The Importance of Component-Wise Stochasticity in Particle Swarm Optimization

Elre T. Oldewage[1,2][0000−0002−0568−8700], Andries P. Engelbrecht[1,3][0000−0002−0242−3539], and Christopher W. Cleghorn[1][0000−0002−7860−0650]

[1] Department of Computer Science, University of Pretoria, Pretoria, South Africa
vze.ezv@gmail.com, engel@cs.up.ac.za, ccleghorn@cs.up.ac.za
[2] Council for Scientific and Industrial Research, Pretoria, South Africa
[3] Institute for Big Data and Data Science, Pretoria, South Africa

Abstract. This paper illustrates the importance of independent, component-wise stochastic scaling values, from both a theoretical and empirical perspective. It is shown that a swarm employing scalar stochasticity is unable to express every point in the search space if the problem dimensionality is sufficiently large in comparison to the swarm size. The theoretical result is emphasized by an empirical experiment, comparing the performance of a scalar swarm on benchmarks with reachable and unreachable optima. It is shown that a swarm using scalar stochasticity performs significantly worse when the optimum is not in the span of its initial positions. Lastly, it is demonstrated that a scalar swarm performs significantly worse than a swarm with component-wise stochasticity on a large range of benchmark functions, even when the problem dimensionality allows the scalar swarm to reach the optima.

1 Introduction

The particle swarm optimization (PSO) algorithm employs stochasticity as an important mechanism to avoid premature convergence to local optima. The stochasticity should (usually) be applied in every dimension (i.e. component-wise) to ensure independence between position updates in each dimension. However, it is a common mistake for scalar stochastic values to be used instead, which restricts the swarm's movement and degrades performance [7, 13, 18, 19].

This paper investigates the effect of using scalar stochasticity, both theoretically and empirically. The paper begins by introducing the PSO algorithm and briefly discussing the importance of component-wise stochasticity in Section 2. Section 3 provides theoretical results to formalize the restriction on the swarm's movement caused by scalar stochasticity. It is shown that there is a problem of "reachability", i.e. a swarm with scalar stochasticity will not be able to reach the optimum if the problem dimensionality is higher than the size of the swarm.

Section 4 examines the empirical effects of reachability on the performance of a swarm employing scalar stochasticity. The section compares a scalar swarm's

performance on a number of constructed benchmark functions with optima defined to be reachable or unreachable by a scalar swarm. Section 5 goes on to compare the performance of a swarm with scalar stochasticity to a swarm with component-wise stochasticity for a wide range of benchmark functions that are not biased towards or against the scalar swarm. Section 6 concludes the paper.

2 Background

This section briefly discusses the PSO algorithm and introduces relevant concepts regarding the importance of component-wise stochasticity.

PSO is a stochastic, population-based optimization algorithm [4] that does not require gradient information and may thus be applied to black box optimization problems. A swarm consists of a number of particles. The position of a particle in the search space represents a potential solution to the optimization problem. The particle moves through the search space for a number of iterations, using local information (the best position encountered by the particle thus far, called the personal best position) and global information (the best position encountered by all the particles within the given particle's logical neighbourhood, called the global or local best position, depending on how the neighbourhood is defined). This paper considers the global best topology, but the findings presented are applicable to arbitrary topologies. Each particle i's position is updated at iteration t according to:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \tag{1}$$

where $\mathbf{x}_i^{t+1}$ denotes the position of particle i at iteration $t+1$ and $\mathbf{v}_i^{t+1}$ denotes its velocity at iteration $t+1$. The particle's initial position, $\mathbf{x}_i^0$, is usually drawn from a uniform random distribution over the search space boundaries (in every dimension). PSO with inertia weight, as introduced in [17], updates particle i's velocity at iteration t in every dimension j as below:

$$v_{ij}^{t+1} = wv_{ij}^t + c_1 r_{1j}(y_{ij}^t - x_{ij}^t) + c_2 r_{2j}(\hat{y}_{ij} - x_{ij}^t) \tag{2}$$

where v_{ij}^{t+1} denotes particle i's velocity in dimension j, w denotes the inertia weight, and c_1 and c_2 denote the cognitive and social acceleration constants respectively. r_{1j}, $r_{2j} \sim U(0,1)$ are random numbers sampled between 0 and 1 at every iteration. y_{ij}^t denotes the personal best position of particle i in the jth dimension and $\hat{y}_{ij}$ denotes jth dimension of the best position found by all the particles in the neighbourhood of particle i. Particle neighbourhoods are usually defined by logical indexing of the swarm. When neighbourhoods are strict subsets of the entire swarm, the algorithm is referred to as a local best PSO. If every particle's neighbourhood consists of the entire swarm, the algorithm is referred to as a global best PSO and $\hat{\mathbf{y}}_i = \hat{\mathbf{y}}$ is called the global best position. Except where otherwise specified, this paper considers a global best PSO.

