International Journal of Innovative Computing, Information and Control
Volume **12**, Number **4**, August **2016**

ICIC International ©2016 ISSN 1349-4198
pp. 1141–1163

A SERENDIPITY-BASED PSO APPROACH TO DELAY PREMATURE CONVERGENCE USING SCOUT PARTICLES

Fábio Paiva[1], José Costa[2] and Cláudio Silva[3]

[1]Campus Parnamirim
Federal Institute of Education, Science and Technology of Rio Grande do Norte
Parnamirim, RN 59143-455, Brazil
fabio.procopio@ifrn.edu.br

[2]Department of Electrical Engineering
[3]Department of Communications Engineering
Federal University of Rio Grande do Norte
Natal, RN 59078-970, Brazil
{ jafcosta; claudio.rmsilva }@gmail.com

Received February 2016; revised June 2016

Abstract. *In metaheuristic algorithms, such as Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), it is common to deal with a problem known as premature convergence. It happens when a population or a particle swarm loses diversity and starts converging too early towards a suboptimal solution. There are many approaches to this problem. This work proposes a new one based on a concept normally applied in the recommender systems domain. This concept is known as serendipity. A formalization for the concepts of serendipity and premature convergence is presented. A Serendipity-Based PSO algorithm is developed and compared with the PSO and some variant approaches available in the literature. After experimentation, the results were promising, since they have showed that the Serendipity-Based PSO outperformed PSO and a variant called PSO-Scout. The experiments were also performed to compare Serendipity-Based PSO with some studies in the literature, considering a fixed number of particles and varying the problem dimensionality and the number of iterations. In all experiments, Serendipity-Based PSO has also showed a better convergence behavior, outperforming PSO and some variants in solution quality, ability to find global optimum, solutions stability and ability to restart the movement of the swarm after stagnation has been detected.*

Keywords: PSO, Serendipity, Scout particles, Premature convergence

1. **Introduction.**

Bio-inspired computing is the use of computers to model the living phenomena in groups and the study of life to improve the usage of computers [1]. Several bio-inspired techniques have been applied to real world optimization problems in different domain areas such as Telecommunications [2, 3], Power Systems [4, 5], Prediction Systems [6, 7] and Automation [8, 9]. There are three groups representing these bio-inspired techniques [10]: Evolutionary Computation algorithms, Natural Ecosystems algorithms and Swarm Intelligence algorithms.

Evolutionary Computation algorithms aim to understand the mechanism of computational systems and to design highly robust, flexible and efficient algorithms to solve real world problems [11]. Some examples of Evolutionary Computation algorithms are: Genetic Algorithm, Genetic Programming, Differential Evolution and Evolutionary Strategy.

Natural Ecosystems comprise the living organisms along with the abiotic environment with which organisms interact such as air, soil, water [12]. There are many interactions among the ecosystem species such as interspecies or intraspecies. From a computational point of view, the ecosystem is an environment that is populated by independent entities (agents) to represent some level of organization [13].

Swarm Intelligence is a widespread concept in bio-inspired computing and involves a set of metaheuristics whose emergent behaviors can result in an ability to solve complex problems [14]. It implements the collective intelligence for groups containing simple agents that are based on the behavior of natural insect swarms. Particle Swarm Optimization [15], for example, is one of the swarm intelligence algorithms that has been widely used in the context of global optimization problems.

Particle Swarm Optimization (PSO) is inspired by the social behavior of birds and fishes, where individuals of a population are represented by a point (or particle) of the search space. According to [16], generally the algorithm is not largely affected by the size and nonlinearity of the problem and, moreover, it converges to optimal solutions, whereas most analytical methods fail during the convergence process.

PSO has some advantages such as easy implementation and fast convergence, but often it faces a problem in which its particles are "trapped" in local optimum. This problem is often known as premature convergence in swarm algorithms. In another area of study, known as recommender systems, there is a concept that can be used as a strategy to delay premature convergence. This concept is called *serendipity*.

In general, serendipity is a term that refers to fortunate findings that apparently were performed by chance. In an exhaustive study, [17] showed that serendipity has a great contribution to the progress of science, technology and art. According to [18], in science and technology, the serendipitous discoveries occur frequently. In the history of science, for example, there are several cases of serendipity such as: 1) X-ray in 1895; 2) radioactivity in 1896; 3) penicillin in 1928; 4) microwave oven in 1945 and many others. The innovations can be considered cases of serendipity since, as a rule, they are performed by individuals who are able to detect patterns whereas others observe randomness.

Recommender systems emerged as a group of approaches to solve the problem of information overload. They are special applications, present in our daily lives, that provide personalized recommendations (for example, products or services) for users that may be interesting [19, 20]. In recent years, the research in recommender systems has focused on the recommendations accuracy [21, 22, 23]. However, high accuracy can cause a specific problem known as over-specialization. Although, this problem occurs when the system refers only to the items related with the user's profile, there may be items which are more appropriate. Serendipity is a strategy that may be used to solve the problem of over-specialization.

When a recommendation system suggests only items related to the user's interest, it converges to recommendations that do not meet the user's expectations [24]. Thus, it fails to suggest items that can be the most adequate. Similarly, when a metaheuristic algorithm converges to a local optimum and does not consider other solutions that are more appropriate than the solution found, the correlation between over-specialization and premature convergence is observed.

Many relevant approaches were proposed in literature to accelerate the convergence of PSO and avoid premature convergence. Operators often used in Genetic Algorithm such as selection and mutation have been incorporated into standard PSO [25, 26, 27] because they are efficient to enhance swarm diversity. Other different approaches have also been proposed [28, 29, 30] such as quantum system and wavelet theory.

These approaches are essentially stochastic, but they can benefit from combinations with other techniques to decrease the randomness of the search process in order to improve the performance of the algorithm. For this purpose, an interesting alternative is the implementation of the serendipity concept that considers a set of patterns to represent the behavior of the fitness values.

This paper proposes a PSO variant called Serendipity-Based PSO that combines the concept of serendipity and scout particles to enhance the exploration within the search space. The scout particles are employed as a way of introducing new exploratory behaviors into the swarm. Some strategies, commonly used in the recommender systems domain, are used to implement two dimensions of serendipity. These dimensions are chance and sagacity. The first strategy is called "blind luck" which uses the exploratory behavior of the scout particles to implement the chance dimension. The other strategy that is called "Pasteur's principle" implements the sagacity dimension by means of a set of behavior patterns.

The paper is organized as follows. In Section 2, it provides a brief background in order to introduce the basic concepts necessary for the reader's understanding. Section 3 presents the proposed algorithm that combines the concept of serendipity with the use of scout particles. Section 4 presents the experiments performed and discusses the results obtained. Finally, in Section 5, the final considerations are presented to conclude the work.

2. **Background.**

This section presents some basic information necessary to the suitable understanding of the algorithm presented in Section 3. The concept of serendipity and three methods to detect premature convergence are presented together with some related work available in the literature.

2.1. **Serendipity.**

The Cambridge Dictionary defines serendipity as *"The fact of finding interesting or valuable things by chance"*. However, unlike many traditional definitions that just use the term as a synonymous of "random", serendipity is closer to a mixture of sagacity and chance [31].

In [32] it expressed the concept of serendipity in a formal way through the identification of different categories. For this, logic equations were used, called *Serendipity Equations*, to present four events that can generate serendipity. These events are associated to different types of serendipity: a) pseudo-serendipity; b) real serendipity; c) serendipity without a metaphor inspiration; and d) serendipity with incorrect knowledge.

In recommender systems, serendipity can be implemented through four strategies [33]: 1) Blind luck, 2) Pasteur's principle ("chance favors the prepared mind"), 3) Anomalies and exceptions and 4) Reasoning by analogy. The *"blind luck"* approach aims to provide recommendations from information randomly generated. "Pasteur's principle" is implemented by means of the user's profile to recommend items that have something to do with the user's profile. The *"Anomalies and exceptions"* approach is partially implemented by low similarities between the user's profile and the recommended items. Finally, the implementation of the "Reasoning by analogy" approach is unknown at the moment.

In the metaheuristic context, it is said that the set F is formed by elements that represent the solutions found during an iteration in a given search space. Thus, we say that f_{ij} is an element belonging to the set F that represents a solution i found during the iteration j. The solution i, whose fitness value is the best value during the iteration j, is represented by the element $f_{ij}^* \in F$.

S is a set whose elements represent the possible solutions in a search space and F is a proper subset of S. So, during the iteration j, when an element of S does not belong to the set F, this element is considered as an *occasional* solution. Then, the set containing the elements that represent occasional solutions during the iteration j is given by the relative complement of F in S:

$$CHANCE = S - F \tag{1}$$

When an element $chance_{ij}$ presents a fitness value better than the fitness presented by the element f^*_{ij}, $chance_{ij}$ is called *serendipitous*. The set SRD is formed by serendipitous elements that represent occasional solutions with fitness value better than the fitness value of the element f^*_{ij}, according to Equation (2):

$$SRD = \{srd_{ij} | fitness(chance_{ij}) < fitness(f^*_{ij})\} \tag{2}$$

The concept of serendipity presented in this paper is adapted to the metaheuristic context. It consists of two essential aspects: *acceptability* and *occasionality*. So, an element of the set SRD can be used to represent a serendipitous solution because it is a) *acceptable*, since it represents a solution whose fitness value is better than fitness value of the element f^*_{ij} and b) *occasional*, since it is an element that does not belong to the set F during the iteration j.

2.2. **Methods to detect premature convergence.** PSO is a technique capable of finding good solutions in a fast convergence time, but the algorithm has the risk to converge prematurely. It is said that the swarm has converged prematurely when the proposed solution approximates to a local optimum, rather than to the global optimum for the problem.

Premature convergence occurs due to decreased diversity in the search space and the result of this process takes the swarm to a state of stagnation. After premature convergence has initiated, the particles will continue to converge within extremely close proximity of one another so that the best and all global personal bests are miniscule within one region of the search space [34].

