Simulated Annealing with Thresheld Convergence

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

Carlos Xudiera
Department of Computer Science and Engineering
York University
Toronto, Canada
xudiera@yorku.ca

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

***Abstract*—Stochastic search techniques for multi-modal search spaces require the ability to balance exploration with exploitation. Exploration is required to find the best region, and exploitation is required to find the best solution (i.e. the local optimum) within this region. Compared to hill climbing which is purely exploitative, simulated annealing probabilistically allows "backward" steps which facilitate exploration. However, the balance between exploration and exploitation in simulated annealing is biased towards exploitation – improving moves are always accepted, so local (greedy) search steps can occur at even the earliest stages of the search process. The purpose of "thresheld convergence" is to have these early-stage local search steps "held" back by a threshold function. It is hypothesized that early local search steps can interfere with the effectiveness of a search technique's (concurrent) mechanisms for global search. Experiments show that the addition of thresheld convergence to simulated annealing can lead to significant performance improvements in multi-modal search spaces.**

Keywords-simulated annealing; thresheld convergence; niching; crowding; exploration; exploitation

I. INTRODUCTION

Imagine a search space with local optimum "wells" of similar size and shape – e.g. a sinusoid superimposed over a linear slope. On average, the difference between two random samples from two different optimum wells in this idealized search space will be equal to the difference between the optima for these two wells. Many heuristic search techniques rely on this correlation as they concentrate their search efforts in the region(s) around the best (random) solution(s) that they have so far discovered.

A simple example is particle swarm optimization (PSO) [1] with a global best/star topology. If it is assumed that every particle in this swarm starts at a random initial position, the initial global best attractor will represent the best individual from a set of random positions. The reason to direct all of the particles to move towards and explore around this global attractor is the inherent belief that the best (local) optimum will eventually be found near it. Specifically, if the local optima around the initial positions have the same relative fitness as the initial random samples, then the concentration of search around the best initial position will lead the swarm towards the best optimum from the original set of optimum wells identified by the initial random positions.

Producing a single local optimum from the best of a small set of random solutions is clearly a highly greedy search strategy [2]. In particular, PSO does not "lock on" to the initial global attractor – the particles follow exploratory trajectories and they can update the global best position if any of them encounters a better position. However, redirecting the search process towards this new global best position again implies the assumption that the best optimum will be found in the region around the best known solution in the search space.

There are two potential problems with directing the search process to finding the (local) optimum nearest to the best known solution in the search space. First, due to sampling errors, the best local optimum found by optimizing all of the current solutions (e.g. from a population) may not be the same as that found by optimizing the best current solution. Second, there is no guarantee that the current best solution is in the optimum well with the global optimum. To continue exploration for the global optimum well, it can be useful to have an "even" selection of sample solutions from each newly explored optimum well.

One example of an ideal sampling is if every solution selected during the exploration phase has the same difference in fitness compared to the optimum in its own optimum well. Since such an ideal sampling is impossible, one goal is to sample as "evenly" as possible. Although two random samples from different optimum wells will on average have a difference in fitness equal to the difference between their two (local) optima, the same cannot be said about the comparison of a random sample from one optimum well with a better-than-random sample from a second optimum well. In particular, it is important to avoid the situation in which a better-than-random sample from a poor local optimum well is better than the expected fitness of a random sample from a good local optimum well. (See Fig. 1.) In this situation, it will become more difficult for a search process that concentrates its search effort around the best current solution to redirect its search effort from the poor optimum well to the better one.

One method to produce a better-than-random sample is to perform local search. Starting from an initial position, let us define any step/change that leads to a position in a new optimum well as an explorative/global search step and any step/change that leads to a position in the same optimum well as an exploitative/local search step. Without any other information, the first solution from an optimum well can be

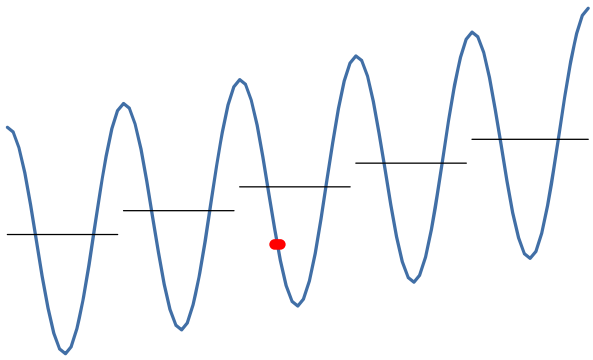

Figure 1. The horizontal lines represent the average fitness of a random sample taken from each optimum well. If an optimum well has a better-than-random solution (see dot), this solution may be fitter than random samples drawn from better optimum wells.

considered to be a random sample. Subsequently, a second solution in the same optimum well that is better than the first solution can be considered to be a better-than-random sample. Referring again to Fig. 1, local search which leads to better-than-random solutions for a given optimum well can interfere with a search technique's ability to perform (concurrent) global search to find new, more promising optimum wells.