The stochastic scaling components, $\mathbf{r}_1$ and $\mathbf{r}_2$ can also be expressed as diagonal matrices, $\mathbf{R}_1$ and $\mathbf{R}_2$, as illustrated below:

$$\mathbf{v}_i^{t+1} = w\mathbf{v}_i^t + c_1\mathbf{R}_1(\mathbf{y}_i^t - x_{ij}^t) + c_2\mathbf{R}_2(\hat{\mathbf{y}} - \mathbf{x}_i^t) \tag{3}$$

where $\mathbf{R}_1$ and $\mathbf{R}_2$ are diagonal matrices, with $\mathbf{r}_1$ and $\mathbf{r}_2$ forming their diagonals.

This paper emphasizes that if r_1 and r_2 are random scalars, then the swarm's movement becomes entirely linear. Every position investigated by the swarm will necessarily be a linear combination of the swarm's initial positions, velocities and personal best positions. If the swarm size is too small relative to the problem dimensionality, then the swarm can only reach a subspace within the larger search space (as is proved in Section 3). Since the swarm is typically initialised randomly, there is a possibility that the optimum can not be expressed as a linear combination of the swarm's initial positions, i.e. the swarm will never be able to find the optimum.

The effect of using scalar r_1 and r_2 values has been discussed in literature [14, 20]. Paquet and Engelbrecht introduced a Linear PSO [14], in order to solve constrained linear optimization problems. If the swarm is initialised so that all positions are within the problem constraints, then using scalar r_1 and r_2 values ensures that the swarm can never leave the feasible space, which forms a subspace of the search space. In [20], purposeful dimensional coupling via shared $\mathbf{r}_1$ and $\mathbf{r}_2$ components was suggested to reduce the unwanted roaming exhibited by PSO in high dimensional problem spaces.

Throughout the remainder of the paper, a swarm that uses scalar values for r_1 and r_2 will be called a "scalar swarm". A swarm that uses vectors for $\mathbf{r}_1$ and $\mathbf{r}_2$ which are multiplied with the cognitive and social components in every dimension (or, alternatively, are diagonal matrices) will be called a "vector swarm".

3 Theoretical Results

This section provides a theoretical discussion regarding the consequences of using a scalar PSO. A number of definitions and key concepts are introduced first [15].

Definition 1. *A set of vectors $\mathcal{I} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_M\} \subset \mathbb{R}^n$ is linearly dependent if there exists a finite number of distinct vectors, $\mathbf{z}_1, \mathbf{z}_2, ...\mathbf{z}_K \in \mathcal{I}$, and scalars, $a_1, a_2, ..., a_K \in \mathbb{R}$, not all zero, such that*

$$a_1\mathbf{z}_1 + a_2\mathbf{z}_2 + ... + a_K\mathbf{z}_K = \mathbf{0} \tag{4}$$

Since at least one scalar is non-zero, say $a_1 \neq 0$, the vector $\mathbf{z}_1$ can be expressed as a linear combination of the other vectors:

$$\mathbf{z}_1 = -\frac{a_2}{a_1}\mathbf{z}_2 - ... - \frac{a_K}{a_1}\mathbf{z}_K \tag{5}$$

Thus, the set $\mathcal{I}$ is linearly dependent if and only if at least one element in $\mathcal{I}$ can be written as a linear combination of the other elements in $\mathcal{I}$.

Definition 2. *A set of vectors $\mathcal{I} = \{\mathbf{z}_1, ..., \mathbf{z}_M\} \subset \mathbb{R}^n$ is linearly independent if*

$$a_1\mathbf{z}_1 + a_2\mathbf{z}_2 + ... + a_M\mathbf{z}_M = \mathbf{0} \tag{6}$$

can only be satisfied by $a_1 = a_2 = ... = a_M = 0$. Thus no element in $\mathcal{I}$ can be written as a linear combination of other elements from $\mathcal{I}$.

Definition 3. *Let $\mathcal{I}$ be a non-empty set of vectors from $\mathbb{R}^n$ (i.e. $\emptyset \neq \mathcal{I} \subset \mathbb{R}^n$). Then the span of $\mathcal{I}$ is the smallest subspace $W \subseteq \mathbb{R}^n$ that contains $\mathcal{I}$. Thus span($\mathcal{I}$) $= W$. The subspace W consists of all linear combinations of elements of $\mathcal{I}$, given below (where $|.|$ denotes set cardinality):*

$$span(\mathcal{I}) = \left\{ \sum_{k=1}^{|\mathcal{I}|} a_k \mathbf{z}_k \;\middle|\; \mathbf{z}_k \in \mathcal{I}, a_k \in \mathbb{R}, k \in \mathbb{N} \right\} \tag{7}$$

