Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)

Defining a Standard for Particle Swarm Optimization

Daniel Bratton
Department of Computing
Goldsmiths College
University of London
London, UK
Email: dbratton@gmail.com

James Kennedy
US Bureau of Labor Statistics
Washington DC, USA
Email: Kennedy.Jim@bls.gov

Abstract— **Particle swarm optimization has become a common heuristic technique in the optimization community, with many researchers exploring the concepts, issues, and applications of the algorithm. In spite of this attention, there has as yet been no standard definition representing exactly what is involved in modern implementations of the technique. A standard is defined here which is designed to be a straightforward extension of the original algorithm while taking into account more recent developments that can be expected to improve performance on standard measures. This standard algorithm is intended for use both as a baseline for performance testing of improvements to the technique, as well as to represent PSO to the wider optimization community.**

I. Introduction

In the years since the introduction of particle swarm optimization (PSO) as a new method for global optimization [1], [2], many researchers have expanded on the original idea with alterations ranging from minor parameter adjustments to complete reworkings of the algorithm. Others have used PSO for comparison testing of other global optimization algorithms, including genetic algorithms and differential evolution [3], [4]. The PSO field has expanded dramatically since its inception, but to this point there has been little to no consensus as to what constitutes the "standard" or "canonical" PSO algorithm. Despite regular usage of the term, the actual implementation of this undefined standard varies widely between publications. Also troublesome is the fact that many of the variations on the particle swarm algorithm that are used for comparison testing do not take into account some of the major developments that have taken place since the original algorithm was proposed.

This lack of cohesion is understandable to an extent - despite its simplicity, the large number of minor variations on the PSO algorithm have led to such a wide assortment of choices that it is often difficult to determine which version can be expected to give the best or most relevant performance for a given research question. Some variations show improved performance on a specific class of problems, while others may have been adapted for use on an entirely different type of search space.

Due to the often overwhelming amount of choice, many researchers have settled for using minor variations on the form of the algorithm as originally proposed over a decade ago. Unfortunately, this original algorithm is no longer in any way representative of the state-of-the-art in PSO and has not been for some time. This paper examines the major improvements that have taken place in the field over the past decade and defines a standard for PSO that takes these into account. It is important to note that this definition should not be viewed as the optimal configuration for PSO across all problem sets, but rather as an evolution of the original algorithm designed to take advantage of subsequent generally applicable improvements. Having a well-known, strictly-defined standard algorithm provides a valuable point of comparison which can be used throughout the field of research to better test new advances.

II. Original PSO

The original PSO algorithm was inspired by the social behavior of biological organisms, specifically the ability of groups of some species of animals to work as a whole in locating desirable positions in a given area, e.g. birds flocking to a food source. This seeking behavior was associated with that of an optimization search for solutions to non-linear equations in a real-valued search space.

In the most common implementations of PSO, particles move through the search space using a combination of an attraction to the best solution that they individually have found, and an attraction to the best solution that any particle in their *neighborhood* has found. In PSO, a neighborhood is defined for each individual particle as the subset of particles which it is able to communicate with. The first PSO model used a Euclidian neighborhood for particle communication, measuring the actual distance between particles to determine which were close enough to be in communication. This was done in imitation of the behavior of bird flocks, similar to biological models where individual birds are only able to communicate with other individuals in the immediate vicinity [5], [6]. The Euclidian neighborhood model was abandoned in favor of less computationally intensive models when research focus was shifted from biological modeling to mathematical optimization. Topological neighborhoods unrelated to the locality of the particle came into use, including what has come to be known as a global neighborhood, or *gbest* model, where

1-4244-0708-7/07/$20.00 ©2007 IEEE

each particle is connected to and able to obtain information from every other particle in the swarm.

An individual particle i is composed of three vectors: its position in the *D*-dimensional search space $\overrightarrow{x}_i = (x_{i1}, x_{i2}, ..., x_{iD})$, the best position that it has individually found $\overrightarrow{p}_i = (p_{i1}, p_{i2}, ..., p_{iD})$, and its velocity $\overrightarrow{v}_i = (v_{i1}, v_{i2}, ..., v_{iD})$. Particles were originally initialized in a uniform random manner throughout the search space; velocity is also randomly initialized.

These particles then move throughout the search space by a fairly simple set of update equations. The algorithm updates the entire swarm at each time step by updating the velocity and position of each particle in every dimension by the following rules:

$$v_{id} = v_{id} + c\epsilon_1 (p_{id} - x_{id}) + c\epsilon_2 (p_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + v_{id} \quad (2)$$

where in the original equations c is a constant with the value of 2.0, ϵ_1 and ϵ_2 are independent random numbers uniquely generated at every update for each individual dimension $d = 1$ to D, and $\overrightarrow{p}_g$ is the best position found by any neighbor of the particle. The update process is summarized in Algorithm 1.

