Article

Comparison Study of the PSO and SBPSO on Universal Robot Trajectory Planning

Issraa Jwad Kazim ^{1,2,*} , Yuegang Tan ¹ and Ruiya Li ^{1,*}

¹ School of Mechanical and Electronic Engineering, Wuhan University of Technology, Wuhan 430070, China; ygtan@whut.edu.cn

² Computer Technology Engineering, ALMustafa University College, Baghdad 10067, Iraq

* Correspondence: israa.cet@almustafauniversity.edu.iq (I.J.K.); liruiya@whut.edu.cn (R.L.)

Abstract: Industrial robots were modified over the years. The benefit of robots is making production systems more efficient. Most methods of controlling robots have some limitations, such as stopping the robots. The robot stops by various reasons, such as collisions. The goal of this study is to study the comparison of improving the Artificial Potential Field (APF) by the traditional Particle Swarm Optimization (PSO) algorithm and the Serendipity-Based PSO (SBPSO) algorithm to control the path of a universal robot UR5 with collision avoidance. Already, the metaheuristic algorithm kinds deal with a premature convergence. This paper presents a new approach, which depends on the concept of serendipity and premature convergence applied to the path of the universal manipulator UR5 and also compares it with traditional the PSO. The features of the SBPSO algorithm prototype are formalized in this paper using the concept of serendipity in two dimensions: intelligence and chance. The results showed that the SBPSO is more efficient and has better convergence behavior than the traditional PSO for controlling the trajectory planning of the UR5 manipulator with obstacle avoidance.

Keywords: Particle Swarm Optimization (PSO); Serendipity-Based PSO (SBPSO); Artificial Potential Field (APF); serendipity; premature convergence; scout particles

Citation: Kazim, I.J.; Tan, Y.; Li, R. Comparison Study of the PSO and SBPSO on Universal Robot Trajectory Planning. *Appl. Sci.* **2022**, *12*, 1518. <https://doi.org/10.3390/app12031518>

Academic Editor:
Alessandro Gasparetto

Received: 24 November 2021

Accepted: 15 January 2022

Published: 30 January 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Using robots in manipulation tasks had a significant growth in the context of industrial production in recent years [1]. The aforementioned diffusion of robots in an industrial environment provided, over the years, that various methods were developed in order to monitor and control manipulator and mobile robots [2]. With this, they acquired the ability to operate in dangerous environments for humans, for example, beyond Earth's atmosphere, in aquatic explorations, and in the transport of materials [3]. The purpose of this paper is to present the simulation of article Swarm Optimization (PSO) and the Serendipity-Based PSO (SBPSO) algorithm to modify APF approach, in order to generate free trajectory of collision of a UR5 universal manipulator robot. These metaheuristics will be used to optimize the repulsion constants of the potential field of the starting point and obstacles and the attraction constant of the end point, which are directly related to the intensity of the force resulting from the potential field.

Particle Swarm Optimization (PSO) is a heuristic algorithm dependent on social behavior of a flock of birds. The method was recommended by Kennedy and Eberhart in 1995 [4]. They had an objective to seek the optimal solution, in a space of search, by exchanging information between individuals of a population to determine which trajectory they should take in the search space. In this paper, the PSO was used because it has a number of advantages including rapid convergence and ease of implementation, but often faces a problem where their particles get “stuck” in great places. This problem is called premature convergence [5]. In this algorithm, individuals called particles function as a set of birds searching for a flight shape and considering the position of each particle within

space; this is due to a curve of evolution within society or set of particles. Each particle will have its success defined by the general trend of the population [5].

Many engineering problems are common to study types of its behavior process, as a rule, such as a dynamic system which can be modeled by a set of equations that consider the same variables and is subjected to a certain evolution over time. A set of techniques which has been applied to this kind of problem is succeeded on what it involves and is known as bio-inspired computing techniques [6]. These techniques applied to the different domains such as: prediction systems [7,8], power systems [9,10], telecommunications [11,12], intrusion detection systems [13,14], and others. In the context of bio-inspired computing, Swarm Intelligence consists of a set of metaheuristics based on some living beings whose behavior emergencies can result in an ability to resolve.

It is important to mention that the work of the PSO and SBPSO algorithm is to enhance the artificial potential field approach in order to obstacle avoidance collision.

In order to enhance work of the PSO by delaying premature convergence, the serendipity concept will be used. Serendipity is a term used to describe a fortunate situation that happens by chance. Recommendation systems are developed as a solution to the problem of information overload. The accuracy of the recommendations has been the focus of recommendation system research [15,16]. Accuracy, on the other hand, resulted in a problem known as overspecialization. This happens when the system only refers to items connected to the user's profile. Serendipity is a strategy that can be used to resolve overspecialization. According to [17], a recommendation system provides products based on the user's interest profile and converges on recommendations when these items do not meet the real expectations of the user. A metaheuristic algorithm converges to a place without considering other more suitable solutions in order to suggest items that may be more suitable. It is possible to observe the correlation between overspecialization and premature convergence. Some approaches in the literature use scout particles with the objective to improve the standard PSO algorithm's performance [18,19]. These approaches are stochastic, but is a benefit when combined with serendipity techniques. This work proposes Serendipity-Based Particle Swarm Optimization (SBPSO) using a combination of the particles Girl Scouts with serendipity to improve the exploration of the algorithm in robotic searches.

Serendipity, according to the Cambridge Dictionary, "is the act of obtaining important or interesting things by chance." Unlike other traditional definitions, which exclusively treat serendipity as a synonym for "chance," serendipity is a mix of chances [20]. In [21], a formal approach was used for multiple categories to present the concept of serendipity. In order to present four occurrences that cause serendipity, logical equations called Serendipity Equations were used. These occurrences are linked to four various kinds of serendipity: (a) pure serendipity; (b) fake serendipity; (c) serendipity with wrong knowledge; and (d) serendipity without an inspiration metaphor.