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

This paper presents an application of thresheld convergence to simulated annealing. A brief background to simulated annealing and other applications of thresheld convergence to particle swarm optimization and differential evolution are presented in Section II. Benchmark results for simulated annealing are presented in Section III before thresheld convergence is added in Section IV. However, no improvement is shown, and it is hypothesized that thresheld convergence requires elitism which is added to simulated annealing in Section V. With the addition of elitism, the opportunity to have increased exploration as provided by thresheld convergence is then shown to lead to significant performance improvements in Section VI. The similarities and differences of thresheld convergence when applied to simulated annealing, particle swarm optimization, and differential evolution are discussed in Section VII before a summary is presented in Section VIII.

II. Background

Simulated annealing is modelled after the physical process of annealing [3]. If an entity such as a molten metallic alloy is cooled too quickly, it can solidify into a sub-optimal crystalline structure. Ideally, there exists a temperature at which the system can easily escape from one optimum to a fitter optimum, but transitions to less fit optima are much less likely at this temperature. It should be noted that the inability to find such a temperature could lead to a given alloy mixture being discarded. Physical annealing is not just a process that solves a problem, but it also helps determine which problems (e.g. alloy mixtures) will be solved in the first place. Further, physical systems have a practical limitation of moving from one state to other nearby states, so a globally convex search space (in which any local optimum can move to the global optimum through a series of transitions to neighbouring optima that have monotonically improving fitness) is the ideal match for annealing-based optimization processes.

In general, any search technique which concentrates its search effort around the best current solution will be most effective in globally convex search spaces. Since the targeted optimum well of these search techniques will only change if a better solution is found and these search techniques concentrate their search efforts around the best found solution, these search processes are most likely to follow a path through neighbouring optimum wells. In globally convex search spaces, there is a path of improving optimum wells from any solution to the global optimum.

In following a path of improving optimum wells, the ability to accurately estimate the relative fitness of the optimum in each well can be beneficial. A key feature of heuristic search is that the fitness of the optimum in a well is often estimated by the fitness of a known solution taken from that well. To accurately compare the potential fitness of two optima by comparing the fitness of two (random) solutions taken from their optimum wells, these two sample solutions should ideally have the same relative fitness within each well. By delaying local search, thresheld convergence helps prevent the sample representing an early optimum well from becoming so optimized that it interferes with the comparison of (random) samples from later optimum wells. An early local optimum which interferes with future exploration is the essence of premature convergence. The diversity of any other sample solutions (e.g. in a population) is wasted if they cannot redirect the search process to concentrate on another (more promising) optimum well.

The first use of thresheld convergence is an application to particle swarm optimization [4]. In standard PSO which uses a ring topology [5], particle trajectories can be drawn towards the personal best positions of two neighbouring particles in the swarm. These attractions concentrate the search effort of the swarm around the best position(s) currently known by the swarm system. When velocities slow, the swarm will conduct a more local/exploitative search around these personal best position(s). Thus, when velocities are faster at the beginning of the search process, it can be viewed that the swarm is performing an explorative search and that each personal best position represents a promising optimum well [6].

In PSO, a particle does not move directly from its current position to a local best attractor. Similar to birds in flight, particles have arcing trajectories that overshoot and loop back to their various attractors. These non-direct trajectories are the basis of exploration, and the identification of a more promising optimum well involves finding a better solution in it than the current personal best position. From the previously introduced

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			

ideas on sampling solutions from optimum wells, this process of exploration may be harmed if the personal best position has a better than random relative fitness within its optimum well.

Local search produces better than random solutions within an optimum well, so a key goal of thresheld convergence is to delay the transition from global search to local search. In the previous application to PSO [4], it was first noted that the concentration of search within a region of the search space occurs when personal best attractors are similarly concentrated in that region. This concentration/convergence of personal best attractors was reduced by disallowing specific updates that cause personal best positions to become closer than a threshold function. The resulting benefits from thresheld convergence led to significant performance improvements for the modified PSO compared to standard PSO on a broad range of multi-modal functions [7].

Thresheld convergence has subsequently been applied to differential evolution [8]. Differential evolution (DE) [9] is most commonly implemented with an elitist population scheme. Therefore, in order for DE to dedicate search effort to a new optimum well, it is necessary to find a (random) sample from the new optimum well that is better than a target solution which represents another optimum well currently under consideration. Again, if the relative quality of the target solution within its optimum well is much better than random, it makes it less likely for a (random) candidate solution to be better – even if it is from an optimum well with a better local optimum (see Fig. 1).

