Simple Adaptive Cognition for PSO

Christopher K. Monson
Google, Inc.
6425 Penn Ave.
Pittsburgh, PA 15206
Email:shiblon@gmail.com

***Abstract*—A simple, effective, computationally cheap, and easily tuned method is presented for improving PSO performance by automatically adapting acceleration coefficients. While this approach can be shown to be effective on its own as a source of swarm diversity on difficult functions, it is also capable of enhancing other adaptive strategies commonly employed with PSO. Significantly, and unlike many other PSO enhancements designed to improve swarm diversity, this approach does not typically harm the performance of the underlying method, allowing it to work well on easy and difficult functions alike.**

I. Introduction

Particle swarm optimization, in its simplest form, consists of initializing a number of "particles" in a limited area of a function's domain, then iteratively updating the state of these particles based on simple rules. These particles keep a modest amount of state, including current position and velocity, the best known position, and the best known function value.

Initialization typically involves scattering the particles uniformly within a cube that is assumed to contain the global optimum and assigning random velocity vectors to each. The basic behavior of particles is described by the following acceleration and velocity equations [12], [24]:

$$\ddot{\mathbf{x}}_{t+1} = \phi_p \, \mathbf{U} \otimes (\mathbf{p}_t - \mathbf{x}_t) + \phi_g \, \mathbf{U} \otimes (\mathbf{g}_t - \mathbf{x}_t) \quad (1)$$

$$\dot{\mathbf{x}}_{t+1} = \omega \dot{\mathbf{x}}_t + \ddot{\mathbf{x}}_{t+1} \quad (2)$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \dot{\mathbf{x}}_{t+1} \quad (3)$$

where the acceleration coefficients in (1) are respectively the "cognitive" and "social" coefficients $\phi_p = \phi_g = 2.05$, $\mathbf{U}$ is a vector whose elements are drawn from a standard uniform distribution each time it is used in a computation, and $\otimes$ represents element-wise vector multiplication. The "momentum" coefficient ω in (2) is typically between 0.4 and 1.0, though values close to 1.0 can cause swarm divergence and velocity explosion[1].

Additionally, after every computation of (2), the absolute value of each component of $\dot{\mathbf{x}}$ is capped by the vector $V_{\max}$, which is typically set to the lengths of the sides of the initialization cube. The diagonal of this cube is called L, and the vector containing the lengths of the cube's sides is $|L|$ (using an element-wise absolute value).

PSO, especially given the simple rules governing individual particle behavior, is surprisingly effective in a variety of optimization settings, including some difficult multi-modal optimization problems. The emergent behavior that is responsible for this success includes a tendency of particles to periodically jump far outside of the current region of exploration ("bursts of outliers") [10], [11], and a tendency of the swarm as a whole to converge to a single point over time [3].

Obviously, each of these behaviors can be good or bad, depending on how and when they are observed. The ability to explore, for example, is of great benefit when trying to avoid getting stuck in local minima, but can be a liability when high levels of precision are desired once the general location of the global minimum is found. Conversely, convergence is beneficial when seeking precision, but converging too early can cause the swarm to get stuck in a local minimum. Balancing these two basic behaviors is critical to a performant swarm and is the subject of a large body of PSO research devoted to swarm diversity.

The method presented here, Simple Adaptive Cognition (SAC) is focused on balancing exploration and convergence by using more of the information available to each particle to guide its motion more intelligently. It is interesting because it does not suffer from some of the common difficulties observed in other diversity-enhancing approaches, such as a tendency to increase diversity too much on easy functions [15], [20] or the introduction of substantial tuning difficulties [23]. SAC is intuitively simple and appealing, requires little if any tuning, uses very little extra particle memory, and does not introduce any new computationally-intense procedures for measuring or improving swarm diversity. Furthermore, it is compatible with many other techniques used to improve PSO; it usually improves on them and does not appear to degrade their performance even on easy functions.

Before fully describing this method, relevant related work will be presented and discussed. As familiarity with the concepts of basic PSO is assumed, these discussions will focus on the broad strokes of various enhancements and will be brief. Simple Adaptive Cognition is then presented and discussed, followed by experimental results on popular benchmark functions before conclusions are drawn and avenues for future work are considered.

[1]The use of a "constriction coefficient" is also popular [3]. It is not explored further here because the use of momentum is equivalent with the right choice of coefficients [7], and because its global affect on the equations makes it much harder to get right when coefficients are changing at every step. This is particularly true of the method described here, which quickly reduces to a set of coefficients that effectively disable constriction altogether.

II. Related Work

The need for balanced diversity in particle swarms, particularly when working with highly multi-modal functions, is universally recognized, and there is therefore a large amount of research on the topic. Though varied, most approaches fit into one of a few high-level categories: altered swarm topology [2], [9], [13], [16], [22], direct diversity detection and injection [4], [15], [20], [23], [28], [29], and methods involving coefficient adjustment [8], [25]–[27], [30].

Each of these approaches, and sometimes several of them together, has its own benefits and liabilities. For example, altered swarm topologies (i.e., anything not fully connected) always reduce the flow of information in a swarm, resulting in slower overall convergence [5], [9], [16], [22]. Sometimes this is desirable (e.g., in highly multimodal functions where convergence is often premature), but often it is not, and it represents yet another decision that requires some a priori global knowledge of the target function (i.e., whether it is close enough to convex that fast convergence is appropriate). Both of these issues become less important when topology is changed in response to swarm performance, TRIBES being one example where performance and tuning considerations are addressed simultaneously by constantly adapting swarm size and topology [2], [19].