The contribution of this paper is to obtain the collision-free trajectories by using the PSO and the SBPSO algorithms, also comparing the effect of the PSO algorithm and SBPSO algorithm to modify APF approach on the trajectory of a universal robot UR5.

2. Materials and Methods

2.1. Artificial Potential Field

The elements and resultant of potential energy, contained from an attractive potential field and repulsive potential field. Equations (1) and (2) represent the attractive potential field and attraction force, respectively, by setting the position of end effector $x = (x, y)^T$ and the point of goal coordinate $x_G = (x_g, y_g)$. The UR5 robot has been attracted to the goal object before reaching it. The goal object has the bigger attraction force.

$$U_{att}(x) = \frac{1}{2} K_{att} \cdot d^2(x, x_g) \quad (1)$$

K_{att} : Attraction potential field constant.

$d(x, x_g) = \|x_g - x\|$: Distant between the first coordinate of manipulator and last coordinate of it (i.e., pick and place operation of the end-effector).

The attraction force of the UR5 robot is the negative gradient of attraction potential field

$$F_{att}(x) = -\nabla U_{att}(x) = k_{att} \cdot d(x, x_g) \quad (2)$$

The direction of attraction force is a line between first coordinate of the end-effector and the goal object coordinate.

In Equations (3) and (4), repulsive potential field and repulsive force respectively [22,23]. The repulsive potential field is dependent on distant. When the UR5 manipulator reaches the obstacle's effect region, it is subjected to repulsion force. The repulsion is greater for the closest (closer) obstacle. The coordinate of obstacle is $x_o = (x_o, y_o)$; the $U_{rep}(x)$ is generated between the manipulator and obstacle based on the distance.

$$U_{rep}(X) = \frac{1}{2} K_{rep} \left(\frac{1}{d(x, x_o)} - \frac{1}{d_o} \right)^2 \quad (3)$$

where $d(x, x_o) \leq d_o$. The $U_{rep}(X) = 0$ when $d(x, x_o) > d_o$.

K_{rep} : The repulsive potential field constant.

$d(x, x_o) = \|x_o - x\|$: Distant between the first coordinate of manipulator and obstacle of it (i.e., in the pick and place operation of the end-effector).

d_o : Influence range of $U_{rep}(x)$.

The repulsive force of the UR5 robot is the negative gradient of repulsion potential field $U_{rep}(X)$.

$$F_{Rep}(x) = -\nabla U_{Rep}(x) = \frac{1}{2} K_{Rep} \left(\frac{1}{d(x, x_o)} - \frac{1}{d_o} \right) \frac{\nabla d(x, x_o)}{d^2(x, x_o)} \quad (4)$$

when $d(x, x_o) \leq d_o$ and zero other where.

The direction of repulsive force is the line between manipulator and obstacle object.

In Equations (5) and (6) the resultant potential field and resultant force respectively effecting on the UR5 manipulator.

$$U(x) = U_{att}(x) + U_{Rep}(x) \quad (5)$$

The resultant force $F(x)$ is

$$F(x) = -\nabla U(x) = -\nabla U_{att}(x) - \nabla U_{Rep}(x) = F_{att}(x) + F_{Rep}(x) \quad (6)$$

$F_{Rep}(x)$ is the repulsive force from more than one obstacle and the $F_{att}(x)$ is the attractive force of goal.

2.2. Particle Swarm Optimization (PSO)

The PSO algorithm can be used to solve problems of trajectory planning using a state function value and an Interactive Optimization by Particle Swarms (IPSO) [24,25]. The position at time (t) is updated through $x_i(t)$ and in the future tense (t + 1) will be updated to $x_i(t + 1)$ as in Equation (7)

$$x_i(t + 1) = x_i(t) + v_i(t + 1) \quad (7)$$

The velocity of particles is $v_i(t)$ [12]. Each particle will have a cognitive component, which will be a relationship of the distance between the optimal solution and the social component, which is the collective's understanding of the particle's existence. For this problem, the PSO was applied globally (Global best PSO), with Equation (8) updating the particle velocity:

$$v_{ij}(t+1) = v_{ij}(t) + C_1 r_1(t) [y_{ij}(t) - x_{ij}(t)] + C_2 r_2(t) [\hat{y}_{ij}(t) - \hat{x}_{ij}(t)] \quad (8)$$

$v_{ij}(t)$ is the particle's velocity in a dimension at time t . The acceleration parameters are C_1 , and C_2 . The y_{ij} and $\hat{y}_{ij}$ which is the best position from the start [12], provide the information of the best particle. A velocity is assigned to each particle in the PSO algorithm. Particles move around the search space at different speeds, which are constantly changed based on their previous actions. Finally, during the search process, particles have a tendency to go via the best research area [12]. The trajectories of the joints of the handler can be optimized using a strategy which divides the robot into several aptitude functions to perform the optimal fit with modified PSO with crossover operator [26]. The simulation results are presented for manipulator trajectory planning of the UR5 using redundant kinematics mounted on a floating space.

Several simulations of the PSO algorithm were carried out with capability of changing the number of generations and particles. The best result of the algorithm was for 30 particles and for 2000 iterations that had the best values for the APF, and there was a convergence between the average of all particles and the best particle. For the PSO algorithm, the following were used element values:

Quantity of particles = 30 particles

Cognitive and social parameters (learning rates - C_1 and C_2) = 2

Iterations = 2000

Initial population generation = $[-1, 1]$;

Inertia factor (w) = 0.5

The initial steps of traditional PSO algorithm represent in Algorithm 1.

