Adaptive Diversity in PSO

Christopher K. Monson
Brigham Young University
Computer Science Department
3361 TMCB, Provo, UT 84602
c@cs.byu.edu

Kevin D. Seppi
Brigham Young University
Computer Science Department
3330 TMCB, Provo, UT 84602
k@cs.byu.edu

ABSTRACT

Spatial Extension PSO (SEPSO) and Attractive-Repulsive PSO (ARPSO) are methods for artificial injection of diversity into particle swarm optimizers that are intended to encourage converged swarms to engage in exploration. While simple to implement, effective when tuned correctly, and benefiting from intuitive appeal, SEPSO behavior can be improved by adapting its radius and bounce parameters in response to collisions. In fact, adaptation can allow SEPSO to compete with and outperform ARPSO. The adaptation strategies presented here are simple to implement, easy to tune, and retain SEPSO's intuitive appeal.

Track Category

Ant Colony Optimization and Swarm Intelligence

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—
nonlinear programming, unconstrained optimization

General Terms

Algorithms

Keywords

Swarm Intelligence, Adaptation, Optimization

1. INTRODUCTION

Particle Swarm Optimization (PSO) is a social or evolutionary optimization algorithm that was discovered during experiments with simulated bird flocking [6]. Its discovery has led to an algorithm which has gained popularity in recent years for its simplicity, relatively small number of tuning parameters, and surprising effectiveness on a large class of functions.

Classical PSO begins by scattering particles in the function domain space, often by means of a uniform distribution bounded by a function-specific region of feasibility. Each particle is a data structure that maintains its current position $\mathbf{x}$ and its current velocity $\dot{\mathbf{x}}$. Additionally, each particle remembers the most fit position it has obtained in the past, denoted $\mathbf{p}$ for "personal best". The most fit $\mathbf{p}$ among all particles is written $\mathbf{g}$ for "global best".

A valuable variant on classical approaches is *constricted PSO*, where each particle updates its state using the following equations (written in a slightly non-traditional way to accentuate the role of acceleration):

$$\ddot{\mathbf{x}}_{t+1} = \phi_1\,\mathrm{U}() \otimes (\mathbf{p} - \mathbf{x}_t) + \phi_2\,\mathrm{U}() \otimes (\mathbf{g} - \mathbf{x}_t) \tag{1}$$

$$\dot{\mathbf{x}}_{t+1} = \chi\,(\dot{\mathbf{x}}_t + \ddot{\mathbf{x}}_{t+1}) \tag{2}$$

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \dot{\mathbf{x}}_{t+1} \tag{3}$$

where $\phi_1 = \phi_2 = 2.05$, U() is a vector whose elements are drawn from a standard uniform distribution, and $\otimes$ represents element-wise multiplication. The constriction coefficient χ is in this case defined to be

$$\chi = \frac{2\kappa}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|} \tag{4}$$

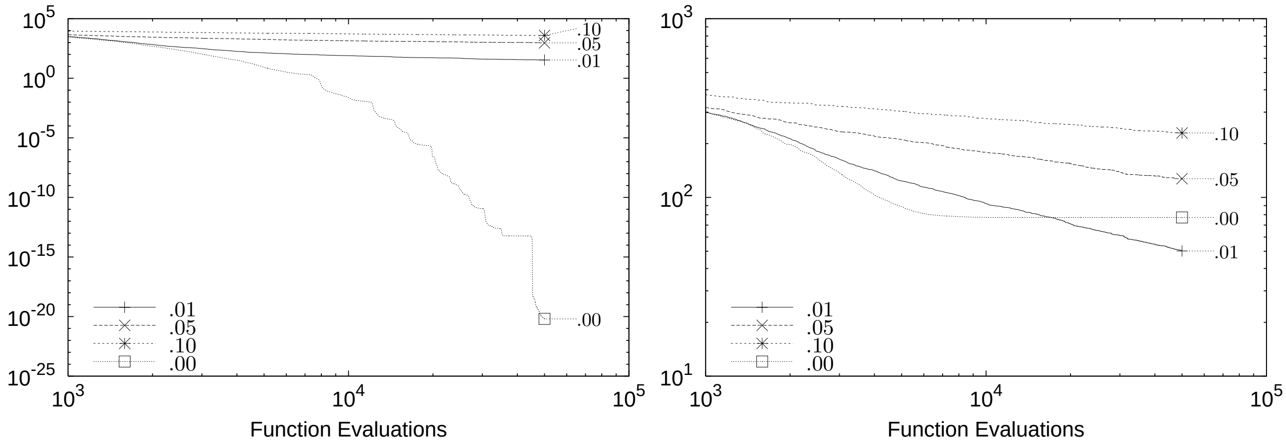
where $\kappa = 1.0$ and $\phi = \phi_1 + \phi_2$ [3].