Definition 4. *A non-empty set of vectors, $\mathcal{I} = \{\mathbf{z}_1, ..., \mathbf{z}_M\}$, is a spanning set for a subspace $W \subseteq \mathbb{R}^n$ if and only if any element in W can be expressed as a linear combination of elements in $\mathcal{I}$. In other words, for any non-zero $\mathbf{z} \in W$, there exist scalars $a_1, a_2, ..., a_M$ with at least one $a_i \neq 0$ such that*

$$\mathbf{z} = a_1\mathbf{z}_1 + a_2\mathbf{z}_2 + ... + a_M\mathbf{z}_M \tag{8}$$

Definition 5. *A basis for a subspace S of $\mathbb{R}^n$ is a set of vectors $\mathcal{B}$ in S where $\mathcal{B}$ is a spanning set for S, and $\mathcal{B}$ is linearly independent.*

Armed with these definitions, it is proven below that if r_1 and r_2 are scalars, then the positions of any particle at any iteration of the search space must be a linear combination of their initial positions, personal best positions, and velocities. The theorem below is for a local best PSO, since global best PSO can be considered as a special case of local best PSO where the neighbourhood is the entire swarm. For the sake of generality, the theorem makes no assumptions regarding the initial particle velocities or personal best positions.

Theorem 1. *For a particle swarm governed by the movement update equations in equations (1) and (2), at any iteration $t \geq 0$, the position $\mathbf{x}_i^t$ of any particle i is in the span of $\mathcal{I}$ where $\mathcal{I} = \{\mathbf{x}_1^0, \mathbf{y}_1^0, \mathbf{v}_1^0, ..., \mathbf{x}_m^0, \mathbf{y}_m^0, \mathbf{v}_m^0\}$.*

Proof. Suppose that particle velocities, positions and personal best positions are initialised randomly within the search space. Let the set of all these initial points be given by $\mathcal{I} = \{\mathbf{x}_1^0, \mathbf{y}_1^0, \mathbf{v}_1^0, ..., \mathbf{x}_{n_s}^0, \mathbf{y}_{n_s}^0, \mathbf{v}_{n_s}^0\}$. Assume that all the elements in $\mathcal{I}$ are unique and non-zero. These assumptions are made without loss of generality: the probability of obtaining a zero vector from a uniform initialisation is zero, since the probability of a continuous random variable being a particular constant is zero. Similarly, the probability of sampling two equal vectors is zero because the set of such points have zero measure.

The position of any particle at $t = 0$ is in $\mathcal{I}$ by the definition of $\mathcal{I}$. Thus, the hypothesis holds for the case $t = 0$.

At iteration $t = 1$, the position of any particle i is given by

$$\mathbf{x}_i^1 = \mathbf{x}_i^0 + \mathbf{v}_i^1 \tag{9}$$

Since $\mathbf{x}_i^0 \in \mathcal{I}$, it is only necessary to prove that $\mathbf{v}_i^1 \in span(\mathcal{I})$ for $\mathbf{x}_i^1$ to be in the span of $\mathcal{I}$. According to the velocity update equation,

$$\mathbf{v}_i^1 = w\mathbf{v}_i^0 + c_1r_1(\mathbf{y}_i^0 - \mathbf{x}_i^0) + c_2r_2(\hat{\mathbf{y}}_i^0 - \mathbf{x}_i^0) \tag{10}$$

$$= w\mathbf{v}_i^0 + c_1r_1\mathbf{y}_i^0 + c_2r_2\hat{\mathbf{y}}_i^0 - (r_1c_1 + r_2c_2)\mathbf{x}_i^0 \tag{11}$$

By definition, $\mathbf{v}_i^0, \mathbf{y}_i^0, \mathbf{x}_i^0 \in \mathcal{I}$. Additionally, the neighbourhood best position is chosen from among the personal best positions of the other particles in the neighbourhood, so that $\hat{\mathbf{y}}_i^0 \in \mathcal{I}$. Thus, if particle i is not the neighbourhood best, then $\mathbf{v}_i^1$ is a linear combination of four distinct elements from $\mathcal{I}$. If particle i is the neighbourhood best, then $\mathbf{y}_i^0 = \hat{\mathbf{y}}_i^0$ and $\mathbf{v}_i^1$ is a linear combination of three distinct elements from $\mathcal{I}$. In either case, $\mathbf{v}_i^1 \in \text{span}(\mathcal{I})$ by definition 3. The fact that $\mathbf{v}_i^1 \in \text{span}(\mathcal{I})$ will be referred to as (*). Therefore, since $\mathbf{x}_i^1$ is the sum of two elements in $span(\mathcal{I})$, $\mathbf{x}_i^1$ is in the span of $\mathcal{I}$. Since this is true for any particle i, all the particles' positions at iteration i must be in the span of $\mathcal{I}$ - this fact will be referred to as (**).