Algorithm 1 The PSO update process

```
for each time step t do
  for each particle i in the swarm do
    update position x_t using eqs 1 & 2
    calculate particle fitness f(x_t)
    update p_i, p_g
  end for
end for
```

Particle velocities in this original algorithm were clamped at a maximum value *vmax*. Without this clamping in place the system was prone to entering a state of *explosion*, wherein the random weighting of the ϵ_1 and ϵ_2 values caused velocities and thus particle positions to increase rapidly, approaching infinity. The *vmax* parameter prevented the system from entering this state by limiting the velocity of all particles to that value.

III. Advances and Defining a Standard

A. Swarm Communication Topology

The *lbest* swarm model, often referred to as a *local topology*, constitutes perhaps the most significant variation to the original PSO algorithm, and was in fact proposed in one of the very first PSO publications [2]. For one reason or another this topology has not been widely used in the community up to this point. Original investigations into this model using the original PSO algorithm showed inferior performance when compared to the global *gbest* model, or *global topology*, shown in Figure 1(a), but more recent research has revealed that lbest swarms return improved results across many standard problem sets when used in conjunction with other improvements to the algorithm [7].

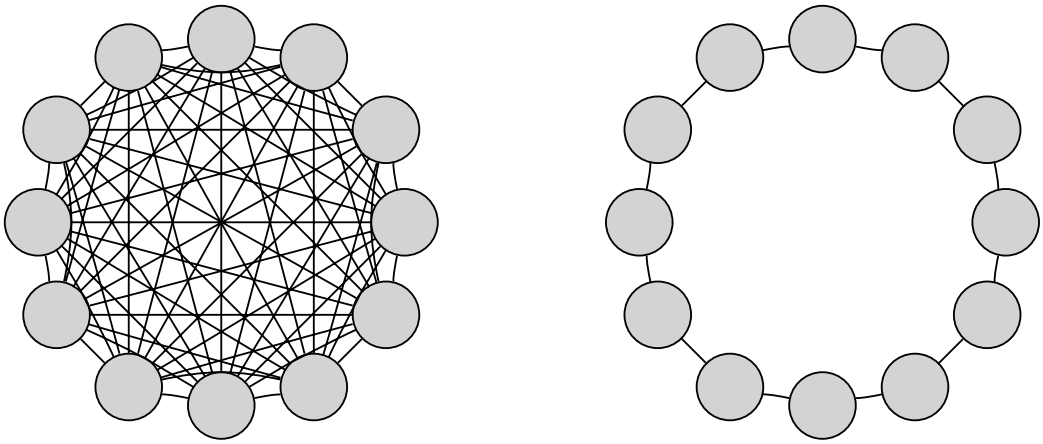
Much of the PSO literature uses the term *local topology* to describe not just a single swarm model, but applies it to any swarm model without global communication. A number of different limited communication topologies have been tested with varying results; the lbest model used here is perhaps the simplest form of a local topology, what is known as the *ring* model, shown in Figure 1(b). The lbest ring model connects each particle to only two other particles in the swarm, in contrast to the gbest model where every particle is able to obtain information from the very best particle in the entire swarm population.

The advantage of the lbest model appears to lie in its slower *convergence* rate relative to the gbest model. Having convergence take place when the swarm has found the global optimum is obviously beneficial, but when the converged-upon location is suboptimal, it is referred to as "premature" and is undesirable as it prevents the algorithm from escaping from an inferior local optimum.

Ironically, it is the slower rate of convergence of the lbest model that is most responsible for the general disregard of it as an alternative up to this point. The much faster convergence of the gbest model seems to indicate that it produces superior performance, but this is misleading. While results for the gbest model are indeed superior for many problems relatively early in the optimization process, the best found fitness for the lbest model quite often surpasses that of the gbest after some number of function evaluations have been performed, particularly on multimodal problems.

Despite the advantages of a local topology, it is important to note that it should not always be considered to be the optimal choice in all situations. The faster convergence rate of a global topology will usually result in better performance on simple unimodal problems than that of any local topology due to the lack of any danger of convergence on a suboptimal local minima. Even on some very complex multimodal problems a gbest model swarm can deliver performance competitive with the lbest model, given proper circumstances. For thorough, rigorous results tests should be run using both topologies, but in the interest of having a single standard PSO algorithm, the superior performance of the lbest model over the majority of benchmarks qualifies it as the better choice for cases where a straightforward means of comparison is desired. In any case, modern research performed using only swarms with a global topology is incomplete at best.

The inclusion of the local ring topology as part of a standard algorithm for particle swarm optimization comes with a caveat, however. Given the slower convergence of the lbest model, more function evaluations are required for the improved performance to be seen. This is especially important on unimodal functions, where the fast convergence of the gbest model combined with a single minima in the feasible search space results in quicker performance than that of the lbest swarm with its limited communication.



