Particle Swarm Optimization with Thresheld Convergence

Stephen Chen
School of Information Technology
York University
Toronto, Canada
sychen@yorku.ca

James Montgomery
Research School of Computer Science
Australian National University
Canberra, Australia
james.montgomery@anu.edu.au

***Abstract*—Many heuristic search techniques have concurrent processes of exploration and exploitation. In particle swarm optimization, an improved *pbest* position can represent a new more promising region of the search space (exploration) or a better solution within the current region (exploitation). The latter can interfere with the former since the identification of a new more promising region depends on finding a (random) solution in that region which is better than the current *pbest*. Ideally, every sampled solution will have the same relative fitness with respect to its nearby local optimum – finding the best region to exploit then becomes the problem of finding the best random solution. However, a locally optimized solution from a poor region of the search space can be better than a random solution from a good region of the search space. Since exploitation can interfere with subsequent/concurrent exploration, it should be prevented during the early stages of the search process. In thresheld convergence, early exploitation is "held" back by a threshold function. Experiments show that the addition of thresheld convergence to particle swarm optimization can lead to large performance improvements in multi-modal search spaces.**

Keywords—particle swarm optimization; thresheld convergence; niching; crowding; exploration; exploitation

I. INTRODUCTION

An attraction basin represents the region of a search space that will lead to a given (local) optimum when greedy local search is used. Let us define the fitness of an attraction basin as the fitness of the optimum within it. During the explorative phase(s) of a search technique, the goal is to find the fittest attraction basin. The exploitative phase(s) will then be responsible for finding the exact optimum within the initially discovered attraction basin. Since precise measurement of the fitness of an attraction basin is not possible without using local search to find the actual optimum, the fitness of a search point's attraction basin is often estimated by the fitness of the search point itself.

Particle swarm optimization (PSO) [1] can be viewed as a system with two populations: a population of current positions which search for better solutions, and a population of *pbest* positions which store the best found solutions. PSO can also be viewed as a system with two phases. During the initial phase when the system has large velocities, the current solutions focus more on exploration. Later, as velocities slow towards zero, the current solutions will focus more on exploitation. This exploitation will occur around the *pbest* positions, so the goal of the initial phase is to find *pbest* positions that are members of the fittest attraction basins.

In PSO, the fitness of an attraction basin is (implicitly) estimated by the fitness of a sample solution found from within that basin. Specifically, given two positions which represent two attraction basins, the position stored by *pbest* (which represents the most promising attraction basin to exploit during the later exploitative phase) will be the position with the better fitness. In order to improve the attraction basin represented by the *pbest* position, it is not sufficient to find a new position in a fitter attraction basin – it is necessary to find a position in a fitter attraction basin that is also fitter than the existing *pbest* position.

For an attraction basin represented by an "average" sample solution, it should be relatively easy to find a fitter sample solution from a fitter attraction basin. Consider a search space with attraction basins that have similar shapes and sizes (e.g. a sinusoid super-imposed over a linear slope). In this search space, the average fitness of a random solution is correlated to the fitness of its attraction basin (see Fig. 1). In particular, the expected difference in fitness between two random solutions from different attraction basins will be equal to the difference in fitness between the optima from these attraction basins. With "average" sample solutions, the task of finding the fittest

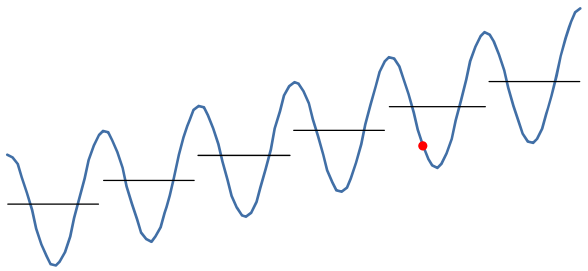

Fig. 1. The horizontal lines represent the average/expected fitness of random sample solutions in each attraction basin. If an attraction basin is represented by a better-than-average solution (see dot), a random solution from a fitter attraction basin may no longer have a better expected fitness.

attraction basin is equivalent to finding the fittest random solution. However, this task becomes more complicated if an existing attraction basin is represented by a better-than-average solution.

