MATIMYÁS MATEMATIKA
ISSN 0115-6926

Journal of the Mathematical Society of the Philippines
Vol. 45 No. 2 (2022) pp. 1-24

A Particle Swarm Optimization Algorithm using Gamma Distribution Function

Nialle Loui Mar "NLM" T. Alcantara
Mathematics and Allied Disciplines Department
Don Mariano Marcos Memorial State University - South La Union Campus
Agoo, La Union
nlalcantara@dmmmsu.edu.ph

Rizavel C. Addawe
Department of Mathematics and Computer Science
University of the Philippines Baguio
Baguio City, Benguet
rcaddawe@up.edu.ph

Joel M. Addawe
Department of Mathematics and Computer Science
University of the Philippines Baguio
Baguio City, Benguet
jmaddawe@up.edu.ph

Abstract

The Particle Swarm Optimization (PSO) is a swarm intelligence-based, population-based meta-heuristic optimization algorithm. In this study, a variant of the PSO is introduced by using the standard gamma distribution function in each particle's movement. The mathematical model

$$S = (S_t)_{t\in\mathbb{N}_0} = ((X_t, V_t, L_t, G_t))_{t\in\mathbb{N}_0} = ((X_0, V_0, L_0, G_0), (X_1, V_1, L_1, G_1), \ldots),$$

where S is the swarm, N the number of particles, and X_t, V_t, L_t, and G_t the properties of each particle of the gamma-based swarm was presented and analysed. Numerical simulations were performed on five (5) test functions to validate the analytical results by identifying the effects of the different values of w and α to the overall performance of the proposed algorithm in a setting that is beneficial to it. Results showed that decreasing the value of w, with their α values constant, will decrease the number of iterations needed to find the global optima. Meanwhile, decreasing the value of α, with w being constant, will decrease the number of iterations needed to find the global optima. These observations, however, have exceptions. Furthermore, numerical experiments on the same benchmark functions to test success rates, average convergence values, and average convergence velocities were conducted to compare performance of the proposed algorithm with the standard PSO. The results suggests that the standard PSO outperforms the proposed algorithm in terms of success rates on some functions, which is also true in terms of average cost value obtained on some functions, although the difference between cost value is relatively small. In terms of average convergence velocity, the proposed algorithm outperformed the standard PSO on some functions.

Keywords: Particle Swarm Optimization, meta-heuristics, optimization, algorithm, gamma distribution
2020 MSC: 90C26, 90C59, 62M99

1 Introduction

The Particle Swarm Optimization (PSO) is a swarm intelligence-based, population-based meta-heuristic optimization method proposed by Kennedy and Eberhart in 1995 [7]. Mainly inspired by the metaphor of social interaction and originally intended for the simulation of bird flock behaviour, this algorithm works by using and maintaining a population of individuals, called "*particles*", that moves around a predetermined multi-dimensional search space [5, 7, 10]. Each of these particles are candidate solution to the optimization problem at hand, changing their state and adjusting their trajectories akin to "flying" until they discover potentially better solutions or until a stopping criteria is reached [10, 11].

A particle n in the swarm S with size N is comprised of three D-dimensional vectors, namely, the *current position* ($\overrightarrow{X^n}$), *local attractor* ($\overrightarrow{L^n}$), and the *current velocity* ($\overrightarrow{V^n}$), where D is the dimensionality of the search space [13, 15, 19]. For some optimization problem with objective function $f : C \subset \mathbb{R}^n \to \mathbb{R}$ being minimized, the movement of a particle n is influenced by the update equation for one element in the vector $\overrightarrow{L^n}$ presented in equation (1) while the update equation for one element in the vector $\vec{G}$ is presented in equation (2), where $n \in N$. These update equations are dependent on the time step $t \in T$.

$$L_{t+1}^n = \begin{cases} L_t^n & \text{if} \quad f(X_{t+1}^n) \geq f(L_t^n) \\ X_{t+1}^n & \text{if} \quad f(X_{t+1}^n) \leq f(L_t^n), \end{cases} \tag{1}$$

$$G_t \in \left\{ L_t^0, L_t^1, \cdots, L_t^N \right\} \left| f(G_t) = \min \left\{ f(L_t^0), f(L_t^1), \cdots, f(L_t^N) \right\} \right. . \tag{2}$$

The positions are identified with search points $X \in S$ and velocities are identified with vectors $V \in \mathbb{R}^n$. The actual movement of the particles is governed by the velocity update equation (3) and the position update equation (4):

$$V_{t+1}^{n,d} = V_t^{n,d} + c_1 \cdot q_t^{n,d} \cdot \left[L_t^{n,d} - X_t^{n,d} \right] + c_2 \cdot r_t^{n,d} \cdot \left[G_t^d - X_t^{n,d} \right], \tag{3}$$

$$X_{t+1}^n = X_t^n + V_{t+1}^n, \tag{4}$$

where $t \in T$ is the current iteration of the algorithm, $d \in D$ denotes the particular dimension the swarm is searching, c_1 and c_2 are the acceleration coefficients that control the influence of the local and global attractor, respectively. Other parameters include the independent random sequences, $q_t^{n,d}, r_t^{n,d} \sim U(0, 1)$, and w, or the inertia weight, that governs how much of the previous velocity is retained.

The movement equations (3) and (4) are then repeated until some fixed termination criterion or the global optimum is reached. The velocity of each particle is repeatedly adjusted so that the particle stochastically fluctuates around the locations of the local and global attractor.

Since each particle of the swarm has no ability to solve any problem by itself, progress occurs only when there is particle interaction present. These particles in the swarm are organized in some sort of communication structure or topology. The members of the swarm that a certain particle can interact with are called its social neighbourhood, and the collection of these social neighbourhoods constitutes the swarm's social network [12, 13, 15, 19]. Algorithm 1 represents how the PSO algorithm generally works.

Algorithm 1

input: Objective function $f : S \to \mathbb{R}$ to be minimized
output: $G \in \mathbb{R}^D$
// Initialization
for $n = 1 \to N$ do
  Initialize position $X^n \in \mathbb{R}^D$ randomly;
  Initialize velocity $V^n \in \mathbb{R}^D$;
  Initialize local attractor $L^n := X^n$;
Initialize $G := argmin_{(L^1,\ldots,L^n)} f$;

// Movement
repeat
for $t = 1 \to T$ do
  for $n = 1 \to N$ do
    for $d = 1 \to D$ do
      $V_{t+1}^{n,d} := w \cdot V_t^{n,d} + c_1 \cdot q_t^{n,d} \cdot \left[L_t^{n,d} - X_t^{n,d}\right] + c_2 \cdot r_t^{n,d} \cdot \left[G_t^d - X_t^{n,d}\right]$;
      $X_{t+1}^n := X_t^n + V_{t+1}^n$;
    if $f(X^n) \leq f(L^n)$ then $L^n := X^n$;
    if $f(X^n) \leq f(G)$ then $G := X^n$;
until *Termination criterion is reached*;
return G;

Table 1: Classical PSO algorithm

The initialization stage is where the position, velocity, local attractor, and global attractor of each particle of the swarm are initialized. The initialization of the coordinates of each particle is taken from $U[-B_{max}, B_{max}]$ which disperses the initial positions of the particles over some bounded search space. Other methods include using pseudo-random number generators, Sobol's sequence, or using other probability distributions [19, 15, 17]. Meanwhile, velocity can be initialized randomly, by setting the initial velocity to zero, or by using the *Half-Diff* method [15]. L^n is usually set to be equal to the initial position of the particle while G follows equation (2).

The movement stage operates by continuously repeating the update equations (1), (2), (3), and (4) of each particle for every iteration. The algorithm stops: (i) when the absolute difference between the known fitness value of the optimum point of the problem and the global attractor that has been found so far is smaller than a given maximum admissible error; or (ii) when the maximum number of fitness evaluations set in advance is reached [3].

PSO method has drawn an increasing number of researches mostly because of its simplicity, ease of implementation, and efficiency. Although the PSO can be used to any function f, it is commonly applied to black box optimization problems. Several real-world problems where the PSO is applied successfully are in (1) biological, medical, and pharmaceutical application; (2) design and optimization of communication networks; and (3) clustering, classification, and data mining, among others [12].

Through the years, improvements were done to the algorithm either by creating a variant that surpasses the performance of original or by solving specific real-world problems. This study considers the possibility of using the standard gamma distribution within the framework of the original PSO and ascertain if the performance of this variant is better than the original [16].

