Strengthening the PSO algorithm with a new technique inspired by the golf game and solving the complex engineering problem

Serkan Dereli¹ · Raşit Köker²

Received: 19 October 2020 / Accepted: 1 February 2021 / Published online: 17 February 2021
© The Author(s) 2021

Abstract

This study has been inspired by golf ball movements during the game to improve particle swarm optimization. Because, all movements from the first to the last move of the golf ball are the moves made by the player to win the game. Winning this game is also a result of successful implementation of the desired moves. Therefore, the movements of the golf ball are also an optimization, and this has a meaning in the scientific world. In this sense, the movements of the particles in the PSO algorithm have been associated with the movements of the golf ball in the game. Thus, the velocities of the particles have converted to parabolically descending structure as they approach the target. Based on this feature, this meta-heuristic technique is called RDV (random descending velocity) IW PSO. In this way, the result obtained is improved thousands of times with very small movements. For the application of the proposed new technique, the inverse kinematics calculation of the 7-joint robot arm has been performed and the obtained results have been compared with the traditional PSO, some IW techniques, artificial bee colony, firefly algorithm and quantum PSO.

Keywords Random descending velocity · Particle swarm optimization · Inertia weight · Robotics · Inverse kinematics · Complex engineering problem

Introduction

Recently, the difficulty and complexity of engineering problems have triggered great motivation in the research world. Because, such problems motivate the research world for many important inventions or techniques [29]. Many methods developed for this purpose, especially mathematically based, have fallen from the eyes of researchers due to their complexity and multiple unknown parameters. Because multiple unknown parameters lead to complex equations and complex equations cause longer solution time and numerical methods are no longer sufficient to solve such equations [43]. For this reason, with the inadequacy of traditional methods in solving complex problems, the research world turned to artificial intelligence techniques for time [42]. Another important factor is the fact that it reaches a solution in a

short time and can be easily applied to many different areas. The concept of artificial intelligence has entered our lives intensively in the 1980s and has secured its place with different methods and ideas until today. Today, it has become one of the most widely used and even essential techniques in almost all fields. The nearest neighborhood [40], threshold acceptance [2], taboo search [16], simulated annealing [37], genetic algorithm [4], particle swarm optimization [11], artificial bee colony [14] and firefly algorithms [21] are some of these techniques and are frequently used in the literature to solve any engineering problem. Figure 1 [1] shows the classifications of these techniques, but techniques that achieve an approximate value are given instead of a single final result in this figure. Because the meta-heuristic optimization techniques including the PSO algorithm, which forms the basis of this study, are not guaranteed to achieve the best results [26]. The reason for this is that these techniques generate random solutions by means of a number of parameters in a given solution space. In this type of algorithms, three important conditions, initial values, parameters and randomness have a direct effect on the results [19].

In this study, after a thorough analysis of the particle swarm optimization strategy, a new IW technique for reinforcing this

✉ Serkan Dereli
dereli@subu.edu.tr

¹ Department of Computer Technology and Programming,
Sakarya University of Applied Sciences, Geyve, Turkey

² Department of Electrical and Electronical Engineering,
Sakarya University of Applied Sciences, Geyve, Turkey

Fig. 1 Classification of optimization techniques

technique is mentioned. As shown in Fig. 1, this technique is a population-based meta-heuristic technique and is also known in the literature as swarm optimization techniques. The swarm is a living community that performs vital activities together in nature, attracts attention with its excellent social organization and, thus, becomes a new structure with superior intelligence [8]. In this sense, many different swarm algorithms have been introduced by being inspired by many living things circulating in land, air and water and these algorithms have been successfully applied in different engineering fields. Whale [25], gray wolf [23], fire-fly [35], artificial bee colony [30], ant colony [32], bat [3] and particle swarm optimization [17] algorithms are some of them.

In this study, particle swarm optimization, which is the first technique working according to swarm intelligence, has been examined in depth and this technique has been improved by being inspired by the movements of the ball during the golf game. The next part of the study is organized as follows: In the section, the necessary parameters and algorithm for this new technique, which is called RDV PSO, which basically reduces the speed of particles randomly according to a particular parameter, are described and the inverse kinematics calculation of the 7-joint serial robot manipulator for the test analyzes of the new technique designed was performed and using the Euclidean distance equation. In the next section, the results obtained with RDV PSO were compared with both classical PSO and other IW techniques. In the following section, the results obtained were analyzed in depth, and some important inferences were revealed. In the last section, the results of the study are explained.

Methodology

In this paper, the new IW technique used in this study performs the inverse kinematics calculation of the newly designed 7-joint serial robot manipulator. The RDV IW

technique has been tested on both simulation and application basis. However, the application stage was limited in terms of the results obtained depending on the resolution of the actuators used in the robot manipulator.