One way to find a better-than-average solution from an attraction basin is to perform local search. Starting from an initial solution, let us define any change that leads to a solution in a new attraction basin as an explorative/global search step and any change that leads to a solution in the *same* attraction basin as an exploitative/local search step. Without any other information, the first solution from an attraction basin can be considered to be a random solution. The expected fitness of a random solution is the average fitness of all solutions in an attraction basin, so a second solution in the same attraction basin that is better than the first solution can be expected to be a better-than-average solution. Referring again to Fig. 1, concurrent exploitation which can lead to better-than-average solutions from an existing attraction basin can interfere with a search technique's explorative processes which are tasked with finding new, more promising attraction basins.

The goal of "thresheld convergence" is to delay local search and thus prevent "uneven" sampling from attraction basins. Convergence is "held" back as (local) search steps that are less than a threshold function are disallowed. As this threshold function decays towards zero, greedier local search steps become allowed. Conversely, until the threshold is sufficiently small, the search technique is forced to focus on the global search aspect of finding the best attraction basin/region of the search space in which a local optimum will eventually be found.

Before applying thresheld convergence to particle swarm optimization, a brief background on the development of threshold functions and other diversification techniques is provided in Section II. Preliminary results for particle swarm optimization with thresheld convergence are then presented in Section III. A simple and robust adaptive threshold function is used to improve performance in Section IV. Performance is again improved by adding an adaptive velocity update in Section V. Finally, all of these results are discussed in Section VI before a summary is given in Section VII.

II. Background

The first work by the authors that used threshold functions to control the rate of convergence was also an application to particle swarm optimization [2]. This application was conceived of and implemented as an efficient version of crowding [3] (applied to the population of *pbest* positions). To prevent crowding in a population, a new solution that is accepted into the population should replace its nearest neighbour. The main weaknesses with crowding is that it is either slow (requiring p = population size distance calculations to find the nearest neighbour) or prone to "replacement errors" (if crowding is applied to only a subset of the population).

Using a threshold function to ensure that new *pbest* positions are kept a minimum distance from all existing *pbest* positions is not more efficient than standard crowding – it still requires p distance calculations. The efficiency gain comes from allowing crowds but disallowing communication within crowds. In differential evolution (DE) [4], crowded solutions create short difference vectors which lead to the creation of new solutions close to existing solutions – i.e. crowding begets more crowding. This "cascading convergence" was reduced by requiring new solutions to be a minimum distance from their base solutions – a requirement which needs only a single distance measurement to enforce [5].

In a particle swarm with a ring topology [1], each particle only communicates with two neighbouring particles. As long as a particle does not create a crowd with its two neighbours, a cascading effect of neighbour after neighbour after neighbour joining this crowd will not occur. In [2], this was implemented by ensuring that a particle would not update its *pbest* position to be within the threshold of its attracting *lbest* position – a requirement which only needs a single distance measurement to enforce.

In the previous work [2], the goal was to prevent *pbest* positions from forming crowds, but it did not prevent the improvement of existing *pbest* positions. This initial implementation has some similarities to niching (e.g. [6]) – as the *pbest* positions are kept a minimum distance apart, they encourage exploration around a more diverse group of attraction basins. However, maintaining diversity among a set of known attraction basins does not address how new attraction basins are discovered and tested.

Recent work with the use of threshold functions to control the rate of convergence has led to the new technique of "thresheld convergence" [7]. Compared to niching and crowding, thresheld convergence prevents both convergence and local search. Similar to niching and crowding, thresheld convergence promotes diversity which increases the chances of finding highly fit (local) optima. The key difference from thresheld convergence to niching and crowding is the prevention of local search which reduces the bias of the search technique to over exploit the current attraction basins (see Fig. 1).

III. Particle Swarm Optimization with Thresheld Convergence

The development of particle swarm optimization (PSO) includes inspirations from "bird flocking, fish schooling, and swarming theory in particular" [8]. Each particle (e.g. a simulated bird) is attracted to its personal best position and the best position of a neighbouring member in the swarm. Rather than a simple line search between a current position and a best position, the velocity and momentum of each particle encourage a more explorative search path. However, elitism is applied to the storage of best positions, so PSO will eventually be highly exploitative around the regions/attraction basins of the best positions as velocities slow to zero.