Suppose for all iterations $s \leq t$, that the positions of all the particles are in the span of $\mathcal{I}$. It will now be proved that the positions of all the particles must still be in the span of $\mathcal{I}$ at iteration $t+1$. The position of any particle i is given by the position update equation:

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1} \tag{12}$$

where $\mathbf{x}_i^t \in span(\mathcal{I})$ by virtue of the inductive assumption. It thus remains to prove that $\mathbf{v}_i^{t+1}$ is in the span of $\mathcal{I}$:

$$\mathbf{v}_i^{t+1} = w\mathbf{v}_i^t + c_1 r_1(\mathbf{y}_i^t - \mathbf{x}_i^t) + c_2 r_2(\hat{\mathbf{y}}_i^t - \mathbf{x}_i^t) \tag{13}$$

$$= w\mathbf{v}_i^t + c_1 r_1 \mathbf{y}_i^t + c_2 r_2 \hat{\mathbf{y}}_i^t - (r_1 c_1 + r_2 c_2)\mathbf{x}_i^t \tag{14}$$

where $\mathbf{x}_i^t \in span(\mathcal{I})$ by the inductive assumption. It remains to prove that $\mathbf{v}_i^t$, $\mathbf{y}_i^t$ and $\hat{\mathbf{y}}_i^t$ are in the span of $\mathcal{I}$. According to the position update equation,

$$\mathbf{x}_i^t = \mathbf{x}_i^{t-1} + \mathbf{v}_i^t \tag{15}$$

$$\implies \mathbf{v}_i^t = \mathbf{x}_i^t - \mathbf{x}_i^{t-1} \tag{16}$$

In other words, $\mathbf{v}_i^t$ is a linear combination of $\mathbf{x}_i^t$ and $\mathbf{x}_i^{t-1}$, both of which are elements in the span of $\mathcal{I}$ by the inductive assumption. Thus, $\mathbf{v}_i^t$ is also in the span of $\mathcal{I}$. The personal best position of any particle i can only be equal to one of the particle's previous positions. But, by the inductive assumption, all the particle's previous positions were in the span of $\mathcal{I}$. Thus, $\mathbf{y}_i^t$ must be in the span of $\mathcal{I}$. Similarly, the neighbourhood best position must be equal to a previous position of some particle in i's neighbourhood, all of which are in the span of $\mathcal{I}$ by the inductive assumption. Therefore, the velocity and also the position of particle i at iteration $t+1$ must be in $span(\mathcal{I})$. □

Theorem 1 implies that, when r_1 and r_2 are random scalar values, the positions of the particles are limited to be in the span of their initial velocities, positions and personal best positions. If either of the assumptions on $\mathcal{I}$ does not hold (e.g. some vectors are multiples of another or some are zero), then the positions of the particles are limited further to being linear combinations of all non-zero, linearly independent initial velocities, positions and personal best positions. The question arises whether any point in the search space can be expressed in terms of such linear combinations. This question is answered by the theorem below:

Theorem 2. *Suppose $\mathcal{I}$ contains m linearly independent vectors and $S = [L, U]^n$. If $m < n$ then span$(\mathcal{I}) \cap S \subsetneq S$. Thus $\mathcal{I}$ can only be a spanning set of S if it contains at least n linearly independent elements.*

Theorem 2 follows from the fundamental theorem of invertible matrices (as given in [15]), the exact proof as given in [10] is not reproduced here. If $\mathcal{I}$ constitutes a spanning set for the search space (i.e. the span of $\mathcal{I}$ is larger than the search space or equal to), then any point in the search space can theoretically be reached by the particles. However, if $\mathcal{I}$ is not a spanning set of the search space (i.e. the span of $\mathcal{I}$ is a strict subspace within the search space), then the particles can not reach every position in the search space. If the global optimum happens to be outside the span of $\mathcal{I}$, then the particles will never be able to find it. Since initial positions and personal best positions are typically generated randomly, this is a realistic scenario in high dimensional spaces.

If the swarm's velocities are initialised to zero and the initial personal best positions are set equal to the initial positions, then the portion of the search space that can be reached by the particles is even smaller. Additionally, the swarm may lose degrees of freedom throughout the search. Since the algorithm is executed on a computer with limited precision, some of the vectors in $\mathcal{I}$ may be cancelled out further in the search. Though unlikely, the span of the swarm may in fact decrease as the search progresses. Thus, if the size of the swarm is much smaller than the dimensionality of the search space, then the swarm will be unable to reach a large part of the search space. Unfortunately, simply increasing the number of particles in the swarm is not an adequate solution, because it greatly increases the computational cost. Additionally, the swarm size parameter influences the swarm's searching behaviour in other ways, so changing the swarm size drastically may have unintended consequences [5, 9].