Direct diversity detection and injection is also a two-edged sword. The general approch involves applying a diversity measure to the swarm and then using that to inform artificial injection of additional diversity. This can, for example, be accomplished by giving particles volume and allowing them to "bounce" off of one another, as in the Spatial Extension PSO (SEPSO) [15], [20], [28]; in this case the detection is a pairwise distance calculation, and the injection is a reversal of velocity accompanied by a reflection of particle position. Another example of artificial detection and injection is the Attraction and Repulsion PSO (ARPSO) [23], where global swarm entropy triggers a temporary change to a "repulsion" mode in the swarm (where acceleration coefficients temporarily reverse sign).

The measure-and-inject approach is popular because the underlying concept is intuitive: if diversity is too low, it is artificially increased. But, these methods can substantially increase computational requirements due to the need to measure global swarm characteristics (a feature that also makes them difficult to parallelize), and they typically introduce several new diversity parameters and thresholds that must be tuned. Tuning is made more difficult still by their tendency to unconditionally suppress convergence, causing the swarm to lose its ability to drill down to the bottom of good local minima with any precision unless the parameter settings are just right for the application. As was the case with topology, however, more successful methods can be obtained by replacing some of the static tuning with automatic parameter adaptation [20].

Another classic technique for changing swarm convergence behavior involves changing the nature of the acceleration (ϕ_p, ϕ_g) and momentum (ω) coefficients [8], [24] (other popular adaptations may provide additional coefficients [17]). These methods are appealing because their implementation is simple and they do not fundamentally change the underlying equations, making them easier to implement, explain, and reproduce: what is left is still easily recognizable as PSO. These are of particular interest in this work, and there are several classes of them.

One very popular technique for improving swarm behavior is linearly decreasing momentum ω over the course of an optimization run. The velocity update equation (2) is changed to something like the following:

$$\dot{\mathbf{x}}_{t+1} = \left(\left(1 - \frac{t}{T}\right)\omega_0 + \frac{t}{T}\omega_1\right)\dot{\mathbf{x}}_t + \ddot{\mathbf{x}}_{t+1} \tag{4}$$

where T is the maximum number of iterations or function evaluations, ω_0 is the start momentum, and ω_1 is the end momentum. Each of these parameters must be tuned. This linear decrease of momentum over the course of an optimization run has been shown to increase PSO effectiveness and is widely used [8], but interestingly, so has a linear *increase* [18], [30], making it unclear which way it should be altered for any given optimization problem. A typical setup might decrease momentum from 0.9 to 0.4, but the range used is problem- and dimensionality-dependent. Furthermore, this approach attaches more than the usual significance to the number of iterations, which makes it hard to adjust running time without observing surprising effects on performance.

Similar changes to ϕ_g and ϕ_p have been proposed in the literature, typically favoring a linear increase or decrease of these coefficients over time, bounded by a maximum number of iterations [26], [27]. Nonlinear versions based on the same idea have also been proposed [14]. Significant improvements have been reported using these methods, but the complexity of finding useful settings for all of the new parameters introduced is considerable.

It is also possible, though not yet substantiated, that the reason for the success of linear changes in momentum and other coefficients has little to do with the specific schedule and more to do with the fact that they happen to pass through good values during an optimization run. This seems evident in the case of the ongoing ω controversy, but becomes considerably more difficult to see when several coefficients are altered at once. This is an interesting avenue for future research.

In the case of topology adaptation or direct manipulation of swarm diversity, ease of use and improvement of performance are obtained when the method uses current swarm characteristics to inform some type of automated parameter adaptation. In other words, things work better with a feedback loop, and coefficient adjustment may benefit from properly-applied feedback, as well. Ideally, the resulting procedure will be informed by easily-obtained swarm state, minimally invasive, easy to understand, minimally tuned, computationally efficient, and decoupled from other useful parameters like T. This paper proposes one such approach.

TABLE I
BENCHMARK FUNCTIONS, WITH ASSOCIATED PER-DIMENSION INITIALIZATION BOUNDS AND OFFSET VECTOR $\mathbf{c}$.

$$\textbf{Parabola (Sphere):}\ (-50,50) \qquad f_{\mathbf{c}}(\mathbf{x}) = \|\mathbf{x}-\mathbf{c}\|_2^2 \tag{5}$$

$$\textbf{Ackley:}\ (-32.768,32.768) \qquad f_{\mathbf{c}}(\mathbf{x}) = 20 + e - 20\exp\left(\frac{-\|\mathbf{x}-\mathbf{c}\|_2}{5\sqrt{D}}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi(x_i - c_i)\right) \tag{6}$$

$$\textbf{Rastrigin:}\ (-5.12,5.12) \qquad f_{\mathbf{c}}(\mathbf{x}) = \|\mathbf{x}-\mathbf{c}\|_2^2 + 10\sum_{i=1}^{D} 1 - \cos(2\pi(x_i - c_i)) \tag{7}$$

$$\textbf{Rosenbrock:}\ (-100,100) \qquad f_{\mathbf{c}}(\mathbf{x}) = \sum_{i=1}^{D-1} 100\left((x_{i+1} - c_{i+1}) - (x_i - c_i)^2\right)^2 + (x_i - c_i - 1)^2 \quad . \tag{8}$$

III. SIMPLE ADAPTIVE COGNITION

Social algorithms of all kinds make use of a simple and powerful idea: complex and useful emergent behavior can be obtained through the use of simple rules applied by individuals of the group. Fundamentally, this means that the complexity in a social algorithm arises from the way that information moves through the system and how that information is used by individuals. Too often, however, the information traverses the social links without any associated qualitative characteristics that would aid an individual in its use of that information; there is no good way for an individual to assess or act upon the reliability of the information it has just received.