The following experiments build from the published source code of a PSO benchmark implemented by El-Abd and Kamel [9]. This benchmark implementation is for a GBest swarm using a star topology, so it requires a slight modification to become an LBest swarm with a ring topology. After this modification, the benchmark becomes an implementation of standard PSO [1] in which each dimension d of a particle's velocity v is updated for the next iteration i+1 by

$$v_{i+1,d} = \chi\left(v_{i,d} + c_1\varepsilon_1\left(pbest_{i,d} - x_{i,d}\right) + c_2\varepsilon_2\left(lbest_{i,d} - x_{i,d}\right)\right) \quad (1)$$

where χ is the constriction factor, c_1 and c_2 are weights which vary the contributions of personal best and local best attractors, ε_1 and ε_2 are independent uniform random numbers in the range [0,1], x is the position of the particle, *pbest* is the best position found by the current particle, and *lbest* is the best position found by any particle communicating with the current particle (i.e. two neighbours in an LBest ring topology). Key parameters in [9] include $\chi = 0.792$, $\chi * c_1 = \chi * c_2 = 1.4944$, i.e. $c_1 = c_2 \approx 1.887$, and $p = 40$ particles.

The application of thresheld convergence requires a threshold function. The threshold function (2) developed in [2] has two parameters: α represents the initial minimum distance as a fraction of the search space diagonal and γ represents the decay factor. For $\gamma = 1$, the threshold decays with a linear slope as the iteration k goes from 0 to the maximum number of allowed function evaluations n.

$$threshold = (\alpha * diagonal) * ([n-k]/n)^{\gamma} \quad (2)$$

The threshold function is used during the update of *pbest* positions. In Algorithm 1, the normal update condition for standard PSO [1] is shown. Algorithm 2 then shows the new update condition for particle swarm optimization with thresheld convergence. Since the two distance measurements are only required when improving positions are found, the addition of thresheld convergence has a negligible effect on the computational efficiency of the underlying implementation of standard PSO.

The following analysis of particle swarm optimization with thresheld convergence focuses on two sets from the Black-Box Optimization Benchmarking (BBOB) functions [10]: set 4, multi-modal functions with adequate global structure, and set 5, multi-modal functions with weak global structure. See Table I for more information on the BBOB functions. To be consistent with previous results (i.e. [2]), the following experiments perform 25 independent trials on each function (5 trials on each of the first 5 instances – each instance has a different randomly shifted location for its global optimum) with a fixed limit of 5000**D* function evaluations. These experiments also use D = 20 dimensions.

Algorithm 1 Normal *pbest* update in PSO

```
if f(x) < f(pbest)
   pbest = x;
end if
```

Algorithm 2 Modified *pbest* update

```
if f(x) < f(pbest)
  AND distance (x, pbest) > threshold
  AND distance (x, lbest) > threshold
   pbest = x;
end if
```

TABLE I
BBOB FUNCTIONS

Set	fn	Function Name	Attribute		
			s	u	gs
1	1	Sphere	X	X	X
	2	Ellipsoidal, original	X	X	X
	3	Rastrigin	X		X
	4	Büche-Rastrigin	X		X
	5	Linear Slope	X	X	
2	6	Attractive Sector		X	
	7	Step Ellipsoidal			X
	8	Rosenbrock, original			
	9	Rosenbrock, rotated			
3	10	Ellipsoidal, rotated		X	X
	11	Discus		X	X
	12	Bent Cigar		X	
	13	Sharp Ridge		X	
	14	Different Powers		X	
4	15	Rastrigin, rotated			X
	16	Weierstrass			X
	17	Schaffers F7			X
	18	Schaffers F7, moderately ill-conditioned			X
	19	Composite Griewank-Rosenbrock F8F2			X
5	20	Schwefel			
	21	Gallagher's Gaussian 101-me Peaks			
	22	Gallagher's Gaussian 21-hi Peaks			
	23	Katsuura			
	24	Lunacek bi-Rastrigin			

Names and selected attributes of the 24 functions in the BBOB problem set – separable (s), unimodal (u), global structure (gs).

In Table II, results for particle swarm optimization with thresheld convergence are presented for γ = 3 and α = 0.01, 0.02, 0.05, 0.10, 0.20, and 0.50. Experiments were also conducted with γ = 2 (the best results in [2] were with γ = 2 or 3), but they were consistently a little worse, so they have been omitted for clarity and brevity. The results show the percent difference (%-diff = (b-a)/b)) in the means for 25 independent trials of particle swarm optimization with thresheld convergence (a) and standard PSO (b).

