2015 IEEE Symposium Series on Computational Intelligence

A Subspace-Based Method for PSO Initialization

ET van Zyl
Department of Computer Science
University of Pretoria
Pretoria, South Africa
Email: evzyl@cs.up.ac.za

AP Engelbrecht
Department of Computer Science
University of Pretoria
Pretoria, South Africa
Email: engel@cs.up.ac.za

***Abstract*—Particle swarm optimization (PSO) is known to suffer under the curse of dimensionality. This paper proposes a novel strategy of particle swarm initialization particularly for high dimensional problems. The initialization strategy encourages the swarm to focus on exploitation rather than exploration, thereby allowing it to find fairly good solutions, even in the face of high dimensionality and very large search spaces. The proposed initialization strategy is compared to a number of other initialization strategies on high dimensional problems. The proposed strategy performed considerably better than all the other initialization strategies for the higher dimensional problems. Reasons for the observed behaviour are also discussed.**

I. Introduction

Particle swarm optimisation (PSO) is a stochastic, population-based optimization algorithm developed by Kennedy and Eberhart [5]. PSO has been used successfully to train neural networks and solve complex optimization problems amongst many other real-world applications [19]. PSO has also been extended for applications in high dimensional problems [20], multi-objective optimisation problems [4], niching problems [3] and dynamic environments [2].

Generally, it is advantageous for the particles' initial positions to cover as much of the search space as possible. The more of the search space is covered, the less the chance that the global optimum is in an uncovered region. This significantly increases the likelihood of the optimum being found, since the only way for the PSO to find an optimum in an uncovered region is for the particles to be carried there by momentum.

Usually, search space coverage is achieved by initializing the particles uniformly throughout the search space so that the initial diversity of the swarm is high.

However, for high dimensional problems, the region of the search space that can be covered by the particles' initial positions is very small because the search space grows exponentially with dimensionality. Even if the particles are distributed uniformly throughout the search space, much of the search space remains effectively uncovered due to its sheer size. The exponential growth of the search space with dimensionality also means that simply increasing the size of the swarms is also not a good solution, since the swarm size would also need to increase exponentially.

The proposed initialization strategy recognizes that effective exploration in high dimensional search spaces is practically impossible. Thus, instead of attempting to promote exploration, the initialization strategy forces the swarm to focus more on exploitation or rather, exploration of only a small part of the search space. So rather than distributing the swarm as uniformly as possible throughout the entire search space, the strategy initializes the swarm in a small subspace of the search space. The swarm does not wander far from the initialized region and is thus able to find a - potentially local - optimum within that region. But, as the results show, the local optimum found by such a swarm is usually better than the optimum found by a swarm that attempts to explore the entire search space.

The remainder of the paper is structured as follows: Section 2 provides a brief overview of the basic PSO algorithm and briefly describes each of the initialization strategies against which the proposed initialization strategy will be compared. Section 3 introduces the proposed initialization strategy. Section 4 describes the experimental method. Section 5 provides the results of the experiments and discusses the observed behaviour. Section 6 concludes the paper.

II. Background

The section provides an explanation of the basic PSO algorithm in Subsection A. The remaining sections describe each of the initialization strategies against which the proposed initialization strategy will be compared.

A. Particle Swarm Optimizer

PSO is a stochastic, population-based optimization algorithm proposed by Kennedy and Eberhart [5] which is based on the flocking behaviour of birds. Every particle in the swarm is a vector representing a candidate solution to the optimization problem. The individuals move or "fly" through the search space for a number of iterations, hopefully converging upon an optimal solution. The position of each particle in the search space changes with every iteration 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 initial position $\mathbf{x}_i^0$ are determined by the chosen initialization strategy.

The particle's velocity at iteration $t+1$ is influenced by the velocity at which the particle was moving in the previous iteration, the best position that the particle has discovered so far (the "personal best" position, denoted $\mathbf{y}_i^t$) and the best location discovered by other members of the particle's neighbourhood (denoted $\hat{\mathbf{y}}$). If the particle's neighbourhood is a proper subset of the swarm, then this location is called the "local best". If the particle's neighbourhood is the entire swarm, then the particle's social component makes use of the best position found thus far by the entire swarm, called the "global best". This paper uses the global best approach. The velocity is thus calculated according to

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

where v_{ij}^{t+1} denotes the velocity of particle i in dimension j at iteration $t+1$. 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 of particle i in the jth dimension and $\hat{y}_{ij}$ denotes the global best position of the swarm in the jth dimension.

B. Uniform Random Initialization

Usually [7], the initial positions of the particles are obtained by sampling from a Uniform distribution as follows

$$\mathbf{x}_i^0 \sim U(x_{min}, x_{max})^n \quad (3)$$

where n is the dimensionality of the search space and x_{max} and x_{min} denote the upper and lower bounds of the search space, respectively. A pseudo-random number generation is usually used to generate these positions.