Algorithm 1: The PSO algorithm is presented below

Initialization:

01: initialize the particle cloud

02: repeat

Loops 1:

03: for $i = 1$ to m

04: if $f(x_i) < f(p_i)$ then $p_i = x_i$

05: if $f(x_i) < f(g)$ then $g = x_i$

06: end if

07: end if

Loop.2:

08: for $j = 1$ to n

09: $r_1 = \text{rand}()$, $r_2 = \text{rand}()$

10: $v_{ij} = wv_{ij} + c_1 r_1(p_i - x_{ij}) + c_2 r_2(g_j - x_{ij})$

11: end to

Function:

12: $x_i = x_i + v_i$

13: end to

14: until you meet the stop criteria

The PSO algorithm's stopping criterion is the algorithm's generations number. In metaheuristic algorithms, elements representing solutions identified in a particular iteration for a given search space are added to the set E . Additionally, e_{ij} is an element that represents a solution i in iteration j , belonging to set E . The element $e_{ij}^* \in E$ represents the fitness value of solution i which is the best value discovered in iteration j . A search space's probable solutions are denoted by the letter S . So, E is a subset of S . Otherwise, this solution is considered an occasional solution given by Equation (9):

$$\text{CHANCE} = S - E \quad (9)$$

Chance_{ij} is a serendipitous element.

When chanceij's fitness value is greater than element of e^*_{ij} fitness value, the SRD forms occasional solutions according to Equation (10):

$$SRD = [(srd_{ij} * fitness(chanceij)) < fitness(e^*_{ij})] \quad (10)$$

The concept of serendipity is presented in this work. There are two important components in this work: irregularity and acceptability.

(a) Irregularity, because it is an element in iteration j that does not belong to set E .

(b) Acceptability because it indicates a solution with a higher fitness value than the element e^*_{ij} .

2.3. Serendipity-Based Particle Swarm Optimization

The creation of serendipity-based recommendation mechanisms is a new and unsolved research challenge. "Something excellent or useful" might be considered as a candidate solution to replace after a specific iteration's number that occurred when the concept of serendipity is applied to metaheuristic algorithms and their optimization applications.

The PSO algorithm has gained a lot of attention from the community's search to solve optimization problems in the real world because of the simplicity of which it would be implemented, and the high quality of its output PSO is a stochastic approach for modeling birds social behavior. The solution in this type of algorithm is represented by a particle that "flies" over the search space looking for the best solution after a set number of iterations.

Particle motion is based on two types of information: pBest and gBest. pBest guides the particle to the swarm's optimal location, whereas gBest impacts the particle's progress to the swarm's best position. After determining pBest and gBest, each particle uses Equations (11) and (12) to update its position and velocity:

$$V_{id}^{(t+1)} = w.V_{id}^t + C_1.r_1(p_{id}^t + X_{id}^t) + C_2.r_2(p_{gd}^t + X_{id}^t) \quad (11)$$

$$X_{id}^{(t+1)} = X_{id}^t + V_{id}^{(t+1)} \quad (12)$$

In Equation (11), the term w represents the inertial velocity of the particle.

X_{id}^t and V_{id}^t are the position and velocity of particle i at time t .

The best fitness value of particle i and the best fitness value of all particles in the swarm up to time t , respectively, are p_{id}^t and p_{gd}^t .

C_1 is a coefficient of the particle self-exploration, while C_2 is a particle's movement in the direction of the global displacement of the swarm in search space.

r_1 and r_2 are random values in the interval $[0, 1]$.

Serendipity presented as an ability to make lucky discoveries by way of intelligence and chance. In this work, the term "intelligence" should be understood as the quality of perceiving an event. This concept of serendipity is useful and interesting since it defines where serendipity can be applied to a traditional PSO algorithm, in addition to intelligence type, to improve the PSO's behavior through serendipity-inspired options. Three natural techniques to implementing this enhancement are defined. The first is based on point random choice modifications accessible in the conventional PSO method. The second requires the addition of a mechanism for implementing a PSO algorithm using serendipity [27,28]. The third approach is a hybrid of the previous two approaches. Because it is important for an optimization technique and the best accessible solution on the current iteration, in all of these techniques, consider the search space's "lucky discovery points."

This work is dependent on the third approach. In [17], a strategy was proposed to generate serendipity dependent on a perceptual model [29], which combines two strategies used in the recommendation systems [30]: Pasteur's principle and blind luck. To detect the capabilities and apply insights to seize the moment, Pasteur's principle is used. The principle uses scout particles in order to inspect regions in the space of unexplored search by the swarm, according to Equations (11) and (12). The blind luck strategy is implemented through the use of Girl Scout particles. They implement the random dimension and

complement the space exploration of the traditional PSO generating additional diversity. These strategies are the implementation of the concept of intelligence or “the ready mind” in order to get a new algorithm called Serendipity-Based Particle Swarm Optimization (SBPSO). This algorithm used scout particles to search on peaks or valleys around the best in iterations points. Scout particles slow down the convergence time of the optimization algorithm and can be utilized to enhance algorithm capacity and scanning. It is worth remembering that adding the scout particles can help the swarm behavior. In the SBPSO method, the Girl Scouts are utilized to provide answers that were better than the swarm’s best particle. Equation (13) represents the velocity of a Girl Scout particle k :

$$V_{kd}^{(t+1)} = w.V_{kd}^t + C_3.r_3(X_{kd}^t + X_{best}^t) \quad (13)$$

C_3 is the variety coefficient and r_3 is a value random in the interval $[-1, 1]$. X_{kd}^t is the position of the Girl Scout particle k and X_{best}^t is the best particle’s position in the swarm at time t . Equation (14) represents the position of a scout particle k :

$$X_{kd}^{(t+1)} = V_{kd}^{(t+1)} - X_{best}^t \quad (14)$$

Algorithm 2 describes the initial step of SBPSO algorithm. The position and speed of each swarm particle and each scout particle which are randomly initialized (lines 1–2). The fitness value of each particle is then calculated to determine which particle is the best (lines 4–11) and for the scout particles (lines 12–19). In the next step the best particle will be found and named the new gBest particle (lines 20–26). After this, the particles start the inspection process at the proximity of the gBest particle (lines 27–31). For this, the algorithm creates n adjacent points (AP) are present in Equation (15):

$$AP_n = best + \alpha \cdot R \quad (15)$$

Best: is the best particle current.