1.1 Why Change the Probability Distribution?

One of the shortcomings of the PSO algorithm is that it is easily trapped in local optima. According to [4], this behaviour is due to the particle's velocity becoming very small with respect to the position of the local optima, which in turn contribute to the inability of the particle to jump out and escape from the local optimum. With how the velocity update equation is set up, if the range from which the random values are drawn is small, the distances will have minimal influence on the updated particle velocity, and in turn cannot increase the chance of the particle to escape local optima. Thus, it is necessary to modify and/or improve the range of the random values.

In [4], they proposed using random values generated by (a) $U[0,1]$, (b) $U[-1,1]$, and $\mathcal{N}(0,1)$ and investigated its effects on the performance of the algorithm. Positive results were produced leading them to conclude that PSO with large-scale random values can avoid falling into local optima. A similar study in [8] used $\mathcal{N}(0,1)$ and set $w = 0$ and $c_1 = c_2 = 1$ in their *Gaussian Swarm*. Results showed that it outperforms the standard PSO algorithm in both convergence velocity and ability to escape from local optima in a suite of benchmark functions. And in [14], the Lévy distribution replaced the uniform distribution in order to induce exploration at any stage of convergence due to the power law behaviour. This enables the swarm to escape from local minima.

1.2 Why use Gamma Distribution?

According to [4] and [8], the performance boost of their variants was attributed in the larger standard deviation of $\mathcal{N}(0,1)$. A larger standard deviation indicates that the values are more spread out over a wider range, which in turn helps in the swarms exploration and adds to its ability to escape local minima. The Lévy distribution produces more points at large distances from the mean which fuels the swarm's ability for exploration, and it also produces more points at very small distances to the mean which fuels the swarm's exploitation ability when compared to a Gaussian distribution.

The gamma distribution has higher standard deviation compared to $U[0,1]$, and $\mathcal{N}(0,1)$. This property of the gamma distribution might be able to contribute in enhancing the ability of the swarm to explore the search space better, and escape the local minima more effectively. Another reason is that changing either of the parameters of the gamma distribution

effectively changes how the random variates are chosen.

The relatively small random values generated by $U(0,1)$ may help in the ability of the swarm to thoroughly search an area of the search space better. However, particles may be susceptible to being trapped in a local minima. For the $\mathcal{N}(0,1)$, having negative random values may also help in the ability of the swarm to thoroughly search an area of the search space better since it enables the particles to "go back" at an area it has previously visited. However, particles may also be susceptible to being trapped in a local minima. The gamma distribution showed that it generates random values that are large enough to help escape being trapped compared to the previous two distributions, but also small enough as to not "overshoot" outside the search space when compared to the outliers produced in a Lévy distribution. It combines the best properties of all the distributions used in previous studies.

2 Gamma-based PSO Algorithm (GbPSO Algorithm)

This study will focus solely on the effects of changing the random distribution used in the velocity update equation: from $q_t^{n,d},\ r_t^{n,d} \sim U(0,1)$ to $\gamma_t^{n,d},\ \beta_t^{n,d} \sim Gamma(\alpha, \lambda = 1)$.

In the initialization phase of the GbPSO, the particles' initial position will be initialized using a uniform distribution while $V_0^{n,d} = 0$, and $L_0^n = X_0^{n,d}$. The topology used is the star topology. Since the initialization phase of the GbPSO is the same as the initialization phase of the original PSO, the same probability space defined in Definition 2.1 is utilized and then relate the positions, velocities, and attractors of the GbPSO algorithm as random variables over the same probability space $\mathcal{R}$.

Definition 2.1 (PSO Probability Space from [15])**.** The probability space $\mathcal{R} = (\Omega, \mathcal{A}, P)$ is defined via

- $\Omega := [0,1]^\infty$
- $\mathcal{A} := \bigotimes_{\mathbb{N}} \mathcal{B}([0,1])$
- $P := L^\infty$

where $[0,1]^\infty = \{(\omega_0, \omega_1, \ldots) | \omega_i \in [0,1]\}$ for every $i \in \mathbb{N}$ is the space of all sequences with values in $[0,1]$, $\bigotimes_{\mathbb{N}} \mathcal{B}([0,1])$ is the product σ-field of countably many instances of $\mathcal{B}([0,1])$ and L^∞ is obtained from the corollary of the Ionesca-Tulcea Theorem by setting $(\Omega_i, \mathcal{A}_i, P_i) = ([0,1], \mathcal{B}([0,1]), \mathcal{L}^1)$.

Definition 2.2 (GbPSO Model)**.** Let $f : \mathbb{R}^D \to \mathbb{R}$ be the objective function. The swarm S of size N of the Gamma-based PSO that moves through the D-dimensional search space $\mathbb{R}^D$ have the stochastic process defined as

$$S = (S_t)_{t \in \mathbb{N}_0} = ((X_t, V_t, L_t, G_t))_{t \in \mathbb{N}_0} = ((X_0, V_0, L_0, G_0).(X_1, V_1, L_1, G_1), \ldots),$$

The stochastic process $(S_t)_{t \in \mathbb{N}_0}$ is determined by the following *movement update equations*:

- Velocity update equation: $V_{t+1}^{n,d} = w \cdot V_t^{n,d} + c_1 \cdot \gamma_{t+1}^{n,d} \cdot (L_t^{n,d} - X_t^{n,d}) + c_2 \cdot \beta_{t+1}^{n,d} \cdot (G_t^{n,d} - X_t^{n,d})$ for $t \geq 0$;
- Position update equation: $X_{t+1}^{n,d} = X_t^{n,d} + V_{t+1}^{n,d}$ for $t \geq 0$;
- Local best update equation: $L_{t+1}^n = \text{argmin}_{(X_{t+1}^n, L_t^n)} f$;
- Global best update equation: $G_t^{n+1} = \text{argmin}_{(L_{t+1}^n, G_t^n)} f$ for $t \geq 0, 1 \leq n \leq N-1$; and
- $G_{t+1}^1 = \text{argmin}_{(L_{t+1}^N, G_t^N)} f$ for $t \geq 0$.

In case of a tie when applying the update equation for the L^n and G^n, the new value prevails. Additionally, if the obtained value is equal to G^n, then G^n is also updated. The parameters w, c_1, and c_2 are the same constants along with the independently drawn random numbers $\gamma_t^{n,d}$, $\beta_t^{n,d} \sim Gamma(\alpha, 1)$.

The PSO algorithm relies on its movement update equations for the swarm to move around the search space. Thus, the focus for the convergence analysis of the GbPSO will be the behaviour of the velocity update equation.

Following the techniques in [6] and [19], velocity update equation needs to be first transformed into a non-homogeneous recurrence relation:

$$\begin{aligned} X_{t+1}^{n,d} = &\left[1 + w - (c_1 \cdot \gamma_{t+1}^{n,d} + c_2 \cdot \beta_{t+1}^{n,d})\right] \cdot X_t^{n,d} - w \cdot X_{t-1}^{n,d} \\ &+ c_1 \cdot \gamma_{t+1}^{n,d} \cdot L_t^{n,d} + c_2 \cdot \beta_{t+1}^{n,d} \cdot G_t^{n,d}. \end{aligned} \tag{5}$$

Since the swarm is composed of n particles that are employed in a D-dimension continuous search space, it is far better to narrow down the analysis by focusing only on a single particle of the swarm.

2.1 1-Particle 1-Dimension Swarm Model

Let f be a function that is to be minimized using GbPSO algorithm with the swarm S with size N in a D-dimensional continuous search space. Assume that at a discrete number of iteration, L_t^n and G_t^n are constant such that $L_t^n =: L^n$ and $G_t^n =: G$.

The stochastic process of the swarm during this discrete time T is $((X_t, V_t))_{t \in T}$. All particles in S now evolve independently. Also, each dimension is updated independently from others. Thus, $((X_t^{n_1,d_1}, V_t^{n_1,d_1}))_{t \in T}$ and $((X_t^{n_2,d_2}, V_t^{n_2,d_2}))_{t \in T}$ are independent of each other if $n_1 \neq n_2$ or $d_1 \neq d_2$. It is then sufficient to study just one of these processes, or equivalently, study a 1-particle, 1-dimension model. The movement update equations, and the corresponding non-homogeneous recurrence relation in equation (5) can be written as

$$V_{t+1} = w \cdot V_t + c_1 \cdot \gamma_{t+1} \cdot (L - X_t) + c_2 \cdot \beta_{t+1} \cdot (G - X_t), \tag{6}$$

$$X_{t+1} = X_t + V_{t+1}, \tag{7}$$

$$X_{t+1} = [1 + w - (c_1 \cdot \gamma_{t+1} + c_2 \cdot \beta_{t+1})] \cdot X_t - w \cdot X_{t-1} + c_1 \cdot \gamma_{t+1} \cdot L + c_2 \cdot \beta_{t+1} \cdot G. \tag{8}$$

The 1-particle swarm model produces the trajectory of the single particle using the movement update equations and that the sequence of points in the search space that plots the trajectory of the single particle are random variables related to the stochastic process $((X_t, V_t))_{t\in T}$. The random variables generated by equation (8) can be regarded as the sequence of random variables, $\{X_t\}$. This sequence is considered as an iterative method. Thus, convergence of $\{X_t\}$ is the convergence of the stochastic process $((X_t, V_t))_{t\in T}$, and by extension, convergence of the 1-particle 1-dimension swarm model.

