swarm algorithm generates different values in Equation (20) based on different values of λ, the ranges of random numbers r_1 and r_2 corresponding to ε_1 and ε_2, respectively, are also different. As the value of λ is usually between 0 and 2, this study breaks λ into 20 values and deduces them one by one at the intervals of 0.1. Table 1 shows the r_1 and r_2 for different λ values according to Equation (20), where the random number range of r_2 is between 0 and 1, and the random number range r_1 will change as per different λ values. Among them, the range of r_1 obtained with λ = 0.3 is the widest and least conservative. Thus, the final range of r_1 and r_2 is

$$\begin{cases} r_1 \in (0.20874\ 1] \\ r_2 \in [0\ 1] \end{cases}. \tag{21}$$

Table 1. Controllable range of r_1 and r_2 from different λ values.

λ	r_1	r_2
0.1	[0.233811 1]	[0 1]
0.2	[0.217891 1]	[0 1]
0.3	[0.208743 1]	[0 1]
0.4	[0.220596 1]	[0 1]
0.5	[0.224792 1]	[0 1]
0.6	[0.238046 1]	[0 1]
0.7	[0.267023 1]	[0 1]
0.8	[0.252359 1]	[0 1]
0.9	[0.280379 1]	[0 1]
1	[0.329923 1]	[0 1]
1.1	[0.284115 1]	[0 1]
1.2	[0.319428 1]	[0 1]
1.3	[0.307748 1]	[0 1]
1.4	[0.333970 1]	[0 1]
1.5	[0.362764 1]	[0 1]
1.6	[0.336742 1]	[0 1]
1.7	[0.340463 1]	[0 1]
1.8	[0.339990 1]	[0 1]
1.9	[0.348875 1]	[0 1]
2	[0.335730 1]	[0 1]

The fractional-order particle swarm algorithm using the random number of Equation (21) is called the controllability fractional-order particle swarm optimizer (CFPSO) algorithm.

This study quotes Lin [21] when it is time to implement the controllable fractional-order particle swarm algorithm. If it meets the conditions, it will be executed. The conditions are as follows:

$$\begin{cases} \|pbest_i(k+1) - pbest_i(k)\| < e_{pbest_i} \\ \|gbest(k+1) - gbest(k)\| < e_{gbet} \end{cases} \tag{22}$$

Among them, e_{pbest_i} and e_{gbet} are set to 10^{-4}. The execution steps of the controllable fractional-order particle swarm algorithm are as follows:

Step 1: Set the number of groups, maximum value ω_{max}, and minimum value ω_{min} of the inertia weight of Equation (7). Then, set the individual learning parameter c_1, group learning parameter c_2, fractional value λ, function evaluations, and maximum number of iterations of Equation (7) $iter_{max}$;

Step 2: Initialize the random particle position $p_i(0)$ and initial velocity of $V_i(0)$ to 0;

Step 3: Calculate particle fitness;

Step 4: Update each particle's best solution and global best solution;

Step 5: Check whether the condition of Equation (22) is satisfied. If it is satisfied, obtain the controllable random number range according to Equation (21) and update Equations (6) and (9);

Step 6: Check whether the stop condition is met; if not, go back to Step 3, Step 4, and Step 5 until the stop condition is met.

3. XGBoost

The integrated machine learning algorithm combines many "weak learners" into one "strong learner" and has two integrated methods. One of the methods is "bagging" [28]. Each weak learner will randomly select some samples for independent training. The final classification result is to calculate the category that all weak learners discriminate the most times (majority voting). The most representative algorithm is random forest [29]. Another method is "boosting" [30]. The weak learner has a sequence relationship. The next weak learner will learn the information that the previous weak learner has not learned. After repeating N times, the N weak learners are weighted and combined into a strong learner. The most representative algorithm is the adaptive boost (AdaBoost) algorithm.