Particle swarm optimization

PSO algorithm has been first used by Kennedy and Eberhart in 1995 [22] and stands out in terms of low number of parameters, fast operation and producing effective results [24]. In addition, PSO is the first of a swarm-based algorithms in which individuals who are in a simple living structure alone come together to form a new structure with superior intelligence. For this reason, it is often preferred to solve nonlinear problems which use intensive mathematical equations in classical methods [15]. As with other swarm-based algorithms, particles begin to look for the most appropriate solution by selecting a random location in the solution space. Thus, they begin a journey towards the optimal solution, taking advantage of both the personal and the swarm's experiences [34].

In particle swarm optimization, the particles move to a new position with a new speed at each iteration, as shown in Fig. 2. Therefore, each time they move closer to the goal that is, the solution of the problem. The fact that the optimum values can be obtained with only two parameters in the algorithm is an indication of how strong this algorithm is. For this reason, it is preferred in many different engineering problems because of its ease of application, simple structure and producing effective results.

They have developed a production planning method by using PSO and GA techniques together in order to increase production efficiency and product quality in a production center [33]. They overcame such problems by running the algorithm with dimensional learning technique in problems such as the phenomena of "oscillation" where the particle

Fig. 2 Illustration of motion of the particles

swarm optimization algorithm was insufficient. [39]. In another study published in the field of energy, the optimization of the parameters produced for the materials used to increase the energy performance of buildings was carried out with the PSO algorithm [6]. In another study conducted in the field of geology, the behavior of a slope during an earthquake was estimated by PSO and ANN [12]. In the other study carried out in the field of transportation, PSO-based traffic control module has been designed to realize the vehicle flow in peak traffic [20]. They developed a PSO-based system to evaluate the characteristics of planar surfaces consisting of four different components [28]. They have developed a PSO-based system for accurate navigation to ensure the real-time movement of unmanned surface vehicles [38]. In their studies in the field of space research, they realized the trajectory planning of the free-floating space robot with the PSO algorithm [36]. They used the PSO algorithm to determine the optimal route of the vessel as a result of the information received from the satellite to prevent collision of ships which are an important study in ocean engineering [18].

Pseudo code of the conventional PSO:

1. Initialize X_i , V_i , iteration, p_{best} , g_{best}
2. Generate random particles (P)
3. **For each** particle (i)
4. Calculate fitness function (f_i)
5. Update p_{best} , g_{best}
6. **End for**
7. **While** iteration
8. **For each** particle I
9. Update V_i , X_i
10. If $X_i > \text{limit}$ then $X_i = \text{limit}$
11. Calculate fitness function f_i
12. Update p_{best} , g_{best}
13. **End for**
14. **End while**

Pseudo code of the conventional PSO is seen above. There are three important points that stand out according to the algorithm steps for particle swarm optimization. These

are parameters (x and v), particle number and fitness function. The use of meta-heuristic algorithms depends on the proper definition of the fitness function. Because, problems in these algorithms are solved by converting them to fitness function which shows the distance of the particles to the best solution [7]. “ v ” is the current velocity of the particle (Eq. 1) and “ x ” is the position (Eq. 2). The convergence of the particle depends on these two parameters. That is, since the convergence of the particle to the solution depends on these two parameters, these parameters are extremely important. The number of particles is a parameter that directly affects the solution. Because, when the number increases, it improves the solution but slows down the operation of the algorithm [9].

$$v_{id} = v_{id} + c_1 \cdot r_1 \cdot (p_{best} - x_{id}) + c_2 \cdot r_2 \cdot (g_{best} - x_{id}), \quad (1)$$

$$x_{id} = x_{id} + v_{id}. \quad (2)$$

In Eqs. 1 and 2; “ d ” is the dimension of the problem; “ i ” is the number of particles; “ c_1 ” and “ c_2 ” personal best and global best weights; “ r_1 ” and “ r_2 ” represent random numbers in the range [0–1]. “ p_{best} ” is the distance of the location of each particle to the food source, i.e., the optimum solution. “ g_{best} ” is the closest distance the swarm has achieved according to the food source during iteration.

RDV IW (random descending velocity inertia weight) PSO

Although PSO was initially proposed for nonlinear problems, it is currently used in many different problems due to its easy applicability, powerful control parameters and effective results. However, there are some disadvantages, such as being stuck in a particular local solution. Furthermore, although the small number of parameters is advantageous in terms of ease of use, it is a major obstacle to improving the algorithm [27]. For this reason, as in this study, researchers who want to improve the particle swarm optimization algorithm have enabled the other parameters to work more

effectively with additional parameters. In this study, the velocity parameter was improved by deriving new parameters inspired by the movements of the ball used in golf game.

A golf game is a competition in which players make the initial hit the most powerful and the next hit slower than the previous one to insert the golf ball into the designated holes. This does not change with the size of the golf course because the main goal in this game is to put the ball into the target hole with fewer strokes.

