CriPS: Critical Particle Swarm Optimisation

Adam Erskine¹ and J. Michael Herrmann¹

¹The University of Edinburgh, School of Informatics, 10 Crichton St, Edinburgh EH8 9AB, U.K.
  a.erskine@sms.ed.ac.uk

Abstract

Particle Swarm Optimisation (PSO) is a metaheuristic used to solve search tasks and is inspired by the flocking behaviour of birds. Traditionally careful tuning of parameters are required to avoid stagnation. Many animals forage using search strategies that show power law distributions in their motions in the form of Lévy flight random walks. It might be expected that when exploring spaces for optima in the absence of any prior knowledge a similar strategy may be useful. Using feedback from swarm metrics, we engineer modifications to the standard PSO algorithm that induce criticality. Such dynamics show long tail distributions in system event sizes. The presence of large (though few) exploratory steps removes the risk of stagnation. The Critical Particle Swarm (CriPS) can be easily combined with many existing PSO extensions.

Introduction

Nature abounds with swarms: herds of mammals, schools of fish, flocks of birds etc. They demonstrate amazing coordinated movements, often showing near simultaneous direction changes across the whole swarm in the response to an approaching threat. Such a loss of length scales is reminiscent of state changes in systems at the point of criticality. For example the Ising model of ferromagnetic materials shows that small magnetic perturbations to the system can generate system wide responses. Particle swarm optimisation (PSO) is a metaheuristic method for obtaining solutions to optimisation problems (Kennedy and Eberhart, 1995). Inspired by the cooperative behaviours of flocks of birds or schools of fish, it employs a number of particles as a swarm of potential solutions. They share knowledge of the problem space in order to improve the performance of the swarm. The position and velocity of particles are iteratively updated using the following rules (Kennedy and Eberhart, 1995)

\[ \mathbf{v}_i' = \omega \mathbf{v}_i + \alpha_1 \mathbf{R}_1 (\mathbf{p}_i - \mathbf{x}_i) + \alpha_2 \mathbf{R}_2 (\mathbf{g} - \mathbf{x}_i) \]  (1)

\[ \mathbf{x}_i' = \mathbf{x}_i + \mathbf{v}_i' \]  (2)

Here \( \mathbf{x}_i \) and \( \mathbf{v}_i \) represent the position and velocity vectors of the \( i^{th} \) particle in the swarm. The velocity update, \( \mathbf{v}_i' \), is a linear combination of three contributions: An inertial term parameterised by \( \omega \), a pull towards the personal best location \( \mathbf{p}_i \) parameterised by \( \alpha_1 \), and a pull towards the global best location \( \mathbf{g} \) parameterised by \( \alpha_2 \). The symbols \( \mathbf{R}_1 \) and \( \mathbf{R}_2 \) denote diagonal matrices whose non-zero entries are uniformly distributed in the unit interval.

In order to perform optimally, the algorithm has to ensure an appropriate mix of local search, i.e. the exploitation of existing knowledge of the problem space, and exploration of areas not yet adequately sampled. The optimal balance of exploration and exploitation will depend on the nature of the problem space. In PSO this mix is achieved via the parameterisation of the algorithm. As metaheuristic algorithms are often applied to problems with little formal specification, trial-and-error search remains the only general option for parameter tuning. Inappropriately set parameters frequently lead to poor results with the swarm prematurely converging to a local minima. The particles are said to stagnate.

The CriPS algorithm ensures that a continuing mix of swarm behaviours occurs throughout its execution. The algorithm includes its own parameters that may be used to adjust its online behaviour, however, it is shown that using values derived from properties of the problem space leads to reasonable results. As CriPS uses the standard PSO update rules to locate better solutions its results do not exceed those of modern metaheuristic methods, but the mechanism should be easily combined with such methods.