Thus, in PSO and many of its popular variants, there is a strong implicit assumption that one piece of information is very much like another. For example, the best location of a neighbor ($\mathbf{g}$) is not typically viewed any differently than the best remembered location of the individual ($\mathbf{p}$). Intuitively, this assumption is almost never true; no two bits of information have the same utility for the optimization problem. The challenge lies in assigning quantifiable quality to information without requiring an inordinate amount of a priori problem knowledge. Unfortunately, the complete and correct answer to the query "How should I use this information to find the minimum?" ultimately requires prior knowledge of the location of that minimum [21].

Information utility, then, is perhaps too general a characteristic to tackle directly. Information *age*, however, can be a good proxy for utility in the context of particle swarm optimization. If a particle has not updated $\mathbf{p}$ in a large number of algorithm steps, then that is an indication that its best remembered location is not in a region of the function that warrants further exploration; ostensibly the particle that last saw that location has already explored it and not found much of value, else it would have been updated earlier. Using this idea, it is possible to efficiently improve PSO by breaking the assumption that information obtained long ago is precisely as useful as information obtained recently[2].

Simple Adaptive Cognition (SAC) does exactly this. It decays the magnitude of vectors toward locations in proportion to how long ago they were discovered. Specifically, the acceleration update equation (1) is changed as follows:

$$\ddot{\mathbf{x}}_{t+1} = \phi_p \gamma^{t-t_p}\, \mathbf{U} \otimes (\mathbf{p}_t - \mathbf{x}_t) + \phi_g \gamma^{t-t_g}\, \mathbf{U} \otimes (\mathbf{g}_t - \mathbf{x}_t) \tag{9}$$

In the above equation, one new parameter and two new measurement variables have been introduced. The adaptation weight $\gamma \in (0,1)$ is typically close to 1.0 (e.g., 0.999), and controls how fast a value is suppressed as the time between updates goes up. The time of the last update of $\mathbf{p}$ is t_p, and the time of the last update of $\mathbf{g}$ is t_g. Each time a particle updates its own $\mathbf{p}_t$ (i.e., whenever $\mathbf{p}_t \neq \mathbf{p}_{t-1}$), it also sets $t_p = t$. Otherwise it is left unchanged. Note that letting $\gamma = 1.0$ restores standard PSO behavior.

The intuitive appeal of this algorithm is obvious: if a best location has not been updated recently, then the vector toward that location tends to have a smaller magnitude. Effectively, this keeps a particle from favoring that location based on how long ago it was discovered. It has practical appeal, as well, since in the case of easy functions, updates are frequent, causing the exponent of γ to be small, restoring standard (and highly effective) PSO behavior for those functions. From there it scales naturally to more difficult functions where updates can happen less frequently.

Furthermore, it has been noted that one of PSO's strengths is its ability to generate chaotic motion at opportune times [3], [10], [11]. Usually, as swarms converge, particles slow down, and the chaotic behavior is naturally suppressed. This simple adaptation tends to selectively restore that behavior if the particles are learning new things as they converge (and not merely converging to stagnant values) by suddenly lengthening the acceleration vector for each particle affected by a recent update. This allows adapted swarms to continue making progress long after other variants would permanently stagnate.

IV. EXPERIMENTS

Simple Adaptive Cognition was tested on some common 100-dimensional benchmark functions in conjunction with several other PSO variants, including standard, decreasing momentum, linearly adapted acceleration coefficients [27], and various parameterizations of particle bouncing [20]. In the case of linear changes in acceleration coefficients, the behavior was

[2]Note that particle age has been explored before [6], but the previous work changed age without regard to fitness: there was no feedback.