Similar to its application in PSO, the goal of applying thresheld convergence to DE is to delay the transition from global search (i.e. finding promising optimum wells) to local search (i.e. finding the best solution within an existing optimum well). Local search occurs when points near existing points are created, and the distance between new search points is affected by the length of the difference vector. By disallowing moves closer to the base solution than the threshold function, thresheld convergence delays these local search steps that can interfere with the effectiveness of concurrent global search steps. The implementation of these modifications suggested by thresheld convergence has also led to significant performance improvements in DE across a broad range of multi-modal functions.

III. SIMULATED ANNEALING

The benchmark implementation of simulated annealing (SA) used in this paper is derived from the `simulannealbnd` function from the MATLAB Global Optimization Toolbox [10]. Using the default implementation, the (maximum) step length is equal to the temperature:

$$T = T_0 * 0.95^k \tag{1}$$

where $T_0 = 100$ is the initial temperature and k is the iteration number. The actual step size is drawn using a Student's distribution with T as the maximum length (see Fig. 2). For a given step which heads in a uniformly random direction, all improving moves are accepted and non-improving moves are accepted with a probability of

$$1 / \left\{ 1 + \exp\left(\frac{\Delta}{\max T} \right) \right\} \tag{2}$$

where Δ is the (positive) difference between the new and old objectives. The termination condition used in the benchmark implementation is a fixed number of function evaluations.

The following analysis of simulated annealing with and without thresheld convergence focuses on two sets from the Black-Box Optimization Benchmarking (BBOB) functions [7]: set 4, multi-modal functions with adequate global structure, and set 5, multi-modal functions with weak global structure. However, for completeness and additional insight, results for all BBOB functions are presented. See Table I for names and selected attributes of the 24 functions in the BBOB problem set – separable (s), unimodal (u), global structure (gs).

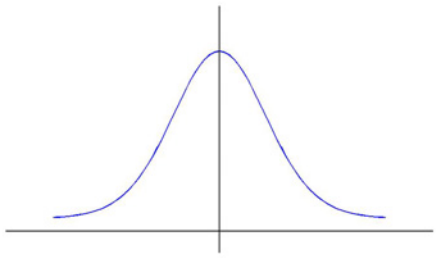

Figure 2. Step sizes are drawn from a Student's distribution in the benchmark implementation of simulated annealing (baseSA).

TABLE II. BENCHMARK SIMULATED ANNEALING RESULTS

Set	fn	simple SA		baseSA		%-diff	*t*-test
		mean	stddev	mean	stddev		
1	1	4.02e+1	1.76e+1	3.46e+1	6.69e+0	14.0%	0.07
	2	3.28e+5	2.38e+5	2.60e+4	6.81e+3	92.1%	0.00
	3	3.17e+2	8.90e+1	2.40e+2	3.02e+1	24.4%	0.00
	4	3.84e+2	7.59e+1	2.83e+2	2.36e+1	26.2%	0.00
	5	1.46e+2	4.10e+1	6.69e+1	1.65e+1	54.1%	0.00
2	6	2.29e+4	3.22e+4	3.30e+2	4.85e+2	98.6%	0.00
	7	1.42e+2	6.02e+1	7.44e+1	1.44e+1	47.7%	0.00
	8	5.83e+3	3.97e+3	4.64e+2	9.51e+1	92.0%	0.00
	9	1.10e+2	8.87e+0	1.17e+2	1.25e+1	-6.4%	0.01
3	10	3.06e+5	1.89e+5	1.96e+4	1.58e+4	93.6%	0.00
	11	2.20e+2	8.51e+1	6.67e+1	1.02e+1	69.7%	0.00
	12	3.00e+7	2.01e+7	1.44e+5	9.48e+4	99.5%	0.00
	13	1.00e+3	2.27e+2	7.32e+2	9.52e+1	26.9%	0.00
	14	2.17e+1	6.39e+0	1.07e+1	1.87e+0	50.7%	0.00
4	15	3.40e+2	1.04e+2	2.37e+2	2.16e+1	30.1%	0.00
	16	2.14e+1	6.37e+0	1.67e+1	2.59e+0	21.6%	0.00
	17	1.25e+1	3.28e+0	6.97e+0	6.09e−1	44.0%	0.00
	18	3.56e+1	9.64e+0	2.31e+1	2.37e+0	35.1%	0.00
	19	2.50e−1	0.00e+0	2.50e−1	0.00e+0	0.0%	0.00
5	20	1.16e+2	2.12e+2	3.21e+0	1.22e−1	97.2%	0.01
	21	7.21e+1	1.23e+1	5.06e+1	6.96e+0	29.9%	0.00
	22	7.70e+1	6.89e+0	5.66e+1	1.13e+1	26.5%	0.00
	23	2.23e+0	7.81e−1	2.00e+0	2.56e−1	10.3%	0.08
	24	2.22e+2	4.33e+1	2.09e+2	1.15e+1	5.8%	0.08