C. Sobol Sequences

It has been found by Schoemann and Engelbrecht [18] that using quasi-random number generators to generate initial particle positions can improve PSO performance. Quasi-random number generators exhibit low discrepancy: for any given set, the proportion of points generated by a quasi-random number generator is approximately proportional to the measure of the set. In other words, the point set generated by such a sequence does not have gaps or clusters.

Quasi-random number generators thus ensure that the swarm is initially distributed evenly throughout the search space, providing the swarm with (relatively) good search space coverage.

The Sobol sequence is one such quasi-random sequence. Further information about the implementation of Sobol sequences is provided by Joe and Kuo in [10].

D. Centroidal Voronoi Tesselations

Richards and Ventura suggested Voronoi tesselations as a particle initialization strategy in [17]. Voronoi tesselations partition the search space into a number of small cells. These cells should be approximately the same size so that their centers are distributed uniformly through the search space.

Such cells are produced by a set of generators. Each generator is assigned all the points in the search space that are closer to that generator than to any of the other generators. In this way, the search space is partitioned into a number of cells, each around one of the generators. For centroidal Voronoi tesselations (CVT), the generators are at the center of their cells.

The CVTs were calculated as described by Ju, et al [11]. Initially, the set of generators are chosen randomly within the search space. The positions of the generators are then adjusted over a number of iterations as follows: At every iteration, a number of sample points are chosen in the search space. Each sample point is allocated to the closest generator. Calculate $\mathbf{a}_i$, the centroid of the sample points assigned to the i_{th} generator. Then move every generator $\mathbf{g}_i$ closer to $\mathbf{a}_i$ by some fraction of the distance between $\mathbf{g}_i$ and $\mathbf{a}_i$.

The pseudo-code for the algorithm is provided below.

```
function CVT(s) returns a set of s initial points
    for all $i$ from 1 to $s$ do
        Choose $g_i$ randomly within search space as $i_{th}$
        generator
    end for
    for all $k$ from 1 to MAX_ITERATIONS do
        Choose a set $Q$ of q random points within search space
            for all $p$ in $Q$ do
        Find $g_i$, the generator closest to $p$
        Add $p$ to $G_i$, the subset of $Q$ containing the points that
        are closer to $g_i$ than any other generator
            end for
            for all $i$ from 1 to $s$ do
        Calculate $a_i$, the average of all the points in $G_i$
        Move $g_i$ closer to $a_i$ by some fraction of the distance
        between $g_i$ and $a_i$
            end for
    end for
end function
```

E. Nonlinear Simplex Method

Parsopoulos and Vrahatis [15], use the Nonlinear Simplex Method (NSM) to initialize particle swarms. The NSM was introduced by Nelder and Mead [12] as a means of function minimization. A n-dimensional simplex is a geometrical figure with $n+1$ vertices. The vertices of a simplex are initialized to be points in the search space, each with a corresponding score according to the fitness function. For a number of iterations, the simplex "walks" around the search space, each time moving the vertex with the lowest score to a point with a higher score. Every step that the simplex takes can be described in terms of reflections, contractions and expansions.

Let $P_0, \ldots, P_n$ denote the $n+1$ vertices of a simplex. Let the vertex with the best score be denoted by B. Let the vertex with the worst score be denoted by W. Also, let $\bar{P}$ denote the centroid calculated using all the vertices except W.

A reflection of P is defined by

$$P^{*} = (1+\alpha)\bar{P} - \alpha P \quad (4)$$

where α is a positive constant known as the reflection coefficient. An expansion of P is defined by

$$P^{**} = \gamma P + (1-\gamma)\bar{P} \quad (5)$$

where γ is known as the expansion coefficient and is a constant greater than one. A contraction of P is defined by

$$P^{***} = \beta P + (1-\beta)\bar{P} \quad (6)$$

where β is between zero and one and is known as the contraction coefficient.

At every iteration, W is reflected to W^*.

If the score of the new position is neither better than B nor worse than W, W is replaced by W^* and the iteration ends.
If the score of W^* is better than B, W^* is expanded to W^{**}.
If the score of W^{**} is better than B, then W^{**} replaces W, otherwise the expansion fails and W^* replaces W and the iteration ends.

Lastly, if W^* is worse than all the other points (except possibly W), then W is replaced by $min\{W, W^*\}$. The new W is then contracted to W^{***}.
If the score of W^{***} is better than W, it replaces W. Otherwise, the contraction fails and all P_i are replaced by $\frac{P_i - B}{2}$ and the iteration restarts.

The initial suggestion for using NSM applied it to problems where the swarm size, s, was larger than the problem dimension, n. In this case, the simplex was initialized randomly in the search space and allowed to step $s-(n+1)$ many times. The points through which the simplex's vertices moved were used as the initial particle positions.

However, the problems examined in this paper were of dimensionality far higher than that of the search space. The simplex was thus allowed to take s steps and the best s points through which the simplex's vertices had moved were selected as the initial particle positions.

III. Proposed Initialization Strategy

The proposed swarm initialization technique initializes the swarm in a small subspace of the entire search space. The swarm does not wander far from this initial subspace, thereby forcing the swarm to focus on exploitation within the small subspace rather than attempting to explore the expanse of the search space.