Figure 3 shows an average golf course and an example game completion. The player first uses all of his energy to perform the first shot called "driving". Because the first shot is extremely important in terms of throwing the ball away as far as possible (Shot 1 in Fig. 3). In this game, a player's first goal is to move the ball to the fairway by freeing the ball from all obstacles or bad ground (Shot 3 in Fig. 3). Now, the player is nearing the final moment. Therefore, the shots must

Fig. 3 A golf course and sample strokes

Fig. 4 Delta parameter value range

be slower or softer than the previous one. The final shots have been made from the area called Putting Green, which has an excellent ground (Shot 4 in Fig. 3). Because, this is an area where players are most likely to complete the game, both in terms of ground quality and proximity to the target hole (Shot 5 in Fig. 3).

RDV PSO pseudo code:

```

1. Initialize  $X_i$ ,  $V_i$ , iteration,  $pbest$ ,  $gbest$ 
2. Generate random particles ( $P$ )
3. For each particle ( $i$ )
4.   Calculate fitness function ( $f_i$ )
5.   Update  $pbest$ ,  $gbest$ 
6. End for
7. While iteration
8.   For each particle  $I$ 
9.     Calculate  $\Delta$ 
10.    If  $\Delta < rand$  then  $\alpha = \alpha * \alpha_{dump}$ 
11.     $W = \alpha$ 
12.    Update  $V_i = V_i * W$ 
13.    Update  $X_i$ 
14.    If  $X_i > limit$  then  $X_i = limit$ 
15.    Calculate fitness function  $f_i$ 
16.    Update  $pbest$ ,  $gbest$ 
17.   End for
18. End while

```

The PSO algorithm including the RDV technique is seen above, with the addition of the pseudo code 9, 10, 11, 12 and 13 lines. The delta parameter calculated in line 9 serves to direct the particles, similar to the control of the ball from the first movement of the ball to the end of the game. It represents a maximum value at the beginning of the algorithm, but decreases over time and completes the algorithm with a value close to zero (Eq. 3).

The Delta parameter tends to decrease linearly between the value of 0.99 and the value of 0.1 from the beginning of

the iteration to the end of the iteration (Fig. 4). This parameter is used to simulate the progress of the particles to the target point to the movements of the ball in a golf game. For example, particles move at normal speeds at the start of the iteration, such as the golf ball moving freely, quickly and uncontrolled on the first hit. However, because there is randomness, the Delta value is compared to a randomly generated value. Since delta has a maximum value at the start of the iteration, it is likely to be greater than the randomly generated value between 0 and 1. This helps the particles to move at normal speeds in the initial stages of iteration. Towards the end of the iteration, the randomly generated value is likely to be larger as the Delta has a minimum value. Thus, the particle velocities are reduced to a certain extent by means of the "*alpha_dump*" parameter, and the change in particle positions is kept at a minimum level.

$$\Delta = e^{\frac{-\text{iteration}}{\text{max_iteration}}} \quad (3)$$

The *alpha* used in the algorithm is a random number between [0, 1] and is taken as 0.3 in this study. *alpha_dump* is between [0.5–1] and is taken as 0.95 in this study. This parameter is used to randomly reduce the *alpha* value in other words the speed value in the PSO algorithm to a certain extent. As shown in the line 10 in Fig. 5, the *alpha* may be reduced if the *delta* value is greater than the randomly obtained number. All these calculations are performed to reduce the speed of the particle. Therefore, *w* parameter is added to Eq. 1 as shown in Eq. 4.

$$v_{id} = w \times v_{id} + c_1 \cdot r_1 \cdot (p_{\text{best}} - x_{id}) + c_2 \cdot r_2 \cdot (g_{\text{best}} - x_{id}). \quad (4)$$

Serial robot manipulator and kinematics analysis

The robot manipulator which has seven rotational joints, designed with nine Dynamixel AX 12-A actuators, is shown in Fig. 5. Since two actuators were used in the second joint to strengthen the joint, eight of them were used for manipulator joints and the other for the end element.

The next stage of the design of the robot manipulators is obviously their control. The basis of this process is kinematic and dynamic analysis [10]. Kinematics, which is called motion science, is actually a science that examines the properties and consequences of motion without questioning the causes of motion [13]. Kinematic performance in robot manipulators is directly related to the structure of the robot and helps the manipulator to operate flexibly and skillfully in the work space. In terms of definition, kinematics in robot manipulators reveals the relationship between joint angles and the distance between these joints and the position of the end element in the working space [31].