In general, the results with thresheld convergence are better and more consistent on the functions with global structure (BBOB set 4) than without global structure (BBOB set 5). Of note are the Gallagher functions [11] (BBOB 21 and 22) – the previous work with threshold functions [2] had large improvements on these functions while the current results have essentially no change in performance. Given the random

TABLE II
EFFECTS OF INITIAL THRESHOLD SIZE

BBOB	α					
fn	0.01	0.02	0.05	0.1	0.2	0.5
15	9.2%	15.9%	12.5%	6.8%	-3.5%	-2.6%
16	20.4%	25.6%	33.8%	21.2%	-1.2%	-3.1%
17	34.1%	40.7%	67.0%	75.2%	61.6%	52.2%
18	13.0%	24.9%	41.1%	55.6%	46.7%	33.5%
19	-1.7%	-0.3%	-0.2%	-8.1%	-6.5%	-4.0%
15-19	15.0%	21.4%	30.8%	30.1%	19.4%	15.2%
20	13.1%	16.1%	18.8%	10.9%	0.4%	-9.0%
21	-5.4%	-1.4%	-20.2%	3.9%	-2.3%	9.3%
22	-3.2%	-13.9%	-16.1%	-18.2%	-20.4%	-6.0%
23	-18.3%	-7.4%	-13.5%	-14.2%	-16.9%	-24.5%
24	2.2%	-0.5%	2.3%	-2.0%	-3.1%	-7.7%
20-24	-2.3%	-1.4%	-5.8%	-3.9%	-8.5%	-7.6%

The best overall results occur with α = 0.05. The benefits of thresheld convergence appear to depend on the global structure of the search space.

TABLE III
BEST RESULTS FOR INITIAL THRESHOLD SIZES

BBOB	standard PSO		with thresholds		α	%-diff	p-value
fn	mean	std dev	mean	std dev			
15	6.05e+1	1.46e+1	5.08e+1	1.49e+1	0.02	15.9%	0.01
16	5.37e+0	1.53e+0	3.55e+0	1.28e+0	0.05	33.8%	0.00
17	6.61e−1	2.64e−1	1.64e−1	1.08e−1	0.10	75.2%	0.00
18	2.87e+0	1.28e+0	1.28e+0	6.70e−1	0.10	55.6%	0.00
19	3.61e+0	4.32e−1	3.62e+0	4.26e−1	0.05	-0.2%	0.47
15-19						36.1%	
20	1.14e+0	1.38e−1	9.22e−1	1.77e−1	0.05	18.8%	0.00
21	1.41e+0	1.21e+0	1.28e+0	1.28e+0	0.50	9.3%	0.35
22	1.69e+0	1.51e+0	1.75e+0	1.77e+0	0.01	-3.2%	0.45
23	1.33e+0	2.49e−1	1.43e+0	2.55e−1	0.02	-7.4%	0.09
24	1.13e+2	1.12e+1	1.10e+2	1.60e+1	0.05	2.3%	0.26
20-24						3.9%	

The addition of thresheld convergence leads to a significant improvement (%-diff > 10% and $p < 0.05$ for the ***t***-test) on four of the five functions in BBOB set 4.

TABLE IV
EFFECTS OF INITIAL THRESHOLD SIZE

BBOB	α					
fn	0.01	0.02	0.05	0.1	0.2	0.5
15	13.0%	10.6%	18.9%	17.7%	5.3%	14.7%
16	16.7%	13.0%	17.1%	5.6%	11.7%	3.8%
17	31.4%	35.8%	66.4%	75.6%	72.3%	64.8%
18	1.0%	20.5%	50.2%	47.1%	52.6%	49.1%
19	-0.1%	1.3%	4.2%	-0.8%	-0.4%	-2.3%
15-19	12.4%	16.3%	31.4%	29.0%	28.3%	26.0%
20	16.2%	14.7%	12.7%	13.0%	5.9%	2.8%
21	37.4%	-1.5%	0.4%	-35.0%	18.9%	-33.8%
22	-7.7%	-42.4%	-7.8%	-17.8%	-37.7%	-12.7%
23	-0.4%	-4.9%	-7.6%	-13.5%	-6.3%	-12.8%
24	7.4%	1.7%	5.1%	4.3%	-1.1%	-1.5%
20-24	10.6%	-6.5%	0.5%	-9.8%	-4.1%	-11.6%

For each value of α, the performance of the adaptive threshold function is remarkably similar to the performance with the scheduled threshold function – see Table II (especially for BBOB set 4).

optima in the Gallagher functions, exploring multiple distinct optima (e.g. through the effects of niching) improves the chances that one of these optima will be highly fit. Conversely, the existence of random attraction basins of different shapes and sizes is a distinct contradiction to the premise of similarly sized and shaped attraction basins used in the development of thresheld convergence (see Fig. 1). The remainder of this paper will thus focus on BBOB set 4.