3 Convergence Analysis of the GbPSO

Convergence analysis will be first applied to the 1-particle 1-dimension swarm model with L and G as constants to find the interval of the parameters in the tuple $\{w, c_1, c_2, \alpha, \lambda = 1\}$ where the behaviour of the swarm converges. Next, assumption of L being constant is removed and the convergence property of the particle is analysed. The N-particle D-dimensional swarm will be recalled and the results obtained in the previous analysis will be applied to it. The techniques used in proving the convergence property of the swarm are derived from the approach in [6].

Definition 3.1 (Swarm Convergence)**.** Swarm S converges if there is almost surely a point z such that the following two conditions hold:

1. $\lim_{t\to\infty} \mathrm{Var}(X_t^n) = 0$ for each $n \in \{1, \ldots, N\}$, the variance of the particles tend to zero;

2. $\lim_{t\to\infty} \mathrm{E}(X_t^n) = z$ for each $n \in \{1, \ldots, N\}$, the expectation of every particle moves towards z.

This is akin to the convergence in mean square of random variables.

From Definition 3.1, the iteration equation (8) needs to satisfy two conditions: (1) expectation of the iteration equation moves toward a particular point in the search space; and (2) variance of the same equation tends towards zero.

Using the definition of expectation, equation (8), and $\lambda = 1$ for the standard gamma distribution, the iteration equation of the expectation of the particle's position is

$$\mathrm{E}[X_{t+1}] = [1 + w - (\alpha(c_1 + c_2))] \cdot \mathrm{E}[X_t] - w \cdot \mathrm{E}[X_{t-1}] + \alpha\,(c_1 \cdot L + c_2 \cdot G)\,, \tag{9}$$

Theorem 3.2. *Given parameters $w \geq 0$ and $c_1, c_2, \alpha > 0$, the iterative method $\{\mathrm{E}[X_t]_{t\in T}\}$ is guaranteed to converge to $\frac{c_1 \cdot L + c_2 \cdot G}{c_1 + c_2}$ if and only if $0 \leq w < 1$ and $0 < c_1 + c_2 < \frac{2(w+1)}{\alpha}$.*

Proof. In the case of the standard PSO where $\gamma_{t+1}, \beta_{t+1} \sim U(0, 1)$, [6] obtained the following recurrence relation for the expected values:

$$\mathrm{E}[X_{t+1}] = \left(1 + w - \frac{c_1 + c_2}{2}\right) \mathrm{E}[X_t] - w\mathrm{E}[X_{t-1}] + \frac{1}{2}(c_1 L + c_2 G). \tag{10}$$

If the random variables $\gamma_{t+1}, \beta_{t+1}$ are both chosen in the same distribution with mean $\hat{\mu}$, then the recurrence relation is

$$\mathrm{E}[X_{t+1}] = (1 + w - \mu(c_1 + c_2))\,\mathrm{E}[X_t] - w\mathrm{E}[X_{t-1}] + \mu(c_1 L + c_2 G). \tag{11}$$

Replacing the parameters (c_1, c_2) in Equation (10) by $(2\hat{\mu}c_1, 2\hat{\mu}c_2)$ leads to Equation (11). With this transformation, the results for Equation (10) stated in [6] can be applied immediately to Equation (11). The sequence of expected values $\{\mathrm{E}[X_t]\}$ will converge to the same limit since

$$\frac{2\hat{\mu}c_1 L + 2\hat{\mu}c_2 G}{2\hat{\mu}c_1 + 2\hat{\mu}c_2} = \frac{c_1 L + c_2 G}{c_1 + c_2},$$

provided that the following transformed stability conditions hold:

$$0 \le w \le 1 \text{ and } \left(0 < 2\hat{\mu}c_1 + 2\hat{\mu}c_2 < 4(w+1) \iff 0 < c_1 + c_2 < \frac{2(w+1)}{\hat{\mu}}\right). \tag{12}$$

In the case where the random values are drawn from the gamma distribution, $\gamma_{t+1}, \beta_{t+1} \sim \Gamma(\alpha, \lambda)$, then we have $\hat{\mu} = \frac{\alpha}{\lambda}$ and the second inequality in the stability condition (12) becomes

$$0 < c_1 + c_2 < \frac{2\lambda(w+1)}{\alpha}, \tag{13}$$

which proves the theorem. It can also be noted that this condition is invariant, or that it remains unchanged, under scaling of parameters in the gamma distribution. That is, the same stability conditions hold for $\Gamma(\alpha, \lambda)$ and $\Gamma(c\alpha, c\lambda)$ for any constant $c > 0$. □

Theorem 3.3. *Given the parameters $w \ge 0$ and $c_1, c_2, \alpha > 0$, iterative method $\{\mathrm{Var}[X_t]\}_{t\in T}$ is guaranteed to converge to*

$$VX = \frac{2\alpha\left(\frac{c_1 c_2}{c_1 + c_2}\right)^2 (G - L)^2 (1 + w)}{h(1)}, \tag{14}$$

where

$$\begin{aligned} h(1) = &\left[-2\alpha(c_1 + c_2)\right] w^2 + \left[\alpha^2(c_1 + c_2)^2 - \alpha(c_1^2 + c_2^2)\right] w \\ &+ \alpha\left[2(c_1 + c_2) - \alpha(c_1 + c_2)^2 - (c_1^2 + c_2^2)\right] > 0. \end{aligned}$$

if and only if $0 \le w < 1$, $c_1 + c_2 > 0$, and $h(1) > 0$ are all satisfied together.

Proof. It was previously shown that the sequence of expected values $\{\mathrm{E}[X_t]\}$ will converge to the same limit $\frac{c_1 L + c_2 G}{c_1 + c_2}$ provided that the applicable stability conditions hold. Consider a random variable $Z_t = X_t - \mu$, where $\mu = \frac{c_1 L + c_2 G}{c_1 + c_2}$. The iteration sequence for Z_t can be written as

$$Z_{t+1} = [1 + w - (c_1 \cdot \gamma_{t+1} + c_2 \cdot \beta_{t+1})] \cdot Z_t - w \cdot Z_{t-1} + \frac{c_1 c_2}{c_1 + c_2}(G - L)(\beta_{t+1} - \gamma_{t+1}). \tag{15}$$

In the case where $\gamma_{t+1}, \beta_{t+1} \sim U(0,1)$, [6] obtained

$$Z_{t+1} = (\psi - R_t) \cdot Z_t - w \cdot Z_{t-1} + Q_t, \tag{16}$$

where $\nu = \frac{c_1+c_2}{2}$, $\psi = 1 + w - \nu$, $R_t = c_1\gamma_{t+1} + c_2\beta_{t+1} - \nu$, and $Q_t = \frac{c_1c_2}{c_1+c_2}(\beta_{t+1} - \gamma_{t+1})(G - L)$. Also, $\mathrm{E}[R_t] = \mathrm{E}[Q_t] = 0$; $\mathrm{Var}[R_t] = \mathrm{E}[R_t^2] = \frac{1}{12}(c_1^2 + c_2^2) = R$; $\mathrm{Var}[Q_t] = \mathrm{E}[Q_t^2] = \frac{1}{6}\left(\frac{c_1c_2}{c_1+c_2}\right)^2 (G - L)^2 = Q$; and $\mathrm{E}[R_tQ_t] = \frac{c_1c_2(c_2-c_1)}{12(c_1+c_2)}(G - L) = T$. By algebraic manipulations, [6] derived the iteration equation of $\{\mathrm{Var}[X_t]_{t\in T}\}$

$$\begin{aligned}\mathrm{Var}[X_{t+2}] =&(\psi^2 + R - w)\mathrm{Var}[X_{t+1}] - w(\psi^2 - R - w)\mathrm{Var}[X_t] + w^3\mathrm{Var}[X_{t-1}] \\ &+ R\left[(\mathrm{E}[X_{t+1}] - \mu)^2 + (\mathrm{E}[X_t] - \mu)^2\right] + Q(1 + w) - 2T\left[(\mathrm{E}[X_{t+1}] - \mu)\right. \\ &\left.+w(\mathrm{E}[X_t] - \mu)\right],\end{aligned} \tag{17}$$

and characteristic equation

$$I^3 - (\psi^2 + R - w)I^2 + w(\psi^2 - R - w)I - w^3 = 0. \tag{18}$$

This characteristic equation is subjected to an eigenvalue analysis in [6] where $h(I) = I^3 - (\psi^2 + R - w)I^2 + w(\psi^2 - R - w)I - w^3$, such that $h(1) > 0$ is proven to be the necessary and sufficient condition for the convergent condition whenever $0 \le w < 1$, and $c_1 + c_2 > 0$ are satisfied.