In this study, DH parameters suggested by Denavit and Hartenberg and shown in Table 1 were used for kinematic analysis. These parameters are four and allow the creation of homogeneous transformation matrices in the work space of each joint [5]. To avoid that the equations appear much

Fig. 5 A 7-DOF robot manipulator used in the study

Table 1 DH parameters for robot manipulator

i	a_i (cm)	α_i (°)	d_i (cm)	θ_i (°) (range)
1	0	−90	$L1=5$	$-90 < \theta_1 < 90$
2	$L2=6.5$	−90	0	$-180 < \theta_2 < 0$
3	$L3=5.5$	−90	0	$-90 < \theta_3 < 90$
4	$L4=6.5$	−90	0	$-90 < \theta_4 < 90$
5	$L5=6.5$	90	0	$-90 < \theta_5 < 90$
6	$L6=6.5$	0	0	$-90 < \theta_6 < 90$
7	$L7=13$	0	0	$-90 < \theta_7 < 90$

longer, "s" is used instead of sine and "c" is used instead of cosine in all equations between Eq. 5 and Eq. 10.

$${}^i_{i-1}T = \begin{bmatrix} \cos \theta_i - \cos \alpha_i \sin \theta_i & \sin \alpha_i \sin \theta_i & a_i \cos \theta_i \\ \sin \theta_i & \cos \alpha_i \cos \theta_i & -\cos \theta_i \sin \alpha_i \\ 0 & \sin \alpha_i & \cos \alpha_i \\ 0 & 0 & 1 \end{bmatrix}, \quad (5)$$

$$A_{\text{End-Effector}} = {}^7T = {}^1T \cdot {}^2T \cdot {}^3T \cdot {}^4T \cdot {}^5T \cdot {}^6T \cdot {}^7T = \begin{bmatrix} n_x & s_x & a_x & P_x \\ n_y & s_y & a_y & P_y \\ n_z & s_z & a_z & P_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (6)$$

$$\begin{aligned} {}^1_0T &= \begin{bmatrix} c\theta_1 & 0 & -s\theta_1 & 0 \\ s\theta_1 & 0 & c\theta_1 & 0 \\ 0 & -1 & 0 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, & {}^2_1T &= \begin{bmatrix} c\theta_2 & 0 & -s\theta_2 & L_2c\theta_2 \\ s\theta_2 & 0 & c\theta_2 & L_2s\theta_2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\ {}^3_2T &= \begin{bmatrix} c\theta_3 & 0 & -s\theta_3 & L_3c\theta_3 \\ s\theta_3 & 0 & c\theta_3 & L_3s\theta_3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, & {}^4_3T &= \begin{bmatrix} c\theta_4 & 0 & -s\theta_4 & L_4c\theta_4 \\ s\theta_4 & 0 & c\theta_4 & L_4s\theta_4 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, & {}^5_4T &= \begin{bmatrix} c\theta_5 & 0 & s\theta_5 & L_5c\theta_5 \\ s\theta_5 & 0 & -c\theta_5 & L_5s\theta_5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \\ {}^6_5T &= \begin{bmatrix} c\theta_6 & -s\theta_6 & 0 & L_6c\theta_6 \\ s\theta_6 & c\theta_6 & 0 & L_6s\theta_6 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, & {}^7_6T &= \begin{bmatrix} c\theta_7 & -s\theta_7 & 0 & L_7c\theta_7 \\ s\theta_7 & c\theta_7 & 0 & L_7s\theta_7 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \end{aligned} \quad (7)$$

Equation 5 shows the general transformation matrix in which each joint angle is formed. Equation 6, which is obtained by multiplying the homogeneous transformation matrices of all joints (Eq. 7), is the forward direction kinematic matrix containing the position and orientation information of the end effector. Since the method used in this study is intended to minimize the position error of the end effector, only these equations (p_x , p_y and p_z) are presented here.

$$\begin{aligned} p_x &= L_2c\theta_1c\theta_2 - L_5s\theta_5(c\theta_3s\theta_1 - c\theta_1c\theta_2s\theta_3) \\ &+ L_3s\theta_1s\theta_3 + L_5c\theta_5(c\theta_4(s\theta_1s\theta_3 + c\theta_1c\theta_2c\theta_3) \\ &+ c\theta_1s\theta_2s\theta_4) - L_6s\theta_6(s\theta_4(s\theta_1s\theta_3 + c\theta_1c\theta_2c\theta_3) \\ &- c\theta_1c\theta_4s\theta_2) - L_7c\theta_7(s\theta_6(s\theta_4(s\theta_1s\theta_3 + c\theta_1c\theta_2c\theta_3) \\ &- c\theta_1c\theta_4s\theta_2) - c\theta_6(c\theta_5(c\theta_4(s\theta_1s\theta_3 + c\theta_1c\theta_2c\theta_3) \\ &+ c\theta_1s\theta_2s\theta_4) - s\theta_5(c\theta_3s\theta_1 - c\theta_1c\theta_2s\theta_3))) \\ &+ L_6c\theta_6(c\theta_5(c\theta_4(s\theta_1s\theta_3 + c\theta_1c\theta_2c\theta_3) + c\theta_1s\theta_2s\theta_4) \\ &- s\theta_5(c\theta_3s\theta_1 - c\theta_1c\theta_2s\theta_3)) - L_7s\theta_7(c\theta_6(s\theta_4(s\theta_1s\theta_3 \\ &+ c\theta_1c\theta_2c\theta_3) - c\theta_1c\theta_4s\theta_2) + s\theta_6(c\theta_5(c\theta_4(s\theta_1s\theta_3 \\ &+ c\theta_1c\theta_2c\theta_3) + c\theta_1s\theta_2s\theta_4) - s\theta_5(c\theta_3s\theta_1 - c\theta_1c\theta_2s\theta_3))) \\ &+ L_4c\theta_4(s\theta_1s\theta_3 + c\theta_1c\theta_2c\theta_3) + L_3c\theta_1c\theta_2c\theta_3 + L_4c\theta_1s\theta_2s\theta_4, \end{aligned} \quad (8)$$