Chen and Guestrin [22] proposed the XGBoost algorithm. It combines the advantages of bagging and boosting and introduces a regularization function to improve the boosting method, and only optimizes the loss function. The regularization function is mainly used to limit the complexity of the model. With the regularization function, the model will be less complicated and is less likely to overfit. XGBoost uses a classification and regression tree (CART) [31] to classify weak learners. CART can be applied to classification tasks because CART uses binary segmentation, and features can be reused to generate trees. Further, it can also be applied to regression tasks. CART uses the maximum Gini index (Gini) as a method to select features to reduce the number of calculations.

For a given training set S, its Gini index is

$$Gini(S) = 1 - \sum_{k=1}^{K} \left(\frac{C_k}{|S|} \right)^2 \tag{23}$$

where C_k denotes a subset of samples belonging to the k-th category in S, and K represents the number of categories. The greater the Gini index, the greater the uncertainty of the data.

CART uses a binary tree as a decision tree and only classifies node features as "yes" or "no." Therefore, the decision tree is equivalent to recursively dicing each feature: divide the feature space into a finite number of units, and determine the prediction probability distribution on these units. Thus, the overall process comprises two steps: decision tree generation and pruning. The generating calculation step is to start from the root node and divide the root node recursively until the stopping conditions are met. The stopping conditions are as follows: (1) the number of samples in the node is less than the preset threshold; (2) the Gini index of the sample set is less than the preset threshold; (3) the depth of the decision tree meets the specified conditions; (4) after the feature is used, it cannot be divided. The pruning step is to start pruning from the bottom of the decision tree T_0 generated using the decision tree generation algorithm and pruning one node at a time until the root node of T_0 forms a subtree sequence $\{T_0, T_1, \cdots T_n\}$. Subsequently, the cross-validation method predicts the subtree sequence in the verification data set, and the best subtree T_α is selected from it.

Suppose that when a new tree f_n is to be constructed in the n-th iteration, the objective function is

$$L^{(n)} = \sum_{i=1}^{\widetilde{T}} l\left(y_i, \hat{y}_i^{(n-1)} + f_n(x_i) \right) + \Omega(f_n) \tag{24}$$

$$\Omega(f_n) = Y\widetilde{T} + \frac{1}{2}\lambda \|w\|^2 \tag{25}$$

$$\hat{y}_i^{(0)} = 0,\ \hat{y}_i^{(1)} = \hat{y}_i^{(0)} + f_1(x_i), \ldots, \hat{y}_i^{(N)} = \hat{y}_i^{(N-1)} + f_N(x_i) \tag{26}$$

where $l(\cdot)$ denotes a loss function that is a convex function; $\Omega(\cdot)$ represents a regularization term; $\hat{y}_i^{(n)}$ denotes the model prediction for the n-th round; $\widetilde{T}$ represents the number of leaf nodes; f_n denotes the structure of the n-th tree; Y represents the penalty coefficient for the number of leaf nodes; λ denotes the penalty coefficient for the leaf node score; w represents the score of each tree leaf node.

4. Results

According to statistics from the Ministry of Health and Welfare, heart disease is ranked second among the top 10 causes of death in Taiwan in 2018. The death toll increased by 4.5% from the previous year. Therefore, adjusting the hyperparameters of XGBoost through the controllable fractional-order particle swarm algorithm is necessary. The trained, reliable prediction model can assist doctors in quickly discerning whether a patient has heart disease so that early treatment can reduce the number of deaths caused by heart damage.

4.1. Heart Disease Data Set

The UCI website provides a heart disease data set [32] with no missing data. This data set offers 13 patient characteristics. There are five continuous features and eight category features to predict whether the patient has heart disease. Table 2 is an introduction to the data characteristics.

Table 2. Data characteristics.