TABLE III. RESULTS WITH THRESHELD CONVERGENCE

Set	fn	α				
		0.001	0.005	0.01	0.05	0.1
1	1	-1.9%	**2.7%**	-2.7%	-12.0%	-15.2%
	2	**16.8%**	-9.6%	-50.2%	-92.4%	-170.8%
	3	-0.8%	-2.5%	0.7%	1.5%	**1.8%**
	4	0.1%	-1.2%	**1.5%**	-3.3%	-4.2%
	5	8.3%	7.3%	6.8%	10.3%	**32.2%**
2	6	72.3%	**80.5%**	78.9%	70.7%	75.3%
	7	3.2%	3.7%	**6.8%**	6.7%	0.3%
	8	25.5%	49.1%	**54.0%**	48.9%	43.5%
	9	**-1.4%**	-5.4%	-5.5%	-4.8%	-5.2%
3	10	**23.8%**	19.2%	-16.8%	-51.1%	-83.0%
	11	**0.0%**	-4.1%	-7.0%	-19.2%	-16.2%
	12	94.0%	**94.9%**	89.0%	46.1%	16.5%
	13	0.7%	6.8%	25.8%	**31.1%**	26.1%
	14	1.7%	1.2%	**4.0%**	-9.7%	-14.9%
4	15	-0.7%	2.3%	**2.7%**	0.3%	-3.9%
	16	-4.3%	-2.4%	-5.3%	-5.1%	-6.2%
	17	**1.6%**	0.5%	-0.2%	-4.7%	-3.9%
	18	-0.2%	**2.1%**	-0.8%	-6.8%	-5.4%
	19	0.0%	0.0%	0.0%	0.0%	0.0%
5	20	1.4%	**5.1%**	4.8%	4.1%	3.3%
	21	**1.3%**	-0.7%	-3.3%	-3.0%	-8.0%
	22	-5.5%	-5.7%	-7.3%	-5.1%	-7.5%
	23	**5.1%**	-0.2%	0.7%	-1.9%	-9.0%
	24	-1.2%	**1.9%**	-1.6%	-3.6%	-9.0%

To be consistent with previous work (e.g. [4][11][12]), the following experiments perform 25 independent trials on each function (5 trials on each of the first 5 instances – each instance has its global optimum in a randomly shifted location) with a fixed limit of 5000**D* function evaluations (FEs). All experiments in this paper use $D = 20$ dimensions which leads to a total of 100,000 FEs. To facilitate the addition of thresheld convergence, we re-implemented MATLAB's version of simulated annealing, and the results (percent difference, %-diff = (b-a)/b) in Table II show that our new version called baseSA (a) compares well with MATLAB's version of simulated annealing called simple SA (b) when reannealing is disabled.

IV. SIMULATED ANNEALING WITH THRESHELD CONVERGENCE

The threshold function (3) developed in [4] has two parameters: α represents the initial minimum distance as a ratio of the search space diagonal and γ represents the decay factor.

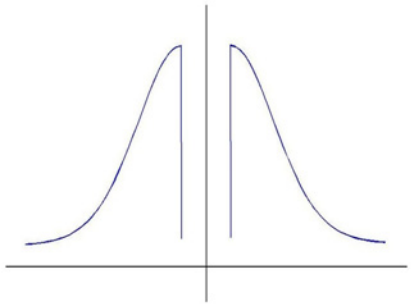

Figure 3. Step sizes are drawn from a Student's distribution with a gap width specified by the threshold function.

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.

$$threshold = (\alpha * diagonal) * ([n-k]/n)^{\gamma} \quad (3)$$

To simplify the generation of new solutions, the threshold is applied to each dimension (and the diagonal is replaced with the range for that dimension). Specifically, compared to the function in Fig. 2, the distribution is "squeezed" at the edges to accommodate the gap created by the *threshold* in the middle (see Fig. 3). When applied to each dimension, this gap leads to a hypercube "tabu" region as opposed to the hypersphere region previously used in PSO [4] and DE [8].

The effects of thresheld convergence on simulated annealing were examined over a range of values for $\alpha = 0.001$, 0.005, 0.01, 0.05, and 0.1 using $\gamma = 2$. The results in Table III show the percent difference ((b-a)/b) in mean performance between baseSA (b) and its performance with thresheld convergence (a). A positive percent difference represents an improvement with thresheld convergence, and the bolded values highlight the best value of α for each function.