$$\begin{aligned} p_y &= L_5s\theta_5(c\theta_1c\theta_3 + c\theta_2s\theta_1s\theta_3) \\ &+ L_7s\theta_7(c\theta_6(s\theta_4(c\theta_1s\theta_3 - c\theta_2c\theta_3s\theta_1) \\ &+ c\theta_4s\theta_1s\theta_2) + s\theta_6(c\theta_5(c\theta_4(c\theta_1s\theta_3 - c\theta_2c\theta_3s\theta_1) \\ &- s\theta_1s\theta_2s\theta_4) - s\theta_5(c\theta_1c\theta_3 + c\theta_2s\theta_1s\theta_3))) \\ &+ L_2s\theta_2s\theta_1 - L_3c\theta_1s\theta_3 - L_5c\theta_5(c\theta_4(c\theta_1s\theta_3 \\ &- c\theta_2c\theta_3s\theta_1) - s\theta_1s\theta_2s\theta_4) + L_6s\theta_6(s\theta_4(c\theta_1s\theta_3 \\ &- c\theta_2c\theta_3s\theta_1) + c\theta_4s\theta_1s\theta_2) - L_4c\theta_4(c\theta_1s\theta_3 \\ &- c\theta_2c\theta_3s\theta_1) - L_6c\theta_6(c\theta_5(c\theta_4(c\theta_1s\theta_3 \\ &- c\theta_2c\theta_3s\theta_1) - s\theta_1s\theta_2s\theta_4) - s\theta_5(s\theta_1c\theta_3 \\ &+ c\theta_2s\theta_1s\theta_3)) + L_7c\theta_7(s\theta_6(s\theta_4(c\theta_1s\theta_3 - c\theta_2c\theta_3s\theta_1) \\ &+ c\theta_4s\theta_1s\theta_2) - c\theta_6(c\theta_5(c\theta_4(c\theta_1s\theta_3 - c\theta_2c\theta_3s\theta_1) \\ &- s\theta_1s\theta_2s\theta_4) - s\theta_5(c\theta_1c\theta_3 + c\theta_2s\theta_1s\theta_3))) \\ &+ L_3c\theta_2c\theta_3s\theta_1 + L_4s\theta_1s\theta_2s\theta_4 \end{aligned} \quad (9)$$

$$\begin{aligned} p_z &= L_1 - L_2s\theta_2 + L_6s\theta_6(c\theta_2c\theta_4 + c\theta_3s\theta_2s\theta_4) \\ &- L_7s\theta_7(s\theta_6(c\theta_5(c\theta_2s\theta_4 - c\theta_3c\theta_4s\theta_2) - s\theta_2s\theta_3s\theta_5) \\ &- c\theta_6(c\theta_2c\theta_4 + c\theta_3s\theta_2s\theta_4 + c\theta_3s\theta_2s\theta_4)) \\ &- L_3c\theta_3s\theta_2 + L_4c\theta_2s\theta_4 + L_6c\theta_6(c\theta_5(c\theta_2s\theta_4 \\ &- c\theta_3c\theta_4s\theta_2) - s\theta_2s\theta_3s\theta_5) + L_5c\theta_5(c\theta_2s\theta_4 \\ &- c\theta_3c\theta_4s\theta_2) + L_7c\theta_7(c\theta_6(c\theta_5(c\theta_2s\theta_4 - c\theta_3c\theta_4s\theta_2) \\ &- s\theta_2s\theta_3s\theta_5) + s\theta_6(c\theta_2c\theta_4 + c\theta_3s\theta_2s\theta_4)) \\ &- L_4c\theta_3c\theta_4s\theta_2 - L_5s\theta_2s\theta_3s\theta_5. \end{aligned} \quad (10)$$

Fitness function

To heuristic algorithms to be used in engineering problems, the problem must be expressed as a fitness function [41]. In this study, the heuristic algorithm first creates seven joint angles at each step and then the position of the end effector in the x , y and z coordinates is calculated using forward kinematics equations of the robot manipulator. The proximity of this calculated point to the desired point is controlled

Fig. 6 Illustration of position error

by the preferred Euclidean distance equation as the fitness function in this study.

$$\text{Pozisyon Hatası} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}. \quad (11)$$

Figure 6 illustrates the position error which is the basis of this study. (x_1, y_1, z_1) calculated by heuristic algorithms represents the current position of the end effector and (x_2, y_2, z_2) which is manually determined is the desired position to reach.