In [35] it presented three methods for detecting the premature convergence in order to avoid stagnation of the swarm. They are:

1) Maximum Swarm Radius – it evaluates the greatest Euclidean distance between a swarm particle and the *gBest* particle. The stagnation occurs when this Euclidean distance is less than a threshold called δ_{stag}. This evaluation method is formally presented by Equation (3):

$$\varphi_j = \frac{\max_{i \in (1,\ldots,s)} \|f_{ij} - f^*_{ij}\|}{|\mu_{\max} - \mu_{\min}|}, \tag{3}$$

where $\|f_{ij} - f^*_{ij}\|$ is the Euclidean norm between a swarm particle i and the *gBest* particle during the iteration j. $\mu_{\max}$ and $\mu_{\min}$ represent the maximum and minimum limit for each dimension of the particle i, respectively.

2) Cluster Analysis – it evaluates the percentage for the number of particles that belong to a cluster C. A particle belongs to set C if the distance between it and *gBest* particle is less than a threshold $\delta_{\min}$. So, when a percentage of the particles in cluster C reaches a threshold $\delta_{\max}$, it is assumed that the convergence occurred.

3) Objective Function Slope – it does not take account of the relative positions of the particles in search space. It is based on the change rate of the fitness function. A normalized value for the fitness function is obtained by Equation (4):

$$f_{ratio} = \frac{f(\hat{y}_j) - f(\hat{y}_{j-1})}{f(\hat{y}_j)}, \tag{4}$$

where $f(\hat{y}_j)$ is the fitness value for the objective function in iteration j and $f(\hat{y}_{j-1})$ in iteration $j-1$. If f_{ratio} is less than a defined threshold, a counter is incremented. When the counter reaches a threshold δ_{conv}, it is assumed that the swarm converged.

Objective Function Slope method is better than the other ones, but it still has some issues [35]. An alternative approach to achieving better results is a combination with Cluster Analysis or Maximum Swarm Radius.

2.3. **Related work.** Serendipity is defined as finding something good or useful while not specifically searching for that in a certain occasion. The design of recommendation engines that considers serendipity is relatively new and it is still an open research problem.

When the concept of serendipity is adapted to the context of metaheuristic algorithms and their optimization applications, "something good or useful" may be seen as a candidate solution that it is better than the current one after a certain number of iterations.

A class of metaheuristic algorithms that has received much attention consists of bio-inspired algorithms. Bio-inspired computing is the use of computers to model the phenomena of life, and simultaneously studying the life to improve the usage of computers, in our case, to improve optimization algorithms. Bio-inspired computing is a major subset of natural computation. Swarm intelligence deals with natural and artificial systems composed of populations. These populations coordinate the swarm using decentralized control and self-organization [36]. The study of swarm intelligence involves the study of the behavior of ants, wasps, termites, fishes, birds and bats. In this context, PSO is one of the well-known algorithms.

PSO has gained much attention from research communities for solving real world optimization problems due to its ease of implementation and good search performance. PSO is a stochastic approach used to model the social behavior of birds. In this kind of algorithm, the solution is represented by a particle that "flies" over the search space looking for an optimum solution during a certain number of iterations [37]. The PSO pseudocode is presented in Algorithm 1.

Algorithm 1 Standard PSO pseudocode

```
1:  for each particle in $S$ do
2:      randomly initialize velocity and position
3:  end for
4:  while stopping criterion not satisfied do
5:      for each particle $i$ in $S$ do
6:          calculate $fitness$ value
7:          if $fit(x_{id}) < fit(pBest_i)$ then
8:              $pBest_i \leftarrow x_{id}$
9:          end if
10:         if $fit(pBest_i) < fit(gBest)$ then
11:             $gBest \leftarrow pBest_i$
12:         end if
13:     end for
14:
15:     /*Update velocity and position of the particle*/
16:     for each particle $i$ in $S$ do
17:         $v_{id}^{(t+1)} = w \cdot v_{id}^t + c_1 \cdot r_1(p_{id}^t - x_{id}^t) + c_2 \cdot r_2(p_{gd}^t - x_{id}^t)$
18:         $x_{id}^{(t+1)} = x_{id}^t + v_{id}^{(t+1)}$
19:     end for
20: end while
```

The movement of the particles is based on two pieces of information: *gBest* and *pBest*. Information *gBest* influences the motion of the particle towards the best position found by the swarm, whereas *pBest* moves the particle to the best position found by itself. After finding *gBest* and *pBest*, each particle updates its velocity and its position by Equations (5) and (6), respectively:

$$v_{id}^{(t+1)} = w \cdot v_{id}^{t} + c_1 \cdot r_1 \left(p_{id}^{t} - x_{id}^{t}\right) + c_2 \cdot r_2 \left(p_{gd}^{t} - x_{id}^{t}\right) \tag{5}$$

$$x_{id}^{(t+1)} = x_{id}^{t} + v_{id}^{(t+1)} \tag{6}$$

In Equation (5), the term w is called inercial factor and represents the particle inertial velocity. v_{id}^{t} and x_{id}^{t} are the velocity and position of the particle i in the instant t, respectively. p_{id}^{t} and p_{gd}^{t} are the best fitness values found by particle i and the best fitness value found among all swarm particles until the instant t, respectively. c_1 is a coefficient that contributes to the auto exploration of the particle, whereas c_2 contributes with the particle movement towards a swarm global dislocation in the search space. r_1 and r_2 are random values uniformly distributed in the range $[0, 1]$.

Different variant approaches have emerged to improve the performance and convergence behavior for Standard PSO algorithm. In [27], it proposed a method that combines the mutation process with the standard PSO. The presented algorithm is called Hybrid PSO with Genetic Mutation (HPSOM) and its particles have a probability of mutating at each iteration. The experiments showed that HPSOM was successful when evaluated on unimodal and multimodal functions.

In [28] it proposed Quantum-Behaved PSO (QPSO). The variant assumes that the PSO is a quantum system, and each particle has a quantum behavior with its quantum state formulated by a wave function. In QPSO, each particle should converge to the local attractor point in order to guarantee that all the particles converge. One particle cannot converge to the best global position without considering its colleagues. Experiments with some benchmark functions showed that QPSO outperformed PSO in most cases.

In [29], an improved QPSO is proposed with weighted mean best position, according to fitness values of the particles called Weighted QPSO (WQPSO). The proposal was tested on benchmark functions and it showed satisfactory results in relation to the Standard PSO and QPSO.

In [30], it proposed a PSO variant that combines mutation operator with wavelet theory. This theory increases the operating space to find the best solutions. Benchmark functions and three industrial applications were employed to evaluate the performance and applicability of the proposed method.

In [38] it proposed a combination of the Particle Swarm with concepts from Evolutionary Algorithms. The method combined the traditional velocity and position update rules with the ideas of Gaussian Mutation. The experiments were conducted on unimodal and multimodal functions.

In [39], the Opposition-Based PSO (OPSO) method is presented to accelerate the convergence of PSO and avoid premature convergence. OPSO employs opposition-based learning and applies dynamic Cauchy Mutation on the best particle of the swarm. The experiments showed that OPSO could successfully deal with multimodal benchmark functions while maintaining fast search speed on simple unimodal functions.

In [40], it presented a hybridization that combines PSO with a modified method of Broyden-Fletcher-Goldfarb-Shanno (BFGS) to improve the ability of local searches using reposition operator and territory of particles.

In [41], a PSO variant called Discrete Particle Swarm Optimization (DPSO) is proposed to be applied to the domain of academic libraries. The proposed algorithm uses scout particles and they correspond to a candidate solution to the problem. The scout particles are employed to enhance the exploration within the solution space. Several computational experiments are designed and conducted. The results are statistically analyzed and demonstrated the effectiveness and efficiency of the algorithm.

In [42], it presented a hybridization that combines PSO with Artificial Bee Colony Optimization. This combination adds the scout bee phase into PSO to regenerate useless particles in order to achieve higher diversity. Four datasets of the medical area are used to validate the proposed algorithm and the results showed good accuracy.

In [43], it proposed a variant that uses scout particles with different roles. The global performance of the algorithm and the roles of the scouts are evaluated by means of bechmark functions. The results showed that the addition of scout particles improved optimization performance and robustness of the algorithm.

3. **Serendipity-Based Particle Swarm Optimization Using Scout Particles.**

In the previous section, serendipity was presented as a kind of ability to perform fortunate discoveries by chance and sagacity. The term "chance" should be understood here as a possibility or probability of anything happening. The term "sagacity", in turn, should be understood as the quality of having good judgment or perception about an event or something related. This concept of serendipity is interesting and useful, since it is the main hint to define where serendipity could be applied to a Standard PSO algorithm and which kind of sagacity should be used to improve PSO behavior with serendipity-inspired decisions.

There are three natural approaches to implement this improvement. The first one is based on modifications in the random decisions points available in the Standard PSO algorithm (presented in Subsection 2.3), according to lines 2, 8, 11, 17 and 18. The second one is based on the inclusion of a complementary method to be used in conjunction with a Standard PSO algorithm to implement a Serendipity PSO-like algorithm. The third one is a mix of the other two. All of them consider "fortunate discoveries" as points in the search space that are useful for an optimization procedure, since they are better than the best solution available found until the current iteration.

This work adopts a variant for the third approach. It proposes a strategy used in [24] to generate serendipity based on a perceptive model [44] that combines two strategies used in the domain of recommender systems [33]. They are: blind luck and Pasteur's principle. The blind luck strategy is implemented using scout particles. They complement exploratory space of the Standard PSO generating additional diversity and implementing the chance dimension. The Pasteur's principle is used to recognize potential and seize the moment in perceptions. It combines a behavior detection with scout particles that inspect unexplored regions by the swarm in the search space, according to Equations (5) and (6). This strategy is the implementation of the concept of sagacity or, in other words, the concept of "prepared mind".

The new algorithm implemented is called Serendipity-Based Particle Swarm Optimization (SBPSO). The behavior detection searches for hill or valley patterns around the optimal points chosen for each iteration. The scout particles delay the convergence time by generating additional diversity to the optimization algorithm.