The larger step sizes taken by simulated annealing as caused by the effects of thresheld convergence (see Fig. 3) lead to some improvements on several unimodal functions (e.g. slope – BBOB fn 5). When the optimal solution is very far from the current solution (e.g. in the corner of the search space), increased exploration can lead to improved performance. However, on the targeted multi-modal functions (BBOB fn 15-24), thresheld convergence has negligible (and generally negative) effects. Increased exploration has not lead to improved performance on these functions.

TABLE IV: RESULTS WITH ELITISM

Set	fn	*r*				
		10,000	1,000	100	10	1
1	1	8.3%	31.2%	73.4%	95.4%	**99.2%**
	2	**1.2%**	-8.1%	0.7%	-9.0%	-1.6%
	3	3.3%	8.0%	26.2%	**36.5%**	32.4%
	4	3.8%	11.5%	25.5%	**32.5%**	27.6%
	5	13.8%	10.4%	8.7%	47.8%	**75.9%**
2	6	-3.1%	56.6%	64.9%	**78.0%**	73.2%
	7	2.9%	28.1%	57.5%	71.9%	**75.6%**
	8	14.9%	41.6%	68.3%	72.7%	**76.4%**
	9	-0.5%	6.1%	61.9%	85.9%	**88.4%**
3	10	-2.3%	-3.6%	-0.7%	**1.1%**	-7.6%
	11	13.3%	47.6%	56.2%	**57.3%**	55.4%
	12	-1.0%	-3.2%	-18.9%	4.6%	**18.8%**
	13	3.0%	16.8%	52.7%	78.2%	**80.6%**
	14	9.3%	30.4%	68.8%	92.9%	**98.2%**
4	15	0.0%	9.7%	22.8%	**40.7%**	37.6%
	16	-4.6%	-0.8%	12.3%	**26.9%**	26.5%
	17	1.3%	13.9%	33.3%	50.8%	**53.1%**
	18	2.8%	7.8%	33.7%	**55.5%**	51.6%
	19	0.0%	0.0%	0.1%	0.1%	**1.2%**
5	20	3.7%	7.5%	20.8%	**24.1%**	20.8%
	21	2.6%	27.9%	68.5%	**80.4%**	77.4%
	22	-1.6%	15.6%	61.5%	80.2%	**87.1%**
	23	-3.7%	5.9%	2.3%	10.9%	**21.9%**
	24	2.5%	6.2%	20.7%	34.2%	**43.3%**

TABLE V: RESULTS WITH ELITISM AND THRESHELD CONVERGENCE

Set	fn	α				
		0.001	0.005	0.01	0.05	0.1
1	1	88.5%	**93.6%**	91.7%	78.4%	69.7%
	2	**4.0%**	-16.8%	-43.7%	-128.8%	-135.2%
	3	11.6%	26.6%	**33.8%**	32.2%	27.4%
	4	9.9%	17.5%	**21.3%**	13.8%	15.3%
	5	-16.7%	65.1%	86.1%	100.0%	**100.0%**
2	6	59.3%	89.8%	**94.6%**	89.0%	90.8%
	7	21.5%	39.5%	**46.0%**	9.9%	5.0%
	8	15.8%	14.3%	**41.5%**	25.1%	40.7%
	9	-11.6%	-75.5%	-88.3%	-171.6%	-354.8%
3	10	**28.0%**	16.4%	10.0%	-47.3%	-76.0%
	11	0.5%	12.5%	15.8%	22.3%	**28.6%**
	12	92.4%	**93.4%**	88.9%	46.9%	23.4%
	13	58.8%	**81.8%**	78.9%	61.2%	60.4%
	14	79.6%	**86.4%**	83.0%	70.1%	65.7%
4	15	7.2%	21.5%	**33.7%**	26.6%	31.3%
	16	17.5%	38.4%	37.5%	**47.1%**	42.1%
	17	-10.0%	11.9%	25.8%	38.1%	**38.8%**
	18	0.6%	17.3%	29.1%	**35.2%**	19.7%
	19	-1.1%	-1.1%	-1.2%	-1.2%	-1.3%
5	20	6.0%	13.2%	19.0%	34.8%	**36.3%**
	21	31.2%	26.3%	33.3%	11.9%	**51.6%**
	22	5.5%	-3.6%	**8.7%**	-58.1%	3.7%
	23	**19.9%**	-0.9%	-2.7%	-14.7%	-21.6%
	24	-2.6%	5.3%	**8.9%**	-10.2%	-16.8%

V. SIMULATED ANNEALING WITH ELITISM

The benchmark implementation of simulated annealing does not include elitism. Like physical annealing, there is no memory in the system – there is only the current state. In this situation, the risks of exploration are much higher. Every attempt to find a better optimum well has an inherent risk of leading to a worse optimum well. Without elitism, the ability to backtrack these steps is not guaranteed, so extra caution should be exercised before large exploratory steps are taken.