Though effective, PSO sometimes suffers from premature convergence on problems with many local minima. Convergence is in general a desirable property, allowing the swarm to search regions near the global minimum at increasing levels of detail as time progresses. Unfortunately, in the context of many local minima, the convergence property may cause a swarm to become trapped in one of them and fail to explore more promising neighboring minima.

Designers of optimization algorithms therefore face a fundamental tradeoff: search the current local minimum in detail through quick convergence, or consume resources exploring other areas of the domain [9]. In an effort to handle this tradeoff more explicitly in PSO, some notable diversity-increasing approaches have been proposed. One such approach, the Spatial Extension PSO (SEPSO), involves endowing each particle with a radius, then causing particles to bounce off of one another [7]. A related approach, called Attractive-Repulsive PSO (ARPSO), measures the global diversity of the swarm, triggering modes of global attraction or repulsion when it crosses predefined thresholds [9]. Though effective when well-tuned, finding good function-specific tuning parameters for these methods is non-trivial.

The tuning parameters in SEPSO and ARPSO alike represent a threshold that dictates when diversity will be artificially added to either a single particle or to the swarm as a whole, respectively. Especially in the case of SEPSO, the

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.



(a) Sphere

(b) Rastrigin

Figure 1: Spatial Extension PSO (SEPSO) with multiple radius settings

threshold is easier to tune and the algorithm's performance improves when a simple adaptation strategy is applied.

We begin by describing SEPSO and demonstrating the issues implicit in setting its radius parameter. We then describe the proposed adaptation methodology used to improve robustness of parameters and performance on multimodal functions. We then briefly describe ARPSO, a successor to SEPSO that is less amenable to improvements using our adaptation strategy and that rarely outperforms the easily implemented SEPSO extensions presented here.

2. SPATIAL EXTENSION PSO

The Spatial Extension PSO (SEPSO) is a simple method of artificially injecting diversity into a swarm. While in classical PSO, particles are conceptually volumeless and therefore never collide with one another, the basic premise of SEPSO is that particles have a spherical volume that is defined by a radius r. Two particles i and j collide when

$$\|\mathbf{x}_i - \mathbf{x}_j\|_2 \leq 2r \ . \tag{5}$$

In the event of a collision, the involved particles "bounce" backwards, effectively moving to a point that is formed by reflecting the intended current position about the previous position, optionally reversing the velocity as well to create a post-bounce position $\mathbf{x}'_{t+1}$ and velocity $\dot{\mathbf{x}}'_{t+1}$:

$$\dot{\mathbf{x}}'_{t+1} = -\dot{\mathbf{x}}_{t+1} \tag{6}$$

$$\mathbf{x}'_{t+1} = \mathbf{x}_t - (\mathbf{x}_{t+1} - \mathbf{x}_t) \ . \tag{7}$$

This approach is simple to implement and has intuitive appeal: if a particle is very close to its neighbors, it is likely to be duplicating work by exploring regions that are covered by other particles and should therefore move away from them. The combination of a radius with associated notions of collisions and bouncing is an effective and intuitive way to accomplish this goal.

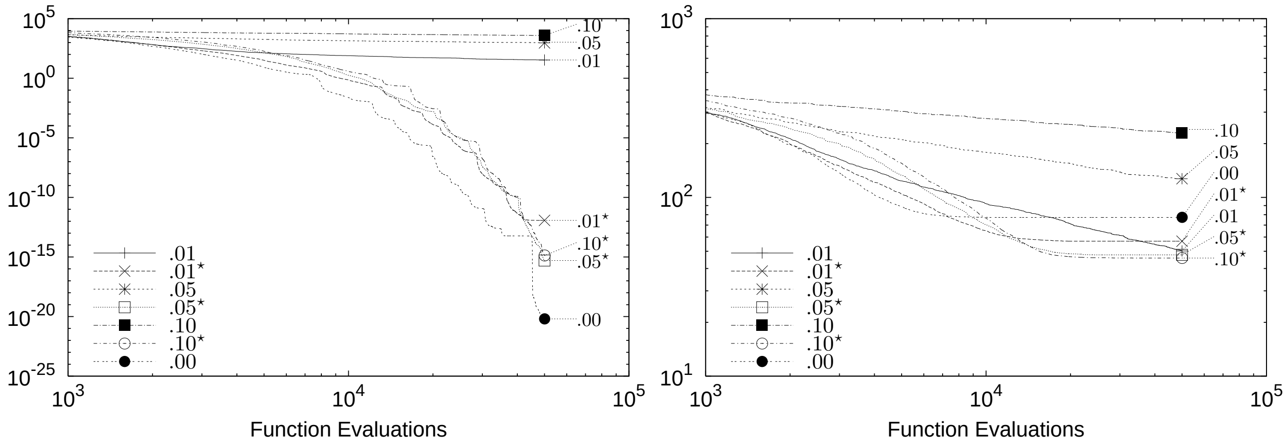
This method of increasing diversity is also appealing because it can be applied to nearly any variant of PSO, including those that do not have an explicit notion of velocity (e.g. Bare Bones PSO [5]): the new location is calculated according to the specified PSO algorithm, then tested against all other new locations; if a collision occurs, that location is reflected before the particle's state changes. Again, velocity may optionally be reversed when present.