α is a real constant positive.

R is a $1 \times \text{dim}$ matrix with entries randomly distributed in the range $[-1, 1]$, where dim is the current particle’s dimension.

According to Equation (16a), for each AP_n that was created, we define a vector $\vec{d}_n$ that represents the segment oriented $\overleftarrow{best}AP_n$:

$$\vec{d}_n = AP_n - best \quad (16a)$$

$best$ and AP_n define a single vector $\vec{d}_n$ which represents a direction, so there are infinite inspection points (IP) with address $\overleftarrow{best}IP_n^m$ which is the same as $\overleftarrow{best}AP_n$. Thus, for each AP_n created, two IPs are chosen dependent on Equation (16b):

$$IP_n^m = best \pm (\|\vec{u}\| / \|\vec{d}_n\|) \cdot \vec{\lambda} \cdot \vec{d}_n \quad (16b)$$

IP_n^m is an inspection point m defined on the straight line that passes through $best$ and AP_n , $\vec{u}$ is the vector $\vec{d}_n$ that has the smallest norm and λ is a non-zero real constant.

Two criteria are defined to find an inspection point that can replace the particle gBest current. The criteria are: (1) IP with fitness value is better than current particle gBest and (2) IP with fitness value is better than others. However, if there is more than one IP that meets these two criteria, a lottery is performed to choose one of them. According to Pasteur’s principle, luck favors the prepared mind. In this work, the concept of “mind prepared” is implemented through inspections around the particle gBest. As a result, there is the implementation of the intelligence dimension. Figure 1 presents an example of the choice of random inspection points in a 2D space.

Algorithm 2: SBPSO pseudocode using scout particles

Initialization:

01: initialize swarm speed and position

02: initialize speed and position of the Girl Scouts

03: while stop criteria not satisfied of the

loop.1:

04: for all particle i in S do

05: calculate fitness value

Condition:

06: if $\text{fit}(x_{id}) < \text{fit}(p\text{Best}_i)$ then $p\text{Best}_i \leftarrow x_{id}$

07: end if

08: end for

Function:

09: $\text{best}_{\text{swarm}} \leftarrow \text{get Best Particle Swarm}()$

loop.2:

10: for all Girl Scout k in S do

function:

11: determine fitness value

condition:

12: if $\text{fit}(\text{Girl scout}_{kd}) < \text{fit}(p\text{Best}_{kd})$ then $p\text{Best}_{kd} \leftarrow \text{Girl scout}_{kd}$

13: end if

14: end for

function

15: $\text{best}_{\text{Girl Scout}} \leftarrow \text{getBest Girl Scout}()$

condition:

16: if $\text{fit}(\text{best}_{\text{swarm}}) < \text{fit}(\text{best}_{\text{Girl Scout}})$ and $\text{fit}(\text{best}_{\text{swarm}}) < \text{fit}(\text{gBest})$ then $\text{gBest} \leftarrow \text{best}_{\text{swarm}}$ 17: else if $\text{fit}(\text{best}_{\text{Girl Scout}}) < \text{fit}(\text{best}_{\text{swarm}})$ and $\text{fit}(\text{best}_{\text{Girl Scout}}) < \text{fit}(\text{gBest})$ then $\text{gBest} \leftarrow \text{best}_{\text{Girl Scout}}$

Girl Scout

18: end if

19: end if

20: creates n adjacent points (AP) using Equation (15)

loop.3

21: for all AP of the

22: define n director vectors $\vec{d}$ using Equation (16)23: for each vector $\vec{d}$ do

24: Create two IP using Equation (17)

25: end for

26: end for

27: $\text{gBest} \leftarrow \text{getBest Inspection Point}()$

Loop.4

28: for all particle i in S do

29: determine speed using Equation (11)

30: update position using Equation (12)

31: end for

Loop.5

32: for each Girl Scout k in S do

33: calculate speed using Equation (13)

34: update position using Equation (14)

35: end for

36: if (swarm stagnated) and (swarm converged) then Resets Girl Scout Position and Speed

37: end if

38: end while

Figure 1. Example of random selection of 8 inspection points around the gBest particle, in a 2D space. This is the result for each of the 4 adjacent points defined.

In Figure 1, the direction of each of the straight lines is stochastic, since they connect the gBest to the inspection point tools. During all iterations, the lines have a reasonable perception of the behavior of the fitness function, even in an n-dimensional space.