In simulated annealing, concurrent local search steps effect a form of caution by reducing the probability of large exploratory steps. Specifically, local search steps which lead to better-than-random samples of the current optimum well will make it more difficult to escape from the current optimum well to explore another (see Fig. 1). Without elitism, this lower level of exploration in baseSA often leads to better results on the multi-modal functions (e.g. BBOB fn 15-24). The idea of "even sampling" presented in Section I implies picking the best from a set of samples, and the current implementation of simulated annealing does not support this assumption.

Elitism can be implemented in simulated annealing by resetting the position to the best known position every r iterations. In the limits, simulated annealing becomes hill climbing when $r = 1$ and elitism has no effect when r is equal to the total number of function evaluations (100,000). In Table IV, the effects of elitism for $r = 1$, 10, 100, 1000, and 10000 are shown as the percent difference ((b-a)/b) between baseSA (b) and baseSA with elitism (a).

In general, the results for simulated annealing improve as the frequency of resets increases. In fact, the best overall results occur with a reset after each iteration – which was originally thought to be the equivalent to hill climbing. However, the difference between "simulated annealing" which never accepts a worsening move and typical implementations of hill climbing is the variable step size. Hill climbing tends to imply a greedy, local search whereas a decreasing step size in baseSA (further) supports a transition from exploration/global search to exploitation/local search.

VI. SIMULATED ANNEALING WITH ELITISM AND THRESHELD CONVERGENCE

Similar to the experiments in Section IV, thresheld convergence has been applied to baseSA with elitism. Building from the best results in Section V with a reset after every iteration (i.e. never accepting a worse move), the results in Table V show the percent difference ((b-a)/b) between baseSA with elitism (b) and simulated annealing with elitism and thresheld convergence (a). The parameters for the threshold function (3) are $\alpha = 0.001$, 0.005, 0.01, 0.05, and 0.1 and $\gamma = 2$.

Exploration for new optimum wells involves the risk of ending up in a worse optimum well. However, this risk is greatly reduced with elitism since the system can always return from the worse optimum well back to the best-known optimum well. As seen in Table IV, the performance of the benchmark implementation of simulated annealing (baseSA) improves with elitism which increases the amount of exploitation in the system. With this increase in exploitation, the performance is further improved by an increase in exploration as effected by the addition of thresheld convergence. Across the full set of BBOB functions, the best result with thresheld convergence delivers statistically significant improvements (as indicated by a t-test with $p < 0.05$) of at least 10% on 18 of 24 functions (see Table VI).

The development of simulated annealing with thresheld convergence is now complete, so comparisons with more

TABLE VI: SUMMARY OF RESULTS

Set	fn	with elitism mean	with elitism stddev	best result mean	best result stddev	α	%-diff	t-test
1	1	2.90e−1	2.22e−1	1.87e−2	9.19e−3	0.005	**93.6%**	**0.00**
	2	2.64e+4	6.30e+3	2.54e+4	9.58e+3	0.001	4.0%	0.33
	3	1.62e+2	4.17e+1	1.07e+2	2.15e+1	0.01	**33.8%**	**0.00**
	4	2.05e+2	5.54e+1	1.61e+2	4.77e+1	0.01	**21.3%**	**0.00**
	5	1.61e+1	6.06e+0	0.00e+0	0.00e+0	0.1	**100.0%**	**0.00**
2	6	8.86e+1	1.15e+2	4.81e+0	7.61e+0	0.01	**94.6%**	**0.00**
	7	1.81e+1	8.44e+0	9.79e+0	5.60e+0	0.01	**46.0%**	**0.00**
	8	1.09e+2	4.07e+1	6.40e+1	5.84e+1	0.01	**41.5%**	**0.00**
	9	1.36e+1	4.87e+0	1.52e+1	1.63e+1	0.001	-11.6%	0.32
3	10	2.11e+4	1.72e+4	1.52e+4	9.78e+3	0.001	28.0%	0.07
	11	2.98e+1	7.08e+0	2.13e+1	9.45e+0	0.1	**28.6%**	**0.00**
	12	1.17e+5	8.04e+4	7.73e+3	3.20e+3	0.005	**93.4%**	**0.00**
	13	1.42e+2	5.32e+1	2.60e+1	8.67e+0	0.005	**81.8%**	**0.00**
	14	1.97e−1	9.91e−2	2.68e−2	7.54e−3	0.005	**86.4%**	**0.00**
4	15	1.48e+2	2.94e+1	9.82e+1	2.56e+1	0.01	**33.7%**	**0.00**
	16	1.23e+1	2.11e+0	6.51e+0	1.90e+0	0.05	**47.1%**	**0.00**
	17	3.27e+0	9.94e−1	2.00e+0	1.14e+0	0.1	**38.8%**	**0.00**
	18	1.12e+1	3.95e+0	7.25e+0	3.50e+0	0.05	**35.2%**	**0.00**
	19	2.47e−1	2.59e−3	2.50e−1	8.37e−4	0.001	-1.1%	0.00
5	20	2.54e+0	3.96e−1	1.62e+0	3.47e−1	0.1	**36.3%**	**0.00**
	21	1.15e+1	9.97e+0	5.55e+0	5.45e+0	0.1	**51.6%**	**0.01**
	22	7.32e+0	6.02e+0	6.68e+0	5.94e+0	0.01	8.7%	0.35
	23	1.56e+0	2.79e−1	1.25e+0	3.07e−1	0.001	**19.9%**	**0.00**
	24	1.18e+2	2.15e+1	1.08e+2	2.71e+1	0.01	8.9%	0.07