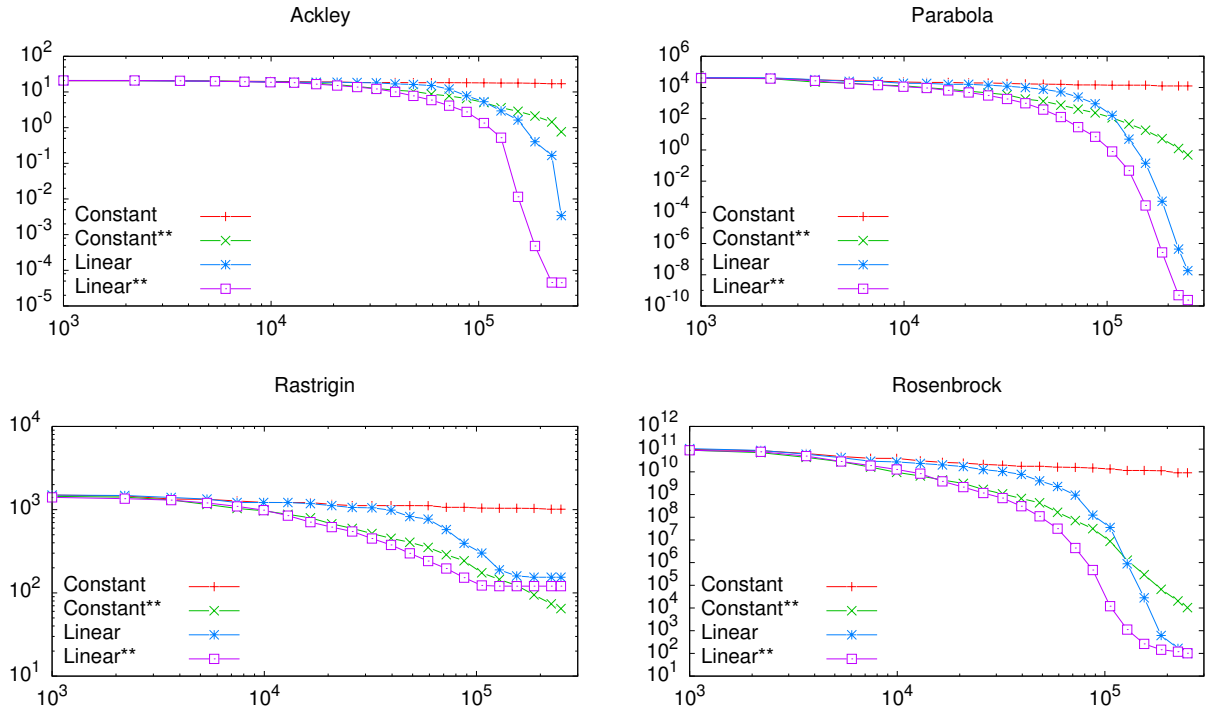

Fig. 1. Performance of constant and linear momentum strategies ("**" denotes use of SAC). X-axis is function evaluations (log-log scale).

sufficiently poor in this setup as to not warrant inclusion in the displayed results; it typically performed only marginally better than standard PSO, and thus occupied the same space on the graphs.

A. Benchmarks

The common benchmark functions selected for comparison are defined in Table I with their associated per-dimension initialization bounds. Because all of the functions have their minimum at or near the origin, each function was shifted to ensure that good results were not obtained because of accidentally-introduced origin-seeking behavior [19]. This involved offsetting the domain in each dimension by $0.25|L|$, e.g., for Parabola, every element of $\mathbf{c}$ is 25.

B. Setup

For all experiments, the dimensionality $D = 100$, the number of particles $N = 5$, and the neighborhood topology is fully-connected[3]. The initialization cube is specified for each benchmark function in Table I, and each initial velocity is generated by multiplying each element of the diagonal vector L by a uniform draw from the interval $(-1, 1)$. For each experiment, the maximum number of function evaluations is $T = 250000$. The motion equations used are repeated here for convenient reference:

$$\ddot{\mathbf{x}}_{t+1} = \phi_p \gamma^{t-t_p}\, \mathbf{U} \otimes (\mathbf{p}_t - \mathbf{x}_t) + \phi_g \gamma^{t-t_g}\, \mathbf{U} \otimes (\mathbf{g}_t - \mathbf{x}_t) \quad (10)$$

$$\dot{\mathbf{x}}_{t+1} = \left(\left(1 - \frac{t}{T}\right)\omega_0 + \frac{t}{T}\omega_1\right)\dot{\mathbf{x}}_t + \ddot{\mathbf{x}}_{t+1} \quad (11)$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \dot{\mathbf{x}}_{t+1} \ . \quad (12)$$

In all cases, $\phi_p = \phi_g = 2.05$.

Where particle bouncing is indicated, it follows the Contracting Radius, Increasing Bounce SEPSO (CRIBS) algorithm [20]. Specifically, a collision occurs when the following is satisfied for particle i:

$$\exists_{j \neq i} . \|\mathbf{x}_i - \mathbf{x}_j\| < (\beta^{b_i} + \beta^{b_j}) r \quad (13)$$

with $\beta = 0.9$, $r = 0.1\|L\|_2$, and b is the number of times a particle has experienced a collision. Note that letting $r = 0$ restores standard non-bouncing behavior, since the above inequality will never be satisfied in that case.

When a collision is detected, the velocity and position of the corresponding particle are changed according to

$$\dot{\mathbf{x}}_{t+1} = -\dot{\mathbf{x}}_{t+1} \quad (14)$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \beta^{-b}(\mathbf{x}_{t+1} - \mathbf{x}_t) \ . \quad (15)$$

Because the methodologies here make use of momentum instead of constriction [3], the absolute value of the velocity vector must be capped at each iteration using the well-known vector $V_{\max}$, each element of which is set to $0.5|L|$ (e.g., for Parabola, every element of $V_{\max}$ is 50).

[3]The dimensionality and swarm size often vary greatly in the literature, as does the selection of topology. No universally correct settings are known for any of these, though some attempts have been made to define standard starting points [1]. Though there is not room to adequately demonstrate it in this work, further testing verified that the algorithm presented here also works well on problems of lower dimensionality and with larger swarm sizes.