Results

In this study, random descending velocity PSO technique is examined in depth in terms of position error and computation time. For this reason, in addition to the results of this technique, comparative results with other techniques have been presented in the test procedures. Since the new

technique introduced is an IW strategy, the comparison process is performed first with IW techniques and then with heuristic algorithms.

Scenario 1

In this scenario, RDV PSO technique is compared with Random IW, linear decreasing IW, random chaotic IW and traditional PSO techniques. Because IW strategies improve the speed of particles, in this scenario is also presented in graphs about the speed averages of particles.

Figure 7 shows the position error values obtained using different IW techniques together with RDV PSO. This figure shows that the values obtained with RDV PSO, which was introduced as the new IW technique within the scope of this study, are decisively ahead of other IW techniques. Therefore, it would not be wrong to say that the RDV IW technique has improved the result achieved by the traditional PSO technique by one billion times.

Figure 8 shows the calculation times during which the IW techniques subjected to the test have achieved a minimum position error. This figure clearly shows that although the RDV IW technique is superior to the other IW techniques in terms of position error, there is no advantage in terms of calculation time.

Figure 9 shows the velocity averages of conventional PSO, random IW techniques, linear decreasing IW and random chaotic IW techniques. As seen in figure, the velocity averages vary in proportion to the obtained position error. According to these graph, it can be said that linear decreasing IW and random chaotic IW techniques produce better value compared to both conventional and random IW techniques. This is because the stabilization of speed averages in the traditional PSO technique occurs after the 200th iteration, and in the random IW technique this happens at about

Fig. 7 Position error of different IW techniques

Fig. 8 Calculation times graph obtained by other IW techniques

Fig. 9 Velocity averages for conventional PSO and random IW techniques

150th iteration. However, stabilization of velocity averages in linear decreasing IW and random chaotic IW techniques was completed in approximately 100 iterations.

Figure 10 shows the velocity averages of the particles approaching the target with the RDV IW technique introduced for the first time in this study. It is clear that speed averages have stabilized in almost the first 20 iterations. Of course, the position error value obtained in parallel with this situation seems to be much better than the other four techniques.

Scenario 2

In this scenario, RDV IW technique is compared with firefly algorithm, artificial bee colony and quantum PSO algorithms. For this purpose, each algorithm was run 100 times in succession and 100 different results were obtained. In the graphs, the best (min), worst (max) and average value (avg) of these 100 values are particularly indicated.

Figure 11 shows the position error values obtained by five different heuristic algorithms, and Fig. 12 shows the

Fig. 10 Velocity averages of RDV IW technique **Fig. 11** Position error values obtained by heuristic algorithms **Fig. 12** Computation time values obtained by heuristic algorithms

calculation times of these algorithms. Of these five algorithms, the best values were obtained by quantum PSO. Afterwards, it is observed that RDV IW technique is very successful in convergence problems. In terms of calculation times, it is not overlooked that RDV IW technique shares the last place with PSO.

Discussion

The new technique introduced in this study is based on keeping changes in particle velocity values as minimum as possible at positions close to the optimum value. Because, the logic that is the basis of this technique is equivalent to the process of getting the ball into the hole in golf game. As with the first hit in the golf game, the randomly generated values will likely be smaller than *Delta*, as *Delta* will be at the maximum value in the first iterations. In this case, the particles will convergence to the target at normal speeds. At the end of the iteration, the randomly generated number will be greater than *Delta* since *Delta* will take its minimum values. In this case, the particle velocities will decrease and they will converge to the target in minimum steps.

The most important parameters providing in this technique are *delta*, *alpha* and *alpha_dump*. The best results were obtained when the $\alpha = 0.3$ and $\alpha_dump = 0.95$ tested in this study. In this way, better results were obtained. In this regard, Fig. 13 below is the best illustration of this.

Figure 13 clearly illustrates how the RDV technique steadily approximates particles to the optimum value. Especially towards the end of the algorithm, that is, as it progresses to the maximum value of the iteration, the change in the classical PSO takes place in its normal course, while in the RDV technique it occurs at a minimum level. Just like in the game of golf. Another important point is that, as can be seen in Fig. 13, the classical PSO search operation is carried out intensely in areas far from the optimum value. However, in RDV technique, the region where the search is concentrated are the last stages of the iteration, that is, positions close to the optimum value.