It should also be noted that the conducted experiments with thresheld convergence included the full set of BBOB functions. On set 1 – separable functions, neither version exploits separability so the results on this set match the results on other sets – thresheld convergence improved performance on the multi-modal functions and had worse performance on the unimodal functions. On set 2 – functions with low or moderate conditioning, neither versions addresses the effects of ill-conditioning so the dominant trend again matches the modality of the underlying functions. On set 3 – unimodal functions with high conditioning, thresheld convergence is explicitly designed for multi-modal functions so it is specifically ill-equipped to perform well on unimodal functions. As the threshold size increases (e.g. α), less exploitation is possible and worse results are obtained (across all functions for all parameter settings). For brevity and clarity, results and analysis for these three sets are omitted from the rest of the paper.

The addition of thresheld convergence is particularly effective on BBOB 17 and 18 – large improvements for all tested values of α. On BBOB 15 and 16, thresheld convergence is effective for smaller values of α, but larger values of α probably lead to too little exploitation. The best results on these four functions are statistically significant as indicated by the ***t***-tests shown in Table III. For all of set 4 (BBOB15-19), the best results with any value of α have a mean improvement of 36.1%. Compared to the best results for a single value of α = 0.05 which has a mean improvement of 30.8%, these best overall results show the room for improvement that should be attainable with adaptive threshold functions.

Algorithm 3 Adaptive threshold function

```
if no pbest updates
   threshold = threshold * decay factor;
end if
```

IV. AN ADAPTIVE THRESHOLD FUNCTION

The key parameter affecting the performance of thresheld convergence is α. On some functions (e.g. BBOB 18 – see Table II), the best results are achieved with larger values of α whereas smaller values of α lead to the best results on other functions (e.g. BBOB 15). It is hypothesized that the ideal threshold value is related to the size of the attraction basins in the search space. For a threshold function with specific α and γ parameters, a certain amount of time will be spent near the ideal threshold value. The development of adaptive threshold functions which can spend more time near this ideal threshold value should improve the performance of thresheld convergence.

An adaptive threshold function has been developed for particle swarm optimization. The basic premise is that a threshold value that is too high will prevent any improvements from being made. Conversely, if improving solutions are being found, the current threshold value may be at the ideal level, so it should be left unchanged. Thus, the number of *pbest* updates that occur during an iteration *i* are recorded (see Algorithm 2), and the threshold value is decreased if the number of updates is zero (see Algorithm 3).

In Table IV, the results on BBOB sets 4 and 5 are given for PSO with the new adaptive threshold function. The initial threshold value (α) is 0.01, 0.02, 0.05, 0.10, 0.20, and 0.50 and the "decay factor" is 0.995 – the threshold decreases by 0.5% after any iteration in which no *pbest* improvements are made. Unreported experiments also tried rates of decrease of 0.25%, 1%, and 2%. Similar but slightly worse results were achieved with 1% while 0.25% and 2% had larger drop-offs in performance. Despite its simplicity, the adaptive threshold function in Algorithm 3 appears to provide relatively stable and predictable performance.

The results in Table IV again show percent difference (%-diff = (b-a)/b) of mean performance between particle swarm optimization with thresheld convergence (a) and standard PSO (b). Compared to the results in Table II, the results in Table IV

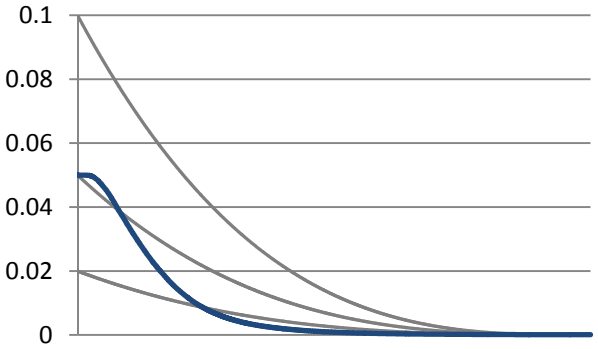

Fig. 2. Adapted threshold values for BBOB 15 with $\alpha = 0.05$. Lighter reference lines show the scheduled threshold function (1) for $\gamma = 3$ and $\alpha = 0.02$, 0.05, and 0.10.