Again, the swarm particles' velocity is calculated, and the scout particles' position is updated (lines 32–35). Finally, the program assesses the swarm's state in terms of premature convergence and stagnation (row 36). The SBPSO restarts the position and velocity of the scout particles in (line 38):

2.4. Methods for Detecting Premature Convergence

The PSO algorithm is a mechanism that can generate good trajectories in a short amount of time. However, the algorithm runs the risk of prematurely converging. The swarm will be converged prematurely when the solution proposal approaches a good location instead of the problem being optimized. Premature convergence is caused by a decrease of variety in the search space, which enables the swarm to become stagnant. After premature convergence begins, the particles continue to shut on each other in a very small region of the search space [27]. In order to avoid swarm stagnation, three strategies for detecting early convergence are proposed in [28]:

1. Determines the greatest Euclidean distance between the gBest particle and a swarm particle. When this distance is smaller than a threshold distance termed δ_{stag} , stagnation develops. Equation (17) is a form to present it:

$$\varphi_j = \frac{\max_{i \in [1, \dots, S]} \|e_{ij} - e^*_{ij}\|}{|\mu_{\max} - \mu_{\min}|} \quad (17)$$

where $\|e_{ij} - e^*_{ij}\|$, is the Euclidean distance between the gBest particle and a particle of the swarm in iteration j . $\mu_{\min}$ and $\mu_{\max}$ represent the minimum and maximum of particle I , respectively.

2. Cluster Analysis consists of evaluating the percentage of the particle's number of a cluster C . If the distance between a particle belongs to C and the gBest particle which is less than a threshold $\delta_{\min}$, the convergence has occurred, when a percentage of particles in C reach a threshold $\delta_{\max}$.
3. Objective Slope Function considers the position of particles in search space based on its normalized value and the change rate of objective function which is obtained through Equation (18):

$$f_{ratio} = \frac{f(\hat{y}_j) - f(\hat{y}_{j-1})}{f(\hat{y}_j)} \quad (18)$$

where $f(\hat{y}_j)$ is the objective function's fitness value on iteration j , and $f(\hat{y}_{j-1})$ is the objective function's fitness value on iteration $j-1$. A counter is incremented if the value of f_{ratio} is less than a predefined threshold.

2.5. Computational Experiments

The computational experiments are presented in this section and implementation results of the APF+PSO and APF+SBPSO algorithm are presented to obtain collision-free trajectories as well as the potential surfaces for each algorithm will represent.

A. APF+PSO simulations

Figure 2 plots a graph of aptitudes and generations, where it is feasible to see when the number of particles is equal to 2000 iterations, the best particle, and the average of all particles converge.

Figure 2. PSO graph converging for 30 particles and 2000 Iterations.

The potential field parameters found in the simulation of the PSO algorithm for 30 particles and 2000 iterations are presented in Table 1. Figure 3 shows the lines of the artificial potential field and the collision-free trajectory, in Cartesian space for the simulation performed, using the parameters in Table 1, obtained from the locations of the obstacles.

Figure 3. Lines of the potential field by the algorithm of APF for the parameters in Table 1.

Table 1. Values of parameters and intensities founded by PSO for 30 particles and 2000 iterations.

Parameter	k_r	K_a	K_o
intensity	2.29	5.250	0.489

In Figure 4, the resulting surface of the field is shown. Potential for the scene is shown in Figure 3.

Figure 4. The field's resultant surface.

B. APF+SBPSO simulations

In this section, some functions of benchmark used to evaluate the proposed algorithm and the parameters used are represented.

The Benchmark Functions

Some functions chosen are often applied in minimization problems to confirm the stagnation and convergence of the algorithm [28]. In [29–31] and several other PSO studies, the functions are described below:

1. **Sphere function**—characterized by being simple, convex and unimodal. Its main function to evaluate the convergence rate of the search process are represented in Equation (19):

$$f_1(x) = \sum_{i=1}^d x_i^2 \quad (19)$$

2. **Rosenbrock function**—has the global minimum in parabolic valley. In this function, the convergence is difficult [32]; this function is represented in Equation (20):

$$f_2(x) = \sum_{i=1}^{d-1} \left[100(x_{i+1} - x_i)^2 + (x_i)^2 \right] \quad (20)$$

3. **Griewank function**—several local minimums regularly distribute and are represented in Equation (21):

$$f_3(x) = \frac{1}{400} \sum_{i=1}^d x_i^2 - \prod_{i=1}^d \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (21)$$

4. **Rastrigin function**—it is a highly multimodal function; however, the locations of the various local minima are regularly distributed and represented in Equation (22):

$$f_4(x) = \sum_{i=1}^d (x_i^2 - 10 \cos(2\pi x_i)) + 10 \quad (22)$$

5. **HappyCat function**—is characterized by being unimodal. This feature makes the algorithms of optimization find it difficult to escape from a “black groove” [33] represented in Equation (23):

$$f_5(x) = \left[\left(\|x\|^2 - d \right)^2 \right]^{\frac{1}{8}} + \frac{1}{d} \left(\frac{1}{2} \|x\|^2 + \sum_{i=1}^d x_i \right) + \frac{1}{2} \quad (23)$$

6. **Ackley function**—there are several local minimums in the parabolic valley. If the function characterized by having an almost flat outer region represent in Equation (24):

$$f_6(x) = -20 \exp\left(-0.2 \sqrt{\frac{1}{d} \sum_{i=1}^d x_i^2}\right) - \exp\left(\frac{1}{d} \sum_{i=1}^d \cos(2\pi x_i)\right) + 20 + \exp(1) \quad (24)$$

For each of the functions presented, Table 2 shows the search space boundaries and the initialization ranges.

Table 2. Benchmark functions characteristics.

Functions	Search Space	Initialization Range
f_1	$-200 \leq x_i \leq 200$	$100 \leq x_i \leq 200$
f_2	$-130 \leq x_i \leq 130$	$75 \leq x_i \leq 130$
f_3	$-300 \leq x_i \leq 300$	$150 \leq x_i \leq 300$
f_4	$-6.10 \leq x_i \leq 6.10$	$3.05 \leq x_i \leq 6.10$
f_5	$-4 \leq x_i \leq 4$	$2 \leq x_i \leq 4$
f_6	$-30.00 \leq x_i \leq 30.00$	$15 \leq x_i \leq 30.00$

The values associated with the parameters w , C_1 , and C_2 , are defined in Equation (11). The standard PSO's performance might vary, also its modifications [34,35].