The behavior that the scout particles implement to explore the search space differs from the typical behavior of the swarm. They may be used to increase the exploration capacity of the algorithm. Scouts spend minimal extra resources without performing global modifications to the algorithm.

It is important to ensure that the inclusion of scout particles do not compromise the habitual behavior of the swarm. In the algorithm proposed, this is transparent because the scout particles are used to identify the solutions that are better than the best swarm particle. The velocity of a scout particle k is given by Equation (7):

$$V_{kd}^{(t+1)} = w \cdot V_{kd}^{t} + c_3 \cdot r_3 (X_{kd}^{t} - x_{best}^{t}), \tag{7}$$

where c_3 is the diversity coefficient and r_3 is a random number uniformly distributed in the range $[-1, 1]$. X_{kd}^{t} and x_{best}^{t} are, respectively, the position of the scout particle k and the position of the best swarm particle at instant t. The position of a scout particle k is given by Equation (8):

$$X_{kd}^{(t+1)} = -x_{best}^{t} + V_{kd}^{(t+1)} \tag{8}$$

The initial steps of the SBPSO algorithm, showed in Algorithm 2, are similar to the Standard PSO. The velocity and position are randomly initialized for each swarm particle and the scout particles, too (lines 2-3). Next, for each swarm particle, the fitness value is calculated to find the best particle (lines 5-11). This also occurs for the scout particles (lines 12-18). After the best swarm particle and the best scout particle are found, the best among them will be the new *gBest* particle (lines 20-26).

The next step is to begin the inspection process near the *gBest* particle (lines 28-30). Therefore, the algorithm randomly creates one Adjacent Point (AP), according to Equation (9):

$$AP = best + \alpha \cdot R \tag{9}$$

where *best* is the *gBest* current particle, α is a positive real constant, R is a matrix $1 \times dim$ whose elements are random values uniformly distributed in the range $[-1, 1]$ and *dim* is the dimension of the *gBest* current particle.

Next, for the AP that was created, the algorithm defines one vector $\vec{d}$ that represents the oriented segment $\overline{bestAP}$, according to Equation (10):

$$\vec{d} = AP - best \tag{10}$$

Since *best* and AP define a single vector $\vec{d}$ representing a direction, then there are infinite Inspection Points (IP) whose direction $\overline{bestIP}$ is the same as $\overline{bestAP}$. For the AP created, two IP can be chosen, according to Equation (11):

$$IP = best \pm \lambda \cdot \vec{d}, \tag{11}$$

where λ is a non-zero real constant.

After setting the two IP, a rules set is evaluated (line 31). The implementation of these rules aims to treat the behavior patterns to define Inspection Points around *gBest* current particle. These new solutions can be obtained from the generation of IP or New Points (NP), according to the rules set showed in Table 1.

Again, the SBPSO steps are similar to Standard PSO. The velocity of the swarm particles is calculated and the position is updated (lines 33-36). This also occurs for scout particles (lines 37-40).

Finally, the algorithm evaluates the situation of the swarm related to the stagnation and premature convergence (lines 42-44). When the following condition is satisfied, SBPSO restarts the swarm:

$$(\varphi_j < \delta_{stag}) \wedge (cont \geq \delta_{conv}), \tag{12}$$

where φ_j is the swarm radius in the iteration j and δ_{stag} is a threshold that evaluates the swarm stagnation, according to Equation (3); *cont* indicates the number of iterations in which the swarm does not show significant improvements and δ_{conv} is the threshold that indicates the swarm convergence, according to Equation (4).

According to Pasteur's principle, chance favors the prepared mind. The concept of "prepared mind" is implemented here using a set of behavior patterns shown in Figure 1. These behavior patterns are considered to implement the sagacity dimension. They

Algorithm 2 SBPSO pseudocode using scout particles

```
learning procedure of behavior patterns
randomly initialize *velocity* and *position* of all swarm particles
randomly initialize *velocity* and *position* of all scout particles
**while** *stopping criterion not satisfied* **do**
  **for each** particle $i$ in $S$ **do**
    calculate $fitness$ value
    **if** $fit(x_{id}) < fit(pBest_i)$ **then**
      $pBest_i \leftarrow x_{id}$
    **end if**
  **end for**
  $best_{swarm} \leftarrow getParticleWithBestFitnessValue()$
  **for each** scout $k$ in $S$ **do**
    calculate $fitness$ value
    **if** $fit(scout_{kd}) < fit(pBest_k)$ **then**
      $pBest_k \leftarrow scout_{kd}$
    **end if**
  **end for**
  $best_{scout} \leftarrow getScoutWithBestFitnessValue()$

  **if** $fit(best_{swarm}) < fit(best_{scout})$ **and** $fit(best_{swarm}) < fit(gBest)$ **then**
    $gBest \leftarrow best_{swarm}$
  **else**
    **if** $fit(best_{scout}) < fit(best_{swarm})$ **and** $fit(best_{scout}) < fit(gBest)$ **then**
      $gBest \leftarrow best_{scout}$
    **end if**
  **end if**

  create adjacent points to *gBest* using Equation (9)
  define onde direction vector $\vec{d}$ using Equation (10)
  create two inspection points using Equation (11)
  execute actions based on rules defined in Table 1

  **for each** particle $i$ in $S$ **do**
    $v_{id}^{(t+1)} = w \cdot v_{id}^t + c_1 \cdot r_1(p_{id}^t - x_{id}^t) + c_2 \cdot r_2(p_{gd}^t - x_{id}^t)$
    $x_{id}^{(t+1)} = x_{id}^t + v_{id}^{(t+1)}$
  **end for**
  **for each** scout $k$ in $S$ **do**
    $V_{kd}^{(t+1)} = w \cdot V_{kd}^t + c_3 \cdot r_3(X_{kd}^t - x_{best}^t)$
    $X_{kd}^{(t+1)} = -x_{best}^t + V_{kd}^{(t+1)}$
  **end for**

  **if** (*stagnant swarm*) **and** (*swarm converged*) **then**
    restart swarm
  **end if**
**end while**
```

must be learned in the initialization procedure of the algorithm. These patterns may be idealized to match fitness behavior along the points in a straight line passing by *gBest* (inside the domain of fitness function) during an iteration i. They may consider several possibilities in matrixes such as $N \times 3$. To simplify, a set of patterns is exemplified considering just a matrix 3×3. For this matrix, there are 27 possibilities ($3 \times 3 \times 3$). 14 of them are redundant and just 13 are enough to represent the behavior patterns considered to implement the sagacity dimension.

Figure 1 shows each pattern considering relative fitness values for three points in a straight line that links *gBest* and AP points. They are IP_1, *gBest* and IP_2. Figure 2 shows an example of matching for behavior Pattern 1 in a typical optimization problem in a 2D-space.

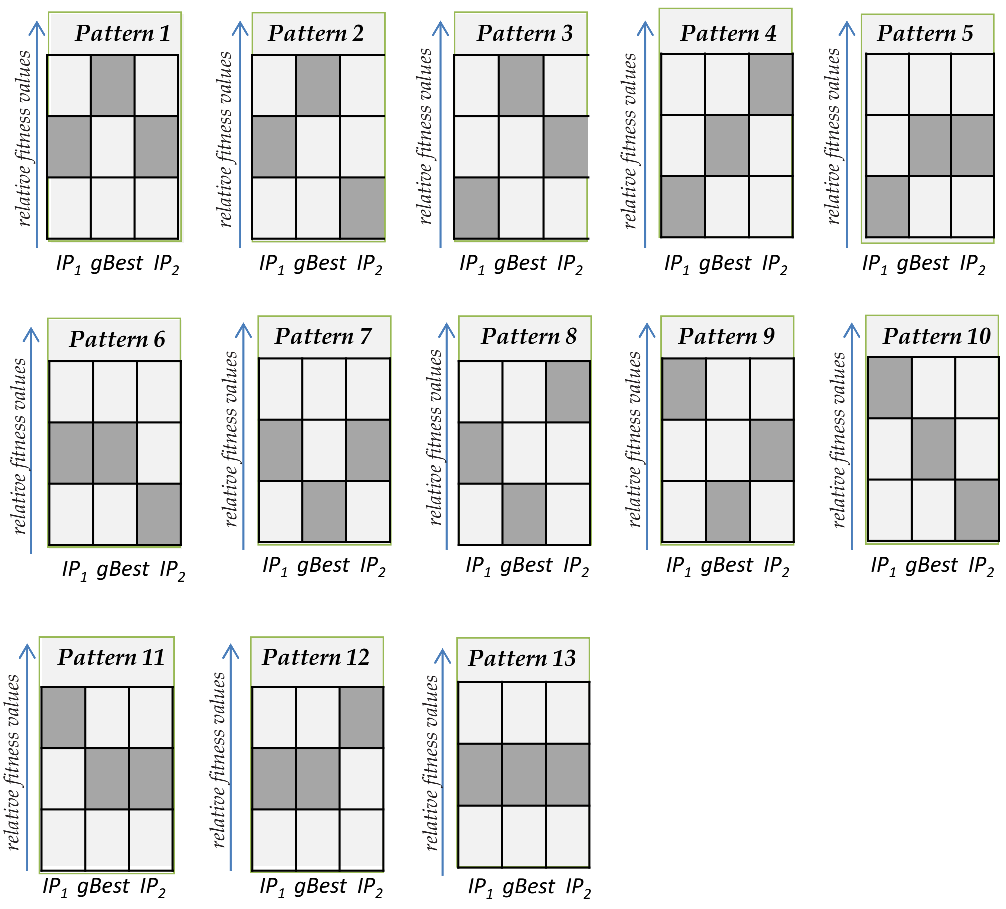


Figure 1. Behavior patterns used to implement the sagacity dimension considering a matrix 3×3

Although the example in Figure 2 is in 2D-space, the match can also occur in a multi-dimensional space. A straight line can implement a cut in the domain of the fitness function in any space. The direction of the straight line is stochastic, since it is the arrow that links *gBest* to a random point AP. In each iteration, this direction changes and during all the iterations it gives a reasonable perception of the fitness function behavior.