4 Illustration of Reachability

This section aims to illustrate the importance of reachability, i.e. that the scalar swarm's performance is severely penalized if the optimum is not in the span of the swarm's initial positions. Section 4.1 describes the experiment's empirical method and Section 4.2 summarizes the results.

4.1 Empirical Method - Reachability

The optimum can be placed inside or outside the swarm's initial span by shifting: $f(\mathbf{x})_{Sh} = f((\mathbf{x} - \boldsymbol{\gamma}))$, where $\mathbf{x}$ denotes the position vector to be evaluated, f denotes the objective function and $\boldsymbol{\gamma}$ denotes the shift vector. If the scalar swarm performs well when the shift places the optimum within its reach, but poorly when the optimum is outside the swarm's span, then the importance of reachability will be empirically justified.

Unfortunately, the performance of a swarm on a function with a reachable optimum can not be compared directly with its performance on an unreachable version of that same function. The swarm would essentially be optimizing different

functions, since applying a particular shift may change the problem's difficulty. Thus, in order to compare the influence of reachability on the scalar swarm's performance, a total of 30 different shifts were generated for each benchmark problem: 15 placed the optimum within the scalar swarm's span and 15 moved the optimum to an unreachable region. The swarm was run 30 times on all 30 versions of each benchmark function. The suite of benchmark functions consisted of Ackley, Absolute Value, Elliptic, Griewank, Quartic, Rastrigin, Rosenbrock, Schwefel 1.20, Schwefel 2.21, Spherical and Weierstrass (as defined in [6]). A total of 30×11 functions were thus under consideration.

The process for generating the shifts is described below. First, the Modified Gram-Schmidt method [12] was applied to the particles' initial positions to produce a basis $\mathcal{B}$ containing m-many vectors (where m is the swarm size, as before). For the shift to be reachable, a direction vector $\mathbf{d}$ was generated by a random linear combination of the vectors in $\mathcal{B}$. The direction vector was normalized and used to define a line passing through the search space center. A random point on that line, γ was then chosen as the shift. To produce an unreachable shift, a new random vector $\mathbf{s}$ was chosen (distributed uniformly over the search space in each dimension, like the particle positions). The vector $\mathbf{s}$ was then orthogonalized relative to $\mathcal{B}$, producing $\tilde{\mathbf{d}}$, a direction vector orthogonal to the swarm's span. As before, γ was chosen to be a random point on the line passing through the center of the search space with direction $\tilde{\mathbf{d}}$.

The experiments used PSO with inertia weight as introduced in [17] with the global best topology. The selected inertia weight, $w = 0.7298$ and the acceleration coefficients $c_1 = c_2 = 1.49618$ are known good values suggested by Clerc [3] that guarantee convergence of the swarm (in terms of expectation and variance of particle positions [2]). As suggested by [1], all personal and global best positions were restricted to be within the search space. Each swarm consisted of 10 particles ($m = 10$), so that m is low enough to test problems of dimensionality 5 times larger than m without venturing into large scale optimization). The particles' initial positions were initialised uniform randomly throughout the search space. Each particle's initial personal best position was set equal to its initial position, and its velocity was initialised to zero. Thus, the scalar swarm was limited to the span of its initial positions. The experiments were repeated for dimensions $n = \{15, 20, 25, 50\}$. Every simulation ran 2000 iterations to allow sufficient time for the swarm to converge.

4.2 Results - Reachability

Table 1 compares the scalar swarm's performance on the 165 (11 functions $\times$ 15) problems with reachable optima against the 165 problems with unreachable optima. As mentioned in Section 4.1, 30 runs were performed on each problem for statistical significance. Every row of the table corresponds to the results for a given problem dimensionality. Friedman tests with a p-value of 0.05 were used to detect statistically significant differences between the scalar and vector PSO's performance (in terms of the best scores attained over all runs on a given function). If the Friedman test indicated a significant difference, pairwise

comparisons were done by Mann-Whitney U tests with a p-value of 0.05. If no statistically significant different was found, the result was recorded as a draw.

Table 1. Scalar Swarm's Performance on Reachable and Unreachable Problems

Dimensionality	Reachable Wins	Draws	Unreachable Wins
$n = 15$	104	26	35
$n = 20$	117	21	27
$n = 25$	120	22	23
$n = 50$	93	28	44

As expected from the theoretical discussion, Table 1 shows that the scalar swarm performed significantly better on the problems with reachable optima. It may still be possible for the swarm to perform better on the benchmark functions with theoretically unattainable optima if the swarm's initial subspace is "sufficiently" close to the shifted optimum. Additionally, a given problem with an unreachable problem may still be easier than a problem with a reachable optimum, as discussed in Section 4.1, resulting in a few wins for the swarm optimizing the unreachable problems. However, the general trend is that the scalar swarm preform better on a given benchmark function when the optimum is within the span of its initial positions, as would be expected from the theory.