To perform a proper comparison, all parameters of the SBPSO and the standard PSO were defined with the same value: w (weight of inertia) which starts at 0.9 and $C_1 = C_2 = 2.0$ and decays linearly ending at 0.3. The maximum velocity $V_{\max}$ of the particle has been defined related to the dimensions of the search space [25].

Other parameters are specific to SBPSO. They are related to quantity and speed of the scout particles, the detection of premature convergence, and the creation of inspection and adjacent points. The number of Girl Scouts was set to 12 percent of the total number of particles. Equations (13), (15), and (17) have additional parameters: $C_3 = 1.5$, which corresponds to 2% of the search space, and $\lambda = 10^{-4}$.

Two thresholds were used: δ_{stag} and δ_{conv} to detect premature convergence. δ_{stag} was assigned the value 10^{-4} and the threshold δ_{conv} corresponds to 6% of the total iterations number. δ_{conv} used to set iterations number whose gBest particle's fitness value has not improved. All of the above parameters were set after repeated simulations and produced good results in the functions which have been studied.

3. Results

The performance of the SBPSO algorithms is evaluated and makes a comparison study with the traditional PSO algorithm applied on UR5 trajectory planning. The result presents the fact that, in the SBPSO, the result of serendipity combined with scout particles, more active than the traditional PSO.

Figures 5–10 show the behavior of the swarm (with 30 particles) in the Sphere, Rosenbrock, Griewank, Rastrigin, HappyCat and Ackley, for 2000 iterations respectively. In the

figures below, the traditional PSO stagnated before iteration 2000, but the SBPSO is still active.

Figure 5. In the Sphere function (30 particles), the traditional PSO stagnated close to iteration 1300 and the SBPSO remains active close to iteration 2000.

Figure 6. In the Rosenbrock function (30 particle), the traditional PSO stagnated close to iteration 870 and the SBPSO remains active near the 2000 iteration.

Figure 7. In the Griewank function (30 particles), the PSO stagnated near the iteration 1010 and the SBPSO remains active near iteration 2000.

Figure 8. In the Rastrigin (30 particles) function, the traditional PSO stagnate close to iteration 1210 and SBPSO remains active near the 2000 iteration.

Figure 9. In the HappyCat (30 particles) function, the traditional PSO has stalled close to iteration 1560 and the SBPSO remains active close to iteration 2000.

Figure 10. In the Ackley function (30 particles), the traditional PSO stagnated close to iteration 1500 and the SBPSO remains active close to iteration 2000.

Tables 3 and 4 show the size of the population, the iterations number, the best fitness value and standard deviation of benchmark functions.

Table 3. The comparison between standard PSO algorithm and SBPSO algorithm by using benchmark functions applied on UR5 robot.

Size of Population	Iteration	Fitness Value of Benchmark Functions											
		Sphere Function		Rosenbrock Function		Griewak Function		Rastrigin Function		HappyCat Function		Ackley Function	
20	1000	PSO	SBPSO	PSO	SBPSO	PSO	SBPSO	PSO	SBPSO	PSO	SBPSO	PSO	SBPSO
		1.2196×10^{-31}	3.7821×10^{-156}	57.7381	1.4324×10^{-4}	0.1692	0	3.9673	0	1.2226×10^{-14}	1.3323×10^{-16}	1.0284×10^{-10}	9.2371×10^{-16}
30	2000	2.9396×10^{-11}	7.3821×10^{-124}	150.7192	2.8091×10^{-2}	0.0185	0	20.6732	0	2.0678×10^{-5}	1.0736×10^{-11}	2.1024	8.8818×10^{-16}
	1000	1.9504×10^{-22}	3.9011×10^{-129}	29.6739	1.0476×10^{-4}	0.0287	0	4.6732	0	3.6771×10^{-14}	2.2204×10^{-16}	0.4030	6.4310×10^{-16}
50	2000	2.0422×10^{-10}	9.7813×10^{-84}	131.6743	1.9692×10^{-2}	0.0532	0	37.8871	0	1.8586×10^{-6}	4.5530×10^{-14}	1.2258	9.7320×10^{-16}
	1000	7.6959×10^{-28}	2.5243×10^{-31}	23.5431	9.9319×10^{-4}	0.0659	0	1.9591	0	4.2327×10^{-15}	2.3592×10^{-16}	6.1342	7.6102×10^{-16}
	2000	1.5353×10^{-17}	8.9143×10^{-120}	64.9304	5.6966×10^{-3}	0.0383	0	33.6462	0	3.9029×10^{-5}	4.6358×10^{-13}	1.8144	8.6098×10^{-16}

Table 4. Comparison between standard deviation of the traditional PSO and SBPSO with a populations of 20 particles.

Functions	Iteration	PSO	SBPSO
Sphere function	1000	3.1403×10^{-22}	3.7821×10^{-156}
	2000	1.4253×10^{-6}	3.6721×10^{-81}
Rosenbrock function	1000	57.7381	1.4324×10^{-4}
	2000	150.7192	2.8091×10^{-2}
Griewank function	1000	0.1692	0
	2000	0.0185	0
Rastrigin function	1000	3.9673	0
	2000	48.7790	0
HappyCat function	1000	4.6100	0
	1500	19.6670	0
Ackley function	1000	4.1558×10^{-11}	0
	2000	4.8007×10^{-11}	0

Figure 11. Tack contour of universal manipulator robot UR5 with SBPSO algorithm.

Figure 11 show the contour path with avoid collision for universal manipulator robot UR5.