The initialization strategy makes use of a "seed set", which is simply a set of u, randomly generated, n-dimensional orthogonal unit vectors. The seed set is obtained by means of the Modified Gram Schmidt method (MGS) [14] where the initial set of linearly independent vectors are generated randomly. The smaller the seed set, the fewer linearly independent vectors are used to initialize the particle positions and personal bests and the smaller the subspace within which the swarm is initialized.

For completeness, a brief discussion of the Gram-Schmidt method follows. The Gram-Schmidt method receives u-many, n-dimensional vectors as input from which it generates a set of $\min\{u, n\}$ orthogonal vectors. The set of orthogonal vectors is generated stepwise as follows:

```
Given a set of $n$-dimensional input vectors, $\{\mathbf{v}_1, ..., \mathbf{v}_u\}$
for all $i$ from 1 to $u$ do
    $\mathbf{a}_i = \mathbf{v}_i - \sum_{j=1}^{i-1} \frac{<\mathbf{v}_i, \mathbf{a}_j>}{<\mathbf{a}_j, \mathbf{a}_j>} \mathbf{a}_j$
    $\mathbf{b}_i = \frac{\mathbf{a}_i}{||\mathbf{a}_i||}$
end for
return $\{\mathbf{b}_1, ..., \mathbf{b}_u\}$
```

In the algorithm above, $< \mathbf{a}, \mathbf{b} >$ denotes the inner product of the two vectors $\mathbf{a}$ and $\mathbf{b}$.

To produce the i_{th} orthogonal vector $\mathbf{b}_i$, the i_{th} seed set vector $\mathbf{v}_i$ is projected onto the subspace generated by all of the orthogonal vectors already calculated, $\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_{i-1}$. The orthogonal vector $\mathbf{b}_i$ is then defined to be the difference between $\mathbf{v}_i$ and its projection. Note that all the $\mathbf{b}_i$ for $i > n$ will be zero vectors.

Intuitively, the projection step finds the components of the seed vector that is shared by the orthogonal vectors calculated thus far. The $\mathbf{b}_i$ is then found by taking away all the shared components from the seed vector.

The Gram-Schmidt method as discussed above is susceptible to rounding errors. The Modified Gram Schmidt method computes the projection step differently so that the process is numerically stable. The implementation of the initialization technique thus uses the MGS method instead.

The initialization strategy generates an n-dimensional point which can be used either as a particle's initial position or as its initial personal best (see Section 4 for further details on how personal bests were chosen in the given experiments). The point is selected randomly on a line, L, that passes through the centre of the search space. The direction of the line, $\mathbf{d}$, is determined by the seed set.

The direction of the line is a random linear combination of all the vectors in the seed set, calculated as

$$\mathbf{d} = c_1\mathbf{b}_1 + c_2\mathbf{b}_2 + ... + c_u\mathbf{b}_u \tag{7}$$

where each $c_i \sim U(0, 1)$ and each $\mathbf{b}_i$ is an element from the seed set.

The point, $\mathbf{p}$, is generated on a line with the equation

$$\mathbf{p} = t\mathbf{d} + \mathbf{c} \tag{8}$$

where $\mathbf{c}$ denotes the centre of the search space and t is a scalar. The value for t is drawn from a uniform distribution with bounds (t_{min}, t_{max}). Let the bounds of the j-th dimension of the search space be $(x_{j,min}, x_{j,max})$. Then the bounds for t are calculated as follows

```
for all $j$ from 1 to $n$ do
    if $d_j == 0$ then continue
    end if
        ▷ Find the point $\mathbf{p}$, the intersection between $L$ and the maximum boundary of the $j_{th}$ dimension
    $t_1 = \frac{x_{j,max} - c_j}{d_j}$
    $\mathbf{p} = t_1\mathbf{d} + \mathbf{c}$
    if !WITHINBOUNDS($\mathbf{p}$, MARGIN) then continue
    end if
        ▷ Find the point $\mathbf{q}$, the intersection between $L$ and the minimum boundary of the $j_{th}$ dimension
    $t_2 = \frac{x_{j,min} - c_j}{d_j}$
    $\mathbf{q} = t_2\mathbf{d} + \mathbf{c}$
    if !WITHINBOUNDS($\mathbf{q}$, MARGIN) then continue
    end if
    return $t_{min} = min(t_1, t_2)$ and $t_{max} = max(t_1, t_2)$
end for
function WITHINBOUNDS($\mathbf{p}$, MARGIN)
    for all $i$ from 1 to $n$ do
        if $|p_i - x_{i,max}|$ or $|p_i - x_{i,min}| >$ MARGIN then return false
        end if
        end for
        return true
    end function
```

The algorithm is illustrated in Figure 1 for a 2-dimensional search space centred at $(0, 0)$. According to the diagram, the algorithm would return bounds $(t_{min}, t_{max}) = (\frac{y_{min}}{d_y}, \frac{y_{max}}{d_y})$.