Particle Swarm Optimisation

To circumvent PSO shortcomings a number of mechanisms for parameter adaption in PSO have been proposed. Various velocity controls have been explored: simple truncation (Kennedy, 1998); or the application of a constriction factor derived from the algorithm’s parameters Clerc and Kennedy (2002). Parameter controls may be applied for instance via linear decreases in the \( \omega \) value as the algorithm progresses (Shi and Eberhart, 1999).
Larger $\omega$ values tend to encourage greater exploration by the swarm of the problem space. By reducing this parameter’s value the swarm is guided to spend more time exploiting the locations that seem promising. Other approaches to varying PSO’s parameters have been explored including, random, increasing, decreasing and chaotic (Bansal et al., 2011). Stagnation tends to still present as a problem. So explicit means to increase diversity have been employed: by using particle repulsion (Riget and Vesterstrøm, 2002; Chowdhury et al., 2013); or random velocities (García-Villoria and Pastor, 2009). The original PSO swarm allowed all particles to share knowledge of global best values. Essentially the swarm was fully connected. Other topologies have been shown to result in performance improvements (Kennedy, 1998; Bratton and Kennedy, 2007).

PSO’s original bird flocking inspiration has been joined by many other bio-inspired approaches. The different search mechanisms of different species bring novel exploration and exploitation methods to the algorithm. One recent approach, Cuckoo search (Yang and Deb, 2009), co-opts an earlier idea of using the power-law distributed movements of Lévy flights to drive local searches.

Algorithms may use an adaptive probability model to choose from a number of strategies in a success-dependent fashion. Self-adaptive algorithms have been proposed for differential evolution (Qin and Suganthan, 2005) and also for PSO (Wang et al., 2011; Zhan et al., 2009). These hyperheuristic methods measure the behaviour of the swarm and use this to determine which algorithm is most appropriate to run at that particular time.

The TRIBES mechanism removes the velocity update component of PSO and the inherent need to set parameter values (Clerc, 2010). Multiple sub-swarms grow and shrink (removing the need to set a swarm size) depending on performance. The swarms exchange information on good locations and new positions are created by weighted sums of the positions of particles representing good solutions.

Recent successful approaches have been derived from the Covariance Matrix Analysis Evolutionary Strategy (CMA-ES). This is a ($\mu$, $\lambda$) evolutionary strategy. A multivariate normal distribution derived from the $\mu$ best parents are used to generate the next generation of solutions ($\lambda$ in number). After each iteration the distribution’s covariance matrix is adapted to direct the future selection toward new and better solutions. The technique may still suffer from premature convergence. However, detection of such stagnation may be used as a signal to trigger algorithmic restarts or other procedures to increase the diversity of the swarm to obtain good results (Loshchilov, 2013).

**Criticality**

Criticality in equilibrium thermodynamics is used to refer to the properties of a system at a transition point between phases. At this point small perturbations can propagate throughout the whole system (Jensen, 1998). Ferromagnetic materials, for example, gain their magnetism ultimately via the alignment of magnetic dipole moments arising from electron spin states. The state of any individual dipole may be influenced by the magnetic field it finds itself in. If this material is at a high temperature, then the thermal noise will be greater than the influence of small perturbations and no effect is seen. At low temperatures the dipole moments are effectively frozen in whichever state they find themselves in. Small external magnetic perturbations will have little or no effect in either case. When the system is tuned by heating it to point between these two states, then any small external magnetic perturbations are able to flip the state of nearby dipoles. Further, these changes can propagate throughout the material resulting in a chain of dipole flips. A characteristic of the such systems is that changes in system states can exhibit power law distributions.

More widely the term criticality may refer to any dynamical system which behaves in a manner like this (Bak et al., 1988). Bak notes that there are many systems in nature that exhibit similar power law type distributions. We see such heavy tail distributions in the earthquakes described by Gutenberg-Richter law. Similarly power laws crop up in: Zipf's law in ranked distributions of word usage in English; fractal geometries in nature; and 1/f type noise in physical systems. The ferromagnetic example above requires tuning but it seems infeasible for all the occurrences of power-laws to require such fine tuning. It is proposed that systems consisting of many interacting units may evolve automatically into a critical state: effectively self-organising to a point poised between order and chaos (Bak, 1997).