(a) The gbest topology

(b) The lbest ring topology

Fig. 1. Particle Swarm Topologies

B. Inertia Weight and Constriction

A few years after the initial PSO publications, a new parameter was introduced in an effort to strike a better balance between global exploration and local exploitation while avoiding clamping the particle velocity through the *vmax* method, which was viewed as both artificial and difficult to balance [8]. A single value for *vmax* was not necessarily applicable to all problem spaces - very large spaces required larger values to ensure adequate exploration, while smaller spaces required very small values to prevent explosion-like behavior on their scale. Finding the appropriate value for *vmax* for the problem being solved was critical, as a poorly-chosen *vmax* could result in extremely poor performance, yet there was no simple, reliable method for choosing this value beyond trial and error.

This new *inertia weight* parameter w was designed to replace *vmax* by adjusting the influence of the previous particle velocities on the optimization process. The velocity update equation was altered to the form:

$$v_{id} = wv_{id} + c_1\epsilon_1 \left(p_{id} - x_{id}\right) + c_2\epsilon_2 \left(p_{gd} - x_{id}\right) \quad (3)$$

By adjusting the value of *w*, the swarm has a greater tendency to eventually constrict itself down to the area containing the best fitness and explore that area in detail. The authors also suggested using *w* as a dynamic value over the optimization process, starting with a value greater than 1.0 to encourage early exploration, and decreasing eventually to a value less than 1.0 to focus the efforts of the swarm on the best area found in the exploration. This control of the "inertia" of particles, similar to the role of friction in a physical setting, resulted in improved performance of the algorithm and removed the need for velocity limiting.

Another method of balancing global and local searches known as *constriction* was being explored simultaneously with the inertia weight method and was occasionally referenced in PSO literature, though the actual research proposing its use was not published until some time later [9]. Similar to the inertia weight method, this method introduced a new parameter χ, known as the constriction factor. χ is derived from the existing constants in the velocity update equation:

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \varphi = c_1 + c_2 \quad (4)$$

It was found that when $\varphi < 4$, the swarm would slowly "spiral" toward and around the best found solution in the search space with no guarantee of convergence, while for $\varphi > 4$ convergence would be quick and guaranteed. While it is possible to weight the velocity update equation to favor the best position of the individual particle p_i or the best position of the entire swarm p_g by adjusting the values of c_1 and c_2, for the sake of simplicity most implementations of constricted particle swarms use equal values for both parameters. Using the constant $\varphi = 4.1$ to ensure convergence, the values $\chi \approx 0.72984$ and $c_1 = c_2 = 2.05$ are obtained. This constriction factor is applied to the entire velocity update equation:

$$\overrightarrow{v}_{id} = \chi \left(v_{id} + c_1\epsilon_1 \left(p_{id} - x_{id}\right) + c_2\epsilon_2 \left(p_{gd} - x_{id}\right)\right) \quad (5)$$

The effects are similar to those of inertia weight, resulting in swarm behavior that is eventually limited to a small area of the feasible search space containing the best known solution. A comparison study of the two methods demonstrated that the PSO algorithm with constriction is in fact a special case of the algorithm with inertia weight in which the values for the parameters have been determined analytically [10]. The parameter values noted above are preferred in most cases when using constriction for modern PSOs due to the proof of stability detailed in [9].

C. Initialization and Boundary Conditions

It has been suggested that many optimization algorithms have what is described as a *bias* toward some area of the space in which the population is initialized. Research has shown that