4. **Computational Experiments.** The proposed algorithm was evaluated on several benchmark functions which are presented in Subsection 4.1. In Subsection 4.2, the parameter settings are presented. In the last subsection, the algorithm is compared with Standard PSO, as well as some variants such as PSO-Scout, HWPSO, WQPSO, QPSO and HPSOM.

Table 1. Rules set that treats the behavior patterns to define Inspection Points around *gBest* particle

Rule	**Condition**	**Action**
01	$fit(IP_1) < fit(IP_2) \wedge$ $fit(IP_2) < fit(gBest)$	$Generate\colon NP_1 = IP_1 - \frac{\lambda}{2} \cdot \vec{d}$ $gBest = \begin{cases} NP_1, & \text{if } fit(NP_1) < fit(IP_1) \\ IP_1, & \text{otherwise} \end{cases}$
02	$fit(IP_2) < fit(IP_1) \wedge$ $fit(IP_1) < fit(gBest)$	$Generate\colon NP_2 = IP_2 + \frac{\lambda}{2} \cdot \vec{d}$ $gBest = \begin{cases} NP_2, & \text{if } fit(NP_2) < fit(IP_2) \\ IP_2, & \text{otherwise} \end{cases}$
03	$fit(IP_1) == fit(IP_2) \wedge$ $fit(IP_1) < fit(gBest)$	$Generate\colon NP_1 = IP_1 - \frac{\lambda}{2} \cdot \vec{d}\,,\ NP_2 = IP_2 + \frac{\lambda}{2} \cdot \vec{d}$ $gBest = \begin{cases} NP_1, & \text{if } fit(NP_1) < fit(NP_2) \wedge fit(NP_1) < fit(IP_1) \\ NP_2, & \text{if } fit(NP_2) < fit(NP_1) \wedge fit(NP_2) < fit(IP_2) \\ raffle(NP_1, NP_2), & \text{otherwise} \end{cases}$
04	$fit(IP_1) < fit(gBest) \wedge$ $fit(gBest) \leq fit(IP_2)$	Idem Rule 01
05	$fit(IP_2) < fit(gBest) \wedge$ $fit(gBest) \leq fit(IP_1)$	Idem Rule 02
06	$fit(IP_1) == fit(IP_2) \wedge$ $fit(IP_2) == fit(gBest)$	*gBest* remains unchanged
07	$fit(gBest) \leq fit(IP_1) \wedge$ $fit(IP_1) \leq fit(IP_2)$	*gBest* remains unchanged
08	$fit(gBest) \leq fit(IP_2) \wedge$ $fit(IP_2) < fit(IP_1)$	*gBest* remains unchanged

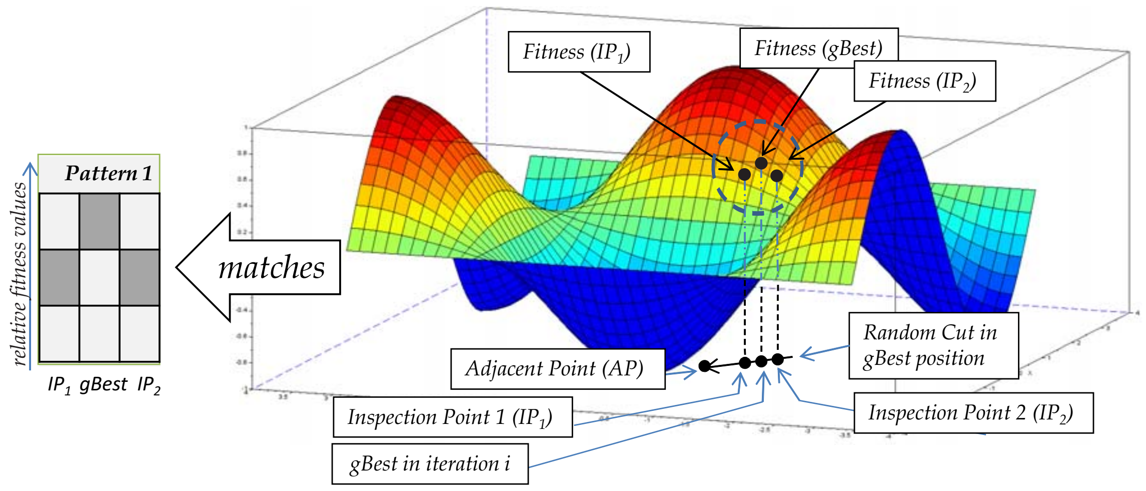


Figure 2. Example for Pattern 1 in 2D-space

4.1. **Benchmark functions.** Four benchmark functions were chosen and associated to high-dimensional problems on which a minimization process is applied. These funtions [35] can be used to investigate the stagnation and the convergence of the algorithm. They have been applied in several studies of PSO such as [27, 29, 30, 40, 45, 46]. The functions are described below:

- Spherical function – it has no interaction between its variables. It is very simple, convex and unimodal:

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

- Rosenbrock function – it has interaction between some variables, whose global minimum is in a parabolic valley. Although it is easy to find the parabolic valley, the convergence to the minimum is difficult [47]:

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

- Griewank function – it has interactions between variables. The function has many widespread local minima, which are regularly distributed:

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

- Rastrigin function – it has several local minima and the locations of the minima are regularly distributed:

$$f_4(x) = \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$$

Table 2 presents the boundaries of the search space, the initialization ranges, the value that represents the global optimum and the optimal solution.

4.2. **Parameter settings.** The values assigned to w, c_1 and c_2, defined in Equation (5), can vary the performance of the Standard PSO and its variants [39]. In order to have a fair comparison, parameters in common of Standard PSO and SBPSO were set to the same value: a) $c_1 = c_2 = 2.0$, b) the inertia weight w that decreases linearly starting at

TABLE 2. Characteristics of the benchmark functions

Function	Search Space	Initialization Range	Global Optimum	Optimal Solution
f_1 – Spherical	$-100 \leq x_i \leq 100$	$50 \leq x_i \leq 100$	$f(x^*) = 0$	$x^* = (0, \ldots, 0)$
f_2 – Rosenbrock	$-30 \leq x_i \leq 30$	$15 \leq x_i \leq 30$	$f(x^*) = 0$	$x^* = (1, \ldots, 1)$
f_3 – Griewank	$-600 \leq x_i \leq 600$	$300 \leq x_i \leq 600$	$f(x^*) = 0$	$x^* = (0, \ldots, 0)$
f_4 – Rastrigin	$-5.12 \leq x_i \leq 5.12$	$2.56 \leq x_i \leq 5.12$	$f(x^*) = 0$	$x^* = (0, \ldots, 0)$

0.9 and ending at 0.4 and c) the maximum velocity $V_{\max}$ of each particle corresponds to half of the length of the search space in one dimension [48].

There are some specific parameters of the SBPSO related to a) the number of scout particles, b) the velocity of the scouts, c) the creation of Adjacent and Inspection Points near the *gBest* particle and d) the detection of premature convergence.

The parameter that defines the number of scouts was set at 10% of the total of the swarm particles. Other parameters were also presented in Equations (7), (9) and (11). Respectively, these parameters were set $c_3 = 1.3$, α was set to be 1% of the length of the search space in one dimension and $\lambda = 0.1$. To detect premature convergence, two thresholds were used: δ_{stag} and δ_{conv}. The threshold δ_{stag} was set at 10^{-3} and δ_{conv} was assigned the value of 5% of the total number of iterations. The threshold δ_{conv} was used to define the number of iterations whose fitness value of the *gBest* particle was not improved significantly. The values assigned to these parameters were empirically chosen after several simulations that presented good results for all evaluated functions.

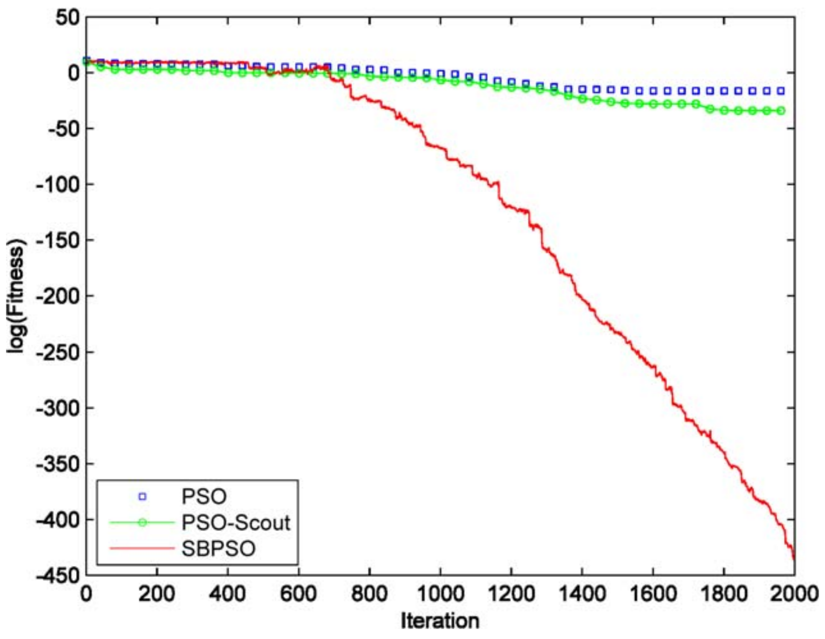
4.3. **Experimental results.** The concept of serendipity, when combined with the use of scout particles, shows that the SBPSO behavior is more active than the Standard PSO and some variants. The performance of the proposed algorithm is evaluated as follows: first, SBPSO is compared with Standard PSO and PSO-Scout. Next, SBPSO is compared with other four PSO variants available in the literature such as QPSO [28], WQPSO [29], HWPSO [30] and HPSOM [27]. The comparison also considers PSO-Scout.