Features	Features Expression	Description
Age	Age	
Sex	Sex	Female: 0, Male: 1
Chest Pain Type (Cp)	Types of chest pain	Typical angina: 0, Atypical angina: 1, Nonangina: 2, Asymptomatic: 3
Resting Blood Pressure (trestbps)	Blood pressure at rest	(mmHg)
Serum Cholesterol (chol)	Serum cholesterol content	(mg/dL)
Fasting Blood Sugar (fbs)	Amount of glucose in the blood on an empty stomach	(>120 mg/dL, false: 0, true: 1)
Resting Electrocardiographic Results (restecg)	ECG information at rest	Normal: 0, Abnormal ST-T fluctuation: 1, Left ventricular hypertrophy: 2
Maximum Heart Rate Achieved (thalach)	Maximum number of heart beats per minute	
Exercise-Induced Angina (exang)	Does angina occur during exercise	No: 0, Yes: 1
Old Peak	ST-segment reduction induced by exercise that is significantly different from relative rest	
Peak Exercise Slope (slope)	Slope of the ST phase at peak exercise	Rise: 1, Flat: 2, Fall: 3
Number of Major Vessels Colored Using Fluoroscopy (ca)	Number of major blood vessels that can be inspected using fluorescence	
Thallium Scan (thal)	Information about the distribution of blood in the blood vessel through thallium test to check whether the blood vessel is defective	Normal: 3, Inherent defect: 6, Repairable defect: 7

4.2. Data Preprocessing

Standard preprocessing methods include sampling, noise reduction, normalization, data cleaning, and feature engineering. The data preprocessing methods used herein include standardization and feature engineering.

4.2.1. Standardization

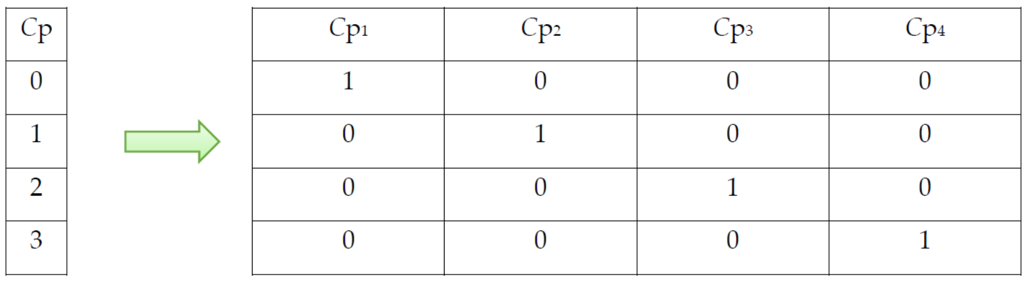
The features in the data have different units, and the distribution ranges are different. Thus, using original features will cause some machine learning algorithms to only focus on the features with larger values that cannot accurately train the model. Therefore, the feature distribution is converted to the same range through a standardized method so that all features have the same influence when the model is learning. The commonly used methods are z-score standardization and maximum and minimum standardization. This study uses maximum and minimum standardization to scale all unique values to between 0–1. The formula is as follows:

$$x_{new} = \frac{x_{ori} - x_{min}}{x_{max} - x_{min}} \quad (27)$$

where x_{ori} denotes the original feature value; x_{min} represents the minimum feature value; and x_{max} denotes the maximum feature value. The calculated x_{new} is the new feature value between 0 and 1 after standardization.

4.2.2. Feature Engineering

In the original data, the values of the feature of discrete data may have no meaning to each other, but they have serial properties when represented. For example, the display of chest pain types in the data set is shown in the left half of Figure 1. Therefore, in the learning process, distance-related algorithms will be affected and lead to erroneous learning. Thus, through one-hot encoding, the original discrete features are expanded into mutually independent and exclusive qualities, as shown in the right half of Figure 1. This will make the features have the same effects on the algorithm. This study uses one-hot encoding to expand the discrete features (such as Cp) that do not contain sequence properties in the original features into four independent features (Cp_1, Cp_2, $Cp_{3,}$ and Cp_4).



Cp
0
1
2
3

Cp_1	Cp_2	Cp_3	Cp_4
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

Figure 1. Schematic of one-hot encoding.