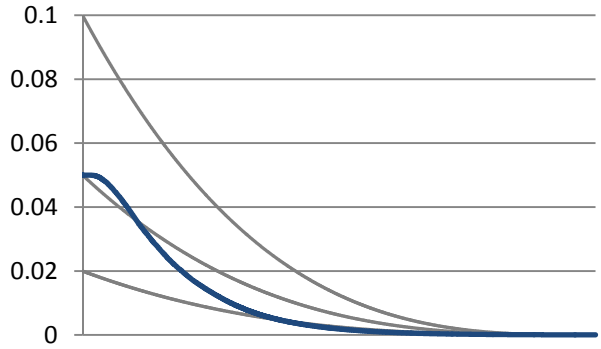

Fig. 3. Adapted threshold values for BBOB 18 with $\alpha = 0.05$. Lighter reference lines show the scheduled threshold function (1) for $\gamma = 3$ and $\alpha = 0.02$, 0.05, and 0.10.

tend to be less dependent on the size of α. In particular, there is less drop-off in performance for $\alpha = 0.50$ (especially for BBOB 15 and 16). It appears that the adaptive threshold function does have some ability to find and maintain an appropriate threshold value regardless of the initial value of α.

The operation of the adaptive threshold function is studied more closely in Figs. 2 and 3. The best overall results in Table IV are for $\alpha = 0.05$ which leads to a mean improvement of 31.4% across the five functions in set 4 (BBOB 15-19). For the scheduled threshold functions (1), BBOB 15 has its best results for $\alpha = 0.02$ and BBOB 18 has its best results for $\alpha = 0.10$ (see Table II). The mean adapted threshold values over all 25 independent trials for BBOB 15 and 18 with $\alpha = 0.05$ are plotted against the scheduled threshold functions for $\gamma = 3$ and $\alpha = 0.02$, 0.05, and 0.10.

In Fig. 2, the threshold value drops off quite quickly and spends a large amount of time below the scheduled threshold function with $\alpha = 0.02$ which led to the best performance on BBOB 15. In Fig. 3, the threshold value drops off more slowly, but it does not approach the scheduled threshold function with $\alpha = 0.10$ (which led to the best performance on BBOB 18) until very late in the search process. In general, there are no visible plateaus (except the initial stage where *pbests* are being improved from their very poor initial random solutions), so the idea of an ideal threshold value may have to be revisited.

V. Another Adaptation

The addition of thresheld convergence to particle swarm optimization increases the distances among the *pbest* positions. With larger distances among the attractors, each particle update (1) will be subjected to larger velocities. These larger velocities can interfere with the convergence and/or fine-grained search properties of the overall process. Another adaptation is proposed to address this issue.

Algorithm 4 Adaptive threshold function with "braking"

```
if no pbest updates
    threshold = threshold * decay factor;
    v = v * v_f
end if
```

Each decrease to the threshold value implies the desire to conduct a finer-grained search. To conduct a finer search/sampling of attraction basins, a smaller step size (created by smaller velocities) should be used. Thus, with each update to the threshold value in Algorithm 3, the velocity is also reduced – see Algorithm 4. (Note: due to the distances among the *pbest* attractors, the particles can reaccelerate. However, this coordinated "tapping on the brakes" still reduces the overall speed within the swarm.)

The results in Table V again show percent difference (%-diff = (b-a)/b) of mean performance between particle swarm optimization with thresheld convergence (a) and standard PSO (b). The values for v_f are 0.95, 0.90, 0.85, 0.80, 0.75, and 0.70 – a smaller value of v_f corresponds to more "braking". The coordinated reduction in speed is quite helpful on functions BBOB 15, 17, 18, 19, and 24, and it has visibly negative effects on BBOB 16 and 20. Overall, the best results for BBOB set 4 are achieved with $v_f = 0.85$.