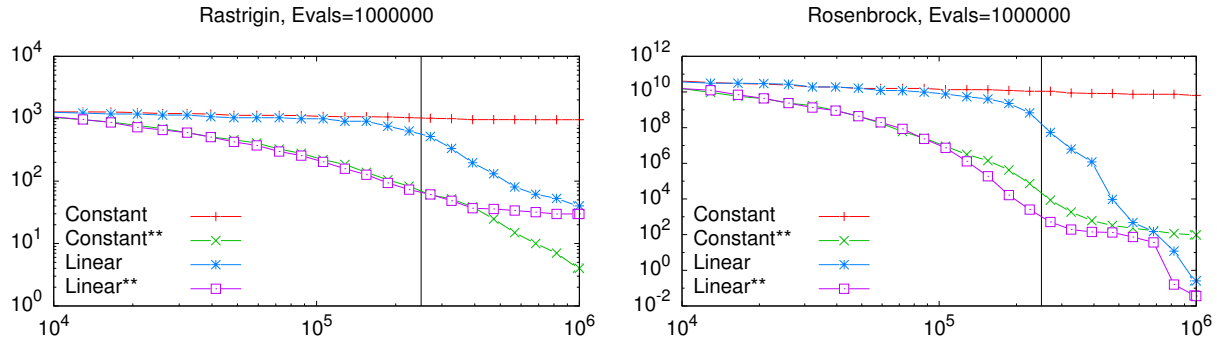

Fig. 2. Performance of best run of momentum strategies ("**" denotes use of SAC) on Rastrigin and Rosenbrock with 1 million function evaluations. The vertical lines denote 250000 evaluations.

C. Results and Analysis

As can be seen in the definitions given in Section IV-B, the various approaches tested, including SAC, are compatible with one another and can be employed in any number of combinations.

For example, it is possible to use any combination of adaptive coefficients and momentum strategies, some of which are shown Fig. 1. In these and all graphs, unless otherwise specified, the values shown are the minimum value of the best particle in the swarm over 20 independent runs. In this particular figure, the following settings apply:

- Constant Momentum: $\omega_0 = \omega_1 = 0.75$
- Linear Momentum: $\omega_0 = 0.75, \omega_1 = 0.4$
- Constant Cognition: $\gamma = 1.0$
- Adaptive Cognition: $\gamma = 0.999$

Again looking at Fig. 1, the use of SAC can improve performance in two important ways: evaluations required to begin obtaining good values, and the final value obtained at the end of the run. In all cases, stopping the process early produces better results when using SAC. This behavior is particularly interesting for the function Parabola, since it is an "easy" function that does not typically respond well to diversity-increasing techniques, as they tend to slow down convergence [20].

In addition to finding better values sooner, SAC also performs as well as or better than its corresponding baseline when the algorithm terminates. In the case of the Rosenbrock function, however, it appears that the standard linearly-decreasing momentum is about to overtake it, suggesting that the algorithm should be run for more iterations before drawing any definitive conclusions. The result of doing this is shown in Fig. 2, where the number of function evaluations T has been increased to 1 million.

The vertical line in Fig. 2 indicates 250000 function evaluations, where previous graphs ended. As might be expected, the constant momentum entries are simply continuing along their previous trajectories; the adaptive version continues to improve while the non-adaptive version continues to stagnate. The linear momentum entries, however, are different: performance is stretched out along the x-axis; at 250000 evaluations, they have not achieved values equal to the earlier experiments in Fig. 1. Given the momentum schedule's dependence on T, this is hardly surprising. They do at least eventually surpass their previous performance, but do not improve on their SAC variants.

While stretching the momentum schedule was ultimately beneficial in this case, it could just as easily have been detrimental to do so; it is always naive to assume, when using a linear schedule tied to T on any coefficient, that one can simply extend the number of function evaluations and watch the swarm continue progressing as before. This is a significant shortcoming of the linear schedule: it conflates a useful and intuitive parameter (T) with a set of parameters that are harder to tune (the coefficient schedule), often resulting in surprising behavior.

The linear schedule for momentum appears to be detrimental when applied to Rastrigin. It is possible that the momentum is simply too small by the time the algorithm is halfway done, indicating that the end momentum ω_1 is probably too low. This is one example of the difficulty inherent in tuning that parameter, and learning to adapt that as well is a subject of future research.

Putting momentum concerns aside for a moment, it is also instructive to compare the behavior of SAC in conjunction with methods that directly measure and increase swarm diversity, such as bouncing particles. Fig. 3 shows the various combinations (bouncing is disabled when $r = 0$ and enabled when $r > 0$ as described previously).

Once again, the use of SAC improves either the bouncing or non-bouncing strategy in both initial speed of improvement and quality of final result. With Rastrigin and Rosenbrock, the adaptive and non-adaptive cases appear to approach one another near the end, but this is less striking once the log scale is noted. In any case, the addition of SAC simultaneously does no harm and allows the algorithm to be stopped much earlier while still obtaining reasonable results.

The behavior of the swarm on Parabola remains notable: not only does SAC improve results on Parabola, it completely obviates the need for bouncing at all. This is notably true for Ackley, as well, which is a difficult multi-modal function.