Moreover, the information that the machine learning model can learn is increased using derivative features. There are many ways to derive features [33]. This research regards the results of the "unsupervised learning" *K*-means algorithm as new features and incorporates them into the original features for machine learning models to learn. Among them, we used the "Euclidean distance" as the calculation method of data grouping and selected the results when the number of groups $K = 2$, $K = 3$, and $K = 4$. Furthermore, the analysis of the principal components of projecting multidimensional features to a lower-dimensional feature coordinate system produces new features orthogonal to each other (which implies that the features are irrelevant). Thus, it is possible to represent the original data with fewer features but still retain the most important information. This study regards the projection of the original data to the new 2D coordinates as a derivative feature for the model to learn.

4.3. Application Controllable Fractional-Order Particle Swarm Algorithm

Herein, the number of populations is set to 30 groups, the number of iterations is 100 generations, and six hyperparameters in the XGBoost model are selected. We used CFPSO to find the best location, i.e., to find the best hyperparameters to minimize the "custom fitness" and compare the differences with the hyperparameters officially recommended by XGBoost (XGBoost, 2021). The six types of hyperparameters include "Learning Rate," the "Max_depth" that each tree can grow, "lowest segmentation threshold (Gamma)" when the leaf node is to be divided into two cotyledon nodes, "data sampling (Subsample)" for the magnification of the data in the training set when each tree is trained, "feature sampling (Colsample_bytree)" for the magnification of the data feature of the training set when each tree is trained, and "L2 regularization parameter (Reg_lambda)."

Table 3 shows the hyperparameter values and search ranges officially recommended by XGBoost.

Table 3. Hyperparameters of XGBoost.

Hyper Parameters	Official Recommended Value	Search Region
Learning Rate	0.3	$[0.1 \quad 0.5]$
Max_depth	6	$[3 \quad 9]$
Gamma	0	$[0.01 \quad 0.2]$
Subsample	1	$[0.5 \quad 1]$
Colsample_bytree	1	$[0.5 \quad 1]$
Reg_lambda	0	$[0.1 \quad 5]$

4.3.1. Custom Fitness

Accuracy is used to evaluate the accuracy of model predictions and is a basis for assessing the overall credibility of the model. $F_{\beta score}$ evaluates and weighs false positives and false negatives through precision and recall. Among them, when $\beta > 1$, recall is β times more important than precision. As the model is applied in the medical field, the false-negative component is more important than the false positive, so this study set β to 2. The formula is as follows:

$$Accurancy = \frac{TP + TN}{TP + TN + FP + FN} \tag{28}$$

$$F_{\beta score} = \frac{1}{\frac{1}{1+\beta}\cdot\frac{1}{Precision} + \frac{\beta}{1+\beta}\cdot\frac{1}{Recall}} = \frac{(1+\beta)\cdot Precision\cdot Recall}{(\beta\cdot precision) + Recall} \tag{29}$$

where $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$. TP denotes true positive; TN represents true negative; FP denotes false positive; FN represents false negative.

This study focuses on the accuracy and $F_{\beta score}$ of the validation set but also on the accuracy and $F_{\beta score}$ of the training set. To prevent the model from learning the model hyperparameters that happen to be performed better when evaluated on the validation set, the custom fitness is set using the following equation:

$$Acc = 0.5\cdot(Train_{Acc} + Val_{Acc}) \tag{30}$$

$$F_{\beta score} = 0.5\cdot\left(Train_{F_{\beta score}} + Val_{F_{\beta score}}\right) \tag{31}$$

$$CustomFitness = \frac{1}{0.5\cdot(Acc + F_{\beta score})} - 1 \tag{32}$$

Custom fitness is within the range $[0 \quad \infty]$. The smaller the value, the better. The execution steps of the experiment are as follows:

Step 1: Load the heart disease data set;

Step 2: Adjust the data set for preprocessing, such as standardization and one-hot encoding, to a type that the machine learning model can learn. Use derivative features such as K-means and PCA's two-dimensional coordinates as new features for model learning;