Compared to standard PSO, the addition of the adaptive threshold function (and the adaptive speed reductions) leads to significant improvements on most of the multi-modal functions with global structure (set 4 – BBOB 15-19) and negligible effects on most of the multi-modal functions without global structure (set 5 – BBOB 20-24) – see Table VI. The results

TABLE V
EFFECTS OF "BRAKING"

BBOB	v_f					
fn	0.95	0.90	0.85	0.80	0.75	0.70
15	**23.2%**	**27.3%**	**35.7%**	**41.9%**	**36.6%**	**31.5%**
16	10.6%	5.5%	6.5%	9.9%	-12.3%	-10.1%
17	**69.9%**	**83.0%**	**84.8%**	**72.9%**	**71.9%**	**75.1%**
18	50.1%	**57.4%**	**59.1%**	**52.4%**	**64.4%**	**60.5%**
19	2.6%	**7.6%**	**18.8%**	**18.1%**	**19.0%**	**14.3%**
15-19	31.3%	36.2%	41.0%	39.0%	35.9%	34.3%
20	11.4%	**13.1%**	2.7%	4.5%	-0.7%	-0.2%
21	-27.6%	**7.7%**	**12.6%**	-12.1%	-30.0%	-0.6%
22	-49.3%	-44.7%	**-2.6%**	**23.0%**	-42.1%	-33.6%
23	**-6.8%**	**-7.1%**	**-1.2%**	**-7.2%**	**-4.8%**	**-0.5%**
24	**17.8%**	**22.0%**	**19.6%**	**28.9%**	**27.1%**	**25.3%**
20-24	-10.9%	-1.8%	6.2%	7.4%	-10.1%	-1.9%

Values in bold are better than the %-diff achieved with $\alpha = 0.05$ in Table IV. On BBOB set 4, the best overall value of $v_f = 0.85$ leads to an additional 10% improvement through the use of "braking".

TABLE VI
FINAL RESULTS

BBOB	standard PSO		with thresholds		%-diff	p-value
fn	mean	std dev	mean	std dev		
15	6.05e+1	1.46e+1	3.89e+1	1.44e+1	35.7%	0.00
16	5.37e+0	1.53e+0	5.02e+0	1.69e+0	6.5%	0.22
17	6.61e−1	2.64e−1	1.00e−1	6.76e−2	84.8%	0.00
18	2.87e+0	1.28e+0	1.18e+0	5.47e−1	59.1%	0.00
19	3.61e+0	4.32e−1	2.93e+0	6.31e−1	18.8%	0.00
15-19					41.0%	
20	1.14e+0	1.38e−1	1.10e+0	1.77e−1	2.7%	0.25
21	1.41e+0	1.21e+0	1.24e+0	1.97e+0	12.6%	0.35
22	1.69e+0	1.51e+0	1.74e+0	1.99e+0	-2.6%	0.47
23	1.33e+0	2.49e−1	1.35e+0	3.04e−1	-1.2%	0.42
24	1.13e+2	1.12e+1	9.05e+1	1.43e+1	19.6%	0.00
20-24					6.2%	

When added to standard PSO, the adaptive threshold function with "braking" leads to consistent and mostly significant improvements (%-diff > 10% and $p < 0.05$ for the ***t***-test) on BBOB set 4.

presented in Table VI are for a single set of parameters (α = 0.05, v_f = 0.85, and a threshold decay rate of 0.5%), and they lead to better overall results (+41.0%) than the best samples from any scheduled threshold function shown in Table II (+36.1%). Although there are still some performance variations with α, v_f, and the threshold decay factor, this result demonstrates that simple, effective, and robust adaptive threshold functions can be developed.

VI. DISCUSSION

There are many ways to improve the performance of PSO from the standard baseline version [1]. For example, multi-swarm techniques (e.g. [12][13][14]) converge quickly to a (local) optimum and then use various restart strategies to find more (local) optima using the allotted function evaluations. These restart strategies add many additional parameters and design decisions (e.g. [15][16]), so the subsequent performance improvements come at the cost of meaningful increased complexity. Nonetheless, these improvements (e.g. 49.1% on BBOB set 4 – see Table II in [16]) are certainly competitive with the current results.

Within a single-swarm system, performance is generally improved by reducing the rate of convergence (e.g. switching from a GBest/star topology [8] to an LBest/ring topology [1]). Convergence occurs in PSO when a particle with zero speed has the same position as all of its *pbest* attractors. Thresheld convergence prevents the co-location of the *pbest* attractors. Other methods to reduce convergence include disallowing zero velocities [17], moving the *pbest* attractors [18][19], and moving the position of a particle away from its *pbest* attractors [20]. A full comparison of the benefits (e.g. in relation to their added complexity) of these techniques is an open area of research, and it should be noted that many of these techniques can be combined with other variations when their specific mechanisms are not in conflict.