The choice of radius r, though not addressed in the original SEPSO work [7], is critical to the performance of the algorithm. Consider Figure 1, which illustrates the relative performance of different radius settings. The radius is set to a constant fraction of the length L of the longest diagonal of the feasible regions for Sphere and Rastrigin (Defined in Table 1). Unless otherwise stated, all figures are generated by averaging 30 runs with constricted PSO as the baseline motion, $D = 30$ dimensions, a fully-connected swarm of size 20, and velocity reversal in the event of SEPSO collisions.

The figure matches intuition. On the simple unimodal function Sphere, for which PSO is already an efficient optimizer, bouncing can only slow down desirable convergence, thereby hurting performance. As the collision radius is decreased, performance gets closer to that achieved by the baseline motion. On the highly multimodal Rastrigin, however, bouncing can be helpful, avoiding the stagnation to which PSO is generally prone for such functions. In this case, a setting of $r = .01L$ represents an improvement over baseline PSO, and the trend that is evident in the radius setting seems to indicate that a smaller radius would provide even better performance. This trend cannot continue indefinitely, however, as setting the radius to 0 simply reproduces the behavior of classical PSO. Finding a good setting for the radius is therefore a problem-dependent exercise; multiple runs may be required to obtain a useful value.

2.1 Adaptive Radius

Convergence, as previously discussed, is a desirable property for PSO, since particles will tend to explore small regions in greater detail as they begin to move more slowly and to converge on a single point in space. This detailed exploration can be important since the scale of the global minimum may not be known before PSO is applied. SEPSO,



(a) Sphere (b) Rastrigin

Figure 2: Contracting Radius SEPSO (CRS) with fixed and adaptive$^{(\star)}$ radius settings

unfortunately, frequently prevents not only premature convergence but useful and appropriate convergence as well.

This is entirely due to the fact that the radius is fixed: whatever else the particles may be doing, they always bounce when within a predefined distance ($2r$) of one another, effectively limiting the scale of the space that may be searched: if they are trying to explore a detailed region of space but are thwarted by collisions, that region will remain unexplored unless a fortunate accident occurs.

Both problems are addressed by giving each particle an individual, adaptable radius. In this case, the detection of a collision causes particles to bounce as before, but the radius of colliding particles is also decreased to make bouncing less likely in the future. This allows particles to escape local minima into which they may become trapped while admitting exploration at increasing levels of detail as time progresses. This idea can be implemented by defining a global adaptation constant $\gamma \in [0, 1]$ and an individual bounce count b for each particle. Each particle's bounce count is initialized to 0 and is incremented whenever the particle is involved in a collision. Collision between particles i and j occurs when

$$\|\mathbf{x}_i - \mathbf{x}_j\| \leq (\gamma^{b_i} + \gamma^{b_j})r \ . \tag{8}$$

No change is made to (6) or (7), leaving bouncing mechanics intact and introducing negligible computational overhead. Adapting the radius in this way results in a new algorithm: the "Contracting Radius SEPSO" (CRS). Results of this approach with $\gamma = 0.8$ and various radius settings are demonstrated in Figure 2, and more will be given later. The superscript * indicates an adaptive result.

Note that when applied to Sphere, CRS performs more closely to the baseline ($r = 0$), which is not unexpected. The radius decreases every time a particle bounces, making it less likely to collide with other particles as time progresses. As a result, the swarm regains the ability to converge, though it does so more slowly than before. Notice also that on Rastrigin the adaptive versions all perform better than the baseline and are clustered more closely (note especially the log scale) around lower values than their non-adaptive counterparts.

Unfortunately, the adaptive version still suffers from premature convergence on Rastrigin. Standard SEPSO with $r = 0.01L$ not only eventually overtakes all of the adaptive versions, it also continues on a downward trend.