Next, [6] computed the value of $h(1)$ to be

$$\begin{aligned}h(1) &= -2\left(\frac{c_1 + c_2}{2}\right)w^2 + \left[\left(\frac{c_1 + c_2}{2}\right)^2 - \frac{1}{12}(c_1^2 + c_2^2)\right] + 2\left(\frac{c_1 + c_2}{2}\right) - \left(\frac{c_1 + c_2}{2}\right)^2 \\ &\quad - \frac{1}{12}(c_1^2 + c_2^2) \\ &= -(c_1 + c_2)w^2 + \left(\frac{1}{6}c_1^2 + \frac{1}{6}c_2^2 + \frac{1}{2}c_1c_2\right)w + c_1 + c_2 - \frac{1}{3}c_1^2 - \frac{1}{3}c_2^2 - \frac{1}{2}c_1c_2 > 0,\end{aligned} \tag{19}$$

and that the sequence of variance $\{\mathrm{Var}[X_t]\}$ will converge to

$$VX = \frac{\frac{1}{6}\left(\frac{c_1c_2}{c_1+c_2}\right)^2 (G - L)^2(1 + w)}{h(1)}, \tag{20}$$

where $h(1)$ is in Equation (19), and that the following stability conditions hold:

$$0 \le w < 1,\ c_1 + c_2 > 0,\ \text{and } h(1) > 0. \tag{21}$$

If the random variable $\gamma_{t+1}, \beta_{t+1}$ are chosen in the same distribution with mean $\hat{\mu}$, and variance $\hat{\sigma}$, then following the process outlined earlier, the iteration sequence for the

random variable Z_t would be the same as in Equation (16), with $\nu = \hat{\mu}(c_1+c_2)$, $\psi = 1+w-\nu$, $R_t = c_1\gamma_{t+1} + c_2\beta_{t+1} - \nu$, and $Q_t = \frac{c_1c_2}{c_1+c_2}(\beta_{t+1} - \gamma_{t+1})(G-L)$. Also, $\mathrm{E}[R_t] = \mathrm{E}[Q_t] = 0$; $\mathrm{Var}[R_t] = \mathrm{E}[R_t^2] = \hat{\sigma}(c_1^2 + c_2^2) = R$; $\mathrm{Var}[Q_t] = \mathrm{E}[Q_t^2] = 2\hat{\sigma}\left(\frac{c_1c_2}{c_1+c_2}\right)^2 (G-L)^2 = Q$; and $\mathrm{E}[R_tQ_t] = \frac{\hat{\sigma}c_1c_2(c_2-c_1)}{c_1+c_2}(G-L) = T$.

The iteration equation for $\{\mathrm{Var}[X_t]\}$ and the characteristic equation are the same from Equations (17), and (18), respectively. Since the characteristic equation is the same, then the same results follow: $h(1) > 0$ is the necessary and sufficient condition for the convergent condition whenever $0 \leq w < 1$, and $c_1 + c_2 > 0$ are satisfied.

Next, the value of $h(1)$ would be

$$\begin{aligned} h(1) = (-2\hat{\sigma}(c_1+c_2))\, w^2 + \left[(\hat{\mu}(c_1+c_2))^2 + \hat{\sigma}(c_1^2+c_2^2)\right] \\ + 2\hat{\mu}(c_1+c_2) - (\hat{\mu}(c_1+c_2))^2 - \hat{\sigma}(c_1^2+c_2^2), \end{aligned} \tag{22}$$

and that the sequence of variance $\{\mathrm{Var}[X_t]\}$ will converge to

$$VX = \frac{2\hat{\sigma}\left(\frac{c_1c_2}{c_1+c_2}\right)^2 (G-L)^2(1+w)}{h(1)}, \tag{23}$$

where $h(1)$ is in Equation (22), and provided that the same stability conditions stated in (21) hold.

Note that replacing the terms $\nu = \frac{c_1+c_2}{2}$, $R = \frac{1}{12}(c_1^2+c_2^2)$, and $Q = \frac{1}{6}\left(\frac{c_1c_2}{c_1+c_2}\right)^2 (G-L)^2$ from the results obtained in [6] by the terms $\nu = \hat{\mu}(c_1+c_2)$, $R = \hat{\sigma}(c_1^2+c_2^2)$, and $Q = 2\hat{\sigma}\left(\frac{c_1c_2}{c_1+c_2}\right)^2 (G-L)^2$, Equation (19) leads to Equation (22), and the sequence of variance $\{\mathrm{Var}[X_t]\}$ will converge to the same limit.

In the particular case where $\gamma_{t+1}, \beta_{t+1} \sim \Gamma(\alpha, 1)$, we have $\hat{\mu} = \alpha$, $\hat{\sigma} = \alpha$, and

$$VX = \frac{2\alpha\left(\frac{c_1c_2}{c_1+c_2}\right)^2 (G-L)^2(1+w)}{h(1)},$$

where

$$\begin{aligned} h(1) = &\left[-2\alpha(c_1+c_2)\right] w^2 + \left[\alpha^2(c_1+c_2)^2 - \alpha(c_1^2+c_2^2)\right] w \\ &+ \alpha\left[2(c_1+c_2) - \alpha(c_1+c_2)^2 - (c_1^2+c_2^2)\right]. \end{aligned}$$

□

The earlier theorems only considered the 1-particle 1-dimension swarm simplification, with the restriction of fixed local and global attractors. In the succeeding theorems, the restriction of fixed local attractor is removed while the global attractor will still remain constant. The relationship between the local and global attractor is defined in the following theorem.

Theorem 3.4. *Given $w \geq 0$, $c_1, c_2, \alpha > 0$, if the iterative method $\{\mathrm{Var}[X_t]\}_{t\in T}$ is guaranteed to converge and $h(1) < 2\alpha c_2^2(w+1)$, then the iterative method $\{L_t\}$ will converge almost surely to G.*

Proof. As defined by the probability space in Definition 2.1, L_t is a random variable, and that L_{t+1} depends on L_t where L_{t+1} is always an improvement of its predecessor. Thus, $\{L_t\}$ is an iterative method.

Suppose that $\{\mathrm{Var}[X_t]\}_{t\in T}$ is guaranteed to converge. This implies that $\{\mathrm{E}[X_t]\}_{t\in T}$ also converges. Thus, the random particle X_t will converge with expectation and variance presented in equations (13) and (14), respectively. No matter the value of L_t, if $h(1) < 2\alpha c_2^2(w+1)$, then

$$\mathrm{Var}[G] = \mathrm{E}[(G - EX)^2] = \left(\frac{c_1}{c_1 + c_2}\right)^2 (G - L)^2 < VX.$$

Hence, $P(X_t = G) > 0$, which leads directly to $P(\lim_{t\to\infty} L_t = G) = 1$. It is then evident that $\{L_t\}$ converges almost surely to G. It should be noted that the condition $h(1) < 2\alpha c_2^2(w+1)$ is only a sufficient condition to ensure convergence. □

The original N-particle D-dimension Gamma-based PSO system is recalled. The value of $L_t^{n,d}$ and $G_t^{n,d}$ are being constantly updated as the system converges toward an optimum. From the analysis, $L_t^{n,d}$ will evolve towards $G_t^{n,d}$ when the latter is fixed under certain conditions. In the event that $G^{n,d}$ changes, then $L_t^{n,d}$ will evolve to the new global attractor. For the convergence property of the GbPSO, the following theorem can be obtained.