5 Extensive Performance Comparison

As seen in the previous section, it may be that the swarm's reachable subspace may lie in a region sufficiently close to the optimum for it to be a mere technicality that the optimum is unreachable. Since the benchmark suites from the previous section were designed either in favour or against the scalar PSOs, this section compares the performance of the vector and scalar swarms on a large suite of unbiased benchmark functions. The optima for these functions are either shifted by a predefined constant (as specified in the corresponding technical papers) or by a random vector, distributed uniformly over the search space. Section 5.1 details the empirical method and Section 5.2 discusses the results.

5.1 Empirical Method - Performance Comparison

The benchmark suite consisted of 28 base functions which are listed in the "Function Name" column of Table 2. A given function f was shifted and rotated to produce f_{ShRot} according to

$$f(\mathbf{x})_{ShRot} = f(Q(\mathbf{x} - \boldsymbol{\gamma})) + \beta \tag{17}$$

where β is a constant scalar, $\boldsymbol{\gamma}$ is either constant or uniform random over the search space in each dimension, and Q is a randomly generated orthogonal matrix. The constants are specified in Table 2.

Table 2. Benchmark Functions

Function Name	**Src**	γ	β	**Rot**	**Function Name**	**Src**	γ	β	**Rot**
Absolute Value	f_1	Rand	0.0	No	Rastrigin	f_{12}	Rand	0.0	No
Ackley	f_2	Rand	0.0	No	Rastrigin Rot	f_{12}	2.0	-330	No
Ackley Sh	f_2	10.0	-140	No	Rastrigin Sh	f_{12}	0.0	0.0	Yes
Ackley Rot	f_2	0.0	0.0	Yes	Rastrigin ShRot	f_{12}	1.0	-330	Yes
Ackley ShRot	f_2	-32.0	-140	No	Rosenbrock	f_{13}	Rand	0.0	No
Alpine	F_7	Rand	0.0	No	Rosenbrock Sh	f_{13}	10.0	390	No
Brown	F_{25}	Rand	0.0	No	Rosenbrock Rot	f_{13}	0.0	0.0	Yes
Dixon-Price	F_{48}	Rand	0.0	No	Salomon	f_{14}	Rand	0.0	No
Egg Holder	f_4	Rand	0.0	No	Schaffer 6	f_{15}	Rand	0.0	No
Elliptic	f_5	Rand	0.0	No	Schaffer 6 ShRot	f_{15}	20.0	-300	Yes
Elliptic Sh	f_5	10.0	-450	No	Schwefel	G_5	0.0	0.0	No
Elliptic Rot	f_5	0.0	0.0	Yes	Schwefel 1.2	f_{16}	Rand	0.0	No
Elliptic ShRot	f_5	10.0	-450	Yes	Schwefel 1.2 Sh	f_{16}	10.0	-450	No
Griewank	f_6	Rand	0.0	No	Schwefel 1.2 Rot	f_{16}	0.0	0.0	Yes
Griewank Sh	f_6	10.0	-180	No	Schwefel 2.21	f_{19}	Rand	0.0	No
Griewank Rot	f_6	0.0	0.0	Yes	Schwefel 2.22	f_{20}	Rand	0.0	No
Griewank ShRot	f_6	-60.0	-180	Yes	Shubert	f_{21}	Rand	0.0	No
HyperEllipsoid	f_7	Rand	0.0	No	Spherical	f_{22}	Rand	0.0	No
Michalewicz	f_8	Rand	0.0	No	Spherical Sh	f_{22}	10.0	-450	No
Norwegian	f_9	Rand	0.0	No	Step	f_{23}	Rand	0.0	No
Powell Singular 2	F_{92}	Rand	0.0	No	Vincent	f_{24}	Rand	0.0	No
Quadric	f_{10}	Rand	0.0	No	Weierstrauss	f_{25}	Rand	0.0	No
Quartic	f_{11}	Rand	0.0	No	Weierstrauss Sh	f_{25}	1.0	-130	No

The "Rot" column in Table 2 indicates whether the function was rotated or not. The transformations provided a total of 46 benchmark functions. The benchmark suite contains uni- and multi-modal functions that are both separable and non-separable. The definitions of the functions and the corresponding bounds were used as in [6], [8] and [16]. The "Src" column of Table 2 lists the identifier of each function according to its source. Function i from [6] is denoted by f_i; function i from [8] is denoted by F_i and function i from [16] is denoted by G_i. The vector and scalar swarms were run on each of the benchmark problems 30 times for statistical significance. Each simulation ran for 2000 iterations.