PSO-Scout is a Particle Swarm Optimization variant that we implemented in this work. This variant improves the Standard PSO performance using the exploratory behavior of the scout particles. Scouts are used both as general mechanisms to globally improve the algorithm and also as a simple approach to incorporate specific knowledge to solve problems. The behavior of such particles can be controlled using Equations (7) and (8), presented in Section 3. The number of scout particles used in this variant is equivalent to 10% of the total of the swarm particles.

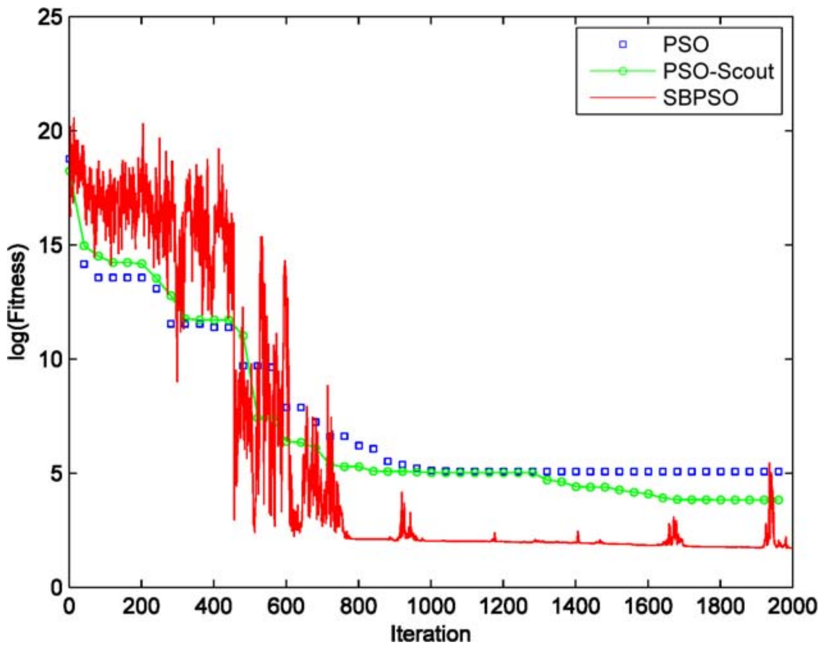
Figures 3(a), 3(b), 4(a) and 4(b) compare the average behavior of the swarm (5 particles) on Spherical, Rosenbrock, Griewank and Rastrigin functions in 2000 iteractions, respectively. The figures show that Standard PSO and PSO-Scout stagnate before iteration 2000, but SBPSO continues its activity.

The scalability of the SBPSO is also investigated. The population size (20, 40 and 80), the dimension of the functions (10, 20 and 30) and the maximum number of iterations (1000, 1500 and 2000) are varied for each benchmark function. In these experiments, the best mean and the standard deviation obtained are recorded.

Figures 5, 6, 7 and 8 show the convergence process averaged on 100 executions. The process occurs in the search space of the Spherical (Figure 5), Rosenbrock (Figure 6), Griewank (Figure 7) and Rastrigin (Figure 8) functions using 20 particles, 30 dimensions and 2000 iterations. Tables 3 and 4 show the population size, the dimensionality of the

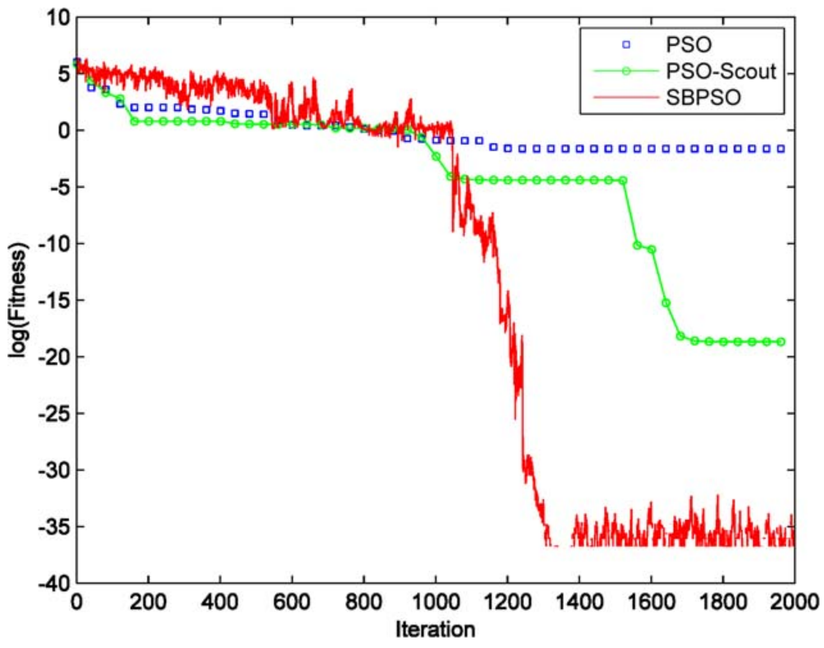


(a)

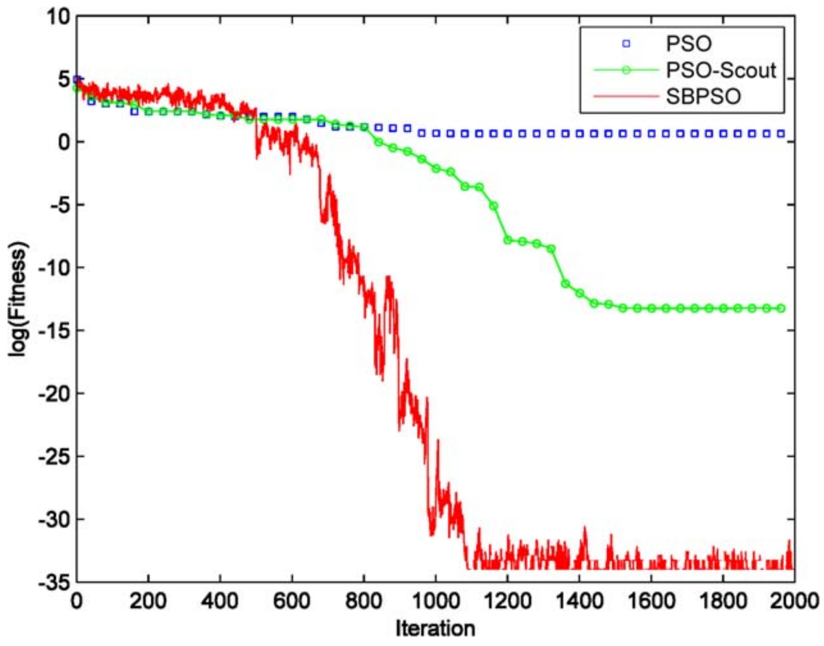


(b)

Figure 3. Average behavior of the swarm (5 particles) during 2000 iterations: (a) for Spherical function, the Standard PSO stagnated next to iteration 1380 and PSO-Scout stagnated next to iteration 1800 and (b) for Rosenbrock function, the Standard PSO stagnated next to iteration 1000 and PSO-Scout stagnated next to iteration 1700. In both figures, it is observed that the particle of the SBPSO remains active next to iteraction 2000.



(a)



(b)

Figure 4. Average behavior of the swarm (5 particles) during 2000 iterations: (a) for Griewank function, the Standard PSO stagnated next to iteration 1200 and PSO-Scout stagnated next to iteration 1710 and (b) for Rastrigin function, the Standard PSO stagnated next to iteration 1000 and PSO-Scout stagnated next to iteration 1580. In both figures, it is observed that the particle of the SBPSO remains active next to iteraction 2000.

function, the number of iterations, the best mean of the fitness value and the standard deviation for each benchmark function.

Wilcoxon Test [49] is applied with significance level 0.05 to evaluate the solutions found by Standard PSO, PSO-Scout and SBPSO. Wilcoxon is a non-parametric statistical test

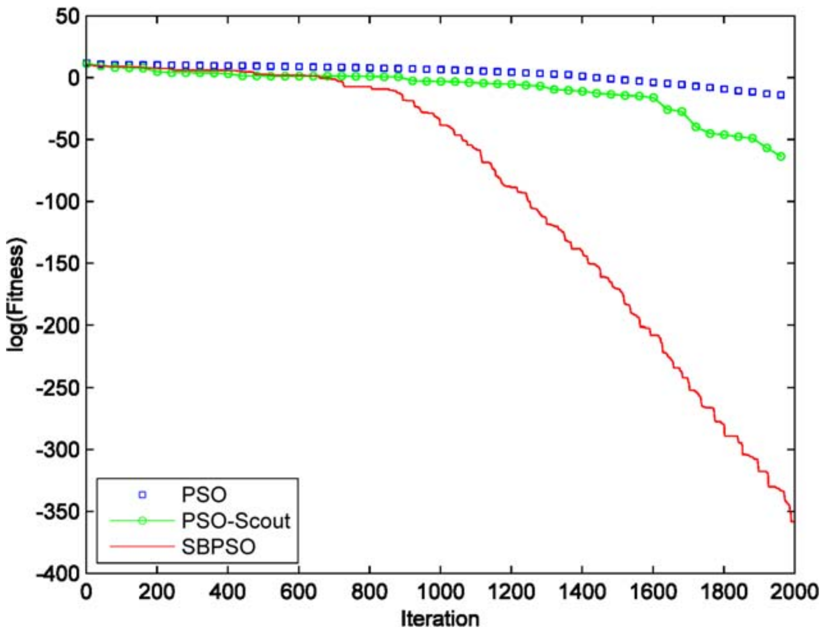


Figure 5. The average convergence process of the Standard PSO, PSO-Scout and SBPSO for Spherical function