Parallel to all of these, Table 2 is another proof that the study is firmly grounded. Because the RDV technique has a separate control stage for the minimum change of particle velocities. Especially towards the end of the iteration, this change decreases even more at every stage. Because at this stage, the *Delta* parameter is at minimum levels, so it is likely to be smaller than the randomly generated value. Thus,

Fig. 13 Comparison of RDV technique and Conventional PSO in terms of particle velocities

Table 2 Comparison of limit exceeding number of particle values in PSO and RDV

	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6	θ_7
Range	$-90 < \theta_1 < 90$	$-180 < \theta_2 < 0$	$-90 < \theta_3 < 90$	$-90 < \theta_4 < 90$	$-90 < \theta_5 < 90$	$-90 < \theta_6 < 90$	$-90 < \theta_7 < 90$
PSO limit count	56,947	45,523	56,938	57,145	56,732	56,815	45,957
RDV limit count	18,505	19,346	17,941	19,892	15,241	15,866	16,301
Reduction rate (~)	3×	2.35×	3.17×	2.87×	3.72×	3.58×	2.82×

the particle values make less attempts in RDV technique compared to PSO to exceed the specified limit values.

Conclusion

In this study, the value obtained by particle swarm optimization algorithm has been improved by one billion times based on the golf ball movements. The most important factor to consider when using this technique is that the next shot is randomly weaker than the previous one. Thus, the particle actually slows down at each iteration, enhancing its convergence and so, introducing the idea of random descending (RDV). Two different scenarios were used to demonstrate the performance of the technique: In one, the technique was compared with some IW strategies; in the other, RDV IW is compared with some heuristic algorithms. In both scenarios, the results were compared in terms of position error and calculation time. As a result of comparison with IW techniques, the best position error values have been obtained by new technique called RDV IW. In the other case, the best values after the results obtained with quantum PSO have been also obtained by RDV IW. In terms of calculation time, there is no significant advantage or weakness of RDV IW technique compared to other techniques or algorithms.

Funding Manuscript is not supported by any person or organization.

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interests regarding the publication of this paper.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Amiri R, Sardroud J, Soto BD (2017) BIM-based applications of metaheuristic algorithms to support the decision-making process: uses in the planning of construction site layout. *Proc Eng* 196:558–564
2. Avci M, Topaloglu S (2015) An adaptive local search algorithm for vehicle routing problem with simultaneous and mixed pickups and deliveries. *Comput Ind Eng* 83:15–29
3. Cai X, Wang H, Cui Z, Cai J, Xue Y, Wang L (2018) Bat algorithm with triangle-flipping strategy for numerical optimization. *Int J Mach Learn Cybern* 9:199–215
4. Chan C, Bai H, He D (2018) Blade shape optimization of the Savonius wind turbine using a genetic algorithm. *Appl Energy* 213:148–157
5. Dash K, Choudhury B, Senapati S (2019) Inverse kinematics solution of a 6-DOF industrial robot. In: *Soft computing in data analytics*. Springer, Singapore, pp 183–192
6. Delgarm N, Sajadi B, Kowsary F, Delgarm S (2016) Multi-objective optimization of the building energy performance: a simulation-based approach by means of particle swarm optimization (PSO). *Appl Energy* 170:293–303
7. Dereli S, Köker R (2018) IW-PSO approach to the inverse kinematics problem solution of a 7-Dof serial robot manipulator. *Sigma J Eng Nat Sci* 36:77–85
8. Dereli S, Köker R (2020) Calculation of the inverse kinematics solution of the 7-DOF redundant robot manipulator by the firefly algorithm and statistical analysis of the results in terms of speed and accuracy. *Inverse Probl Sci Eng* 28:601–613
9. Esmin A, Coelho R, Matwin S (2015) A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data. *Artif Intell Rev* 44:23–45
10. Fernandes JJ (2018) Kinematic and dynamic analysis of 3PUU parallel manipulator for medical applications. *Proc Comput Sci* 133:604–611
11. Fu H, Li Z, Liu Z, Wang Z (2018) Research on big data digging of hot topics about recycled water use on micro-blog based on particle swarm optimization. *Sustainability* 10:2488–2493
12. Gordan B, Armaghani D, Hajihassani M, Monjezi M (2016) Prediction of seismic slope stability through combination of particle swarm optimization and neural network. *Eng Comput* 32:85–97
13. Gogate GR (2016) Inverse kinematic and dynamic analysis of planar path generating adjustable mechanism. *Mech Mach Theory* 102:103–122
14. Gorkemli B, Karaboga D (2019) A quick semantic artificial bee colony programming (qsABCP) for symbolic regression. *Inf Sci* 502:346–362
15. Guedria N (2016) Improved accelerated PSO algorithm for mechanical engineering optimization problems. *Appl Soft Comput* 40:455–467
16. Harada R, Takano Y, Shigeta Y (2016) TaBoo search algorithm with a modified inverse histogram for reproducing biologically relevant rare events of proteins. *J Chem Theory Comput* 12:2436–2445