Theorem 3.5. *Given $w \geq 0$, $c_1, c_2, \alpha > 0$, if $0 \leq w < 1$, $c_1 + c_2 > 0$, and $0 < h(1) < 2\alpha c_2^2(w+1)$ are all satisfied together, the Gamma-based PSO system determined by the parameter tuple $\{w, c_1, c_2, \alpha, \lambda = 1\}$ will converge.*

Proof. From the results of Theorems (3.2) and (3.3), given the parameters $w \geq 0$, $c_1, c_2, \alpha > 0$, if the local attractor and the global attractor are kept constant during a discrete time T, then if $0 \leq w < 1$, $c_1 + c_2 > 0$, and $h(1) > 0$ are all satisfied together, then for particle n in every dimension d

$$\{\mathrm{E}[X_t^{n,d}]\}_{t\in T} \to \frac{c_1 L^{n,d} + c_2 G^d}{c_1 + c_2} \quad \text{and} \quad \{\mathrm{Var}[X_t^{n,d}]\}_{t\in T} \to \frac{2\alpha\left(\frac{c_1 c_2}{c_1 + c_2}\right)^2 (G^d - L^{n,d})^2(1+w)}{h(1)}.$$

From the velocity and position update equations, each dimension of particle n is independently updated from others.

$$\{\mathrm{E}[X_t^n]\}_{t\in T} \;\to\; \frac{c_1L^n + c_2G}{c_1+c_2} \quad \text{and} \quad \{\mathrm{Var}[X_t^n]\}_{t\in T} \;\to\; \frac{2\alpha\left(\frac{c_1c_2}{c_1+c_2}\right)^2(G-L^n)^2(1+w)}{h(1)}.$$

From the results of Theorem (3.4), if $h(1) < 2\alpha c_2^2(w+1)$, then each local attractor converges almost surely to the global attractor. Thus, in the event that $L = G$:

$$\{\mathrm{E}[X_t^n]\}_{t\in T} \;\to\; G \quad \text{and} \quad \{\mathrm{Var}[X_t^n]\}_{t\in T} \;\to\; 0.$$

From Definition (3.1), each sequence $\{X_t\}$ will stochastically evolve toward G until it converges in mean square to G. Since this applies to every particle in the swarm, it can be concluded that the Gamma-based PSO swarm determined by the parameter tuple $\{w, c_1, c_2, \alpha, \lambda = 1\}$ will converge to the global attractor G. □

The analysis shown for the convergence of the GbPSO algorithm only affirm that each particle in the swarm would converge in mean square to the best position found by the swarm so far (G). However, this does not mean that this convergent position is the optimal one, or even a local optimal one.

4 Experimental Results

This study used the extensible research toolkit for the PSO called PySwarms developed using the Python programming language [9] to devise the GbPSO algorithm. It is then used to optimize some commonly used benchmark functions presented in Table 2.

Function Name	Test Function	Search Space	Particle Bound	Best Solution	Best Result
Sphere	$f_1(x) = \sum_{i=1}^{d} x_i^2$	$[-100, 100]$	$[-100, 100]$	$[0, \cdots, 0]$	0
Schaffer N.2	$f_2(x) = 0.5 + \frac{\sin^2(x_1^2 - x_2^2) - 0.5}{\left[1+0.001(x_1^2+x_2^2)\right]^2}$	$[-100, 100]$	$[-100, 100]$	$[0, 0]$	0
Three-hump Camel	$f_3(x) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} + x_1x_2 + x_2^2$	$[-5, 5]$	$[-5, 5]$	$[0, 0]$	0
Rastrigin	$f_4(x) = 10d + \sum_{i=1}^{d}\left[x_i^2 - 10\cos(2\pi x_i)\right]$	$[-5.12, 5.12]$	$[-5.12, 5.12]$	$[0, \cdots, 0]$	0
Rosenbrock	$f_5(x) = \sum_{i=1}^{d-1}\left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right]$	$[-30, 30]$	$[-30, 30]$	$[1, \cdots, 1]$	0

Table 2: Benchmark Functions

These five well-known benchmark functions are a combination of unimodal (sphere, and Schaffer N.2), and multimodal (Rastrigin, Rosenbrock, and Three-hump Camel) functions. It is possible to test the exploitation capability of the proposed algorithm using unimodal functions. Meanwhile, the multimodal functions used are difficult to optimize because they have several local minima, which can trap the particles in the swarm, and produce premature convergence.

In all parts of the experiment, the following parameters are set to be the same for each benchmark function until otherwise stated: (1) number of particles is $N = 100$; (2) dimension of the search space is $D = 2$; and (3) maximum number of function evaluations

is 10,000. The results of all experiments are averaged over 30 independent runs to eliminate random discrepancy, while the initial position of each particle are randomly drawn from $U[-B, B]^2$ using the same seed, where B is the search space bound stated in each benchmark function, and the initial velocity of each particle is set to zero. The researchers consider particles to have *converged* to a particular region whenever the worst particle is within a tolerance value of 1×10^{-5} of the best particle in terms of cost value.

4.1 Effects of Different w and α values on the GbPSO

This experiment numerically validates the results in the convergence analysis. The parameter values of the algorithm were computed according to the equation below, with $c_1 = c_2$

$$c = \left[\text{round off}\left(\frac{1 - w^2}{\frac{1}{2}w + \alpha + \frac{1}{2} - \alpha w}\right)\right] - 0.05. \tag{24}$$

The aim of this part of the experiment is to identify the effect of different values of w and α to the overall performance of the GbPSO in a setting beneficial to it. Table (3) shows the different combinations used in this experiment.

Shape	Inertia weight (w)									
Parameter (α)	**0.90**	**0.80**	**0.70**	**0.60**	**0.50**	**0.40**	**0.30**	**0.20**	**0.10**	**0.00**
1.00	0.1310	0.2773	0.3935	0.6000	0.5500	0.5962	0.6241	0.6357	0.6328	0.6167
1.50	0.1227	0.2500	0.3423	0.4071	0.4500	0.4750	0.4853	0.4833	0.4711	0.4500
2.00	0.1152	0.2269	0.3017	0.3500	0.3786	0.3921	0.3939	0.3864	0.3713	0.3500
2.50	0.1083	0.2071	0.2688	0.3056	0.3250	0.3318	0.3292	0.3192	0.3036	0.2833
3.00	0.1020	0.1900	0.2414	0.2700	0.2833	0.2860	0.2809	0.2700	0.2546	0.2357
3.50	0.0962	0.1750	0.2184	0.2409	0.2500	0.2500	0.2435	0.2324	0.2176	0.2000
4.00	0.0907	0.1618	0.1988	0.2167	0.2227	0.2210	0.2138	0.2026	0.1886	0.1722
4.50	0.0857	0.1500	0.1818	0.1962	0.2000	0.1971	0.1895	0.1786	0.1652	0.1500
5.00	0.0810	0.1395	0.1670	0.1786	0.1808	0.1770	0.1693	0.1587	0.1460	0.1318

Table 3: Different parameter combinations for the GbPSO

The termination condition for this part will be: (i) If at least one particle managed to find the global optima, and when the worst performing particle is within the aforementioned tolerance value; or (ii) when the maximum number of fitness evaluations set in advance is reached. These will allow the swarm to show its ability to escape premature convergence and find the global minimum.

4.1.1 Simulation Results

The first criteria tested presents the exploration-exploitation capability of the algorithm and its ability to escape local minima. An optimization run is successful whenever the algorithm obtained the optimal solution and converged to it, otherwise, it is a failed run. In the event of a failed run, performance is measured based on the proximity of the obtained value to the optimal value.

For the Schaffer N.2 and Rastrigin functions, 100% of the parameter combinations had successful runs in both functions in all 30 independent runs. For the Sphere function, 96.67% of all parameter combinations succeeded in all their runs. For the remaining 3.33%,

1.11% failed in all its runs while 2.22% failed in some of their optimization runs. For the Three-hump Camel function, similar observations to the Sphere function can be noted in the results: 96.67% of all parameter combinations succeeded in all their runs. For the remaining 3.33%, 1.11% failed in all its runs while 2.22% failed in some of their optimization runs. For the Rosenbrock function, 87.78% of all parameter combinations succeeded in all their runs. The remaining 12.22% failed in some of their runs.

The results show that the algorithm has the ability to find the global minima and converge to it, along with its ability to escape local minima which can be observed in its performance in the Rastrigin function - a function that has several local minima that are regularly distributed from each other.

The second criteria shows how fast the swarm converges to the global minimum. The convergence velocity of every run of each parameter combination were averaged. Parameter combinations which failed in some or all of their runs were excluded when identifying the fastest converging combination.