Step 3: Divide the data set into three parts: 70% training data set, 15% validation data set, and 15% test data set;

Step 4: Initialize the controllable fractional-order particle swarm optimizer (CFPSO) position and use it as the initial input of the model hyperparameters;

Step 5: Use the training set to train the machine learning model according to the hyperparameter settings and evaluate the accuracy and $F_{\beta score}$ of the model training set and validation set to integrate into a custom objective function (custom fitness);

Step 6: Update the controllable fractional-order particle swarm algorithm's own best solution and global best solution according to the minimum value of the self-defined objective function;

Step 7: Repeat Steps 5 and 6 until the stop condition is met;

Step 8: Use the global best solution as the model's best hyperparameters to train the model and use the test set as the final evaluation result.

4.3.2. Experimental Results

The experimental results are shown in Table 4, which shows the evaluation index results of the hyperparameters officially recommended by XGBoost and the best hyperparameters found by CFPSO on the training and validation sets. Table 5 shows the two results in the test data set, and Table 6 lists the best hyperparameter values found by CFPSO.

Table 4. Evaluation index results of training and validation sets.

	Accuracy	$F_{\beta score}$	**Custom Fitness**
XGBoost officially recommended hyperparameters	91.4%	89.5%	0.1056
CFPSO best hyperparameters	98.7%	98.9%	0.0120

Table 5. Evaluation index results of the test set.

	Accuracy	$F_{\beta score}$	**Custom Fitness**
XGBoost officially recommended hyperparameters	84.7%	88.2%	0.1559
CFPSO best hyperparameters	91.3%	94.3%	0.0776

Table 6. The best hyperparameters found using CFPSO.

Hyperparameters	**Best Hyperparameters**
Learning Rate	0.265201
Max_depth	7
Gamma	0.194108
Subsample	0.95
Colsample_bytree	0.63
Reg_lambda	4.846424

The experimental results show that the model trained using CFPSO to obtain the best hyperparameters is better than the model learned using the hyperparameters officially recommended by XGBoost. Furthermore, the accuracy, $F_{\beta score}$, and custom fitness are better than the original hyperparameter model, so CFPSO is more suitable as a reference basis for assisting doctors in determining whether a patient has heart disease.

5. Conclusions

This study replaced PSO with FPSO. We used the control theory viewpoint to deduce the CFPSO algorithm, rewrote the system into an uncertain linear system, and proved its controllability. As the fractional value λ will affect the random number range, this research disassembled λ into 20 values and deduced them one by one at intervals of 0.1. The range obtained when λ is 0.3 was the least conservative. Therefore, the random number selected had a wide range. This study used CFPSO to find the best hyperparameters of the model when predicting the heart disease data set. The experimental results show that the best hyperparameters found through CFPSO were better than the hyperparameters officially recommended by XGBoost. Therefore, CFPSO is more suitable for helping physicians quickly determine whether a patient has heart disease through the learned model. In future work, variable-order fractional operators (VOFO) can be involved in this topic for coping with complicated real-world issues because VOFO can provide the robustness and flexibility characteristics more in control theory [34–38].

Author Contributions: Conceptualization, F.-I.C. and T.-H.H.; methodology, P.-Y.Y.; software, C.-H.L.; validation, P.-Y.Y. and C.-H.L.; formal analysis, W.-H.H.; investigation, J.-H.C.; resources, J.-H.C.; data curation, T.-C.L.; writing—original draft preparation, F.-I.C.; writing—review and editing, T.-H.H.; visualization, W.-H.H.; supervision, J.-H.C.; project administration, J.-H.C.; funding acquisition, J.-H.C. All authors have read and agreed to the published version of the manuscript.

Funding: Publication costs are funded by the Ministry of Science and Technology, Taiwan, under grants MOST 110-2221-E-992-091-MY3, MOST 110-2221-E-037-005, and MOST 110-2221-E-153-010. The design and part writing costs of the study are funded by NKUST-KMU JOINT RESEARCH PROJECT (#NKUSTKMU-110-KK-001) and the "Intelligent Manufacturing Research Center" (iMRC) from the Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available in a publicly accessible repositoryThe data presented in this study are openly available in UCI Machine Learning Repository at http://archive.ics.uci.edu/ml, reference number [32].