TABLE VII: RESULTS VS. MATLAB'S SIMULATED ANNEALING

Set	fn	MATLAB SA mean	MATLAB SA stddev	best result mean	best result stddev	α	%-diff	t-test
1	1	6.93e+0	2.04e+0	1.87e−2	9.19e−3	0.005	99.7%	0.00
	2	2.65e+3	1.39e+3	2.54e+4	9.58e+3	0.001	-855.4%	0.00
	3	1.65e+2	6.17e+1	1.07e+2	2.15e+1	0.01	35.1%	0.00
	4	2.31e+2	6.16e+1	1.61e+2	4.77e+1	0.01	30.1%	0.00
	5	6.49e+1	8.87e+0	0.00e+0	0.00e+0	0.1	100.0%	0.00
2	6	1.10e−1	7.89e−2	4.81e+0	7.61e+0	0.01	-4270.2%	0.00
	7	1.09e+1	4.19e+0	9.79e+0	5.60e+0	0.01	9.9%	0.22
	8	2.82e+1	2.65e+1	6.40e+1	5.84e+1	0.01	-126.9%	0.00
	9	7.34e+0	4.99e+0	1.52e+1	1.63e+1	0.001	-107.3%	0.01
3	10	2.70e+3	1.46e+3	1.52e+4	9.78e+3	0.001	-462.8%	0.00
	11	6.74e+1	2.65e+1	2.13e+1	9.45e+0	0.1	68.4%	0.00
	12	4.52e+0	7.98e+0	7.73e+3	3.20e+3	0.005	-1.7e+5%	0.00
	13	4.88e+0	5.47e+0	2.60e+1	8.67e+0	0.005	-431.6%	0.00
	14	7.31e+0	1.21e+0	2.68e−2	7.54e−3	0.005	99.6%	0.00
4	15	1.84e+2	6.79e+1	9.82e+1	2.56e+1	0.01	**46.6%**	**0.00**
	16	8.30e+0	2.04e+0	6.51e+0	1.90e+0	0.05	**21.6%**	**0.00**
	17	6.55e+0	7.63e−1	2.00e+0	1.14e+0	0.1	**69.5%**	**0.00**
	18	1.69e+1	3.11e+0	7.25e+0	3.50e+0	0.05	**57.0%**	**0.00**
	19	2.50e−1	0.00e+0	2.50e−1	8.37e−4	0.001	0.2%	0.00
5	20	2.23e+0	1.63e-1	1.62e+0	3.47e−1	0.1	**27.2%**	**0.00**
	21	2.64e+1	1.07e+1	5.55e+0	5.45e+0	0.1	**79.0%**	**0.00**
	22	3.70e+1	1.47e+1	6.68e+0	5.94e+0	0.01	**81.9%**	**0.00**
	23	8.51e−1	2.18e−1	1.25e+0	3.07e−1	0.001	-47.1%	0.00
	24	1.56e+2	5.35e+1	1.08e+2	2.71e+1	0.01	**31.0%**	**0.00**

sophisticated versions of simulated annealing are now meaningful. Specifically, MATLAB's `simulannealbnd` function [10] implements adaptive simulated annealing based on [13]. In Table VII, the best result with thresheld convergence is compared against MATLAB's implementation of simulated annealing.