TABLE I
BENCHMARK FUNCTIONS

Equation	Name	D	Feasible Bounds
$f_1 = \sum_{i=1}^{D} x_i^2$	Sphere/Parabola	30	$(-100, 100)^D$
$f_2 = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	Schwefel 1.2	30	$(-100, 100)^D$
$f_3 = \sum_{i=1}^{D-1} \left\{ 100 \left(x_{i+1} - x_i^2 \right)^2 + (x_i - 1)^2 \right\}$	Generalized Rosenbrock	30	$(-30, 30)^D$
$f_4 = -\sum_{i=1}^{D} x_i \sin\left(\sqrt{x_i}\right)$	Generalized Schwefel 2.6	30	$(-500, 500)^D$
$f_5 = \sum_{i=1}^{D} \left\{ x_i^2 - 10\cos(2\pi x_i) + 10 \right\}$	Generalized Rastrigin	30	$(-5.12, 5.12)^D$
$f_6 = -20\exp\left\{ -0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2} \right\} - \exp\left\{ \frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i) \right\} + 20 + e$	Ackley	30	$(-32, 32)^D$
$f_7 = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	Generalized Griewank	30	$(-600, 600)^D$
$f_8 = \frac{\pi}{D}\left\{ 10\sin^2(\pi y_i) + \sum_{i=1}^{D-1} (y_i - 1)^2 \left\{ 1 + 10\sin^2(\pi y_{i+1}) \right\} + (y_D - 1)^2 \right\}$ $+ \sum_{i=1}^{D} \mu(x_i, 10, 100, 4)$ $y_i = 1 + \frac{1}{4}(x_i + 1)$ $\mu(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	Penalized Function P8	30	$(-50, 50)^D$
$f_9 = 0.1\left\{ \sin^2(3\pi x_i) + \sum_{i=1}^{D-1} (x_i - 1)^2 \left\{ 1 + \sin^2(3\pi x_{i+1}) \right\} + (x_D - 1)^2 \times \left\{ 1 + \sin^2(2\pi x_D) \right\} \right\} + \sum_{i=1}^{D} \mu(x_i, 5, 100, 4)$	Penalized Function P16	30	$(-50, 50)^D$
$f_{10} = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	Six-hump Camel-back	2	$(-5, 5)^D$
$f_{11} = \left\{ 1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2 \right) \right\} \times \left\{ 30 + (2x_1 - 3x_2)^2 \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2 \right) \right\}$	Goldstein-Price	2	$(-2, 2)^D$
$f_{12} = -\sum_{i=1}^{5} \left\{ \sum_{j=1}^{4} (x_j - a_{ij})^2 + c_i \right\}^{-1}$	Shekel 5	4	$(0, 10)^D$
$f_{13} = -\sum_{i=1}^{7} \left\{ \sum_{j=1}^{4} (x_j - a_{ij})^2 + c_i \right\}^{-1}$	Shekel 7	4	$(0, 10)^D$
$f_{14} = -\sum_{i=1}^{10} \left\{ \sum_{j=1}^{4} (x_j - a_{ij})^2 + c_i \right\}^{-1}$	Shekel 10	4	$(0, 10)^D$

some algorithms can return superior performance when the global optimum of the problem being solved is located in or near the center of the area in which the swarm is initialized [11].

This is especially problematic in algorithm comparison: as a basic example, it is simple to construct an algorithm that instantly finds the point at the center of the feasible bounds through a single mathematical equation. If that point is also the global optimum, the simple algorithm would appear to show superior optimization performance to any other heuristic algorithm, when in actuality the algorithm's ability to find that optimum would be totally dependent upon it being located at the easily-calculated center of the search space. Benchmarking problems with the global optima at the center of the feasible bounds are common in optimization literature, so adjustment is necessary for accurate performance testing.

The common method for negating any centrist bias in an optimization algorithm is to *shift* the function when the optimum is defined at the center of the feasible search space. This is also known as the *center offset* method. Moving the optimum away from the point which the algorithm is supposedly biased toward eliminates any inappropriate advantage that may be gained.

Another proposed method of alleviating this potential bias is population initialization within a subspace of the entire feasible search space that does not contain the global optimum [12], often referred to as *region scaling*. Ensuring that the global optimum does not lie within this subspace forces the swarm to expand its search beyond its initial limits, and eliminates its ability to immediately converge to a single point without exploring the entire space to find the global optimum of the problem. This method is most applicable as a research standard for performance testing and algorithm comparison when both the problem and its optimum are well-known and understood, but for practical optimization applications it is unnecessary.

It can be shown that a bias toward the center can arise in algorithms similar to PSO when the trajectory of a particle is artificially limited when it reaches the boundary of the search space. To avoid any effect on the performance of the algorithm, the simplest and most straightforward method for handling particles which cross the boundary of the feasible search space is to leave their velocity and infeasible position unaltered. The fitness evaluation step is then skipped, thereby preventing the infeasible position from being set as a personal and/or global best. Using this method, particles outside the feasible search space will eventually be drawn back within the space by the influence of their personal and neighborhood bests. As it is possible under certain rare circumstances for particles to develop extremely high velocities when they are allowed to continue past the edge of the defined search space, the use of a very generous *vmax* value has been recommended

to keep the particle from going too far beyond the feasible region [10].

These boundary conditions, often referred to as "letting the particles fly", prevent this factor from contributing to any potential chance of a bias toward the center of the search space; non-uniform swarm initialization and shifting of a centrally-located optimum should substantially reduce the risk of a centrist bias altogether.

D. Number of Particles

Empirical results have shown that the number of particles composing the swarm can influence the resulting performance by a varying amount, depending on the problem being optimized. Some test functions show slightly improved performance as the size of the swarm is increased, while others tend to be better optimized by smaller swarms. There seems to be no definitive value for the swarm size that is optimal across all problems, so to avoid tuning the algorithm to each specific problem, a compromise must be reached.