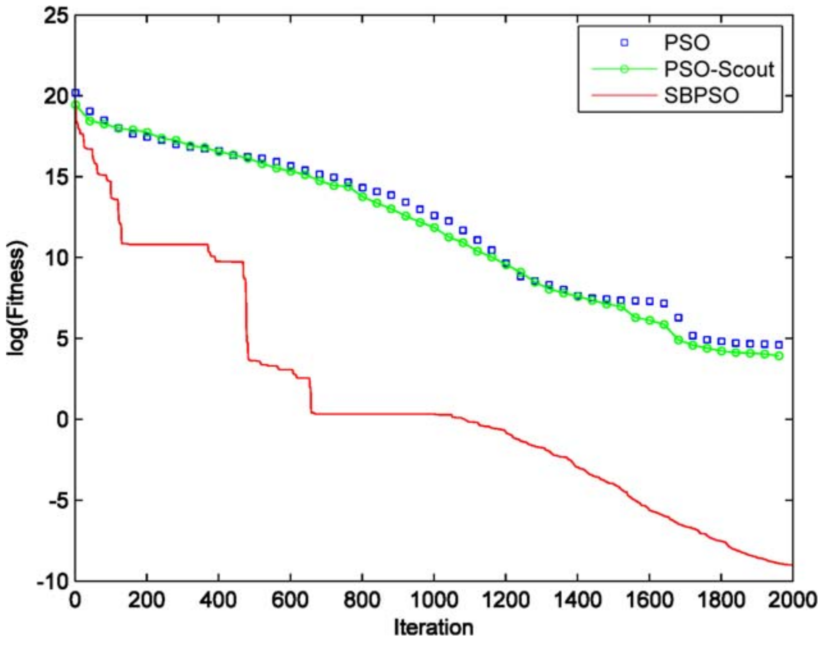


Figure 6. The average convergence process of the Standard PSO, PSO-Scout and SBPSO for Rosenbrock function

used to compare two independent samples. The null hypothesis H_0 indicates that two samples come from the same population, whereas the alternative hypothesis H_1 indicates that one has higher values than the other. When *p-value* is less than the significance level, then it decides to reject H_0, i.e., there is significant difference between the samples.

Table 5 presents *p-values* obtained by the Wilcoxon Test when the SBPSO is compared with the Standard PSO and PSO-Scout. The test is used to verify if the SBPSO results are statistically significant compared with the Standard PSO and PSO-Scout. It is observed that all experiments present *p-values* lower than the significance level set at 0.05. In such

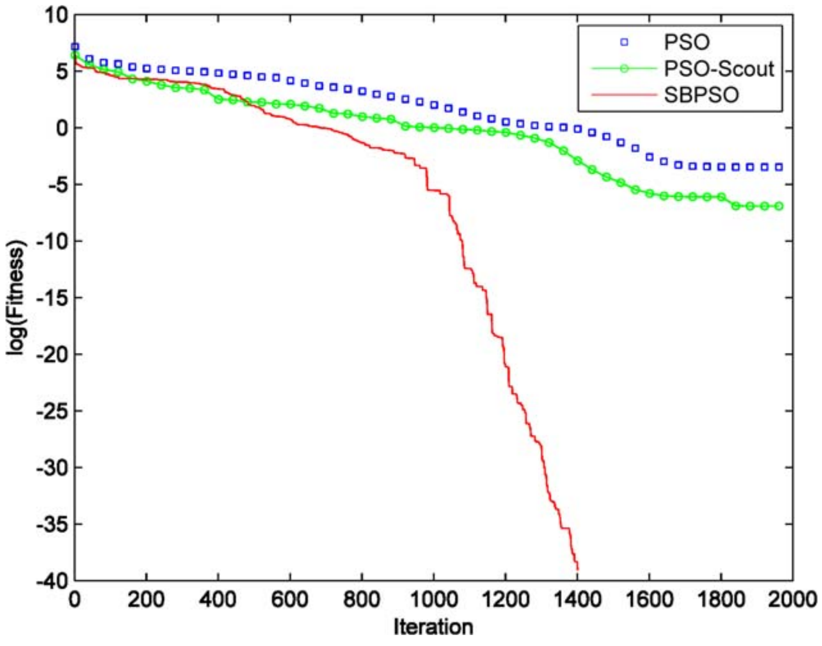


FIGURE 7. The average convergence process of the Standard PSO, PSO-Scout and SBPSO for Griewank function

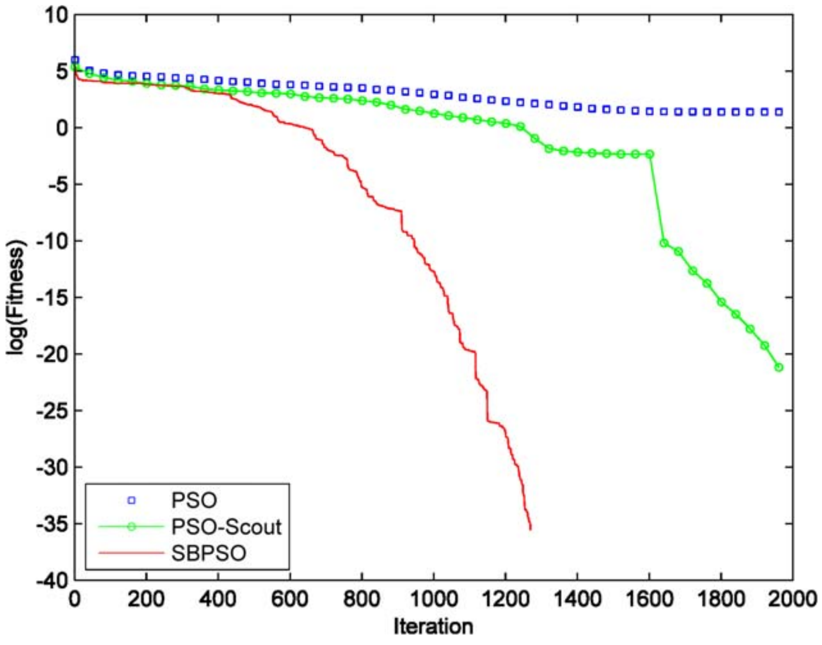


FIGURE 8. The average convergence process of the Standard PSO, PSO-Scout and SBPSO for Rastrigin function

cases, the hypothesis H_0 is rejected. This ensures that there is no equality between the solutions obtained by the methods. This means that SBPSO statistically obtained better results than the Standard PSO and PSO-Scout.

Table 6 compares the SBPSO results with some available variants in the literature such as HWPSO, WQPSO, QPSO and HPSOM. For each benchmark function, the dimensions (10, 20 and 30) and the maximum number of iterations (1000, 1500 and 2000) are varied. Only the population size is fixed to 20 particles. The mean best fitness and the standard deviation obtained by the algorithms cited previously are recorded.

Table 3. Comparison between Std. PSO, PSO-Scout and SBPSO for the unimodal functions: Spherical and Rosenbrock

Part	**Dim**	**Iter**	**Spherical (f_1)**			**Rosenbrock (f_2)**		
			Std. PSO	PSO-Scout	SBPSO	Std. PSO	PSO-Scout	SBPSO
20	10	1000	1.2368E-020 (3.1403E-020)	1.0056E-022 (5.5127E-022)	1.3900E-081 (1.2772E-080)	58.3417 (133.7896)	40.6746 (103.9401)	9.5785E-08 (5.7832E-07)
	20	1500	2.9396E-011 (1.8370E-010)	3.4132E-018 (1.9818E-017)	3.1139E-124 (2.6196E-123)	104.9516 (162.9876)	99.4901 (186.9198)	2.9085E-07 (8.6492E-06)
	30	2000	4.6804E-008 (1.3386E-007)	5.0619E-021 (5.0275E-020)	2.9522E-156 (2.9520E-155)	151.5238 (239.0893)	104.9869 (151.4036)	2.0347E-05 (7.6421E-04)
40	10	1000	2.2365E-024 (1.3369E-023)	1.3691E-027 (1.0608E-026)	3.1786E-084 (3.1719E-083)	30.80349 (114.8610)	15.6310 (38.1135)	3.7120E-08 (8.8402E-07)
	20	1500	8.1334E-015 (2.5881E-014)	2.9916E-028 (2.9840E-027)	2.2798E-119 (1.9630E-118)	81.5949 (141.4203)	48.1244 (60.5719)	6.8410E-07 (6.0376E-06)
	30	2000	8.0544E-011 (1.3452E-010)	5.4840E-034 (2.4924E-033)	5.5811E-129 (2.3349E-128)	132.6704 (204.0919)	112.7430 (13.9928)	3.7810E-05 (1.9304E-04)
80	10	1000	1.5394E-028 (4.8125E-028)	1.5496E-030 (1.0748E-029)	9.7470E-081 (9.7469E-080)	20.5744 (43.8337)	13.9724 (34.8846)	4.8522E-08 (4.3964E-07)
	20	1500	5.1700E-018 (1.4221E-017)	2.3584E-026 (1.9487E-025)	5.6513E-120 (5.6513E-119)	65.7612 (105.0775)	52.7690 (92.4566)	6.9654E-07 (3.0942E-06)
	30	2000	1.4817E-013 (2.9377E-013)	2.2470E-031 (2.2468E-030)	1.1080E-162 (1.1113E-161)	86.87974 (123.5531)	69.8944 (92.4215)	3.8413E-05 (7.7475E-04)

Table 4. Comparison between Std. PSO, PSO-Scout and SBPSO for the multimodal functions: Griewank and Rastrigin

Part	**Dim**	**Iter**	**Griewank (f_3)**			**Rastrigin (f_4)**		
			Std. PSO	PSO-Scout	SBPSO	Std. PSO	PSO-Scout	SBPSO
20	10	1000	0.1012 (0.0516)	0.0216 (0.0456)	0 (0)	4.5782 (2.1132)	0.9514 (0.6391)	0 (0)
	20	1500	0.0334 (0.0336)	4.2475E-04 (5.1756E-03)	0 (0)	22.8061 (10.0912)	0.0475 (0.3913)	0 (0)
	30	2000	0.0146 (0.0171)	2.4458E-04 (1.4592E-03)	0 (0)	49.7192 (13.7956)	0.0253 (0.1905)	0 (0)
40	10	1000	0.0845 (0.0438)	0.0192 (0.0396)	0 (0)	3.9224 (1.5118)	0.2854 (0.4404)	0 (0)
	20	1500	0.0276 (0.0291)	7.9024E-04 (3.7688E-03)	0 (0)	16.1082 (1.5722)	8.7266E-04 (8.7266E-03)	0 (0)
	30	2000	0.0592 (0.1690)	1.7253E-04 (1.2263E-03)	0 (0)	38.6344 (2.2039)	9.5142E-04 (9.5142E-03)	0 (0)
80	10	1000	0.0772 (0.0396)	0.0188 (0.0339)	0 (0)	2.2854 (0.4780)	0.1141 (0.3123)	0 (0)
	20	1500	0.0357 (0.0325)	3.7898E-04 (1.4178E-03)	0 (0)	12.3630 (2.2804)	3.5527E-04 (6.6714E-03	0 (0)
	30	2000	0.0127 (0.0133)	1.0129E-04 (9.8582E-03)	0 (0)	31.5714 (6.2924)	7.6115E-04 (3.2228E-03)	0 (0)

The numerical results presented in Table 6 show that SBPSO outperforms other variants of the Standard PSO when evaluated on benchmark functions presented in Subsection 4.1. In the evaluated unimodal functions, the algorithm obtains good solutions with high accuracy. The algorithm also shows a good performance for functions with multiple local minima.