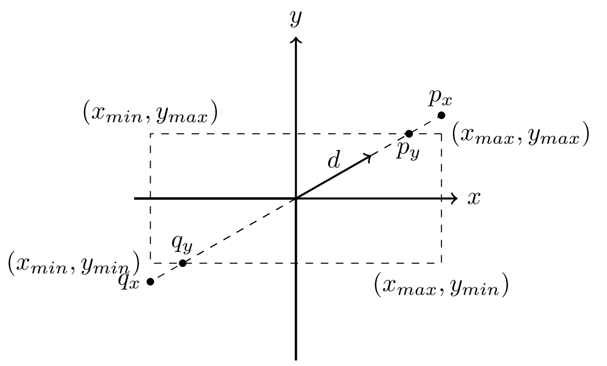

Fig. 1. Determining bounds for the scalar t in a 2-dimensional space centred at $(0, 0)$, given a direction vector $\mathbf{d}$

Since the relative sizes of the direction vector's components differ, the bounds (t_{min}, t_{max}) may define a very small range that causes the generated positions to be near the origin in all dimensions except the few for which the direction vector is relatively large. In order to prevent all of the generated points from being concentrated near the centre of the search space, the points were allowed to be generated outside of the search space in a given dimension by a small margin, denoted by *MARGIN* in the algorithm above.

IV. Experimental Procedure

This section describes the experimental procedure followed.

Each of the algorithms were evaluated on a benchmark suite of 22 base functions which are listed in the "Function Name" column of Table IV. Since the proposed initalization strategy initializes particles on lines that pass through the centre of the search space, it would not be representative to use benchmark functions with optima near the centre of the search space. Instead, many of the benchmark functions were also shifted by varying degrees so that the benchmark suite is not biased towards the proposed initialization strategy. A given function f was shifted and rotated to produce f^{ShR} according to

$$f(\mathbf{x})^{ShR} = f(R(\mathbf{x} - \gamma)) + \beta \tag{9}$$

where γ and β are constants specified in Table 1 and R is a randomly generated orthogonal matrix. The "Rotated" column in Table 1 indicates whether the function was rotated or not. This provided a total of 38 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 [8], [9] and [16]. The "Source" column of Table 1 lists the identifier of each function according to its source. Function i from [8] is denoted by f_i; function i from [9] is denoted by F_i and function i from [16] is denoted by G_i. All of the functions were minimized in $\{500, 1000, 2000\}$ dimensions.

The PSOs made use of synchronous updates and the star neighbourhood topology. Personal bests and the global best were only updated if they were within the bounds of the search space, thereby ensuring that the particles' attractors remained inside the search space. Velocity clamping was not used. Initial velocities were set to zero. The values of the velocity equation parameters were used as suggested by [6] to ensure convergence with $w = 0.729844$ and $c_1 = c_2 = 1.49618$.

The initial personal best and initial positions of the particles were initialized by generating two vectors within the search space (according to the chosen initialization strategy) and evaluating both. The vector with the better value for the objective function was assigned to the particle's personal best and the particle's initial position was set to the other vector.

The pseudo-random number generator used for the Uniform Random strategy was a Merzenne Twist. The quasi-random number generator used for the Sobol sequence initialization strategy was the implementation provided in the the Apache Commons Math package version 3.6 [1].

The CVT strategy was allowed to run for 5 iterations. With every iteration, 50000 sample points were generated as suggested in [17]. Every generator was moved 0.6 of the distance between it and the centroid of the sample points assigned to it. The NSM was implemented as described in the previous section with values $\alpha = 1.0$, $\gamma = 2.0$ and $\beta = 0.5$ as suggested in [13].

The proposed initialization strategy was implemented as described in Section 3. Particle positions were generated with a *MARGIN* of one tenth the range of the search space, $(0.1(x_{max} - x_{min}))$. Personal bests were generated with a *MARGIN* of zero to ensure that the particle's attractor remains inside the search space. Three different values for u were tested, namely $\{1, 5, 25\}$.

A swarm size of 25 was used in all cases. Each simulation was allowed to run for 2000 iterations. Every simulation was repeated on each benchmark function 30 times to achieve statistical significance. The performance of an algorithm on a given benchmark function was characterized in terms of the best fitness value attained in a given simulation. The swarm diversity was characterized by means of the average distance from the swarm centre as suggested in [13].

Friedman tests with a p-value of 0.05 were used to detect statistically significant differences among the algorithms' performance. If the Friedman test indicated significant differences, further pairwise comparisons were done by means of Mann-Whitney U tests with a p-value of 0.05.

V. Results and Discussion

This subsection provides the comparative results of the experiments as well as a detailed discussion of the observed outcomes.