17. Hasanipanah M, Naderi R, Kashir J, Noorani S, Qaleh A (2017) Prediction of blast-produced ground vibration using particle swarm optimization. *Eng Comput* 33:173–179
18. İnan T, Baba A (2019) Particle swarm optimization-based collision avoidance. *Turk J Electr Eng Comput Sci* 27:2137–2155
19. Jafari S, Nikolaidis T (2018) Meta-heuristic global optimization algorithms for aircraft engines modelling and controller design; A review, research challenges, and exploring the future. *Prog Aersp Sci* 104:40–53
20. Kareem E, Mejbel A (2018) Traffic light controller module based on particle swarm optimization (PSO). *Am J Artif Intell* 2:7–15
21. Kaveh A, Javadi S (2019) Chaos-based firefly algorithms for optimization of cyclically large-size braced steel domes with multiple frequency constraints. *Comput Struct* 214:28–39
22. Kennedy J, Eberhart R (1995) Particle swarm optimization (PSO). In: *IEEE international conference on neural networks*, Perth, pp 1942–1948
23. Kumar A, Pant S, Ram M (2017) System reliability optimization using gray wolf optimizer algorithm. *Qual Reliab Eng Int* 33:1327–1335
24. Marini F, Walczak B (2015) Particle swarm optimization (PSO). A tutorial. *Chemometr Intell Lab Syst* 149:153–165
25. Mirjalili S, Lewis A (2016) The whale optimization algorithm. *Adv Eng Softw* 95:51–67
26. Naji N (2017) A review of the metaheuristic algorithms and their capabilities (particle swarm optimization, firefly and genetic algorithms). *Int J Curr Eng Technol* 7:921–925
27. Şahin O, Akay B (2016) Comparisons of metaheuristic algorithms and fitness functions on software test data generation. *Appl Soft Comput* 49:1202–1214
28. Pathak V, Kumar S, Nayak C, Rao N (2017) Evaluating geometric characteristics of planar surfaces using improved particle swarm optimization. *Meas Sci Rev* 17:187–196
29. Patriarca R, Hosseini S (2019) Modeling and quantification of resilience in complex engineering systems. *Complexity*. <https://doi.org/10.1155/2019/1038908>
30. Rao R, Waghmare G (2017) A new optimization algorithm for solving complex constrained design optimization problems. *Eng Optim* 49:60–83
31. Rayner RM, Sahinkaya MN, Hicks B (2017) Improving the design of high speed mechanisms through multi-level kinematic synthesis, dynamic optimization and velocity profiling. *Mech Mach Theory* 118:100–114
32. Sama M, Pellegrini P, D'Ariano A, Rodriguez J, Pacciarelli D (2016) Ant colony optimization for the real-time train routing selection problem. *Transport Res Part B Methodol* 85:89–108
33. Shang J, Tian Y, Liu Y, Liu R (2018) Production scheduling optimization method based on hybrid particle swarm optimization algorithm. *J Intell Fuzzy Syst* 34:955–964
34. Şevkli A, Sevilgen FE (2010) StPSO: strengthened particle swarm optimization. *Turk J Electr Eng Comput Sci* 18:1095–1114
35. Wang H, Wang W, Cui L, Sun H, Zhao J, Wang Y et al (2018) A hybrid multi-objective firefly algorithm for big data optimization. *Appl Soft Comput* 69:806–815
36. Wang M, Luo J, Walter U (2015) Trajectory planning of free-floating space robot using particle swarm optimization. *Acta Astronaut* 112:77–88
37. Wei L, Zhang Z, Zhang D, Leung S (2018) A simulated annealing algorithm for the capacitated vehicle routing problem with two-dimensional loading constraints. *Eur J Oper Res* 265:843–859
38. Xin J, Li S, Sheng J, Zhang Y, Cui Y (2019) Application of improved particle swarm optimization for navigation of unmanned surface vehicles. *Sensors*. <https://doi.org/10.3390/s19143096>
39. Xu G, Cui Q, Shi X, Ge H, Zhan ZH, Lee HP, Liyang Y, Tai R, Wu C (2019) Particle swarm optimization based on dimensional learning strategy. *Swarm Evolut Comput* 45:33–51
40. Yang P, Miao L, Xue Z, Ye B (2015) Variable neighborhood search heuristic for storage location assignment and storage/retrieval scheduling under shared storage in multi-shuttle automated storage/retrieval systems. *Transport Res Part E Logist Transport Rev* 79:164–177
41. Zhang Y, Wang S, Ji G (2015) A comprehensive survey on particle swarm optimization algorithm and its applications. *Math Probl Eng*. <https://doi.org/10.1155/2015/931256>
42. Zhang Q, Lu J, Jin Y (2020) Artificial intelligence in recommender systems. *Complex Intell Syst*. <https://doi.org/10.1007/s40747-020-00212-w>
43. Zhao W, Zhang Z, Wang L (2020) Manta ray foraging optimization: an effective bio-inspired optimizer for engineering applications. *Eng Appl Artif Intell*. <https://doi.org/10.1016/j.engappai.2019.103300>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.