While it may certainly be beneficial to tune this parameter based on the problem at hand, generally speaking it is of minor importance. Swarms of all sizes have been shown to perform acceptably on standard benchmarks. 50 particles were used in the tests presented here, as swarms of this size performed best by a very slight margin when averaged across the entire range of test problems. It should be taken into account, however, that this improved performance is tied directly to the benchmark that was used, and even then is still an average - no one value was clearly optimal on all problems. Full analysis and determination of the optimal swarm size is beyond the scope of this paper, but it can be reported that under testing, no swarm size between 20 - 100 particles produced results that were clearly superior or inferior to any other value for a majority of the tested problems.

E. Statistical Analysis of Results

Having gathered some empirical data, differences in performance between several versions of algorithms can become notable. One version may be more likely than another to reach a criterion, or the final function result after some number of function evaluations or iterations may be better, averaged over some number of trials. But is it significantly better - in other words, is there a real difference between the two versions, or did chance play the deciding role?

It is not necessary, in analyzing computer-generated results, to do anything especially complicated, difficult, or time-consuming. Many researchers have adopted the practice of performing t-tests on pairs of groups of data; these tests give a *p-value* which is compared to a constant called α to determine whether a difference is significant or not.

There is a problem, however, in conducting multiple significance tests. Because they are probabilistic, it is possible that some results are due simply to chance - even random data generated from the same distribution will differ "significantly" sometimes. Statisticians have addressed this problem in various ways. Corrections known in the literature include

TABLE II
OPTIMA AND INITIALIZATION RANGES FOR ALL FUNCTIONS

Function	Feasible Bounds	Optimum	Initialization
f_1	$(-100, 100)^D$	0.0^D	$(50, 100)^D$
f_2	$(-100, 100)^D$	0.0^D	$(50, 100)^D$
f_3	$(-30, 30)^D$	1.0^D	$(15, 30)^D$
f_4	$(-500, 500)^D$	420.9687^D	$(-500, -250)^D$
f_5	$(-5.12, 5.12)^D$	0.0^D	$(2.56, 5.12)^D$
f_6	$(-32, 32)^D$	0.0^D	$(16, 32)^D$
f_7	$(-600, 600)^D$	0.0^D	$(300, 600)^D$
f_8	$(-50, 50)^D$	-1.0^D	$(25, 50)^D$
f_9	$(-50, 50)^D$	1.0^D	$(25, 50)^D$
f_{10}	$(-5, 5)^D$	$(-0.0898, 0.7126)$, $(0.0898, -0.7126)$	$(2.5, 5)^D$
f_{11}	$(-2, 2)^D$	$(0, -1)$	$(1, 2)^D$
f_{12}	$(0, 10)^D$	4.0^D	$(7.5, 10)^D$
f_{13}	$(0, 10)^D$	4.0^D	$(7.5, 10)^D$
f_{14}	$(0, 10)^D$	4.0^D	$(7.5, 10)^D$

Duncan, Bonferroni, Sheffé, Fisher, and Tukey adjustments, among others. These approaches typically manipulate α or the p-value in some way that corrects for the likelihood of forming spurious conclusions due to chance.

Unfortunately, many of these corrections are too conservative - the researcher ends up rejecting good results simply because the adjustment was too severe. A good alternative is a modified Bonferroni procedure, which manipulates the α value in a way that protects against capitalization on chance, without penalizing the researcher [13].

A number of t-tests are conducted, and p-values recorded. These are then ranked from smallest to largest, and the ranks recorded. These ranks are then inverted, so that the highest-p-value gets an inverse-rank of 1, and the test with the lowest p-value gets an inverse-rank of N, the number of tests performed.

Next, α is divided by the inverse rank for each observation. Assuming the desired $\alpha = 0.05$, then the first observation received a new value of $0.05/N$, and so on, with the worst one receiving a new α of $0.05/1 = 0.05$.

The researcher then goes down the list, comparing each p-value to the new alpha until a nonsignificant result is found. If $p < \alpha$, then a test is reported as significant. When the first nonsignificant result is encountered, the subsequent observations are reported as nonsignificant.

This procedure is used to validate t-tests, which compare pairs of groups. In comparing more than two groups over a set of functions, it may be necessary to perform a great number of t-tests. In this case, it may be wise to reject certain tests at the beginning, for instance if one group is always worse than others, or if some means are identical. This cuts down on the number of tests and keeps α from becoming too small.