The properties of such systems may be explored by looking at simple models such as the proposed sandpile model (Bak et al., 1988). Sand trickled onto a pile results in gradients that are at some critical angle. Adding a single extra grain may release an avalanche whose size is shown to follow a power law distribution. Similar dynamics have been noted earthquake magnitudes (Olami et al., 1992), punctuated evolution (Bak and Sneppen, 1993), neuronal avalanches (Eurich et al., 2002; Beggs and Plenz, 2003; Levina et al., 2007). The presence of criticality in a system can be shown to optimise the system in some way. Shew et al. (2011) showed the criticality of cortical neuronal avalanches results in optimal information capacity and transmission.

In such systems, the presence of power laws in system event sizes imply that events of any size are possible. Practical limits arise from the finite size of systems. If we were to engineer critical dynamics into the swarm size of a PSO algorithm then we would be assured that the swarm would sooner or later extend throughout the whole problem...
space and thus avoid stagnation. A random search strategy would likewise avoid stagnation, but our approach assures that the algorithm can still favour exploitation of the problem space over exploration.

Previously there have been approaches to adding criticality to PSO. A sandpile-like approach was employed by adding an additional counter to each particle (Lovbjerg and Krink, 2002). If a particle came close to another it incremented its counter. Once over a threshold the particle relocated within the problem space and redistributed its accrued value to other particles allowing avalanches to occur. There was limited evidence that the swarm behaved in a critical manner and the problem of setting parameter values remained. Richer and Blackwell (2006) implemented a PSO algorithm inspired by the Lévy flight random walk behaviour of many foraging animals. They modified a Gaussian PSO algorithm, which performs velocity updates by drawing from Gaussian distributions scaled by distance of particle from local and global best locations, to use Lévy distributions instead. The nature of this power-law approach produced a greater number of outliers in a given problem space, resulting in a more powerfully exploring swarm. Their results showed that this Lévy swarm outperformed both standard PSO and Gaussian PSO approaches. More recently an approach was made where numbers drawn from a power-law distribution were used to modify the PSO parameters directly on each iteration (Fernandes et al., 2012). Again, improvements over standard PSO were apparent.

Our CriPS algorithm aims to remove the need to set parameters. Our approach is to make the algorithm responsive to its environment so that it is able to self-tune to the current problem being explored. The technique presented here was first explored by Cordero (2012). A measure of the diversity of the particle swarm is made. This acts as a feedback signal modifying the PSO algorithm parameters on each iteration.

The CriPS Algorithm

We modify the swarm dynamics of the PSO algorithm such that exploration and exploitation of the problem space are statistically balanced automatically and on-line during the optimisation process. Intuitively, we induce a more stable behaviour should the swarm tend to diverge, but change the parameters of the algorithm towards the unstable regime if the swarm is likely to collapse. In this way we hope to achieve an optimal compromise between local fine-grained search near candidate optima and large scale exploration. The swarm will stay near a critical regime between stability and instability.

Our algorithm uses the dynamic of the swarm itself as a signal to modify its future behaviour. In order to maintain the responsiveness of the swarm to the objective function, we will control the parameters based on changes in the swarm’s diversity. The diversity is most obviously a measure of the swarm size. The average distance between every particle and the centroid of the swarm, for instance, could be calculated on successive iterations and this change used as a feedback signal. We could similarly use intra-particle distances as our diversity measure. A third possibility is to measure the average velocity norm of all particles. This measures the dynamics of the swarm in a slightly different way. The speed of a particle is in essence telling us about the ability of the particles to move around in the problem space. Higher speeds represent more exploratory particles. Averaging over all particles gives us a measure of the swarms diversity. The difference of this metric between two successive iterations quantifies the exploratory and exploitative behaviour of the swarm. If the difference is positive, the swarm is accelerating and increasing its tendency to explore, if the difference is negative, the swarm explores less, exploits more. This third metric is used as our measure of swarm diversity.