Tables II, III and IV provides comparative results for the different initialization strategies for 500, 1000 and 2000 dimensional problems respectively. Each table may be interpreted as follows: The value of the cell in row number R and column

TABLE I. BENCHMARK FUNCTIONS

Function Name	Source	Domain Shift	Range Shift	Rotated	Function Name	Source	Domain Shift	Range Shift	Rotated
Absolute Value	f_1	0.0	0.0	No	Rastrigin Sh	f_{12}	0.0	0.0	Yes
Dixon-Price	F_{48}	0.0	0.0	No	Rastrigin ShR	f_{12}	2.0	-330.0	Yes
Egg Holder	f_4	512.0	0.0	No	Rosenbrock	f_{13}	0.0	0.0	No
Elliptic	f_5	0.0	0.0	No	RosenbrockSh	f_{13}	10.0	390.0	No
Elliptic Sh	f_5	10.0	-450.0	No	RosenbrockR	f_{13}	0.0	0.0	Yes
Elliptic R	f_5	10.0	-450.0	Yes	Salomon	f_{14}	0.0	0.0	No
Elliptic ShR	f_5	0.0	0.0	Yes	Schaffer 6	f_{15}	0.0	0.0	No
Griewank	f_6	0.0	0.0	No	Schaffer 6S hR	f_{15}	20.0	-300.0	Yes
Griewank Sh	f_6	10.0	-180.0	No	Schwefel	G_5	0.0	0.0	No
Griewank R	f_6	0.0	0.0	Yes	Schwefel 1.2	f_{16}	0.0	0.0	No
Griewank ShR	f_6	-60.0	-180.0	Yes	Schwefel 1.2 Sh	f_{16}	10.0	-450.0	No
HyperEllipsoid	f_7	0.0	0.0	No	Schwefel 1.2 R	f_{16}	0.0	0.0	Yes
Michalewicz	f_8	0.0	0.0	No	Schwefel 2.21	f_{19}	0.0	0.0	No
Norwegian	f_9	0.0	0.0	No	Spherical	f_{22}	0.0	0.0	No
Powell Singular 2	F_{92}	5.0	0.0	No	Spherical Sh	f_{22}	0.0	0.0	No
Quadric	f_{10}	0.0	0.0	No	Step	f_{23}	0.0	0.0	No
Quartic	f_{11}	0.0	0.0	No	Vincent	f_{24}	0.0	0.0	No
Rastrigin	f_{12}	0.0	0.0	No	Weierstrauss	f_{25}	0.0	0.0	No
Rastrigin R	f_{12}	2.0	-330.0	No	Weierstrauss Sh	f_{25}	1.0	-130.0	No

TABLE II. COMPARISON WITH SUBSPACE-BASED PSO ON $n = 500$

	$>$ Subspace-u1	=	$<$Subspace-u1
Uniform Random	0	0	38
Sobol Sequence	21	1	16
CVT	3	7	28
NSM	0	0	38

TABLE III. COMPARISON WITH SUBSPACE-BASED PSO ON $n = 1000$

	$>$ Subspace-u1	=	$<$Subspace-u1
Uniform Random	0	0	38
Sobol Sequence	2	5	31
CVT	6	9	23
NSM	0	0	38

number C reflects the number of benchmark functions for which the algorithm listed in row R performed significantly better than the algorithm in column C in terms of the best score attained by the swarm, averaged across all runs. In all cases, the subspace-based initialization strategy with a seed set of size 1 was used for comparison.

Note that for the lower dimensional problems (500 dimensions), the Sobol sequence is the only strategy that performed better than the subspace-based strategy - the rest almost never performed significantly better than the subspace-based strategy. In contrast, for the higher dimensional problems (1000 and 2000 dimensions), the only initialization strategy in competition with the subspace-based strategy was CVT and even then the subspace-based strategy outperformed CVT for more than half of the benchmark problems.

The observed behaviour can be explained by the hypothesis that for high dimensional problems, initializing particles evenly throughout the search space may not be beneficial. As explained in Section III, it is proposed that forcing the swarm to search within a small part of the search space may yield better results than attempting to explore the entire search space.

TABLE IV. COMPARISON WITH SUBSPACE-BASED PSO ON $n = 2000$

	$>$ Subspace-u1	=	$<$Subspace-u1
Uniform Random	0	0	38
Sobol Sequence	2	2	34
CVT	8	8	22
NSM	0	1	37

The Sobol sequence initialization strategy ensures that the particle positions are initialized in such a way that the particles do not cluster together and that there are no large gaps between them initially. This appears to be a useful strategy in the lower dimensional cases. However, as the problem dimensionality increases, the swarms are no longer able to explore the exponentially growing search space effectively and the swarm's performance deteriorates. A typical profile of best score achieved by the swarm per iteration is shown in Figure 2. The figure shows that the best solution found by the Sobol sequence was discovered within the first few iterations and then never improved upon; the good solutions found by the swarm were thus not due to gradual exploration of the search space, but rather because the solution found by the swarm was serendipitously near one of the swarm's initial points. Such propitious initialization becomes less likely as the problem dimensionality increases.