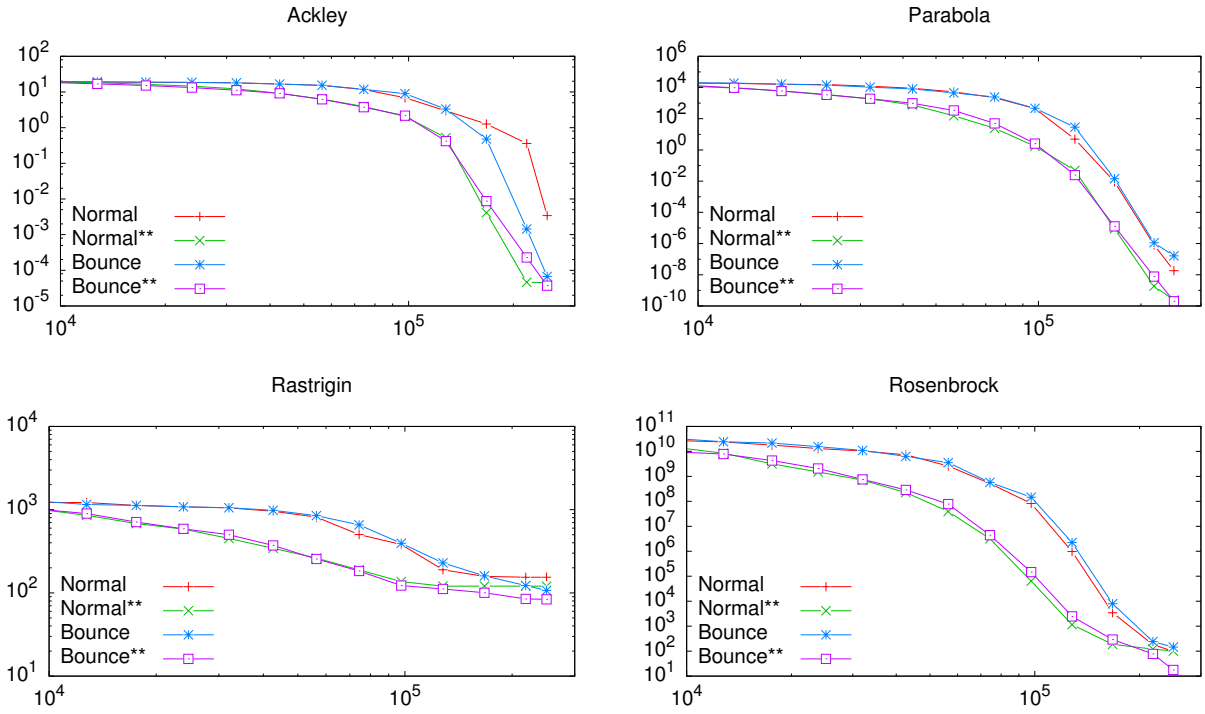

Fig. 3. Performance of bouncing ("**" denotes use of SAC). X-axis is function evaluations (log-log scale).

Finally, over all methods evaluated, the best runs consistently came from a method that was combined with SAC, as shown in Fig. 4. Furthermore, the methods used in conjunction with SAC were almost always more consistent (sometimes substantially so) in their performance than methods used without it, shown in Fig. 5. The one notable exception is on the Ackley function, where there is no disernible difference in either case.

V. Conclusion and Future Work

Particle swarm algorithms implicitly assign a sort of quality measure to the information obtained by each particle: the age of the best known location. Making use of that quality to inform a particle's motion is the underlying principle behind Simple Adaptive Cognition (SAC).

As the magnitudes of vectors toward old information are suppressed, the particles in the swarm tend to avoid less interesting regions of the domain and converge instead toward more recently-updated locations, often providing substantial benefits when optimizing both easy and hard functions alike.

This simple approach is intuitive, consistently performant, and practically cost-free in terms of computation and memory. Furthermore, it combines elegantly with and improves upon other PSO variants. Also, when information is updated frequently, it reduces gracefully to the behavior of the underlying method. This particular feature allows it to perform as well as or better than standard PSO on easy functions, and allows it to serve as a non-intrusive enhancement for harder ones, making it a compelling addition to any compatible PSO algorithm. Finally, the fact that it introduces exactly one new parameter that appears to require little if any tuning is an added bonus.

The study of this approach has opened up a few new questions, the pursuit of which is left for future work. For example, the new interactions between momentum and the now-decreasing coefficients need to be studied in more detail. This was particularly evident when optimizing Rastrigin.

Additionally, having applied the concept of information quality to the acceleration coefficients in SAC, it seems natural to wonder whether the momentum term in the velocity equation would benefit from similar treatment. The linear schedule has some definite drawbacks, so making momentum dependent on intrinsic information has obvious appeal. As there is no update rule in direct evidence there, the selection of a reasonable proxy for information quality is not as straightforward.

Finally, how this approach will fare on fully informed swarms [17] is an open question, since they tend to provide many more disparate sources of information to each particle. This and other questions remain to be studied in detail, but until then the method is simple and compelling enough to make it a solid addition to any swarm optimization toolbox.

References

[1] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS 2007)*, Honolulu, HI, 2007, pp. 120–127.

[2] M. Clerc, "TRIBES - un exemple d'optimisation par essaim particulaire sans paramètres de contrôle," in *Optimisation par Essaim Particulaire (OEP 2003)*, Paris, France, 2003.

[3] M. Clerc and J. Kennedy, "The particle swarm: Explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, February 2002.