We use the change in the metric value between iterations to provide a feedback signal to update the parameter values. For any metric $S$ as listed above, the change in its value is

$$
\Delta S = S(t+1) - S(t).
$$

We update the parameters of the PSO algorithm using

$$
\theta(t+1) = \theta(t) - \varepsilon f(\Delta S),
$$

where $\theta$ is each of the PSO parameters: $\omega$, $\alpha_1$, or $\alpha_2$. All parameters are updated on each iteration. Alternatively one may update a single stochastically chosen parameter on each iteration. This variant allows exploitation of the full PSO parameter space and appears to further improve results. We wish our updates to be approximately the same order of magnitude as PSO parameters (typically a little smaller). We therefore choose a function $f(\Delta S)$ to provide our update signal within the range [-1, 1]. We used a sigmoid function given in equation 5. The $\sigma$ parameter adjust the sensitivity of the sigmoid response to the size of changes in the metric. A small value will make the response function tend to return larger update values for smaller changes in the swarm metric used. The $\varepsilon$ parameter is used to scale the maximum size of the update.

The rescaled sigmoid, we use here, is given by

$$
f(\Delta S) = \tanh \left( \frac{\Delta S}{2\sigma} \right)
$$

For our experiments we used the mean velocity norm of the particles as our metric $S$, our $\sigma$ is chosen to be less than or equal to the maximum extent of the problem space so that the sigmoid returns either -1 or 1 for swarms where the change in the mean velocity norm exceeds the extent of the problem space per iteration.
The change in the metric provides a negative feedback signal to the PSO parameters. Thus an expanding swarm results in smaller $\omega$, $\alpha_1$ and $\alpha_2$ values. Conversely a shrinking swarm leads to increased values. We can consider how this may affect the dynamic of our swarm. A large quickly expanding swarm is exploring the problem space. To encourage exploitation it needs to be brought under control. By decreasing the parameters we will reduce the inertial term ($\omega v_i$) of the update equation. This will have the effect of reducing the expansion if $v_i$ is outward. If inward then this component is still reduced, yet it will still contribute to the swarm’s shrinkage. The $\alpha$ parameters are also reduced, however their contribution are multiplied by the large distance between the particle and the known good locations already found. The pull of these terms is therefore large. In the opposite case, that of a small relatively stagnant swarm, the $\alpha$ terms are multiplied by small distances and contribute little to changing the particles’ dynamics. Increasing $\omega$ over several iterations will accelerate the particles in some direction, ultimately outward. This requires $\omega$ to increase above unity. In this way the feedback mechanism in equation 4 allows the swarm to utilise the changing swarm metric as a feedback signal to control the balance of exploration and exploitation of the problem space. CriPS is shown in Algorithm 1.

Many PSO variants restrict their particles to remain within the spatial extent that the test objective function is defined over. We do not wish to constrain our swarm’s motion. Instead for locations outside the problem space the objective function returns a maximum real value. In this way the particles can explore all space, but will never locate better locations outside the defined benchmark’s spatial extent. The attraction of local and global best locations within the defined problem space always draws the particles back.

Not shown in this algorithm are two components: these apply exponential forces to the parameters should the swarm tend to zero size or a size much greater (fifty times) the size of the problem space. These are required for reasons of practicality. As we apply no constraints to the swarm size a large swarm size change can expand well beyond the problem space. Whilst the dynamics of our update mechanism would sooner or later return the swarm to within the confines of the defined problem, there is the potential to waste time and function evaluations. A similar situation may apply for very small swarms. In reality neither force appears to be much required.

**Results**

The parameters $\varepsilon$ and $\sigma$ in equations 4 and 5 scale the size of the PSO parameter updates. First we look at the effect these have on the swarm dynamics. As our aim is to remove the need to set any parameters we will fix the values of all parameters before we explore the evidence of criticality in these swarm dynamics. Next we show that stagnation is avoided. Finally by comparison with modern algorithms and recent benchmark tests we assess the performance of CriPS. Dynamics are explored using simple but non-trivial functions in two dimensions, e.g. Schwefel and Griewank functions. We present typical examples here. The benchmarks tests are detailed later use a wider range of problem functions. For our PSO implementation we choose $\omega = 0.815$, $\alpha_1 = 1$ and $\alpha_2 = 1$. These are also used as the initial values for our CriPS algorithm. There may be better values that could be chosen for PSO, but we are interested here in comparing the behaviours between the two approaches. Performance comparisons are made later with algorithms that competed in the CEC2013 metaheuristic benchmarking competition.