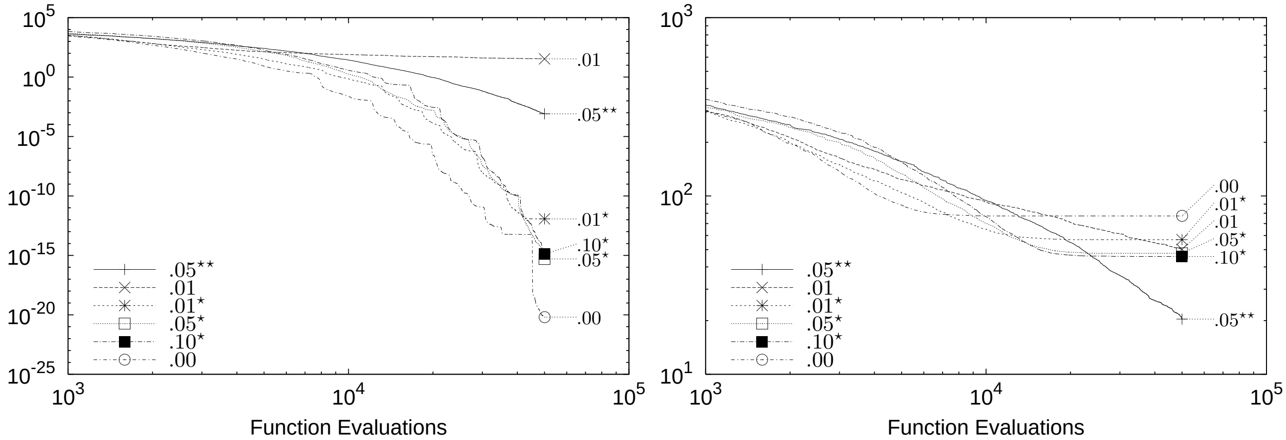
2.2 Adaptive Distance

CRS's observed premature convergence behavior is present in other multimodal functions, prompting an additional extension to CRS: the "Contracting Radius, Increasing Bounce SEPSO" (CRIBS), where individual bounce distance is also adapted. Although various bounce distances have been attempted by the SEPSO authors without noticeable improvement [7], individually increasing particle bounce distance while decreasing collision radius has merit; as particles converge, their diversity decreases and the locations to which they bounce will tend to be in the same local minimum. Therefore, as the adaptive radius *decreases*, a good indicator for convergence, the bounce distance should *increase* to make the act of bouncing more effective. This is accomplished through a simple change to (7):

$$\mathbf{x}'_{t+1} = \mathbf{x}_t - \gamma^{-b}(\mathbf{x}_{t+1} - \mathbf{x}_t) \tag{9}$$

In other words, while the radius is *decreased* via multiplication by γ^b, the distance is similarly *increased* by γ^{-b}. Employing a bounce distance that is inversely proportional to collision radius may be expected to hurt performance on unimodal functions by wasting function evaluations on distant points; however, it should be expected to improve performance on multimodal functions by increasing each particle's odds of escaping a local minimum. These predictions are verified in Figure 3 where it is shown that performance suffers for Sphere while significantly improving for Rastrigin.

It should be noted that while only one radius setting is shown for CRIBS to avoid clutter, far more data were collected than can be presented in this setting. Those data make it clear that the initial radius becomes less important when adaptation is present; on Rastrigin, for example, CRIBS always outperformed CRS by a large margin.



(a) Sphere

(b) Rastrigin

Figure 3: Contracting Radius, Increasing Bounce SEPSO (CRIBS) with adaptive radius$^{(\star)}$ and distance$^{(\star\star)}$

2.3 Remarks and Additional Results

The robustness of the initial parameter settings is affected by adapting those parameters over time. In the case of SEPSO, the radius setting has a dramatic impact on the performance of the algorithm, and it is clear in Figure 1 that the parameters selected for the experiment are not low enough for Sphere but are beginning to approach appropriate values for Rastrigin. Adaptation, however, makes the choice of initial radius far less important, as illustrated in Figure 2. In each case, adapting the radius evens out the differences between the initial parameter settings, allowing them all to perform reasonably well.

In the case of multimodal functions, adapting the distance is productive because it allows a slow-moving, nearly-converged particle to jump out of its current local minimum, facilitating search in other areas of the domain. Significantly, even CRIBS retains the ability to converge, but does so more slowly than CRS or baseline PSO.

Results for the benchmarks defined in Table 1 are found in Figure 4. DeJongF4, like Sphere, is smooth and unimodal. Griewank, while multimodal, begins to appear unimodal as the dimensionality increases. Ackley, SchafferF6, and SchafferF7 are highly multimodal and symmetric like Rastrigin; Rosenbrock is multimodal and asymmetric but appears unimodal when not in the region of the global minimum.