Spherical function is used to test the ability of the SBPSO for local searches. Rosenbrock is widely used to test the ability of the algorithms in local and global searches. Its global

Table 5. Results of the Wilcoxon Test with significance level 0.05

Part	**Dim**	**Iter**	**p-Value (Std. PSO $\times$ SBPSO)**				**p-Value (PSO-Scout $\times$ SBPSO)**			
			f_1	f_2	f_3	f_4	f_1	f_2	f_3	f_4
20	10	1000	1.5777E-30	1.1453E-21	1.5777E-30	1.4551E-10	1.5777E-30	1.6713E-17	1.0444E-17	3.6379E-06
	20	1500	1.5777E-30	3.9312E-20	1.5777E-30	1.5777E-30	1.5777E-30	1.0622E-07	5.6843E-15	3.3881E-21
	30	2000	1.5777E-30	9.3528E-18	1.5777E-30	1.5777E-30	1.5777E-30	5.8320E-15	7.4505E-13	2.5849E-23
40	10	1000	1.5777E-30	2.3377E-19	1.5777E-30	1.9073E-12	1.5777E-30	5.8344E-15	5.2939E-23	6.1035E-06
	20	1500	1.5777E-30	9.2688E-17	6.3108E-30	6.7762E-21	1.5777E-30	9.5982E-14	5.8207E-11	9.5367E-07
	30	2000	1.5777E-30	1.5949E-16	1.5777E-30	1.5777E-30	1.5777E-30	3.2755E-17	1.4901E-08	2.2204E-16
80	10	1000	1.5777E-30	1.1296E-19	1.5777E-30	4.8828E-10	1.5777E-30	4.4341E-16	6.6174E-24	3.1250E-06
	20	1500	1.5777E-30	6.5128E-18	4.0389E-28	1.7763E-12	1.5777E-30	1.6376E-15	7.4505E-09	1.4210E-06
	30	2000	1.5777E-30	7.3438E-22	1.5777E-30	1.5777E-30	1.5777E-30	1.3199E-17	1.9073E-06	2.7755E-17

minimum is located in a parabolic valley in a region of difficult convergence. Griewank and Rastrigin are employed to test the ability of the algorithm to perform global searches.

5. **Conclusion.**

This work presented a PSO variant called Serendipity-Based Particle Swarm Optimization to delay premature convergence in typical metaheuristic optimization problems, particularly considering the PSO methods. The algorithm prototype used an approach that considers two dimensions for the concept of serendipity: chance and sagacity. Chance dimension was implemented with the use of scout particles to enhance the exploration within the search space. Sagacity dimension was implemented by the use of a machine learning technique known as classification.

After the experimentation, the results were promising and showed that SBPSO outperformed the Standard PSO and the PSO-Scout variant. Two criteria were analyzed in the comparisons: convergence and stagnation. In convergence criteria, the algorithm prototype found global optimum solutions before the other algorithms on Griewank and Rastrigin functions. For Spherical and Rosenbrock functions, none of the algorithms found optimal solutions, but the SBPSO found a better local optimum than the other ones. When the stagnation criteria was evaluated for Spherical function, the proposed algorithm delayed the stagnation in 31% of the number of iterations when compared with Standard PSO and 10% in comparison with PSO-Scout, approximately. For Rosenbrock function, it delayed in 50% and 15% in relation to Standard PSO and PSO-Scout, respectively. For Griewank function, SBPSO delayed the stagnation in 40% when compared with Standard PSO and 14.5% in comparison with PSO-Scout, respectively. For Rastrigin function, it delayed in 50% and 21% in relation to Standard PSO and PSO-Scout, respectively.

The experiments were also performed to compare SBPSO with some studies in the literature, considering the same size of population (20 particles) and the same number of variables and iterations. In all experiments, SBPSO also showed a better convergence behavior, outperforming the Standard PSO and some variants regarding the solution quality, the ability to find global optimum, the solutions stability and the ability to restart the movement of the swarm after stagnation has been detected.

In general, the results were promising for the context of metaheuristic algorithms, since the premature convergence was really delayed in most of the experiments. However, it was observed that the prototype may need some adjustments to improve its run time, since there is an increase of time in the use of SBPSO.

Future research may investigate the viability of adapting and implementing other strategies for serendipity. One of them would be an approach based on “Anomalies and exceptions”, since it considers an important feature called dissimilarity that could be used to guide the movement of the scout particles in searching of the opposite direction of the swarm particles. This feature is an adaptation of the concept of similarity and could be

Table 6. Comparison between different PSO variants with 20 particles for benchmark functions

Function	Dim	Iter	Std. PSO	HWPSO	WQPSO	QPSO	HPSOM	PSO-Scout	SBPSO
f_1	10	1000	1.2368E-20 (3.1403E-20)	6.2868E-56 (1.506E-55)	2.2922E-56 (1.7300E-58)	1.3909E-41 (1.4049E-043)	2.2400E-56 (1.7300E-58)	1.0056E-22 (5.5127E-22)	1.3900E-81 (1.2772E-80)
	20	1500	2.9396E-11 (1.8370E-10)	6.2830E-45 (2.0330E-44)	2.9451E-40 (2.8717E042)	3.5103E-022 (3.5452E-24)	2.1449E-49 (1.6891E-43)	3.4132E-18 (1.9818E-17)	3.1139E-124 (2.6196E-123)
	30	2000	4.6804E-08 (1.3386E-07)	3.7940E-36 (1.4060E-35)	3.9664E-33 (3.8435E-35)	5.3183E-014 (5.3623E-16)	6.5764E-034 (5.6809E-36)	5.0619E-21 (5.0275E-20)	2.9522E-156 (2.9520E-155)
f_2	10	1000	58.3417 (133.7896)	36.4736 (0.1844)	35.8436 (0.2843)	51.9761 (0.4737)	6.9688 (0.2730)	40.6746 (103.9401)	9.5785E-08 (5.7832E-07)
	20	1500	104.9516 (162.9876)	65.6678 (0.5870)	62.7696 (0.4860)	136.8782 (0.6417)	17.3033 (0.2210)	99.4901 (186.9198)	2.9085E-07 (8.6492E-06)
	30	2000	151.5238 (239.0893)	70.7275 (0.4813)	70.9525 (0.4283)	157.4707 (0.8287)	27.5645 (0.2939)	104.9869 (151.4036)	2.0347E-05 (7.6421E-04)
f_3	10	1000	0.1012 (0.0516)	0.13333 (0.33993)	5.6353E-04 (5.5093E-04)	5.5093E-04 (0.0657)	4.32057E-11 (3.1216E-11)	5.0968E-02 (6.5753E-02)	0 (0)
	20	1500	0.0334 (0.0336)	2.9333 (2.7439)	2.1318E-04 (1.0402E-04)	1.0402E-04 (0.0211)	4.00370E-11 (3.13852E-11)	2.2766E-02 (2.1534E-02)	0 (0)
	30	2000	0.0146 (0.0171)	9.2333 (6.1455)	2.1286E-04 (1.2425E-04)	1.2425E-04 (0.0110)	5.1756E-11 (3.0143E-11)	1.0973E-02 (1.4329E-02)	0 (0)
f_4	10	1000	4.5782 (2.1132)	4.6100 (2.5364)	4.0567 (0.0094)	4.8274 (0.0015)	4.1558E-11 (3.2202E-11)	1.1796 (0.9522)	0 (0)
	20	1500	22.8061 (10.0912)	19.6670 (6.7661)	12.1102 (0.0287)	16.0519 (0.0414)	4.1403E-11 (3.2402E-11)	6.2384 (2.8271)	0 (0)
	30	2000	49.7192 (13.7956)	44.7230 (13.9680)	23.5593 (0.0713)	33.7218 (0.0114)	4.8007E-11 (3.1883E-11)	11.9819 (4.7205)	0 (0)

combined with the "blind luck" strategy to improve the exploratory behavior of the scout particles. It would also be interesting to evaluate the performance and the run time of the algorithm for a large number of inspection points.

REFERENCES

[1] C.-M. Pintea, Advances in bio-inspired computing for combinatorial optimization problems, *Intelligent Systems Reference Library*, vol.57, 2014.

[2] C. R. M. Silva, H. W. C. Lins, S. R. Martins, E. L. F. Barreto and A. G. D'Assuncao, A multiobjective optimization of a UWB antenna using a self organizing genetic algorithm, *Microwave and Optical Technology Letters*, vol.54, no.8, pp.1824-1828, 2012.

[3] C. R. M. Silva and S. R. Martins, An adaptive evolutionary algorithm for UWB microstrip antennas optimization using a machine learning technique, *Microwave and Optical Technology Letters*, vol.55, no.8, pp.1864-1868, 2013.