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

Appendix A

This appendix illustrated the proving process to the controllability of the linear system of Equations (13) and (14).

Proof. Given $rank(E) = rank\left(S^{-1}U^{H}EV\right)$, the rank of E_I (Rank) is equivalent to discussing the rank of the following Equation (A1):

$$\begin{bmatrix} I_{n^2} & 0_{n^2\times n(m-1)} \end{bmatrix} + \sum_{j=1}^{n} \varepsilon_j F_j \tag{A1}$$

where $F_j = S^{-1}U^{T}E_jV, j = 1, 2, \cdots \overline{n}$. If the matrix has at least one nonsingular $n^2 \times n^2$ submatrix, then the matrix whose rank is at least the above formula has a rank of n^2 and the sufficient condition is that the following formula is a nonsingular matrix:

$$G = I_{n^2} + \sum_{j=1}^{n} \varepsilon_j \widetilde{F}_j \tag{A2}$$

where $\widetilde{F}_j = S^{-1}U^{T}E_jV\begin{bmatrix} I_{n^2} & 0_{n^2\times n(m-1)} \end{bmatrix}^{T}$.

According to the lemma proposed by Desoer and Vidyasagar [39].

Let $\|\cdot\|$ represent the norm of the induction matrix defined on $C^{n\times n}$. For $A \in C^{n\times n}$, the following properties of the matrix measure $\mu_{\text{measure}}(\cdot)$ are all true:

1. $\mu_{\text{measure}}(\pm I) = \pm 1$, for the identity matrix I;
2. $-\|A\| \le -\mu_{\text{measure}}(-A) \le Re(\lambda(A)) \le \mu_{\text{measure}}(A) \le \|A\|$;
3. $\mu_{\text{measure}}(A + B) \le \mu_{\text{measure}}(A) + \mu_{\text{measure}}(B)$, for any two matrices $A, B \in C^{n\times n}$;
4. $\mu_{\text{measure}}(\alpha A) = \alpha\mu_{\text{measure}}(A), \forall\alpha > 0, \alpha \in R$.

where $\lambda(A)$ represents the eigenvalues of the matrix A, and $Re(\lambda(A))$ represents the real number of $\lambda(A)$. The following formula is obtained:

$$\begin{aligned} &\mu_{\text{measure}}\left(-\textstyle\sum_{j=1}^{n} \varepsilon_j \widetilde{F}_j\right) = \mu_{\text{measure}}\left(-\textstyle\sum_{j=1}^{n} \varepsilon_j \left(S^{-1}U^{T}E_jV\begin{bmatrix} I_{n^2} & 0_{n^2\times n(m-1)} \end{bmatrix}\right)\right) \\ &\le \textstyle\sum_{j=1}^{\overline{n}} \mu_{\text{measure}}\left(-\varepsilon_j\left(S^{-1}U^{T}E_jV\begin{bmatrix} I_{n^2} & 0_{n^2\times n(m-1)} \end{bmatrix}\right)\right) = \textstyle\sum_{j=1}^{n} \varepsilon_j\varphi_j < 1 \end{aligned} \tag{A3}$$

Therefore, according to the lemma proposed by Lin [40], let $A \in C^{n\times n}$. If $\mu_{\text{measure}}(-A) < 1$, then $det(I + A) \neq 0$, and the following formula is obtained:

$$det(G) = det\left(I_{n^2} + \sum_{j=1}^{n} \varepsilon_j \widetilde{F}_j\right) \neq 0 \tag{A4}$$

Therefore, the matrix of Equation (A1) is nonsingular. It can be observed that the matrix E_I is full rank n^2. Furthermore, according to the lemma proposed by Rosenbrock [41], the uncertain linear systems of Equations (13) and (14) are robust and controllable. End of proof. □