As expected, CRS and CRIBS are less effective on unimodal functions than baseline PSO. The Griewank function is interesting because it is unimodal until the proper level of detail is achieved, a fact that is evident in the slow initial drop but eventual good performance of CRIBS. With the possible exception of Rosenbrock, CRIBS works best on multimodal functions, and even on Rosenbrock it remains competitive.

Clearly, if it is known that the target function is smooth and unimodal, any kind of bouncing is a bad idea. When working with multimodal functions, however, using bouncing with both adaptive collision radius and bounce distance serves to improve performance while retaining reasonable convergence properties.

Table 1: Common benchmark functions

Ackley: $(-32.768, 32.768)$

$$f(\mathbf{x}) = 20 + e - 20\exp\left(\frac{-\|\mathbf{x}\|_2}{5\sqrt{D}}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right)$$

DeJongF4: $(-20, 20)$

$$f(\mathbf{x}) = \sum_{i=1}^{D} i x_i^4$$

Griewank: $(-600, 600)$

$$f(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

Rastrigin: $(-5.12, 5.12)$

$$f(\mathbf{x}) = \|\mathbf{x}\|_2^2 + 10\sum_{i=1}^{D} 1 - \cos(2\pi x_i)$$

Rosenbrock: $(-100, 100)$

$$f(\mathbf{x}) = \sum_{i=1}^{D-1} 100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2$$

SchafferF6: $(-100, 100)$

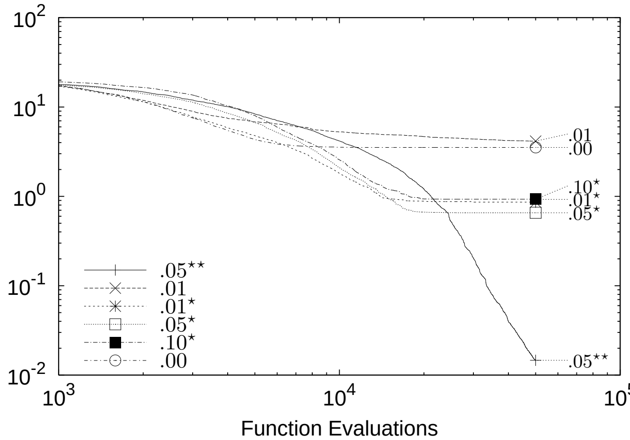
$$f(\mathbf{x}) = \frac{1}{2} + \frac{\sin^2\|\mathbf{x}\|_2 - \frac{1}{2}}{\left(1 + \frac{1}{1000}\|\mathbf{x}\|_2^2\right)^2}$$

SchafferF7: $(-100, 100)$

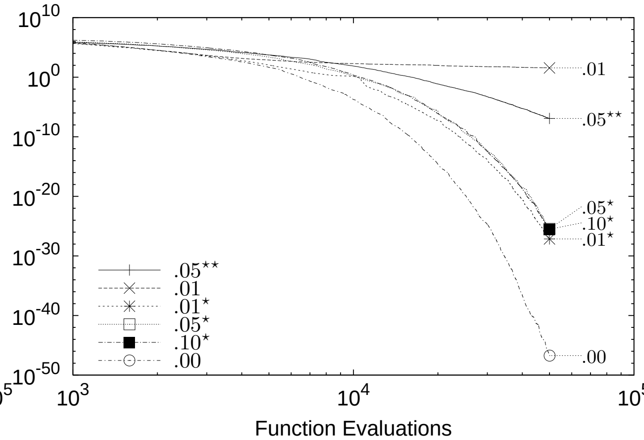
$$f(\mathbf{x}) = \sqrt{\|\mathbf{x}\|_2}\left(1 + \sin^2 50\sqrt[5]{\|\mathbf{x}\|_2}\right)$$