All of the functions were minimized in 5, 10, 15, 20, and 25 dimensions. As before, the swarm size was set to 10. If the hypothesis proved in the previous section holds, then it is expected for the scalar swarm's performance to deteriorate as the problem dimensionality exceeds the number of particles in the swarm.

5.2 Results - Performance Comparison

Table 3 summarizes the results of the wide performance comparison.The scalar swarm consistently performed better than the vector swarm on the Quadric function (f_{10}) for $n > 5$. However, the vector swarm outperforms the scalar swarm on nearly all of the benchmark functions, even when the problem dimensionality is low enough for the scalar swarm to reach the optimum. Although the reachability of the optimum also plays a role (as shown previously), the scalar swarm's linear movement prevents the swarm from finding good solutions even inside the swarm's initial subspace.

Table 3. Comparison of Vector and Scalar Swarms Across Dimensionality

Dimensionality	Scalar Wins	Draws	Vector Wins
$n = 5$	0	2	44
$n = 10$	1	0	45
$n = 15$	1	0	45
$n = 20$	1	2	43
$n = 25$	1	1	44

The strong restriction imposed on the scalar swarm becomes apparent in Figures 1 to 4, which plot typical profiles of the swarm diversity (as defined in [11]), averaged over all runs for $n = 5$ and $n = 25$. As the problem dimensionality increases, the vector PSO's swarm diversity also increases. In contrast, the scalar PSO's diversity profile remains unchanged even as the dimensionality increases.

As shown before [20], restricting the swarm's movement may be beneficial in high dimensional spaces for the very reason that the initial velocity explosion is mitigated, causing the unchanged diversity profile observed here. However, in low dimensional spaces, the vector swarm outperforms the scalar swarm.

6 Conclusion

This paper demonstrated the importance of employing component-wise stochasticity both theoretically and empirically. Section 3 showed that a swarm's movement is severely restricted by using scalar values for r_1 and r_2. In particular, it is emphasized that a scalar swarm is limited to the span of its initial particle positions, personal best positions and velocities. Thus, the swarm may not be able to reach the optimum. Section 4 shows that reachability is not merely a theoretical problem, but can also be illustrated empirically. The section constructs benchmark functions with optima explicitly defined to be reachable or unreachable by a scalar swarm. The scalar swarm is shown to perform significantly better on benchmark problems with reachable optima, as expected from the theory.

Since the benchmarks in Section 4 were biased in favour of, or against the scalar swarms, the artificial benchmarks would not provide a fair comparison

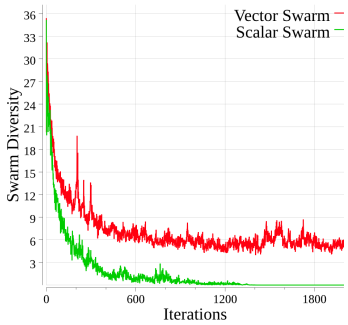

Fig. 1. Diversity, Ackley Shr ($n = 5$)

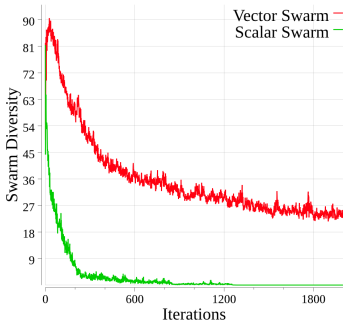

Fig. 2. Diversity, Ackley Shr ($n = 25$)

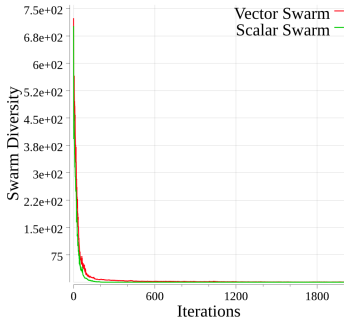

Fig. 3. Diversity, Griewank ($n = 5$)

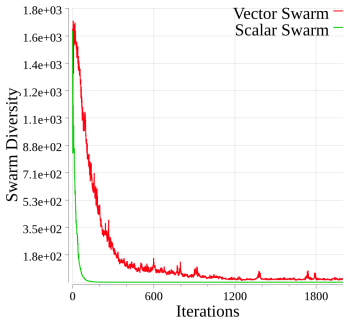

Fig. 4. Diversity, Griewank ($n = 25$)

between scalar and vector swarms. Towards this end, Section 5 demonstrated the performance difference between scalar and vector swarms on an extensive range of benchmarks. It was shown that the vector swarm performs significantly better on almost all of the benchmark functions than the scalar swarm, even when the dimensionality is low enough for the scalar swarm to reach the optimum.