References

1. Abiodun, O.I.; Kiru, M.U.; Jantan, A.; Omolara, A.E.; Dada, K.V.; Umar, A.M.; Linus, O.U.; Arshad, H.; Kazaure, A.A.; Gana, U. Comprehensive review of artificial neural network applications to pattern recognition. *IEEE Access* **2019**, *7*, 158820–158846. [CrossRef]
2. Duchi, J.; Hazan, E.; Singer, Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.
3. Tseng, P. Approximation accuracy, gradient methods, and error bound for structured convex optimization. *Math. Program.* **2010**, *125*, 263–295. [CrossRef]
4. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; pp. 1942–1948.
5. Khan, M.K.; Nystrom, I. A Modified particle swarm optimization applied in image registration. In Proceedings of the 20th International Conference on Pattern Recognition, Istanbul, Turkey, 23–26 August 2010; pp. 2302–2305.
6. Naderi, E.; Narimani, H.; Fathi, M.; Narimani, M.R. A novel fuzzy adaptive configuration of particle swarm optimization to solve large-scale optimal reactive power dispatch. *Appl. Soft Comput.* **2017**, *53*, 441–456. [CrossRef]
7. Yang, C.I.; Chou, J.H.; Chang, C.K. Hybrid Taguchi-based particle swarm optimization for flowshop scheduling problem. *Arab. J. Sci. Eng.* **2014**, *39*, 2393–2412. [CrossRef]
8. Shi, Y.; Eberhart, R. A modified particle swarm optimizer. In Proceedings of the IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat "No. 98TH8360"), Anchorage, AK, USA, 4–9 May 1998; pp. 69–73.
9. Shi, Y.; Eberhart, R.C. Empirical study of particle swarm optimization. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat "No. 99TH8406"), Washington, DC, USA, 6–9 July 1999; pp. 1945–1950.
10. Suganthan, P.N. Particle swarm optimiser with neighbourhood operator. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat "No. 99TH8406"), Washington, DC, USA, 6–9 July 1999; pp. 1958–1962.
11. Clerc, M. The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. In Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat "No. 99TH8406"), Washington, DC, USA, 6–9 July 1999; pp. 1951–1957.
12. Clerc, M.; Kennedy, J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Computat.* **2002**, *6*, 58–73. [CrossRef]
13. Shi, Y.; Eberhart, R.C. Particle swarm optimization: Developments, applications and resources. In Proceedings of the 2001 Congress on Evolutionary Computation (IEEE. Cat "No. 01TH8546"), Seoul, Korea, 27–30 May 2001; pp. 81–86.
14. Ratnaweera, A.; Halgamuge, S.K.; Watson, H.C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Trans. Evol. Computat.* **2004**, *8*, 240–255. [CrossRef]
15. Chatterjee, A.; Siarry, P. Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. *Comput. Oper. Res.* **2006**, *33*, 859–871. [CrossRef]
16. Ko, C.N.; Chang, Y.P.; Wu, C.J. An orthogonal-array-based particle swarm optimizer with nonlinear time-varying evolution. *Appl. Math. Comput.* **2007**, *191*, 272–279. [CrossRef]
17. Chen, J.; Pan, F.; Cai, T.; Tu, X. Stability analysis of particle swarm optimization without Lipschitz constraint. *J. Control. Theor. Appl.* **2003**, *1*, 86–90. [CrossRef]
18. Emara, H.M.; Fattah, H.A. Continuous swarm optimization technique with stability analysis. In Proceedings of the 2004 American Control Conference, Boston, MA, USA, 30 June–2 July 2004; pp. 2811–2817.
19. Fan, W.; Cui, Z.; Ceng, J. Inertia weight selection strategy based on Lyapunov stability analysis. In Proceedings of the Ninth International Conference on Hybrid Intelligent Systems, Shenyang, China, 12–14 August 2009; pp. 504–509.
20. Koguma, Y.; Aiyoshi, E. Statistical stability analysis for particle swarm optimization dynamics with random coefficients. *Electron. Commun. Jpn.* **2012**, *95*, 31–42. [CrossRef]
21. Lin, C.C. Study on the Controllability of Particle Swarm Optimization Algorithm and Its Applications. Master's Thesis, National Kaohsiung First University of Science and Technology, Kaohsiung City, Taiwan, 2014.

22. Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 785–794.
23. XGBoost. Available online: https://xgboost.readthedocs.io/en/latest/parameter.html (accessed on 1 September 2021).
24. Ma, R.; Yang, L.; Zhang, Z. Analysis the characteristic of C1, C2 based on the PSO of iterative shift and trajectory of particle. *Math. Comput.* **2013**, *2*, 109–115.
25. Solteiro Pires, E.J.; Tenreiro Machado, J.A.; Moura Oliveira, P.B.; Boaventura Cunha, J.; Mendes, L. Particle swarm optimization with fractional-order velocity. *Nonlinear Dyn.* **2010**, *61*, 295–301. [CrossRef]
26. Solteiro Pires, E.J.; Tenreiro Machado, J.A.; de Moura Oliveira, P.B. *Fractional Particle Swarm Optimization. Mathematical Methods in Engineering*; Fonseca Ferreira, N.M., Tenreiro Machado, J.A., Eds.; Springer: Dordrecht, The Netherlands, 2014; pp. 47–56.
27. Chen, S.H.; Chou, J.H. Controllability robustness of linear interval systems with/without state delay and with unstructured parametric uncertainties. *Admin. Appl. Anal.* **2013**, *2013*, 1–10. [CrossRef]
28. Breiman, L. Bagging predictors. *Mach. Learn.* **1996**, *24*, 123–140. [CrossRef]
29. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
30. Kearns, M. Thoughts on Hypothesis Boosting. 1988; Unpublished manuscript.
31. Breiman, L.; Friedman, J.; Stone, C.J.; Olshen, R.A. *Classification and Regression Trees*; CRC Press: Boca Raton, FL, USA, 1984.
32. UCI. Heart Disease Data Set. Available online: https://archive.ics.uci.edu/ml/datasets/Heart+Disease (accessed on 1 September 2021).
33. Zheng, A.; Casari, A. *Feature Engineering for Machine Learning*; O'Reilly Media, Inc.: Newton, MA, USA, 2018.
34. Zheng, X.; Wang, H. A Hidden-Memory Variable-Order Time-Fractional Optimal Control Model: Analysis and Approximation. *SIAM J. Control. Optim.* **2021**, *59*, 1851–1880. [CrossRef]
35. Zheng, X.; Wang, H. An Error Estimate of a Numerical Approximation to a Hidden-Memory Variable-Order Space-Time Fractional Diffusion Equation. *SIAM J. Numer. Anal.* **2020**, *58*, 2492–2514. [CrossRef]
36. Patnaik, S.; Hollkamp, J.P.; Semperlotti, F. Applications of variable-order fractional operators: A review. *Proc. R. Soc. A Math. Phys. Eng. Sci.* **2020**, *476*, 20190498. [CrossRef]
37. Lorenzo, C.F.; Hartley, T.T. Variable Order and Distributed Order Fractional Operators. *Nonlinear Dyn.* **2020**, *29*, 57–98. [CrossRef]
38. Samko, S.G.; Ross, B. Integraton and differentiation to a variable fractional order. *Integral. Transform. Spec. Funct.* **1993**, *1*, 277–300. [CrossRef]
39. Desoer, C.A.; Vidyasagar, M. *Feedback Systems: Input-Output Properties*; SIAM: Philadelphia, PA, USA, 1975.
40. Lin, C.L. *Mathematics of Modern Control Theory*; Kaun Tang International Publications Ltd.: Taipei, Taiwan, 2007.
41. Rosenbrock, H.H. *State-Space and Multivariable Theory*; Thomas Nelson: London, UK, 1970.