Sphere: $(-50, 50)$

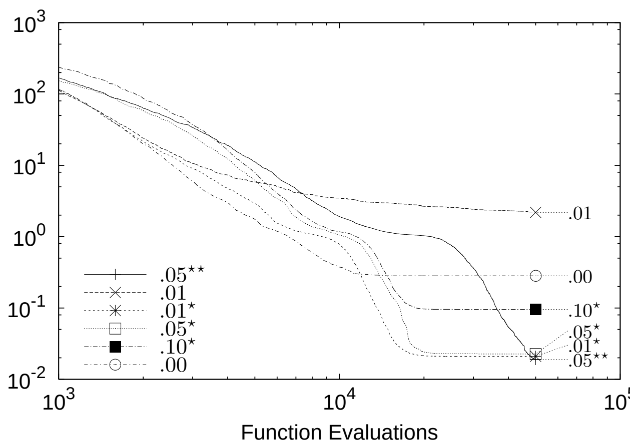
$$f(\mathbf{x}) = \|\mathbf{x}\|_2^2$$



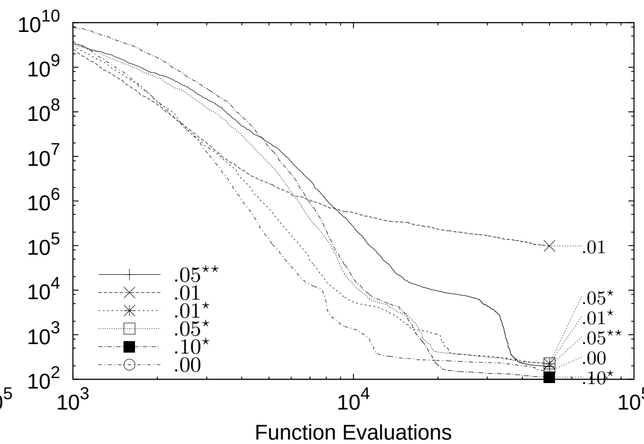
(a) Ackley



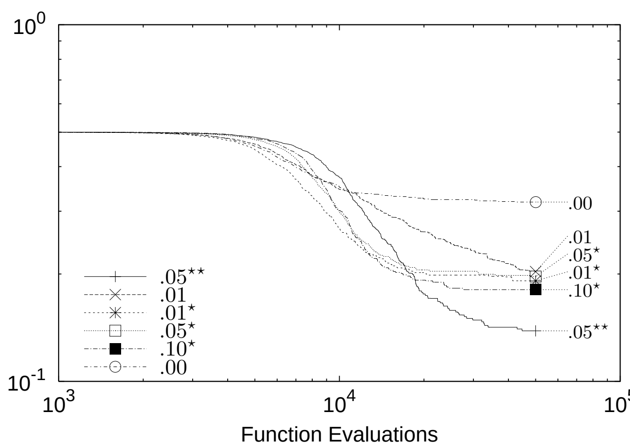
(b) DeJongF4



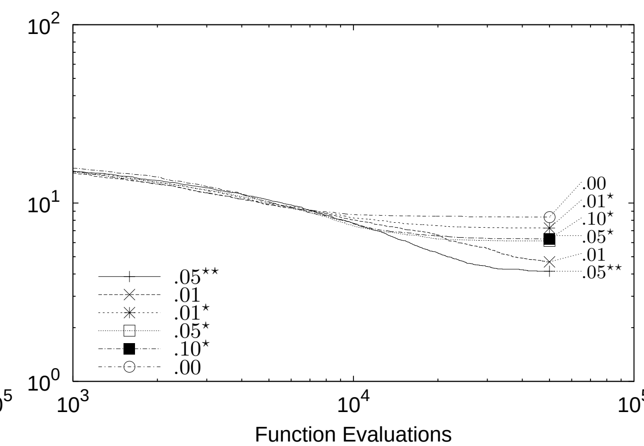
(c) Griewank



(d) Rosenbrock



(e) SchafferF6



(f) SchafferF7

Figure 4: Additional results for CRIBS

3. ATTRACTIVE-REPULSIVE PSO

SEPSO is one of many diversity-increasing methods for PSO. The same authors later introduced the "Attractive and Repulsive PSO" (ARPSO), which uses a global diversity metric to guide a swarm's exploratory behavior [9]. The diversity of a swarm S is

$$\mathrm{diversity}(S) = \frac{1}{|S|L} \sum_{i=1}^{|S|} \|\mathbf{x}_i - \bar{\mathbf{x}}\|_2 \tag{10}$$

which is essentially a measure of the average Euclidean distance of each particle from the center of mass:

$$\bar{\mathbf{x}} = \frac{1}{|S|} \sum_{i=1}^{|S|} \mathbf{x}_i \ . \tag{11}$$

Diversity is scaled by L, the length of the longest diagonal in the feasible region. The metric[1] is calculated globally at each iteration of PSO and is used to artificially inject diversity when needed, via "repulsion".

When the diversity falls below η^-, particles switch into repulsion mode; this is intended to make them fly away from each other, increasing diversity and allowing them to escape local minima. When the diversity has exceeded η^+, the particles are switched back to their normal behavior of attraction and the swarm begins once again to converge.