Since the Sobol sequence swarm and the uniform random PSO swarm are initialized evenly throughout the search space, there will be large distances between the initial positions of the particles. Although the NSM is not initialized evenly throughout the search space, the initial particle positions are the s points with the highest scores through which the n dimensional simplex passes. In high dimensions, this becomes equivalent to a brute-force approach for finding a favourable region in the search space, since the number of points generated by the simplex's iterative walk is much smaller than the number of its initial vertices ($n + 1$), which are chosen randomly. The distance between the initial particle positions are thus also quite large. This is supported by Figure 3.

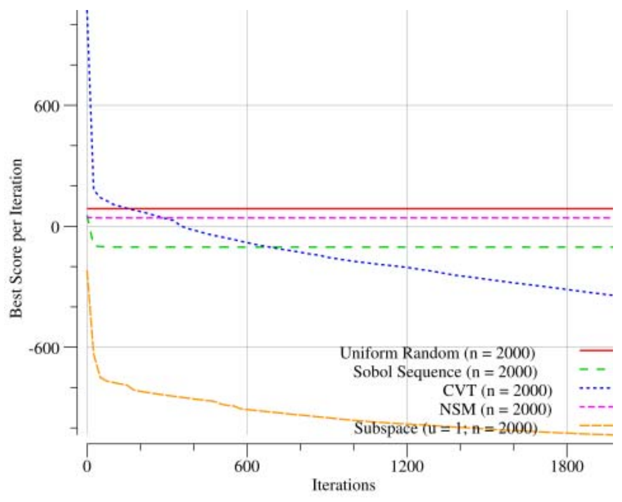

Fig. 2. Best Score per Iteration of Uniform Random, Sobol Sequence, CVT, NSM and Subspace PSOs on Vincent for 2000 Dimensions over 2000 Iterations

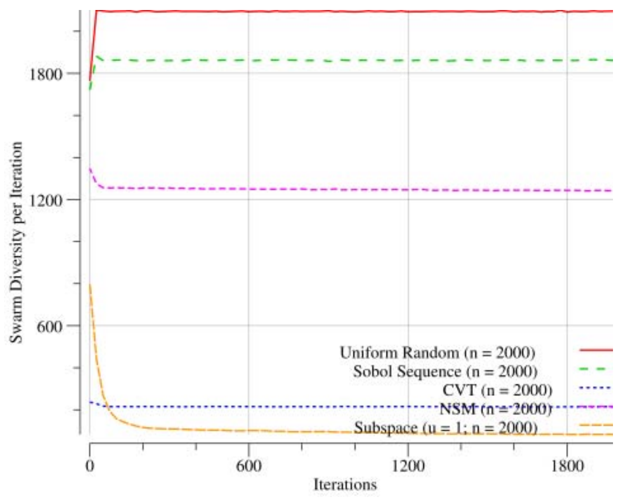

Fig. 3. Swarm Diversity per Iteration of Uniform Random, Sobol Sequence, CVT, NSM and Subspace PSOs on Elliptic, Rotated for 2000 Dimensions over 2000 Iterations

The momentum of the particles will thus be fairly large, due to the large initial distances between the particles. Even if a good solution is found, the particles will be unable to exploit the surrounding area effectively due to their large velocities, which will prevent fine-grained searching.

The argument above is supported by Figures 2 and 3. Figure 2 shows that for the uniform random PSO and then NSM PSO, the best score per iteration fails to improve after initialization and, for the Sobol sequence PSO, the score improves very little after initialization. Additionally, as illustrated in Figure 3, after an initial drop, the average diversity of the swarm stays relatively high for the remainder of the search. This implies that the swarms converge to a local minimum that is found within the first few iterations of the search. However, the sustained high diversity indicates that the swarms were unable to exploit further within the discovered region.

Both CVT and the subspace-based method caused particles to be initialized closer together, thereby giving the swarm a much lower initial diversity than the other methods. The momentum components of the particles' velocities would thus be smaller, enabling the particles to perform fine-grained exploration in a selected region. The reasons for the small initial swarm diversities exhibited by the subspace-based and CVT methods are explained below.

The CVT method makes use of sample points to determine the cell centroids and to ensure that they are distributed evenly throughout the search space. However, the number of sample points remain fixed regardless of the problem dimensionality. For high dimensional problems, the chosen number of sampling points are not enough to ensure even distribution of the cell centroids, particularly if the number of iterations also remains fixed. This, in turn, led to a relatively low initial swarm diversity (see Figure 4).