Acknowledgments. This work is based on the research supported by the National Research Foundation (NRF) of South Africa (Grant Number 46712). The opinions, findings and conclusions or recommendations expressed in this article is that of the author(s) alone, and not that of the NRF. The NRF accepts no liability whatsoever in this regard.

References

1. Bratton, D., Kennedy, J.: Defining a standard for particle swarm optimization. In: Proceedings of the IEEE Swarm Intelligence Symposium. pp. 120–127. IEEE Computer Society (2007). https://doi.org/10.1109/SIS.2007.368035

2. Cleghorn, C.W., Engelbrecht, A.P.: Particle swarm stability: a theoretical extension using the non-stagnate distribution assumption. Swarm Intelligence (Sep 2017). https://doi.org/10.1007/s11721-017-0141-x
3. Clerc, M., Kennedy, J.: The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation **6**(1), 58–73 (Feb 2002). https://doi.org/10.1109/4235.985692
4. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science. pp. 39–43 (Oct 1995). https://doi.org/10.1109/MHS.1995.494215
5. Engelbrecht, A.P.: Fitness function evaluations: A fair stopping condition? In: Proceedings of the IEEE Symposium on Swarm Intelligence. pp. 1–8 (Dec 2014). https://doi.org/10.1109/SIS.2014.7011793
6. Engelbrecht, A.: Particle swarm optimization: Global best or local best? In: Proceedings of the BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence (BRICS-CCI CBIC). pp. 124–135 (Sept 2013). https://doi.org/10.1109/BRICS-CCI-CBIC.2013.31
7. Han, F., Liu, Q.: A diversity-guided hybrid particle swarm optimization based on gradient search. Neurocomputing **137**, 234 – 240 (2014). https://doi.org/https://doi.org/10.1016/j.neucom.2013.03.074, advanced Intelligent Computing Theories and Methodologies
8. Jamil, M., Yang, X.S.: A literature survey of benchmark functions for global optimization problems. Int. Journal of Mathematical Modelling and Numerical Optimisation **4**(2), 150194 (2013)
9. Malan, K., Engelbrecht, A.P.: Algorithm comparisons and the significance of population size. Proceedings of the IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence) pp. 914–920 (2008)
10. Oldewage, E.: The Perils of Particle Swarm Optimisation in High Dimensional Problem Spaces. Master's thesis, University of Pretoria, Pretoria, South Africa (2018)
11. Olorunda, O., Engelbrecht, A.P.: Measuring exploration/exploitation in particle swarms using swarm diversity. In: Proceedings of the IEEE Congress on Evolutionary Computation. pp. 1128–1134 (June 2008). https://doi.org/10.1109/CEC.2008.4630938
12. Paige, C.C., Rozlozník, M., Strakos, Z.: Modified gram-schmidt (mgs), least squares, and backward stability of mgs-gmres. Society for Industrial and Applied Mathematics Journal Matrix Analysis and Applications **28**(1), 264–284 (May 2006). https://doi.org/10.1137/050630416
13. Pandey, S., Wu, L., Guru, S.M., Buyya, R.: A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In: 2010 24th IEEE International Conference on Advanced Information Networking and Applications. pp. 400–407 (April 2010). https://doi.org/10.1109/AINA.2010.31
14. Paquet, U., Engelbrecht, A.P.: Particle swarms for linearly constrained optimisation. Fundamental Informatics **76**(1-2), 147–170 (Feb 2007), http://dl.acm.org/citation.cfm?id=1232695.1232705
15. Poole, D.: Linear Algebra: A Modern Introduction, Third Edition. Cengage Learning, Canada (2011)
16. Ramezani, F., Lotfi, S.: The modified differential evolution algorithm (mdea). In: Pan, J.S., Chen, S.M., Nguyen, N. (eds.) Intelligent Information and Database Systems, Lecture Notes in Computer Science, vol. 7198, pp. 109–118. Springer Berlin Heidelberg (2012). https://doi.org/10.1007/978-3-642-28493-9_13

17. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: Proceedings of the IEEE International Conference on Evolutionary Computation. pp. 69–73 (May 1998). https://doi.org/10.1109/ICEC.1998.699146
18. Yoshida, H., Kawata, K., Fukuyama, Y., Takayama, S., Nakanishi, Y.: A particle swarm optimization for reactive power and voltage control considering voltage security assessment. IEEE Transactions on Power Systems **15**(4), 1232–1239 (Nov 2000). https://doi.org/10.1109/59.898095
19. Zahara, E., Kao, Y.T., Su, J.R.: Enhancing particle swarm optimization with gradient information. In: 2009 Fifth International Conference on Natural Computation. vol. 3, pp. 251–254 (Aug 2009). https://doi.org/10.1109/ICNC.2009.711
20. van Zyl, E., Engelbrecht, A.: Group-based stochastic scaling for pso velocities. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC). pp. 1862–1868 (07 2016)