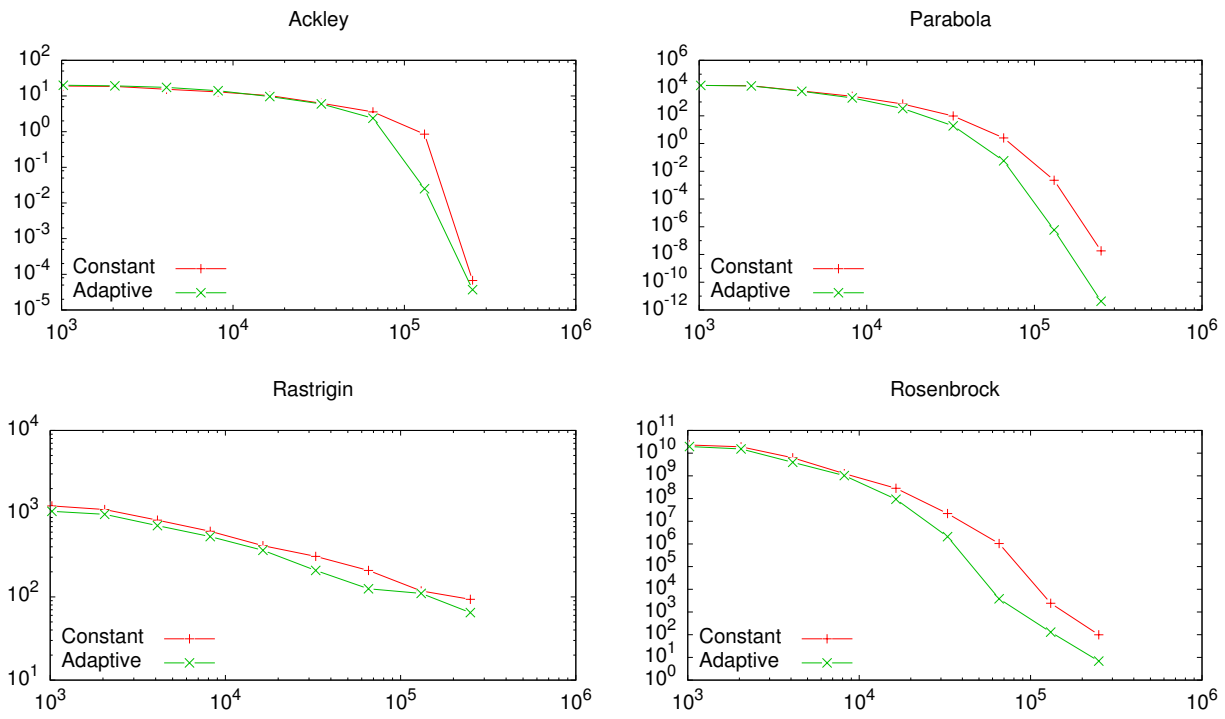

Fig. 4. Best run over all algorithms with and without SAC. X-axis is function evaluations (log-log scale).

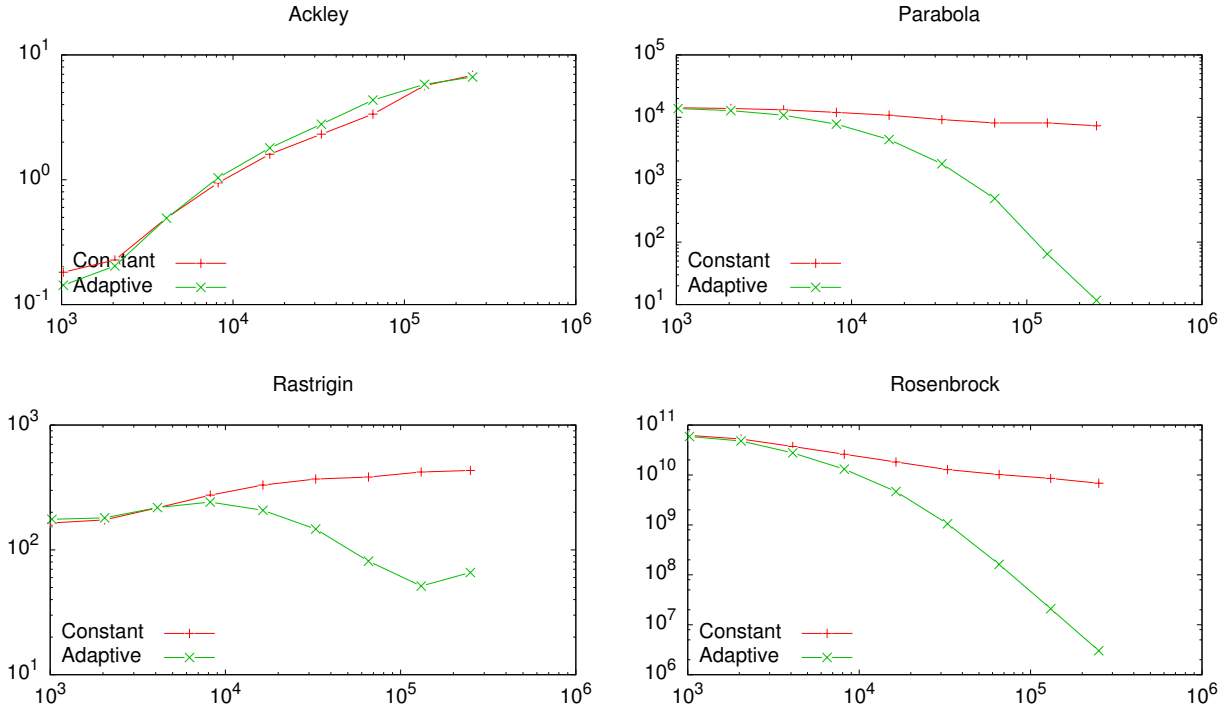

Fig. 5. Standard deviation across all algorithms with and without SAC plotted against function evaluations (log-log scale).