The observations regarding the convergence velocity of the algorithm can be loosely generalized as follows: (1) For parameter combinations grouped according to their α, as w decreases, the number of iterations needed to find the global optima also decreases as seen in Figure 1; and (2) For parameter combinations grouped according to their w, as α increases, the number of iterations needed to find the global optima decreases as seen in Figure 2. Also, it should be mentioned that there are exceptions to these observations on some of the benchmark functions.

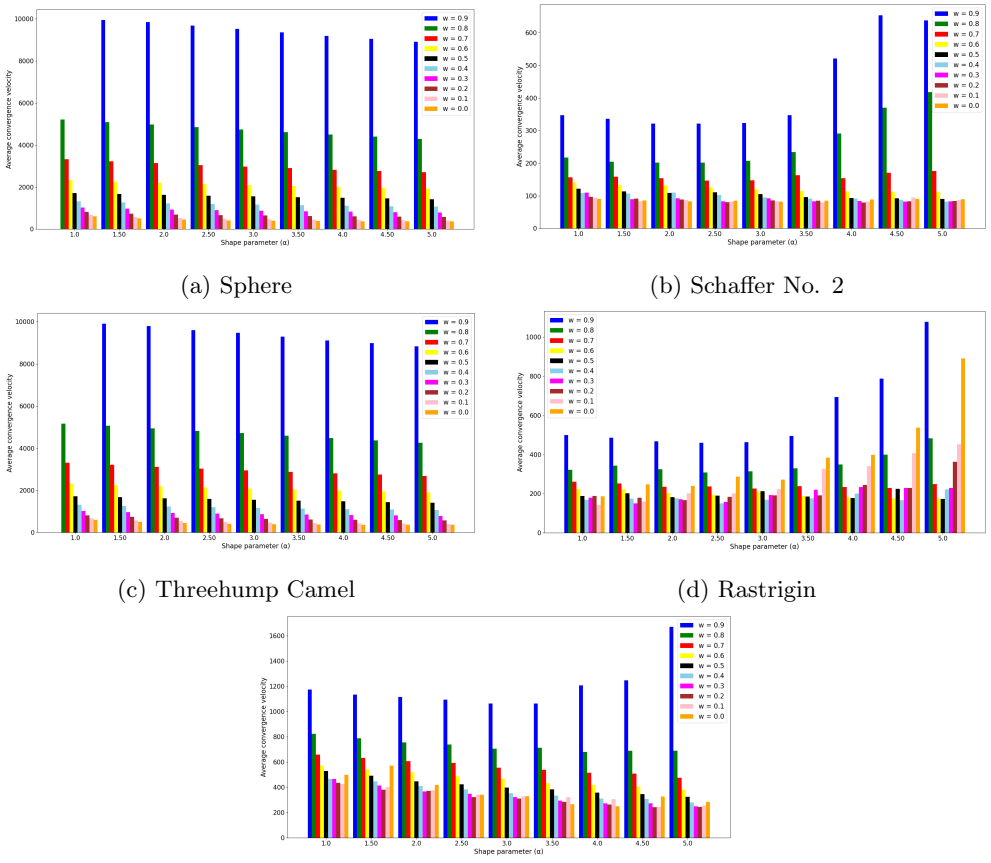
4.2 Comparison between GbPSO and WCPSO

The same experiment was also done using the standard PSO with parameter combinations taken in the literature, namely, **WC1**: $c_1 = c_2 = 1.49617, w = 0.72984$ [2]; **WC2**: $c_1 = 2.04355, c_2 = 0.9487, w = 0.72984$ [1]; **WC3**: $c_1 = c_2 = 1.7, w = 0.6$ [18], to determine the best performing combination and compare it to the best performing combination of the GbPSO.

The best performing parameter combination of the GbPSO for each benchmark function based on the two criteria is as follows: GbPSO$(4.50, 0.0)$ for Sphere, GbPSO$(4.0, 0.2)$ for Schaffer N.2, GbPSO$(4.0, 0.0)$ for Three-hump Camel, GbPSO$(1.0, 0.1)$ for Rastrigin, and GbPSO$(4.50, 0.2)$ for Rosenbrock function. Meanwhile, WC3 is the best performing parameter combination for the standard PSO in all benchmark functions used.

Comparison was done using the same criteria in Part 1. However, the number of independent runs will be 30/50/100, and swarm sizes will be 100/200/1000. The termination criteria for this phase is also different: (i) If the worst performing particle is within the aforementioned tolerance value from the best performing particle of the swarm; and (ii) when the maximum number of fitness evaluations set in advance is reached.

The termination conditions were modified for this part in order to simulate real-world problem optimization and to show how likely the GbPSO and the standard PSO may prematurely converge to a point in the search space with respect to the tolerance value. For this part, an optimization run is successful whenever the algorithm managed to find the



(a) Sphere

(b) Schaffer No. 2

(c) Threehump Camel

(d) Rastrigin

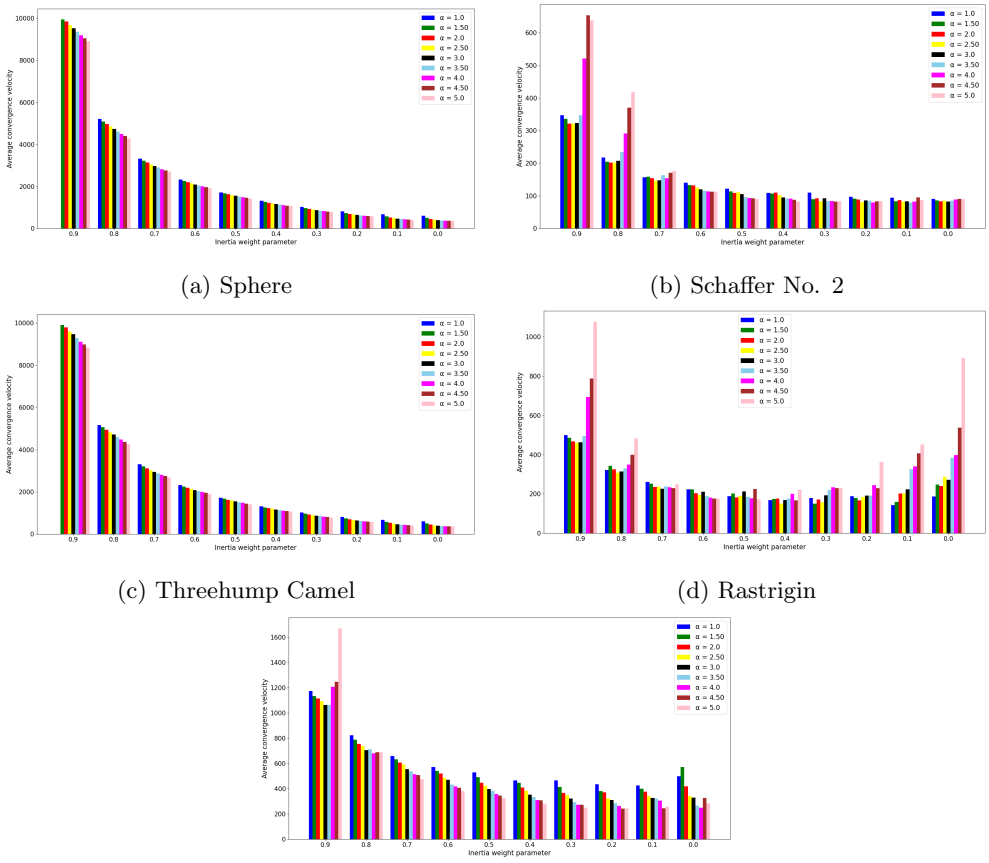
(e) Rosenbrock

Figure 1: Average convergence velocity of all parameters with respect to their α in all benchmark functions

optimal value and converge to it. Failure means that the algorithm either did not find the optimal value and converged prematurely on any point in the search space, or the maximum number of iterations were exhausted without the swarm converging.

4.2.1 Simulation Results

Success Rates. As indicated in Table 4, both algorithms failed to converged to the optimal value in the Sphere and Three-hump Camel functions in all the different combinations of runs and swarm sizes. This result is completely different for the Schaffer N.2 and Rastrigin functions where both algorithms have 100% success rates in all the different runs and swarm sizes. For the Rosenbrock function, both algorithms have a 100% success rate in converging in all the different runs when the swarm size is 1000 particles. However, this decreases to approximately half in the runs with a swarm size of 200 particles, and further decreases to about $5\% - 10\%$ in the runs with a swarm size of 100 particles. This "decrease in successful runs as the population size decreases" behaviour observed in both algorithms is only notice-



(a) Sphere

(b) Schaffer No. 2

(c) Threehump Camel

(d) Rastrigin

(e) Rosenbrock

Figure 2: Average convergence velocity of all parameters with respect to their w in all benchmark functions

able in the Rosenbrock function.