The comparisons show that reannealing and adaptive step sizes can be very effective at improving exploitation (e.g. unimodal functions BBOB fn 10-14). However, exploitation is easily achieved by local optimization (e.g. gradient descent), so the more difficult task is usually exploration. The enhanced exploration provided by thresheld convergence is demonstrated on the multi-modal functions (e.g. BBOB fn 15-24) where statistically significant improvements (as indicated by a t-test with $p < 0.05$) of at least 10% are achieved on 8 of the 10 functions in BBOB sets 4 and 5.

VII. DISCUSSION

Thresheld convergence has similarities to niching (e.g. [14]) and crowding [15]. A key difference is that thresheld convergence affects how new candidate solutions are created as opposed to which candidate solutions are kept (e.g. as part of a population). One advantage of this difference is efficiency – thresheld convergence can be implemented with a single distance measurement [4][8] while niching and crowding can use up to P (the population size) distance measurements [15]. A second advantage is the ability to apply thresheld convergence to non-population-based search techniques such as simulated annealing.

The current application with simulated annealing provides new insights into the operation of thresheld convergence that were not possible with previous implementations [4][8]. Specifically, the founding principle of "even sampling" involves an implicit use of elitism. Without elitism, the risks of exploration become much greater – any attempt to find a better optimum well may sacrifice the opportunity to exploit the current optimum well. The results in Section IV show that the increase in exploration caused by thresheld convergence often leads to worse results in standard simulated annealing which does not have elitism.

The limits on exploration caused by a lack of elitism can also be seen in genetic algorithms (GAs) with generational replacement schemes [16]. Compared to steady-state GAs [17], generational GAs often perform better with smaller crossover rates. This reduced rate of crossover can be viewed as a form of elitism since it increases the chance that (good) solutions can pass unchanged from one generation to the next.

In simulated annealing, it is possible to leave a good optimum well for a worse one. Without elitism to provide a guarantee that these "disruptive" search steps can be undone, it becomes important to limit these steps. Small local search steps which improve the fitness of the current location also reduce the probability that large exploratory steps (which can cause large gains in the evaluation function) will be accepted. To limit "disruption" also limits exploration, and one of the key goals of thresheld convergence is to reduce the interference between the mechanisms of exploration and exploitation that can occur when these processes operate concurrently.

It should be noted that the application of thresheld convergence in simulated annealing completely eliminates all (local) search steps that are smaller than the threshold function. This differs from the first implementation of thresheld convergence to particle swarm optimization [4]. In the PSO implementation, the threshold function prevented a particle from updating its personal best position to be within the

threshold from its local best position. However, positions within the threshold could still be evaluated and stored (in the local best position). Since this PSO implementation allowed local search throughout the search process, it could still converge with relatively large values for α. Applications of thresheld convergence that completely eliminate local search will likely require smaller values for α and a truncated threshold function.

More than α, the key "parameter" in thresheld convergence is the required "gap" between new sample solutions. The threshold function (3) is just a preliminary and relatively crude means of adapting the threshold. The development of adaptive threshold functions is a promising area for future research.

VIII. Summary

Many heuristic search techniques have concurrent processes of exploration and exploitation. Since exploration often depends on estimating the quality of a new region of the search space based on a few random samples, it is important to evaluate these samples fairly. In particular, it is not reasonable to compare a random solution from one region of the search space with a locally optimized solution from another. Concurrent exploitation can thus interfere with exploration. The goal of thresheld convergence is to help separate the processes of exploration and exploitation, and its addition has been successful in search techniques like simulated annealing, particle swarm optimization, and differential evolution.

References

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

[2] S. Chen, K. Miura, and S. Razzaqi, "Analyzing the role of "smart" start points in coarse search-greedy search", ACAL, 2007, pp. 13–24.

[3] S. Kirkpatrick, C.D. Gelatt, Jr., and M.P. Vecchi, "Optimization by simulated annealing," Science, vol. 220, pp. 671–680, 1983.

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

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

[6] 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.

[7] 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.

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

[9] 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.

[10] http://www.mathworks.com/help/toolbox/gads/bq2g2yi-15.html – Nov. 14, 2011.

[11] S. Chen and J. Montgomery, "Selection strategies for initial positions and initial velocities in multi-optima particle swarms," GECCO, 2011, pp. 53–60.

[12] S. Chen and Y. Noa Vargas, "Improving the performance of particle swarms through dimension reductions – a case study with locust swarms," IEEE CEC, 2010, pp. 2950–2957.

[13] L. Ingber, "Adaptive simulated annealing (ASA): Lessons learned," Control and Cybernetics, vol. 25, pp. 33–54, 1996.

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

[15] 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.

[16] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning. Reading, MA: Addison Wesley, 1989.

[17] L.D. Whitley and T. Starkweather, "GENITOR II: a distributed genetic algorithm," J. Experimental and Theoretical Artificial Intelligence, vol. 2, pp. 189–214, 1990.