Attraction is the default behavior of PSO, so repulsion is achieved by adding a sign coefficient to the acceleration term in (2):

$$\dot{\mathbf{x}}_{t+1} = \chi\left(\dot{\mathbf{x}}_t + s\ddot{\mathbf{x}}_{t+1}\right) \ . \tag{12}$$

Letting $s = -1$ switches the swarm into repulsion mode while $s = 1$ restores normal attraction behavior. Note that repulsion generally only has an effect on moving (less converged) particles.

Because ARPSO attempts to quantify the amount of clustering of the swarm in order to detect appropriate times to inject diversity, the use of Euclidean distance as the core of the diversity measure is not, in general, appropriate [1]. This issue is less of a concern with SEPSO because it makes local decisions rather than detecting global clustering, but it nevertheless merits future study.

Figure 5 shows a direct comparison with the published ARPSO results for several 50-dimensional problems, averaged over 50 runs. The underlying algorithm for CRIBS is constricted PSO with default constriction parameters as described in the introduction, effectively eliminating problem-dependent parameter tuning. The initial radius and adaptation factor are in all cases set conservatively at $r = 0.5L$ and $\gamma = 0.9$, respectively. All other results are reproduced directly from the ARPSO paper, including the use of a linear scale instead of the log scale ubiquitously employed in this work; all that is known about the published ARPSO results is that a variant of constricted PSO was used as the underlying motion methodology, and problem-dependent parameters such as swarm size, and maximum velocity, and inertia weights were carefully tuned for each function [9].

[1]The published definition uses $\mathbf{p}$ instead of $\mathbf{x}$ in (10) and (11), but this is unlikely to be correct: in order for diversity to increase, at least one particle must quickly find a better $\mathbf{p}$ while accelerating away from $\mathbf{g}$, a highly unlikely event; implemented this way, ARPSO behavior is equivalent to that of standard PSO until entering repulsion mode, where it remains indefinitely without re-entering attraction mode or improving further over time.

The initial radius r and adaptation constant γ for the ARPSO comparisons in Figure 5 are intentionally different than those reported elsewhere in this work; the way that they were determined illustrates an important point: they were set conservatively (huge r, large γ, and standard constricted PSO) with no exploratory tuning. It was assumed that adaptation would adjust for any problems with the initial parameter settings, and it did. This characteristic of CRIBS makes its successful application to problems easy because it is robust to various parameter settings.

Even though ARPSO is reported to be using optimized parameter settings for each experiment, the untuned CRIBS performed at least as well on all problems but Rosenbrock, and on that benchmark CRIBS exhibits the same behavior that makes ARPSO itself attractive: it is avoiding premature convergence and is continuing on a downward trend.

It is natural to ask whether it is useful to adapt η^- and η^+ in the same way that r is adapted in CRS and CRIBS. While our efforts in this regard did result in some improvement, results were not consistent and even the improved versions of ARPSO failed to reliably outperform CRIBS.

4. CONCLUSIONS

Artificial diversity injection for a convergent algorithm like PSO is an interesting idea, but can require the manipulation of parameters that are nontrivial to tune. In addition, care must be taken to avoid eliminating desirable convergence that allows a swarm to explore the domain at decreasing scales, thereby gaining increasingly detailed information about a local minimum over time.

These issues are simultaneously addressed by allowing diversity to be injected less frequently as time progresses. In SEPSO, this is achieved by reducing the radius after every collision (CRS). The algorithm has the advantage of being simple to implement and more effective than its non-adaptive counterparts early in a run, especially on unimodal functions. Adapting the bounce distance (CRIBS) improves performance on multimodal functions while continuing to ensure eventual convergence. ARPSO does not appear to benefit from adaptation in the same way, and initial results suggest that CRIBS is a more robust approach in any case. Other diversity injection approaches (e.g. charged swarms [2]) may benefit from adaptation, an interesting topic for future research.

The adaptation of diversity parameters is simple to implement and has intuitive appeal, providing an effective way of increasing the exploration capabilities of PSO while retaining its desirable convergence properties.

5. REFERENCES

[1] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional space. *Lecture Notes in Computer Science*, 1973:420–434, 2001.

[2] Tim M. Blackwell and Peter J. Bentley. Dynamic search with charged swarms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, pages 19–26, New York, New York, 2002.