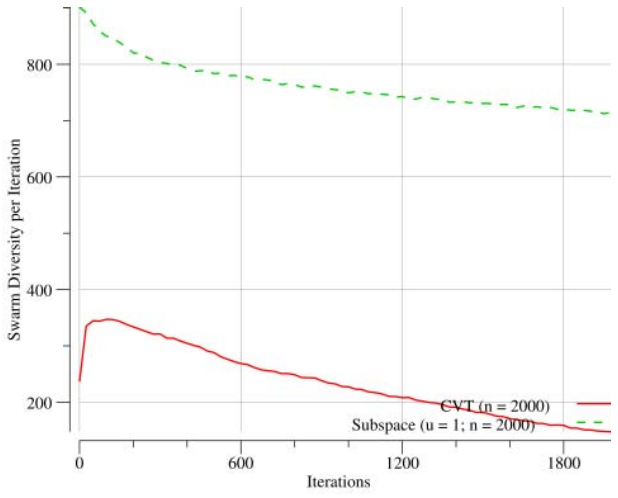

Fig. 4. Swarm Diversity per Iteration of CVT and Subspace PSOs on Schaffer6 for 2000 Dimensions over 2000 Iterations

The subspace method also causes the swarm to have a low initial diversity due to the way in which the scalar t is chosen. Although t- the scalar by which the direction vector $\mathbf{d}$ is scaled - is chosen randomly, $t\mathbf{d}$ must still be approximately within the bounds of the search space. If the value of d is relatively large for only one dimension, this will force t to be small, thereby confining the particle position to a smaller initial space in all the other dimensions.

As illustrated in Figure 5 and explained above, a low initial diversity may have been an advantage. In many of the cases that the CVT method outperforms the subspace-based method, the CVT swarm started with a lower initial diversity than the subspace-based swarm.

There were, of course, also cases in which the subspace-based swarm outperformed the CVT method, even though the CVT had lower initial diversity. This shows that simply initializing particle unevenly is not an appropriate strategy. This may be because the CVT created position clusters in unfavourable areas - parts of the search space with no good solutions - or because the CVT left gaps in favourable areas of the search space. However, since the subspace within which the subspace-based PSOs are initialized is chosen randomly,

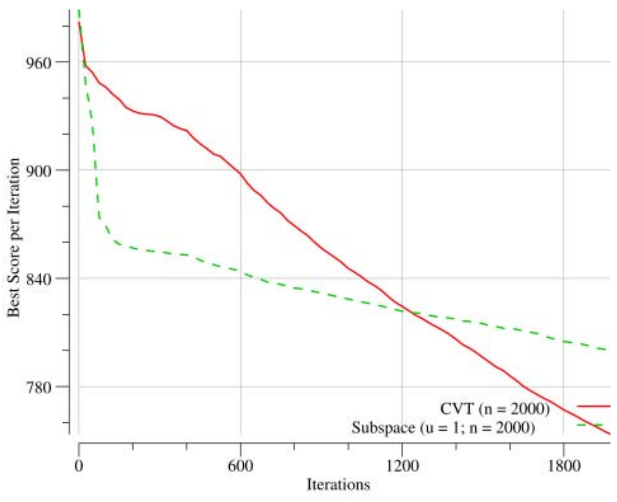

Fig. 5. Best Score per Iteration of CVT and Subspace PSOs on Schaffer6 for 2000 Dimensions over 2000 Iterations

TABLE V. COMPARISON AMONGST SUBSPACE INITIALIZATION STRATEGIES FOR VARYING SEED SET SIZES

	$>$ Subspace u=1	$=$	$<$Subspace u=1
Subspace u=5	3	35	0
Subspace u=25	11	26	1

it should be expected that the subspace may frequently be unfavourable and that the best score of the subspace-PSOs should have high standard deviation.

This was indeed seen by examining the standard deviations of the subspace-based PSOs for different values of u. As u is increased, the initial subspace in which the swarm is initialized becomes larger and so the chances of initialization in a poor region are slightly lower. Nevertheless, even for $u = 1$, the subspace-based PSO still exhibited lower standard deviations than the other initialization strategies for 28 of the 38 benchmark functions.

The swarms with different seed sizes performed very similarly for problems of lower dimensionality. For problems of dimensionality 2000, Table V shows that swarms with larger seed sets performed better than swarms with smaller seed sets. Swarms with larger seed sets are initialized in a larger subspace of the search space which decreases the chances of the particles being initialized in a poor region. However, even the largest seed size tested is still initialized in a much smaller subspace than the other initialization strategies, thereby allowing the swarm to focus on exploring a smaller part of the search space.

VI. CONCLUSION

The paper introduced a novel particle swarm initialization strategy that is particularly suited to high dimensional problems. The proposed initialization strategy forces the swarm to focus on exploration within a small subspace of the search space, rather than attempting to explore the entire search space. The proposed initialization strategy is compared to a number of other initialization strategies on problems of high dimensionality.

It was found that the proposed strategy performed much better than uniform random initialization and NSM. The proposed strategy was outperformed by Sobol sequences for the lower dimensionality problems, but in turn performed far better than CVT for the higher dimensional problems. The only strategy that performed comparably to the proposed subspace-based strategy was CVT. However, the proposed strategy still performed significantly better than CVT on more than half of the benchmark suite for all problem dimensionalities, though the two strategies performed more similarly at higher dimensions.

In addition, the effect of different seed set sizes - the number of random, linearly independent vectors used to generated the initial particle positions - was briefly examined. It was found that PSOs initialized with different seed set sizes behaved similarly at dimensions below 2000. At higher dimensions, PSOs with larger seed sizes performed better.