4. Conclusions

This paper developed a Serendipity-Based Particle Swarm Optimization (SBPSO) variant of the traditional PSO in order to prevent the premature convergence of optimization procedures. The algorithm based on an approach with two dimensions of the serendipity concept: intelligence and chance. The intelligence dimension implemented by inspections at the adjacencies of the gBest particle. The chance dimension implemented by using the particles to improve search space exploration. After the studies were completed, it was shown that the SBPSO outperformed the standard PSO. To make these comparisons, two criteria were analyzed: convergence and stagnation. In the convergence criterion, when the Griewank and Rastrigin were evaluated, the SBPSO algorithm found the global solutions faster than the PSO algorithm. For the Sphere, Rosenbrock, HappyCat and Ackley functions, none of them found the optimal solutions. However, the SBPSO algorithm found very satisfactory solutions when compared with PSO. The evaluation of the stagnation criterion in the Sphere function showed that when the SBPSO was compared with the traditional PSO, it was delayed at least 21%. For the Rosenbrock function, the stagnation delay was at least 30% of the number of iterations compared to the traditional PSO. To the Griewank function, the SBPSO slowed the stagnation by less 20% when compared to the traditional PSO. For the Rastrigin function, stagnation was delayed at least 34% compared to the traditional PSO. In the HappyCat function, when the SBPSO was compared to the traditional PSO, there was a delay at less than 20% of the maximum iterations number. Finally, in the Ackley function, the SBPSO delayed the stagnation at least 20%, in relation to the traditional PSO algorithm. In these experiments, the SBPSO algorithm gave a better convergence behavior, overcoming some variations in relation to the quality of the solution, such as ability to find the global optimum, the stability of solutions and the ability to restart the swarm movement in stagnation case. In general, the results proved to solve the optimization problem of universal robot UR5 trajectory planning and find the optimal path. In future work, we will apply the SBPSO algorithm to solve more real-world optimization problems, such as vehicle localization, wireless sensor networks, and velocity estimation.

Author Contributions: Formal analysis, I.J.K.; investigation, I.J.K.; methodology, I.J.K.; supervision Y.T. and R.L.; writing—original draft, I.J.K., Y.T., and R.L.; writing—review, I.J.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Open Research Foundation of Advanced Innovation Center for Intelligent Robotics and Systems under Grant Number 2019IRS18, and the Fundamental Research Funds for the Central Universities of China under Grant Number 2021IVA019.

Acknowledgments: Thanks to my mother, my husband (Omer W. Taha), my family, prof. Salim Ali Abass, Shaimaa Shukri. Almustafa university college and the reviewers and proof reader for this paper.

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

References

1. Chappell, P. *Mechatronic Hands Prosthetic and Robotic Design*; Institution of Engineering and Technology: London, UK, 2016; 192p.
2. LaValle, S.M. *Planning Algorithms*; Cambridge University Press: Cambridge, UK, 2006. [\[CrossRef\]](#)
3. Pinto, M.F.; Mendonça, T.R.F.; Olivi, L.R.; Costa, E.B.; Marcato, A.L.M. Modified approach using variable charges to solve inherent limitations of potential fields method. In Proceedings of the 2014 11th IEEE/IAS International Conference on Industry Applications, Juiz de Fora, Brazil, 7–10 December 2014; pp. 1–6. [\[CrossRef\]](#)
4. Kaya, İ. Optimal and Analytical Tuning of I-PD Controllers for Controlling Stable Processes with Inverse Response. *Balk. J. Electr. Comput. Eng.* **2020**, *8*, 260–265. [\[CrossRef\]](#)
5. Paiva, F.A.P.; Costa, J.A.F.; Silva, C.R.M. A Serendipity-Based Approach to Enhance Particle Swarm Optimization Using Scout Particles. *IEEE Lat. Am. Trans.* **2017**, *15*, 1101–1112. [\[CrossRef\]](#)
6. Pintea, C.M. Advances in Bio-inspired Computing for Combinatorial Optimization Problems. In *Intelligent Systems Reference Library*; Springer: Berlin/Heidelberg, Germany, 2014. [\[CrossRef\]](#)