[4] A. Esmin, G. Lambert-Torres and A. de Souza, A hybrid particle swarm optimization applied to loss power minimization, *IEEE Trans. Power Systems*, vol.20, pp.859-866, 2005.

[5] S. Chiaradonna, F. Di Giandomenico and N. Murru, On enhancing effciency and accuracy of particle swarm optimization algorithms, *International Journal of Innovative Computing, Information and Control*, vol.11, no.4, pp.1165-1189, 2015.

[6] A.-P. Chen, C.-H. Huang and Y.-C. Hsu, A novel modified particle swarm optimization for forecasting financial time series, *IEEE International Conference on Intelligent Computing and Intelligent Systems*, vol.1, pp.683-687, 2009.

[7] L. Jin and Y. Huang, A particle swarm optimization-neural network prediction model for typhoon intensity based on isometric mapping algorithm, *The 5th International Joint Conference on Computational Sciences and Optimization*, pp.857-861, 2012.

[8] X. Dong, Y. Zhao, Y. Xu, Z. Zhang and P. Shi, Design of PSO fuzzy neural network control for ball and plate system, *International Journal of Innovative Computing, Information and Control*, vol.7, no.12, pp.7091-7103, 2011.

[9] A. V. Dhanraj and D. Nanjundappan, Design of optimized PI controller with ideal decoupler for a non linear multivariable system using particle swarm optimization technique, *International Journal of Innovative Computing, Information and Control*, vol.10, no.1, pp.341-355, 2014.

[10] S. Binitha and S. S. Sathya, A survey of bio inspired optimization algorithms, *International Journal of Soft Computing & Engineering*, vol.2, pp.137-151, 2012.

[11] X. Yao and Y. Xu, Recent advances in evolutionary computation, *J. Comput. Sci. Technol.*, vol.21, no.1, pp.1-18, 2006.

[12] W. Gurney and R. M. Nisbet, *Ecological Dynamics*, Oxford University Press, 1998.

[13] R. L. Olson and R. A. Sequeira, Emergent computation and the modeling and management of ecological systems, *Computers and Eletronics im Agriculture*, vol.12, no.3, pp.183-209, 1995.

[14] E. Silva and C. Bastos-Filho, PSO efficient implementation on GPUs using low latency memory, *Latin America Transactions, Revista IEEE America Latina*, vol.13, pp.1619-1624, 2015.

[15] J. Kennedy and R. Eberhart, Particle swarm optimization, *Proc. of IEEE International Conference on Neural Networks*, vol.4, pp.1942-1948, 1995.

[16] Y. Del Valle, G. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez and R. Harley, Particle swarm optimization: Basic concepts, variants and applications in power systems, *IEEE Trans. Evolutionary Computation*, vol.12, pp.171-195, 2008.

[17] P. V. Andel, Anatomy of the unsought finding. Serendipity: Origin, history, domains, traditions, appearances, patterns and programmability, *The British Journal for the Philosophy of Science*, vol.45, no.2, pp.631-648, 1994.

[18] T. S. Kuhn, *The Structure of Scientific Revolutions*, 2nd Edition, University of Chicago Press, Chicago, 1970.

[19] F. A. P. Paiva, J. A. F. Costa and C. R. M. Silva, A hierarchical architecture for ontology-based recommender systems, *Computational Intelligence and the 11th Brazilian Congress on Computational Intelligence*, pp.362-367, 2013.

[20] F. A. P. Paiva, J. A. F. Costa and C. R. M. Silva, An ontology-based recommender system architecture for semantic searches in vehicles sales portals, *Hybrid Artificial Intelligence Systems, Lecture Notes in Computer Science*, vol.8480, pp.537-548, 2014.

[21] M. Ge, C. Delgado-Battenfeld and D. Jannach, Beyond accuracy: Evaluating recommender systems by coverage and serendipity, *Proc. of the 4th ACM Conference on Recommender Systems*, New York, NY, USA, pp.257-260, 2010.

[22] K. Oku and F. Hattori, Fusion-based recommender system for improving serendipity, *CEUR Workshop Proceedings*, vol.816, pp.19-26, 2011.

[23] P. Adamopoulos and E. Tuzhilin, On unexpectedness in recommender systems: Or how to expect the unexpected, *Proc. of RecSys the 11th Intl. Workshop on Novelty and Diversity in Recommender Systems*, 2011.

[24] F. A. P. Paiva, J. A. F. Costa and C. R. M. Silva, UMA meta-heuristica alternativa de inteligencia de enxames baseada em serendipidade guiada, *The 2nd LA-CCI (Latin American) and the 12th CBIC Brazilian Congress on Computational Intelligence*, 2015.

[25] C. Li, Y. Liu, A. Zhou, L. Kang and H. Wang, A fast particle swarm optimization algorithm with cauchy mutation and natural selection strategy, *Advances in Computation and Intelligence*, pp.334-343, 2007.

[26] W. Jiao, G. Liu and D. Liu, Elite particle swarm optimization with mutation, *Asia Simulation Conference – The 7th International Conference on System Simulation and Scientific Computing*, pp.800-803, 2008.

[27] A. Esmin and S. Matwin, HPSOM: A hybrid particle swarm optimization algorithm with genetic mutation, *International Journal of Innovative Computing, Information and Control*, vol.9, no.5, pp.1919-1934, 2013.

[28] J. Sun, B. Feng and W. Xu, Particle swarm optimization with particles having quantum behavior, *Congress on Evolutionary Computation*, vol.1, pp.325-331, 2004.

[29] M. Xi, J. Sun and W. Xu, An improved quantum-behaved particle swarm optimization algorithm with weighted mean best position, *Applied Mathematics and Computation*, vol.205, no.2, pp.751-759, 2008.

[30] S.-H. Ling, H. H. C. Iu, K. Y. Chan, H.-K. Lam, C. W. Yeung and F. H. F. Leung, Hybrid particle swarm optimization with wavelet mutation and its industrial applications, *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, vol.38, no.3, pp.743-763, 2008.

[31] S. Catellin, *Srendipit: Du Conte au Concept*, Seuil, Paris, 2014.

[32] J. Campos and A. D. Figueiredo, Programming for serendipity, *Proc. of Fall Symposium on Chance Discovery – The Discovery and Management of Chance Events*, pp.48-60, 2002.

[33] E. Toms, Serendipitous information retrieval, *The 1st DELOS Network of Excellence Workshop on Information Seeking, Searching and Querying in Digital Libraries*, pp.11-12, 2000.

[34] G. I. Evers and M. B. Ghalia, Regrouping particle swarm optimization: A new global optimization algorithm with improved performance consistency across benchmarks, *SMC*, pp.3901-3908, 2009.

[35] F. Van Den Bergh, *An Analysis of Particle Swarm Optimizers*, Ph.D. Thesis, Pretoria, South Africa, 2002.

[36] M.-X. Zhang, B. Zhang and Y.-J. Zheng, Bio-inspired meta-heuristics for emergency transportation problems, *Algorithms*, vol.7, no.1, p.15, 2014.

[37] G. R. Souza, E. F. G. Goldbarg, M. C. Goldbarg and A. M. P. Canuto, A multiagent approach for metaheuristics hybridization applied to the traveling salesman problem, *Proc. of the 2012 Brazilian Symposium on Neural Networks*, Washington, DC, USA, pp.208-213, 2012.

[38] N. Higashi and H. Iba, Particle swarm optimization with gaussian mutation, *Proc. of the 2003 IEEE Swarm Intelligence Symposium*, pp.72-79, 2003.

[39] H. Wang, H. Li, Y. Liu, C. Li and S. Zeng, Opposition-based particle swarm algorithm with cauchy mutation, *IEEE Congress on Evolutionary Computation*, pp.4750-4756, 2007.

[40] S. Li, M. Tan, I. Tsang and J.-Y. Kwok, A hybrid PSO-BFGS strategy for global optimization of multimodal functions, *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, vol.41, pp.1003-1014, 2011.

[41] Y.-L. Wu, T.-F. Ho, S. J. Shyu and B. M. T. Lin, Discrete particle swarm optimization with scout particles for library materials acquisition, *The Scientific World Journal*, vol.2013, pp.1-11, 2013.

[42] H. Koyuncu and R. Ceylan, Scout particle swarm optimization, *The 6th European Conference of the International Federation for Medical and Biological Engineering*, Dubrovnik, Croatia, pp.82-85, 2014.

[43] A. Silva, A. Neves and T. Gonçalves, Using scout particles to improve a predator-prey optimizer, *Proc. of the 11th International Conference on Adaptive and Natural Computing Algorithms*, Lausanne, Switzerland, pp.130-139, 2013.

[44] J. Lawley, Maximising serendipity: The art of recognising and fostering unexpected potential – A systemic approach to change, *Neuro Linguistic Psychotherapy and Counselling Association*, 2013.

[45] J. Sun, W. Fang, X. Wu, V. Palade and W. Xu, Quantum-behaved particle swarm optimization: Analysis of individual particle behavior and parameter selection, *Evolutionary Computation*, vol.20, pp.349-393, 2012.

[46] C. Sun, J. Zeng, J. Pan, S. Xue and Y. Jin, A new fitness estimation strategy for particle swarm optimization, *Inf. Sci.*, vol.221, pp.355-370, 2013.

[47] V. Picheny, T. Wagner and D. Ginsbourger, A benchmark of kriging-based infill criteria for noisy optimization, *Structural and Multidisciplinary Optimization*, vol.48, no.3, pp.607-626, 2013.

[48] W.-J. Zhang and X.-F. Xie, DEPSO: Hybrid particle swarm with differential evolution operator, *IEEE International Conference on Systems, Man and Cybernetics*, vol.4, pp.3816-3821, 2003.

[49] J. Demšar, Statistical comparisons of classifiers over multiple data sets, *The Journal of Machine Learning Research*, vol.7, pp.1-30, 2006.