[4] Z. Cui, J. Zeng, and X. Cai, "A new stochastic particle swarm optimizer," in *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*. Portland, Oregon: IEEE Press, 20-23 June 2004, pp. 316–319.

[5] M. A. M. de Oca, T. Stützle, M. Birattari, and M. Dorigo, "Frankenstein's pso: A composite particle swarm optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1120–1132, October 2009.

[6] S. Dehuri, A. Gosh, and R. Mall, "Particles with age for data clustering," in *Proceedings of the 9th International Conference on Information Technology (ICIT)*, Bhubaneswar, December 2006, pp. 221–224.

[7] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2000)*, San Diego, California, 2000, pp. 84–88.

[8] ——, "Particle swarm optimization: Developments, applications, and resources," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2001)*, Seoul, Korea, 2001.

[9] J. Kennedy, "Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance," in *Proceedings of the Congress of Evolutionary Computation*, P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and Z. Zalzala, Eds., vol. 3. IEEE Press, 1999, pp. 1931–1938.

[10] ——, "Probability and dynamics in the particle swarm," in *Proceedings of the Congress on Evolutionary Computation (CEC 2004)*, vol. 1, June 2004, pp. 340–347.

[11] ——, "Dynamic-probabilistic particle swarms," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2005)*, vol. 1, June 2005, pp. 201–207.

[12] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *International Conference on Neural Networks IV*. Piscataway, NJ: IEEE Service Center, 1995, pp. 1942–1948.

[13] J. Kennedy and R. Mendes, "Neighborhood topologies in fully-informed and best-of-neighborhood particle swarms," in *Proceedings of the 2003 IEEE SMC Workshop on Soft Computing in Industrial Applications (SMCia03)*. Binghamton, New York: IEEE Computer Society, June 2003.

[14] C.-N. Ko, Y.-P. Chang, and C.-J. Wu, "An orthogonal-array-based particle swarm optimizer with nonlinear time-varying evolution," *Applied Mathematics and Computation*, vol. 191, no. 1, pp. 272–279, August 2007.

[15] T. Krink, J. S. Vestertroem, and J. Riget, "Particle swarm optimisation with spatial particle extension," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, Honolulu, Hawaii, 2002.

[16] R. Mendes, J. Kennedy, and J. Neves, "Watch thy neighbor or how the swarm can learn from its environment," in *Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003)*, Indianapolis, Indiana, 2003, pp. 88–94.

[17] ——, "The fully informed particle swarm: Simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, June 2004.

[18] C. K. Monson and K. D. Seppi, "Bayesian optimization models for particle swarms," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2005)*, vol. 1, Washington, D.C., 2005, pp. 193–200.

[19] ——, "Exposing origin-seeking bias in pso," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2005)*, vol. 1, Washington, D.C., 2005, pp. 241–248.

[20] ——, "Adaptive diversity in pso," in *Proceedings of the 8th Annual conference on Genetic and Evolutionary Computation (GECCO 2006)*. Seattle, Washington: ACM, 2006, pp. 59–66.

[21] ——, "A graphical model for evolutionary optimization," *Evolutionary Computation*, vol. 16, no. 3, pp. 289–313, 2008.

[22] M. Richards and D. Ventura, "Dynamic sociometry in particle swarm optimization," in *International Conference on Computational Intelligence and Natural Computing*, September 2003.

[23] J. Riget and J. S. Vesterstrøm, "A diversity-guided particle swarm optimizer — the ARPSO," Department of Computer Science, University of Aarhus, Tech. Rep. 2002-02, 2002.

[24] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1998)*, Piscataway, New Jersey, 1998.

[25] ——, "Particle swarm optimization with fuzzy adaptive inertia weight," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2001)*, Seoul, Korea, 2001.

[26] P. K. Tripathi, S. Bandyopadhyay, and S. K. Pal, "Multi-objective particle warm optimization with time variant inertia and acceleration coefficients," *Information Sciences*, vol. 177, no. 22, pp. 5033–5049, November 2007.

[27] Z. Wu and J. Zhou, "A self-adaptive particle swarm optimization algorithm with individual coefficients adjustment," in *2007 International Conference on Computational Intelligence and Security (CIS 2007)*. Harbin, China: IEEE, 2007, pp. 133–136.

[28] J. Zhang, Y. Tan, and X. He, "Concentric spatial extension based particle swarm optimization inspired by brood sorting in ant colonies," in *Swarm Intelligence Symposium (SIS 2009)*. Nashville, Tennessee: IEEE, 2009, pp. 9–15.

[29] S.-Z. Zhao and P. N. Suganthan, "Diversity enhanced particle swarm optimizer for global optimization of multimodal problems," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2009)*. Trondheim, Norway: IEEE, 2009, pp. 590–597.

[30] Y. Zheng, L. Ma, L. Zhang, and J. Qian, "Empirical study of particle swarm optimizer with an increasing inertia weight," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2003)*, Canbella, Australia, 2003, pp. 221–226.