Comparing the success rates of both algorithms in the Rosenbrock function, standard PSO is more likely to converge to the optimal value than the GbPSO in all the other different combinations of runs and swarm sizes, except in the combination of 30 runs/200 particles. However, their difference can be considered minimal: 19 successes out of 30 runs for the GbPSO compared to 18 successes out of 30 runs for the WCPSO.

Average Cost Value. In terms of average cost value for functions where both algorithms failed to converge to the optimal value, both showed the behaviour where the average cost value improves (decreases) as the swarm size increases. Also, the standard PSO outperformed the GbPSO in terms of average cost value in all combinations of runs and swarm sizes with some exceptions: 50 runs/1000 particles, and 100 runs/1000 particles. However, the difference between the average cost value obtained by both algorithms is relatively small. The summary of the average cost values obtained are presented in Table 5.

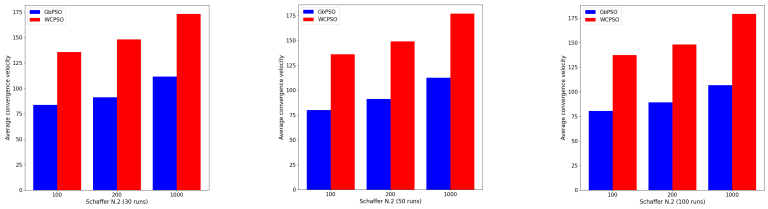
Function	Runs	30			50			100		
	Size	100	200	1000	100	200	1000	100	200	1000
Sphere	GbPSO	0%	0%	0%	0%	0%	0%	0%	0%	0%
	WC3	0%	0%	0%	0%	0%	0%	0%	0%	0%
Schaffer N2	GbPSO	100%	100%	100%	100%	100%	100%	100%	100%	100%
	WC3	100%	100%	100%	100%	100%	100%	100%	100%	100%
Three-hump	GbPSO	0%	0%	0%	0%	0%	0%	0%	0%	0%
	WC3	0%	0%	0%	0%	0%	0%	0%	0%	0%
Rastrigin	GbPSO	100%	100%	100%	100%	100%	100%	100%	100%	100%
	WC3	100%	100%	100%	100%	100%	100%	100%	100%	100%
Rosenbrock	GbPSO	0%	63.33%	100%	2%	42%	100%	8%	50%	100%
	WC3	6.67%	60%	100%	10%	60%	100%	13%	63%	100%

Table 4: Comparison between GbPSO and standard PSO: Success Rates

Function	Runs	30			50			100		
	Size	100	200	1000	100	200	1000	100	200	1000
Sphere	GbPSO	9.79E-12	8.13E-14	8.48E-18	1.58E-11	2.79E-14	1.25E-17	1.11E-11	8.20E-14	6.50E-18
	WCPSO	1.76E-15	1.48E-18	1.29E-22	3.46E-16	4.03E-18	3.95E-22	2.03E-15	6.78E-18	4.90E-22
Schaffer N2	GbPSO	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	WCPSO	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Three-hump	GbPSO	2.40E-12	9.77E-15	8.42E-24	4.04E-12	2.90E-15	6.06E-23	7.21E-13	1.79E-14	9.98E-23
	WCPSO	2.43E-16	1.48E-17	2.32E-21	1.73E-15	1.34E-16	1.45E-21	5.70E-16	3.75E-17	1.38E-21
Rastrigin	GbPSO	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	WCPSO	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Rosenbrock	GbPSO	5.49E-19	6.76E-26	0.0	1.48E-18	5.21E-28	0.0	8.26E-17	8.73E-26	0.0
	WCPSO	1.04E-23	1.02E-26	0.0	2.46E-22	5.60E-29	0.0	6.10E-23	2.13E-28	0.0

Table 5: Average cost values for functions where GbPSO and standard PSO failed to converge to optimal value

Average Convergence Velocity. In terms of average convergence velocity for their successful runs in the Schaffer N.2, and Rastrigin functions, both algorithms showed that as the swarm size increases, the average convergence velocity also increases as seen in Figures 3, and 4 for the Schaffer N.2, and Rastrigin functions, respectively. Since more particles are present in the swarm, more iterations are needed for all of them to converge to a particular point in the search space. This observation is true in all the varying runs used in both algorithms. Another observation that is true in both Schaffer N.2 and Rastrigin functions is that the GbPSO is faster to converge to the optimal value than the standard PSO.

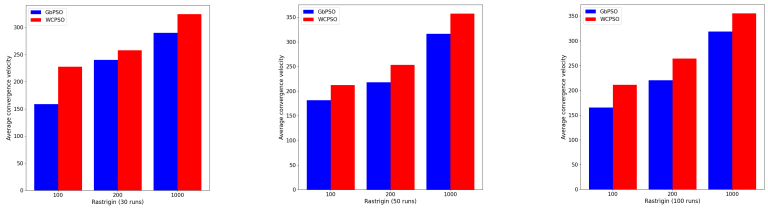


(a) Average convergence velocity in 30 runs

(b) Average convergence velocity in 50 runs

(c) Average convergence velocity in 100 runs

Figure 3: Average convergence velocity in all runs and population size - Schaffer N.2



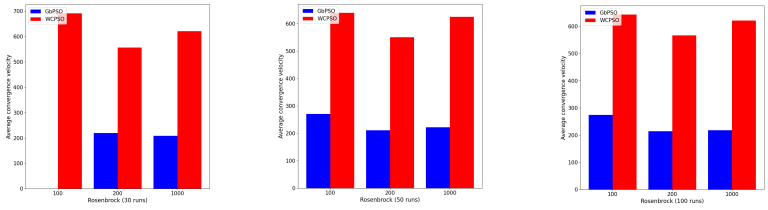
(a) Average convergence velocity in 30 runs

(b) Average convergence velocity in 50 runs

(c) Average convergence velocity in 100 runs

Figure 4: Average convergence velocity in all runs and population size - Rastrigin

For the Rosenbrock function, the observation regarding the relationship in the behaviour of the average convergence velocity and the swarm size can still be considered true for runs with more than a 20% success rate. Average convergence velocity of the GbPSO in this function is still lower than that of the standard PSO. These observations are presented in Figure 5.



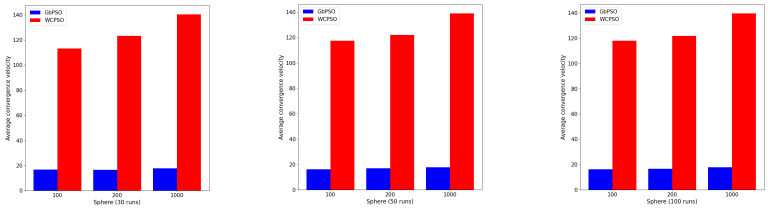
(a) Average convergence velocity in 30 runs

(b) Average convergence velocity in 50 runs

(c) Average convergence velocity in 100 runs

Figure 5: Average convergence velocity in all runs and population size - Rosenbrock

For the Sphere and Three-hump Camel functions where both algorithms failed to converge to the optimal value, the GbPSO converges to a point in the search space faster than the standard PSO, with higher particle count runs also having higher number of iterations needed to converge to that point.

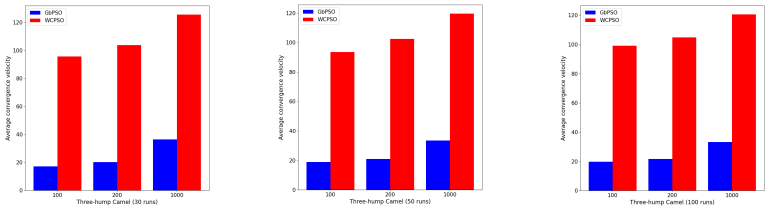


(a) Average Iterations (Failure) in 30 runs

(b) Average Iterations (Failure) in 30 runs in 50 runs

(c) Average Iterations (Failure) in 30 runs in 100 runs

Figure 6: Average Iterations (Failure) in 30 runs in all runs and population size - Sphere

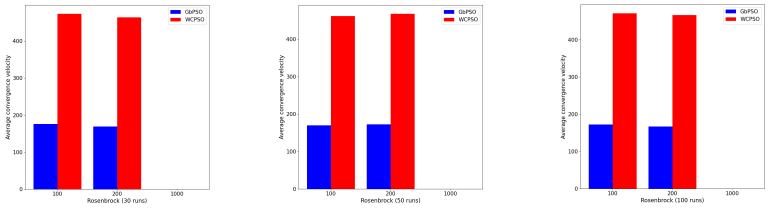


(a) Average Iterations (Failure) in 30 runs

(b) Average Iterations (Failure) in 30 runs in 50 runs

(c) Average Iterations (Failure) in 30 runs in 100 runs

Figure 7: Average Iterations (Failure) in 30 runs in all runs and population size - Three-hump Camel



(a) Average Iterations (Failure) in 30 runs

(b) Average Iterations (Failure) in 30 runs in 50 runs

(c) Average Iterations (Failure) in 30 runs in 100 runs

Figure 8: Average Iterations (Failure) in 30 runs in all runs and population size - Rosenbrock

4.3 Summary of Experimental Results

There were two numerical simulations done in this study. The first experiment numerically validates the results of the convergence analysis. It identified the effects of changing the values of w and α in the performance of the GbPSO in terms of its exploration-exploitation ability, and how fast the swarm converges to the global minimum. Successful and failed runs were observed on each parameter combinations, on each benchmark functions tested. Said results were summarized in Table 6.