7. Silva, C.R.M.; Lins, H.W.C.; Martins, S.R.; Barreto, E.L.F.; D'Assunção, A.G. A multiobjective optimization of a UWB antenna using a self organizing genetic algorithm. *Microw. Opt. Technol. Lett.* **2012**, *54*, 1824–1828. [\[CrossRef\]](#)
8. Dosciatti, E.R.; Junior, W.G.; Foronda, A. TQ/PSO—A new scheduler to optimize the time frame with PSO in WiMAX networks. *IEEE Lat. Am. Trans.* **2015**, *13*, 365–376. [\[CrossRef\]](#)
9. Khan, A.; Hizam, H.; Wahab, N.I.b.A.; Othman, M.L. Optimal power flow using hybrid firefly and particle swarm optimization algorithm. *PLoS ONE* **2020**, *15*, e0235668. [\[CrossRef\]](#)
10. Wang, Y.; Wu, D.; Chi, J. Current Situation and Development Trend of Hydrogen Energy and Hydrogen Production Technology. *Monogr. Rev.* **2001**, *20*, 3. [\[CrossRef\]](#)
11. Chen, A.P.; Huang, C.H.; Hsu, Y.C. A novel modified particle swarm optimization for forecasting financial time series. In Proceedings of the 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems, Shanghai, China, 20–22 November 2009; Volume 1. [\[CrossRef\]](#)
12. Jin, L.; Huang, Y. A Particle Swarm Optimization-Neural Network Prediction Model for Typhoon Intensity Based on Isometric Mapping Algorithm. In Proceedings of the 2012 Fifth International Joint Conference on Computational Sciences and Optimization, Harbin, China, 23–26 June 2012; pp. 857–861. [\[CrossRef\]](#)
13. Lv, Z.; Wang, L.; Guan, Z.; Wu, J.; Du, X.; Zhao, H.; Guizani, M. An Optimizing and Differentially Private Clustering Algorithm for Mixed Data in SDN-Based Smart Grid. *IEEE Access* **2019**, *7*, 45773–45782. [\[CrossRef\]](#)
14. Machado, R.B.; Boukerche, A.; Sobral, J.B.M.; Juca, K.R.L.; Notare, M.S.M.A. A hybrid artificial immune and mobile agent intrusion detection based model for computer network operations. In Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, Denver, CO, USA, 4–8 April 2005. [\[CrossRef\]](#)
15. Caballe, S.; Xhafa, F.; Abraham, A. Towards an automatic real-time assessment of online discussions in Computer-Supported Collaborative Learning practices. In Proceedings of the 2008 Third International Conference on Digital Information Management, London, UK, 13–16 November 2008; pp. 470–475. [\[CrossRef\]](#)
16. Oku, K.; Hattori, F. Fusion-based recommender system for improving serendipity. *J. Jpn. Soc. Fuzzy Theory Intell. Inform.* **2011**, *25*, 524–539.
17. Adamopoulos, P.; Tuzhilin, A. On unexpectedness in recommender systems. *ACM J.* **2011**, *5*, 1–32. [\[CrossRef\]](#)
18. Silva, A.; Neves, A.; Gonçalves, T. Using scout particles to improve a predator-prey optimizer. In Proceedings of the International Conference on Adaptive and Natural Computing Algorithms, Lausanne, Switzerland, 4–6 April 2013. [\[CrossRef\]](#)
19. Wu, Y.L.; Ho, T.F.; Shyu, S.J.; Lin, B.M.T. Discrete particle swarm optimization with scout particles for library materials acquisition. *Sci. World J.* **2013**, *2013*, 636484. [\[CrossRef\]](#)
20. Campos, J.; de Figueiredo, A.D. Programming for Serendipity. *SSRN Electr. J.* **2011**, *5*, 195–202. [\[CrossRef\]](#)
21. Chakraborti, T.; Briggs, G.; Talamadupula, K.; Zhang, Y.; Scheutz, M.; Smith, D.; Kambhampati, S. Planning for serendipity. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015. [\[CrossRef\]](#)
22. Lee, K.; Choi, D.; Kim, D. Incorporation of potential fields and motion primitives for the collision avoidance of unmanned aircraft. *Appl. Sci.* **2021**, *7*, 3103. [\[CrossRef\]](#)
23. Weerakoon, T.; Ishii, K.; Nassiraei, A.A.F. An Artificial Potential Field Based Mobile Robot Navigation Method To Prevent From Deadlock. *J. Artif. Intell. Soft Comput. Res.* **2015**, *5*, 189–203. [\[CrossRef\]](#)
24. Chen, Q.; Sun, J.; Palade, V.; Wu, X.; Shi, X. An improved Gaussian distribution based quantum-behaved particle swarm optimization algorithm for engineering shape design problems. *Eng. Optim.* **2021**. [\[CrossRef\]](#)
25. Ling, S.H.; Iu, H.H.C.; Chan, K.Y.; Lam, H.K.; Yeung, B.C.W.; Leung, F.H. Hybrid particle swarm optimization with wavelet mutation and its industrial applications. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **2008**, *38*, 743–763. [\[CrossRef\]](#) [\[PubMed\]](#)
26. Mason, K.; Duggan, J.; Howley, E. Multi-objective dynamic economic emission dispatch using particle swarm optimisation variants. *Neurocomputing* **2017**, *270*, 188–197. [\[CrossRef\]](#)
27. Evers, G.I.; Ghalia, M.B. Regrouping particle swarm optimization: A new global optimization algorithm with improved performance consistency across benchmarks. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, San Antonio, TX, USA, 11–14 October 2009. [\[CrossRef\]](#)
28. van den Bergh, F. An Analysis of Particle Swarm Optimizers. Ph.D. Thesis, University of Pretoria, Pretoria, South Africa, 2001.
29. Sun, J.; Feng, B.; Xu, W. Particle swarm optimization with particles having quantum behavior. In Proceedings of the Congress on Evolutionary Computation, Portland, OR, USA, 19–23 June 2004. [\[CrossRef\]](#)
30. Xi, M.; Sun, J.; Xu, W. An improved quantum-behaved particle swarm optimization algorithm with weighted mean best position. *Appl. Math. Comput.* **2008**, *205*, 751–759. [\[CrossRef\]](#)
31. Sharma, M.; Chhabra, J.K. Sustainable automatic data clustering using hybrid PSO algorithm with mutation. *Sustain. Comput. Inform. Syst.* **2019**, *23*, 144–157. [\[CrossRef\]](#)
32. Church, K.W. Benchmarks and goals. *Nat. Lang. Eng.* **2019**, *26*, 579–592. [\[CrossRef\]](#)
33. Beyer, H.G.; Finck, S. HappyCat—A simple function class where well-known direct search algorithms do fail. In Proceedings of the International Conference on Parallel Problem Solving from Nature, Taormina, Italy, 1–5 September 2012. [\[CrossRef\]](#)

-
34. Patwal, R.S.; Narang, N.; Garg, H. A novel TVAC-PSO based mutation strategies algorithm for generation scheduling of pumped storage hydrothermal system incorporating solar units. *Energy* **2018**, *142*, 822–837. [[CrossRef](#)]
 35. Kazim, I.J.; Gang, T.Y.; Qaseer, L. Integration of DE Algorithm with PDC-APF for Enhancement of Contour Path Planning of a Universal Robot. *Appl. Sci.* **2021**, *14*, 6532. [[CrossRef](#)]