TABLE III
MEAN FITNESS AFTER 30 TRIALS OF 300000 EVALUATIONS

Algorithm	f_1	f_2	f_3	f_4	f_5	f_6	f_7
Original PSO	2.7562	5.4572	54.6867	4211	400.7194	20.2769	1.0111
(Std Err)	($\pm$0.0448)	($\pm$0.1429)	($\pm$2.8570)	($\pm$44)	($\pm$4.2981)	($\pm$0.0082)	($\pm$0.0031)
Constricted GBest	***0.0***	***0.0***	***8.1579***	3508	***140.4876***	17.6628	0.0308
	($\pm$0.0)	($\pm$0.0)	($\pm$2.7835)	($\pm$33)	($\pm$4.8538)	($\pm$1.0232)	($\pm$0.0063)
Constricted LBest	***0.0***	0.1259	12.6648	***3360***	144.8155	***17.5891***	***0.0009***
	($\pm$0.0)	($\pm$0.0178)	($\pm$1.2304)	($\pm$34)	($\pm$4.4066)	($\pm$1.0264)	($\pm$0.0005)
Algorithm	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}
Original PSO	8.8695	0.1222	1.7770	1.08e-007	5.0981	5.3152	5.4079
(Std Err)	($\pm$0.3829)	($\pm$0.0021)	($\pm$0.2886)	($\pm 2.66e-008$)	($\pm 6.40e-006$)	($\pm 6.53e-006$)	($\pm 6.19e-006$)
Constricted GBest	0.1627	0.0040	***0.0***	***0.0***	4.5882	4.4747	3.8286
	($\pm$0.0545)	($\pm$0.0016)	($\pm$0.0)	($\pm$0.0)	($\pm$0.2840)	($\pm$0.3744)	($\pm$0.4674)
Constricted LBest	***0.0***	***0.0***	***0.0***	***0.0***	***2.5342***	***1.0630***	***0.5409***
	($\pm$0.0)	($\pm$0.0)	($\pm$0.0)	($\pm$0.0)	($\pm$0.4708)	($\pm$0.3948)	($\pm$0.3013)

IV. RESULTS

Three PSO algorithms were compared in the interest of demonstrating the performance gains granted by improvements to the technique: the original 1995 algorithm, a constricted swarm using a global topology, and a constricted swarm using a local topology. All swarms were randomly initialized in an area equal to one quarter of the feasible search space in every dimension that was guaranteed not to contain the optimal solution. Initialization ranges for all functions can be found in Table II.

Each algorithm was run on an array of common benchmarks, shown in table I, for 300000 evaluations per function. These benchmarks were chosen for their variety - functions $f_1 - f_3$ are simple unimodal problems, $f_4 - f_9$ are highly complex multimodal problems with many local minima, and $f_{10} - f_{14}$ are multimodal problems with few local minima. Performance was measured as the minimum error $|f(x) - f(x^\star)|$ found over the trial, where $f(x^\star)$ is the optimum fitness for the problem. Results were averaged over 30 independent trials, and are displayed in Table III. Convergence graphs for selected functions are shown in Figure 2. In cases where a value was $< 10^{-8}$ it was rounded to 0.0 in order to accommodate reproduction using programming languages that may not include floating point precision at smaller values.

Statistical tests were performed on these results to show whether performance improvements are significant. As the performance of the original PSO algorithm is poorer than the others for all functions, tests were limited to comparisons of the two constricted swarm models over all functions. These results are shown in Table IV. The lbest swarm showed significantly superior performance over the gbest swarm for six of the fourteen test functions, while the gbest swarm was significantly superior on one unimodal function. There was no significant difference in performance between the two swarm models for the other seven test functions.

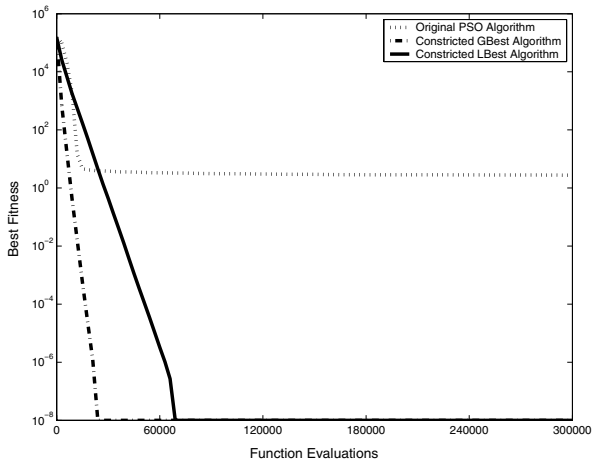
These results show that both the global and the local models of constricted swarms return improved performance over the original PSO algorithm. Further, it is clear that in many of the test cases, the lbest model swarm can be reliably expected to return better results than the gbest model swarm. It is important to note, however, than performance comparisons were done strictly on average performance over a fixed number of function evaluations. On the three problems where both of the constricted swarms found the global optimum, the number of evaluations required was not recorded.