The current variation of thresheld convergence appears to be effective mostly on multi-modal function with global structure. This unexpected result is inconsistent with previous work involving threshold functions [2][5][7] which showed broad benefits across the full range of multi-modal functions (i.e. BBOB sets 4 and 5). As previously discussed in Section III, it appears that a "niching effect" may be more suitable for the Gallagher functions (BBOB 21 and 22) than the current implementation of thresheld convergence. Future research will study the Gallagher functions more closely with an emphasis on achieving the simultaneous benefits of niching and thresheld convergence.

Future research will also study the effects of each parameter more closely (e.g. α, v_f, and the threshold decay factor). Although the preliminary work presented here has been quite successful, there are still large variations in performance on some functions for different parameter settings. This variation suggests that more improvements can be achieved through the development of improved (adaptive) threshold functions.

One aspect of the current adaptive threshold function that may be difficult to improve is its simplicity. In general, simple modifications (e.g. the switch from a GBest/star topology to an LBest/ring topology [1]) are more likely to gain widespread adoption than more complex modifications (e.g. niching [6]). The development of adaptive threshold functions to replace scheduled threshold functions is a definite improvement in terms of simplicity. The large potential benefits, computational efficiency, and general ease of adding thresheld convergence make improved threshold functions a promising area for further research.

VII. SUMMARY

The addition of thresheld convergence to particle swarm optimization can lead to large performance improvements on multi-modal functions with adequate global structure. A simple, effective, and robust adaptive threshold function has been developed to replace the originally developed scheduled threshold functions. The simplicity and effectiveness of the proposed modifications make thresheld convergence a promising area for further research.

REFERENCES

[1] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," IEEE SIS, 2007, pp. 120–127.

[2] S. Chen and J. Montgomery, "A simple strategy to maintain diversity and reduce crowding in particle swarm optimization," Austalasian AI, 2011, pp. 281–290.

[3] K. A. De Jong, An analysis of the behavior of a class of genetic adaptive systems, PhD thesis. Dept. of Computer and Communication Sciences, University of Michigan, 1975.

[4] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," J. Global Optimization, vol. 11, pp. 341–359, 1997.

[5] J. Montgomery and S. Chen, "A simple strategy to maintain diversity and reduce crowding in differential evolution," IEEE CEC, 2012, pp. 2692–2699.

[6] R. Brits, A. P. Engelbrecht, and F. Van den Bergh, "A niching particle swarm optimizer," SEAL, 2002, pp. 692–696.

[7] S. Chen, C. Xudiera, and J. Montgomery, "Simulated annealing with thresheld convergence," IEEE CEC, 2012, pp. 1946–1952.

[8] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," IEEE ICNN, 1995, pp. 1942–1948.

[9] M. El-Abd and M. S. Kamel, "Black-box optimization benchmarking for noiseless function testbed using particle swarm optimization," GECCO, 2009, pp. 2269–2273.

[10] N. Hansen, S. Finck, R. Ros, and A. Auger, "Real-parameter black-box optimization benchmarking 2009: noiseless functions definitions," INRIA Technical Report RR-6829, 2009.

[11] M. Gallagher and B. Yuan, "A General-Purpose Tunable Landscape Generator", IEEE TEC, vol. 10(5), pp. 590–603, 2006.
[12] T. Hendtlass, "WoSP: a multi-optima particle swarm algorithm," IEEE CEC, 2005, pp. 727–734.
[13] J. J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer," IEEE SIS, 2005, pp. 124–129.
[14] S. Chen, "Locust swarms – a new multi-optima search technique," IEEE CEC, 2009, pp. 1745–1752.
[15] A. Bolufé Röhler and S. Chen, "An analysis of sub-swarms in multi-swarm systems," Austalasian AI, 2011, pp. 271–280.
[16] S. Chen and J. Montgomery, "Selection strategies for initial positions and initial velocities in multi-optima particle swarms," GECCO, 2011, pp. 53–60.
[17] T. Hendtlass, "Particle swarm optimisation and high dimensional problem spaces," IEEE CEC, 2009, pp. 1988–1994.
[18] M. G. Epitropakis, V. P. Plagianakos, and M. N. Vrahatis, "Evolving cognitive and social experience in particle swarm optimization through differential evolution," IEEE CEC, 2010, pp. 2400–2407.
[19] S. S. Pace and C. Woodward, "Diversity preservation using excited particle swarm optimisation," GECCO, 2011, pp. 61–68.
[20] S. Chen, "Particle swarm optimization with pbest crossover," IEEE CEC, 2012, pp. 1234–1239.