The paper proposes that low initial swarm diversity may be beneficial to the swarm's searching ability in high dimensional problems because the particles will have lower momentum, enabling the swarm to perform more fine-grained exploration within the initialized region.

It is suggested that optimization of high dimensional problems should be focused more on finding good local minima within a small region of the search space, rather than attempting to trawl the entire space in search of a global minimum. Although the shift in focus to exploitation rather than exploration may seem counter-intuitive, the results indicate that it may be a fruitful approach.

Further research may attempt to generalize these results to other population-based algorithms such as differential evolution or evolutionary programming. Additionally, further research may attempt to investigate the importance of particle momentum for high dimensional problems; techniques such as velocity clamping or gradually decreasing inertia weights may allow swarms to perform effective large-scale as well as fine-grained exploration.

Future work may also investigate the advantages the initialization strategy may offer with regard to niching problems. Slight modifications to the initialization strategy may allow different swarms to be initialized in othogonal subspaces within the search space, thereby allowing each swarm to discover a different local optimum.

REFERENCES

[1] The Apache Software Foundation. *Commons Math: The Apache Commons Mathematics Library*, 2015.

[2] T. Blackwell. Particle swarm optimization in dynamic environments. In S. Yang, Y.-S. Ong, and Y. Jin, editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, volume 51 of *Studies in Computational Intelligence*, pages 29–49. Springer Berlin Heidelberg, 2007.

[3] R. Brits, A. P. Engelbrecht, and F. van den Bergh. A niching particle swarm optimizer. In *Proceedings of the Conference on Simulated Evolution And Learning*, pages 692–696, 2002.

[4] C. A. Coello Coello and M. Lechuga. Mopso: a proposal for multiple objective particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation*, volume 2, pages 1051–1056, 2002.

[5] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, Oct 1995.

[6] R. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the 2000 Congress on Evolutionary Computation*, volume 1, pages 84–88 vol.1, 2000.

[7] A. P. Engelbrecht. *Computational Intelligence, An Introduction*, page 297. John Wiley & Sons, Ltd, 2nd edition, 2007.

[8] A. P. Engelbrecht. Particle swarm optimization: Global best or local best? In *Proceedings of the 11th Brazilian Congress on Computational Intelligence*, pages 124–135, Sept 2013.

[9] M. Jamil and X.-S. Yang. A literature survey of benchmark functions for global optimization problems. *Int. Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150194, 2013.

[10] S. Joe and F. Y. Kuo. Remark on algorithm 659: Implementing sobol's quasirandom sequence generator. *Association for Computing Machinery Trans. Math. Softw.*, 29(1):49–57, Mar. 2003.

[11] L. Ju, Q. Du, and M. Gunzburger. Probabilistic methods for centroidal voronoi tessellations and their parallel implementations. *Parallel Computing*, 28(10):1477 – 1500, 2002.

[12] J. A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7(4):308–313, 1965.

[13] O. Olorunda and A. P. Engelbrecht. Measuring exploration/exploitation in particle swarms using swarm diversity. In *IEEE Congress on Evolutionary Computation*, pages 1128–1134, June 2008.

[14] C. C. Paige, M. Rozlozník, and Z. Strakos. Modified gram-schmidt (mgs), least squares, and backward stability of mgs-gmres. *Society for Industrial and Applied Mathematics Journal Matrix Anal. Appl.*, 28(1):264–284, May 2006.

[15] K. E. Parsopoulos and M. N. Vrahatis. Initializing the particle swarm optimizer using the nonlinear simplex method. In *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, pages 216–221. WSEAS Press, 2002.

[16] F. Ramezani and S. Lotfi. The modified differential evolution algorithm (mdea). In J.-S. Pan, S.-M. Chen, and N. Nguyen, editors, *Intelligent Information and Database Systems*, volume 7198 of *Lecture Notes in Computer Science*, pages 109–118. Springer Berlin Heidelberg, 2012.

[17] M. Richards and D. Ventura. Choosing a starting configuration for particle swarm optimization. In *Proceedings of the IEEE International Joint Conference on Neural Networks*, volume 3, pages 2309–2312 vol.3, July 2004.

[18] I. Schoeman and A. P. Engelbrecht. Effect of particle initialization on the performance of particle swarm niching algorithms. In M. Dorigo, M. Birattari, G. Di Caro, R. Doursat, A. P. Engelbrecht, D. Floreano, L. Gambardella, R. Gro, E. ahin, H. Sayama, and T. Sttzle, editors, *Swarm Intelligence*, volume 6234 of *Lecture Notes in Computer Science*, pages 560–561. Springer Berlin Heidelberg, 2010.

[19] D. Sedighizadeh and E. Masehian. Particle swarm optimization methods, taxonomy and applications. *International Journal of Computer Theory and Engineering*, 1(4), Oct 2009.

[20] F. van den Bergh and A. P. Engelbrecht. A cooperative approach to particle swarm optimization. *IEEE Transaction on Evolutionary Computation, IEEE Transactions on*, 8(3):225–239, June 2004.