TABLE IV
SIGNIFICANCE FOR GBEST VS LBEST

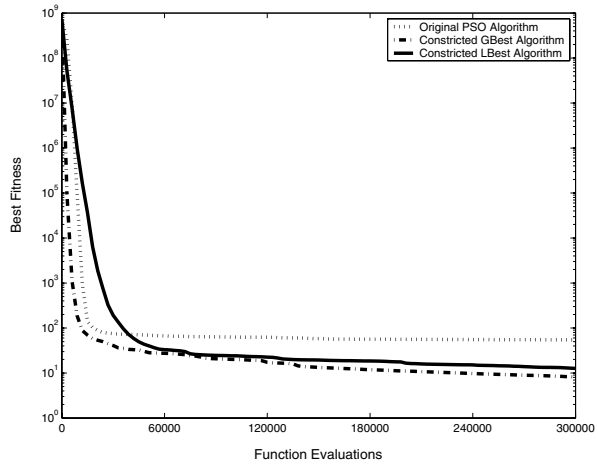
Func	**p-value**	**Rank**	**Inverse rank**	**New α**	**Significant**
f_2	0	1	14	0.003571	Yes
f_{13}	0	2	13	0.003846	Yes
f_{14}	0	3	12	0.004167	Yes
f_7	0.00002	4	11	0.004545	Yes
f_{12}	0.00043	5	10	0.005	Yes
f_4	0.002	6	9	0.005556	Yes
f_8	0.004	7	8	0.00625	Yes
f_9	0.016	8	7	0.007143	No
f_3	0.14	9	6	0.008333	No
f_5	0.51	10	5	0.01	No
f_6	0.96	11	4	0.0125	No
f_1	1	12	3	0.016667	No
f_{10}	1	13	2	0.025	No
f_{11}	1	14	1	0.05	No

V. CONCLUSION

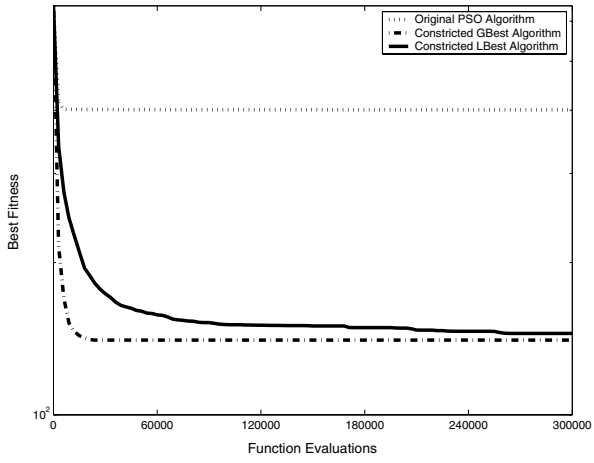
Since its relatively recent inception, PSO has spawned a considerable amount of research into modifications and variations on the original algorithm, many of which have been shown to be significant improvements on the original algorithm while still being generally applicable. Given the wide array of choice, several improvements which should have



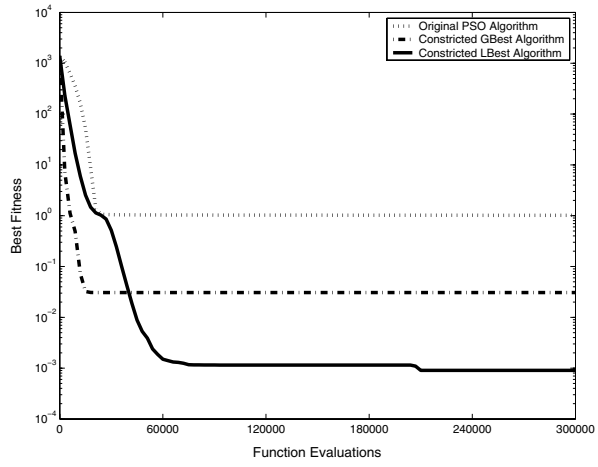
(a) f01 (Sphere/Parabola)



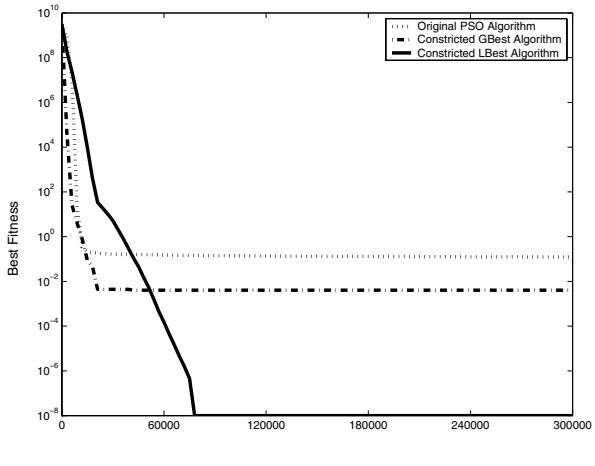
(b) f03 (Generalized Rosenbrock)