[3] Maurice Clerc and James Kennedy. The particle swarm: Explosion, stability, and convergence in a

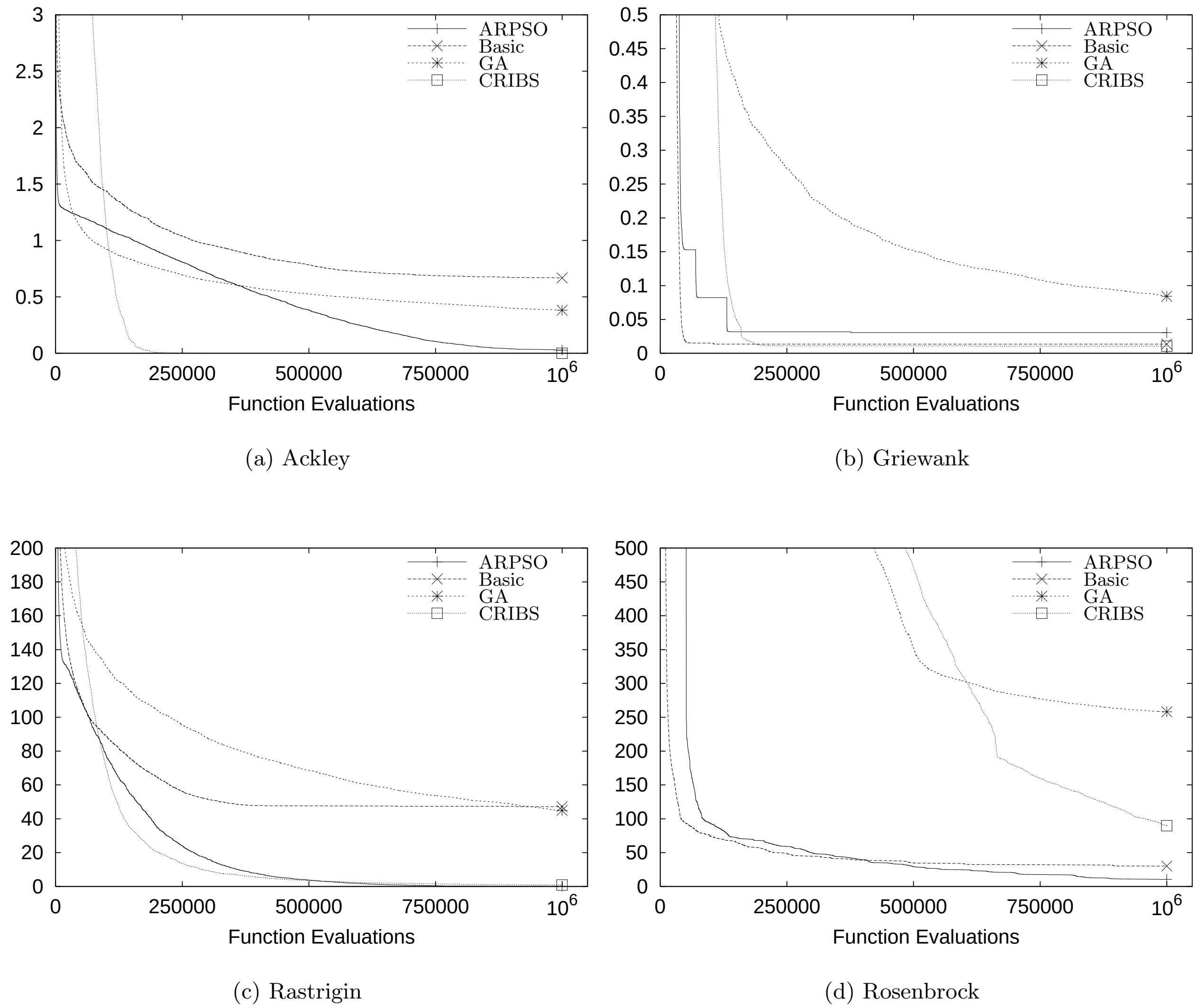


Figure 5: ARPSO compared with other techniques

multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, February 2002.

[4] James Kennedy. Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. In P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and Z. Zalzala, editors, *Proceedings of the Congress of Evolutionary Computation*, volume 3, pages 1931–1938. IEEE Press, 1999.

[5] James Kennedy. Bare bones particle swarms. In *Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003)*, pages 80–87, Indianapolis, Indiana, 2003.

[6] James Kennedy and Russell C. Eberhart. Particle swarm optimization. In *International Conference on Neural Networks IV*, pages 1942–1948, Piscataway, NJ, 1995. IEEE Service Center.

[7] Thiemo Krink, Jakob S. Vestertroem, and Jacques Riget. Particle swarm optimisation with spatial particle extension. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, Honolulu, Hawaii, 2002.

[8] Morten Løvbjerg. Improving particle swarm optimization by hybridization of stochastic search heuristics and self-organized criticality. Master's thesis, Department of Computer Science, University of Aarhus, 2002.

[9] Jacques Riget and Jakob S. Vesterstrøm. A diversity-guided particle swarm optimizer — the ARPSO. Technical Report 2002-02, Department of Computer Science, University of Aarhus, 2002.