For the second criteria, two observations were noted: (1) when parameter combinations are grouped according to their α, the number of iterations needed to find the global optima decreases as w decreases; and (2) when parameter combinations grouped according to their w, the number of iterations needed to find the global optima decreases as α increases. These observations have exceptions on some of the benchmark functions.

Benchmark Function	**Runs = 30**		
	Success (all runs)	Success (some runs)	Failure (all runs)
Sphere	96.67	2.22	1.11
Schaffer N.2	100	-	-
Three-Hump	96.67	2.22	1.11
Rastrigin	100	-	-
Rosenbrock	87.78	-	12.22

Table 6: Success rates of the different parameter combinations of GbPSO

For the second experiment, the GbPSO and WCPSO were compared in terms their success rates in finding the global optima, average cost value, and average convergence velocity in the same benchmark functions used in the first experiment. In terms of their success rates in finding the global optima, both algorithms performed the same in all the functions - succeeded in the Schaffer N.2 and Rastrigin; failed in the Sphere and Three-hump Camel. For the Rosenbrock function, the WCPSO had a higher success rate than the GbPSO.

Functions where both algorithms failed to converge (Sphere and Three-hump Camel), the average cost value of both algorithms were compared. Both algorithms showed the behaviour where the average cost value improves as the swarm size increases. However, the GbPSO exhibited inferior average cost value compared to the WCPSO, albeit with a relatively small difference.

In terms of their average convergence velocity, the GbPSO managed to converged faster to the optimal value of Schaffer N.2 and Rastrigin functions compared to the WCPSO. For the remaining functions, the GbPSO still converged to a point faster than the WCPSO.

5 Conclusion

In this study, an alternative particle swarm optimization (PSO) algorithm (called Gamma-based PSO or GbPSO) was developed by replacing the probability distribution used in its velocity update equation - from $U(0,1)$ to the standard gamma distribution. This simple change was motivated by the studies of [4, 8, 14] to take advantage of the innate properties of the $\mathcal{N}(0,1)$ distribution (larger standard deviation than uniform distribution) and Lévy

distribution (fatter tails than a Gaussian distribution) to alleviate one of the weakness of the standard PSO in its inability to escape local minima.

A mathematical model of the GbPSO was also introduced and defined using techniques in [15]. This model was transformed to a 1-particle 1-dimension swarm to analytically show that the GbPSO has the ability to converge to a point in the search space whenever its parameters were chosen according to a set of guidelines presented and proven throughout the study. To further test the results of the convergence analysis, numerical simulations were done. Said simulations showed positive results, with a majority of the parameter combinations used successfully obtaining the optimal value for a number of the benchmark functions. The best performing parameter combinations were then compared to the standard PSO, with its parameters taken from the literature. Results of this comparison showed that the GbPSO outperformed the standard PSO in all of the benchmark functions in terms of their convergence velocity. However, the WCPSO outperformed the GbPSO in terms of average cost value, and percentage of successful runs.

References

[1] Anthony Carlisle and Gerry Dozier. "An off-the-shelf pso". In: *Proceeding of Workshop on Particle Swarm Optimization*. 2001.

[2] M. Clerc and J. Kennedy. "The particle swarm - explosion, stability, and convergence in a multidimensional complex space". In: *IEEE Transactions on Evolutionary Computation* 6.1 (2002), pp. 58–73. DOI: 10.1109/4235.985692.

[3] Maurice Clerc. "Standard Particle Swarm Optimisation". 15 pages. Sept. 2012. URL: https://hal.archives-ouvertes.fr/hal-00764996.

[4] Hou-Ping Dai, Dong-Dong Chen, and Zhou-Shun Zheng. "Effects of Random Values for Particle Swarm Optimization Algorithm". In: *Algorithms* 11.3 (2018), p. 23. ISSN: 1999-4893. DOI: 10.3390/a11020023.

[5] F. Heppner and U. Grenander. "A stochastic nonlinear model for coordinated bird flocks". In: *The ubiquity of chaos*. Ed. by Saul Krasner. Washington, USA: AAAS, 1990, pp. 233–238.

[6] M. Jiang, Y.P. Luo, and S.Y. Yang. "Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm". In: *Information Processing Letters* 102.1 (2007), pp. 8–16. ISSN: 0020-0190. DOI: https://doi.org/10.1016/j.ipl.2006.10.005. URL: https://www.sciencedirect.com/science/article/pii/S0020019006003103.

[7] James Kennedy and Russell Eberhart. "Particle swarm optimization". In: *Proceedings of IEEE International Conference on Neural Networks* IV (1995), pp. 1942–1948. DOI: 10.1109/icnn.1995.488968.

[8] Renato A. Krohling. "Gaussian swarm: a novel particle swarm optimization algorithm". In: *IEEE Conference on Cybernetics and Intelligent Systems, 2004.* Vol. 1. 2004, 372–376 vol.1. DOI: 10.1109/ICCIS.2004.1460443.

[9] Lester James V Miranda, Aaron Moser, and Siobhán K Cronin. *Welcome to PySwarms's documentation!* Available at https://pyswarms.readthedocs.io/en/latest/index.html. Date accessed: Jun. 2019. URL: https://pyswarms.readthedocs.io/en/latest/index.html.

[10] E. Ozcan and C.K. Mohan. "Particle swarm optimization: surfing the waves". In: *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406).* Vol. 3. 1999, 1939–1944 Vol. 3. DOI: 10.1109/CEC.1999.785510.

[11] M.E.H. Pedersen. *Good Parameters for Particle Swarm Optimization.* Tech. rep. HL1001. Hvass Laboratories, 2010.

[12] Riccardo Poli. "Analysis of the Publications on the Applications of Particle Swarm Optimisation". In: *Journal of Artificial Evolution and Applications* 2008 (Feb. 2008), p. 10. DOI: 10.1155/2008/685175.

[13] Riccardo Poli, James Kennedy, and Tim Blackwell. "Particle Swarm Optimization: An Overview". In: *Swarm Intelligence* 1 (Oct. 2007). DOI: 10.1007/s11721-007-0002-0.

[14] T.J. Richer and T.M. Blackwell. "The Lévy Particle Swarm". In: *2006 IEEE International Conference on Evolutionary Computation.* 2006, pp. 808–815. DOI: 10.1109/CEC.2006.1688394.

[15] Berthold Immanuel Schmitt. "Convergence Analysis for Particle Swarm Optimization". PhD thesis. FAU University Press, 2015.

[16] Yuhui Shi and Russell Eberhart. "A modified particle swarm optimizer". In: *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)* (1998), pp. 69–73. DOI: 10.1109/ICEC.1998.699146.

[17] Radha Thangaraj, Millie Pant, and Kusum Deep. "Initializing PSO with probability distributions and low-discrepancy sequences: The comparative results". In: *2009 World Congress on Nature Biologically Inspired Computing (NaBIC).* 2009, pp. 1121–1126. DOI: 10.1109/NABIC.2009.5393814.

[18] Ioan Cristian Trelea. "The particle swarm optimization algorithm: convergence analysis and parameter selection". In: *Information Processing Letters* 85.6 (2003), pp. 317–325. ISSN: 0020-0190. DOI: https://doi.org/10.1016/S0020-0190(02)00447-7.

[19] Frans Van Den Bergh and A. P. Engelbrecht. "An Analysis of Particle Swarm Optimizers". PhD thesis. ZAF: University of Pretoria, 2002.

This page is intentionally left blank