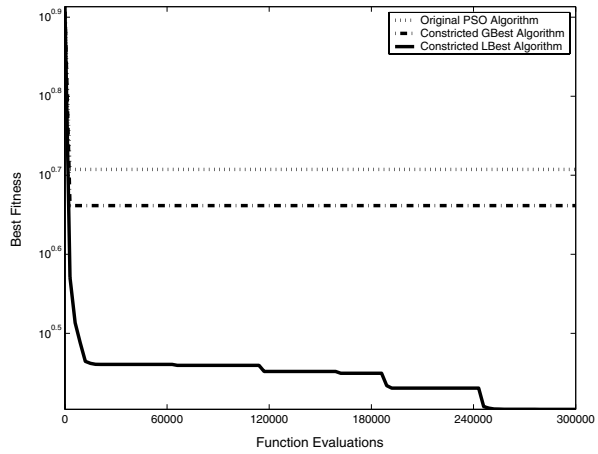
(c) f05 (Generalized Rastrigin)



(d) f07 (Generalized Griewank)



(e) f09 (Penalized Function P16)



(f) f12 (Shekel 5)

Fig. 2. Algorithm convergence for selected problems. Results are averaged over 30 runs.

become commonly known and used have not been universally adopted. By defining a standard for the field we are able to ensure that all research performed takes advantage of the same techniques.

As defined above, the standard PSO algorithm includes:

- a local ring topology,
- the constricted update rules in equations 5 and 2,
- 50 particles,
- non-uniform swarm initialization, and
- boundary conditions wherein a particle is not evaluated when it exits the feasible search space.

The intent of this definition is not to discourage exploration and alterations to the PSO algorithm, but rather to give researchers a common grounding to work from. This baseline will provide a means of comparison for future developments and improvements and will prevent unnecessary effort being expended on "reinventing the wheel" on rigorously tested enhancements that are being used at the forefront of the field.

To reiterate, this new definition for PSO should not be considered as the "best possible option" for all problem sets. There are a huge number of variations and hybridizations of the original algorithm that potentially offer better performance on certain problems. What is offered here is both a standard for algorithm comparison and a standard to represent modern PSO in the global optimization community.

ACKNOWLEDGMENT

Daniel Bratton's work is supported by EPSRC XPS grant GR/T11234/01

REFERENCES

[1] J. Kennedy and R. Eberhart (1995). Particle Swarm Optimization. *Proceedings of the 1995 IEEE International Conference on Neural Networks* (Perth, Australia): IEEE Service Center, Piscataway, NJ, IV: pp 1942-1948.

[2] R. Eberhart and J. Kennedy (1995). A New Optimizer using Particle Swarm Theory. *Proceedings of the Sixth International Symposium on Micro Machine and Human Science* (Nagoya, Japan): IEEE Service Center, Piscataway, NJ: pp 39-43.

[3] J. Vesterstrøm and R. Thomsen (2004). A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems. *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, Volume 2, pp. 1980 - 1987

[4] R. Eberhart and Y. Shi (1998). Comparison between Genetic Algorithms and Particle Swarm Optimization. *EP '98: Proceedings of the 7th International Conference on Evolutionary Programming VII*, pp. 611-616.

[5] C. W. Reynolds (1987). Flocks, Herds, and Schools: A Distributed Behavioral Model. *Computer Graphics*, Volume 21, pp. 25-34.

[6] F. Heppner and U. Grenander (1990). A stochastic nonlinear model for coordinated bird flocks. In S. Krasner, Ed., *The Ubiquity of Chaos*, AAAS Publications, Washington, DC.

[7] J. Kennedy and R. Mendes (2006). Neighborhood topologies in fully informed and best-of-neighborhood particle swarms. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, Volume 36, Issue 4, pp. 515-519.

[8] Y. Shi and R. Eberhart (1998). A Modified Particle Swarm Optimizer. *Proceedings of the 1998 IEEE Congress on Evolutionary Computation*, Anchorage, AK.

[9] M. Clerc and J. Kennedy (2002). The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1): pp. 58-73.

[10] R. Eberhart and Y. Shi (2000). Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization. *Proceedings of the 2000 IEEE Congress on Evolutionary Computation*, Volume 1, pp. 84-88.

[11] C. Monson and K. Seppi (2005). Exposing Origin-Seeking Bias in PSO. *Proceedings of the 2005 Genetic and Evolutionary Computation Conference*, pp 241-248.

[12] D. K. Gehlhaar and D. B. Fogel (1996). Tuning evolutionary programming for conformationally exible molecular docking. *Evolutionary Programming,* pp. 419-429.

[13] J. Jaccard and C. K. Wan (1996). *LISREL approaches to interaction effects in multiple regression*, Sage Publications, Thousand Oaks, CA.