**Swarm Dynamics**

We measure the swarm size by calculating the mean distance of all particles from the swarm centroid (MSD). A typical evolution of this measure is shown in figure 1. For standard PSO it is typical that the swarm gradually shrinks as the algorithm runs, homing in on a preferred solution. This arises from the tendency of

---

**Algorithm 1:** CriPS algorithm.

**Input:** Objective function $F()$, Maximum iterations $I$, Target Fitness value $V$, $N$ particles with position and velocity state $x_j, v_j$ where $j \in [1, N]$.

**Output:** Returns location of best evaluation of $F()$ achieved in $I$ iterations.

**Initialise:**

- $\text{minSize} \leftarrow$ lower limit of $F()$’s spatial bounds;
- $\text{maxSize} \leftarrow$ upper limit of $F()$’s spatial bounds;
- $x_j \leftarrow$ uniform random $[\text{minSize}, \text{maxSize}]$;
- $v_j \leftarrow$ uniform random $[\text{minSize}, \text{maxSize}]$;
- $\omega \leftarrow 0.815$;
- $\alpha_1 \leftarrow 1$;
- $\alpha_2 \leftarrow 1$;
- $\varepsilon \leftarrow 1$;
- $\sigma \leftarrow \text{maxSize} - \text{minSize}$;
- $S \leftarrow$ mean velocity norm;

**while** Termination conditions are not met **do**

- Update PSO swarm velocities;
- Update PSO swarm positions;
- Calculate $F(x_j) \forall j$;
- Update personal and global bests;
- $S' \leftarrow$ mean velocity norm;
- $\Delta S \leftarrow S' - S$;

**foreach** PSO parameter $\Theta$ **do**

- $\Delta \Theta \leftarrow \varepsilon \times \tanh \left( \frac{\Delta S}{2\sigma} \right)$;
- $\Theta \leftarrow \Theta - \Delta(\Theta)$;

**end**

$S \leftarrow S'$;

**end**

particles to become trapped in local minima within the problem space. Whether, or how quickly, this occurs depends on the choice of parameter values. It is not possible to know in advance for an arbitrary function which values will yield the best results. CriPS, in contrast, shows a swarm dynamic that features periods where the swarm is relatively small, with occasional large bursts. These include bursts larger than the problem space explored. It is from this dynamic that stagnation is avoided.

Assessment of criticality Investigating the presence of power-laws requires the study of larger PSO systems. We increased the number of particles in the swarm to 250 and studied the dynamics for 50000 iterations. We plot the distribution of changes in swarm size between iterations. Our diversity is measured by the mean swarm distance from the swarm centroid (MSD). We can plot the frequency of changes in this ($\Delta MSD$). Figure 2 shows this on a log-log plot. The bottom graph (with $\varepsilon = 0.15$) shows a straight-line portion suggestive of critical behaviour. Although it is not unusual that event distributions show a deviation from a power-law outside a lower and an cut-off, it is often required that the distribution behaves linearly in the log-log plot for at least two decades which is not reached here. It is possible that we need to increase the particle numbers or the size of the region of interest to allow any power-law to become apparent over larger scale.

This figure gives limited evidence for a power-law of the form

$$y \sim x^{-2.3}. \quad (6)$$

If the straight line represents a true power-law relationship then we should suspect that it is possible to tune the system via a parameter variation from subcritical, through critical, to supercritical behaviours. Using the $\varepsilon$ parameter we can tune the swarm behaviour in this manner, see the upper plots in figure 2. Large values ($\varepsilon = 0.5$) appear to result in a subcritical swarm with a shortage of large swarm movements, whilst a small value ($\varepsilon = 0.075$) results in an excess of large moves making the swarm appear supercritical.

As the intent is to remove the need to set any parameter values we set $\varepsilon = 1$ and $\sigma$ equal to the size of the problem space. The PSO parameters are altered on each iteration and are given arbitrary initial values. Figure 3 shows an example of the log-log plot for this configuration to show that behaviour still maintains the mix of behaviours seen previously.

Stagnation The swarm dynamics suggest stagnation may be avoided, but we wish to see continued improvements are being found. In figure 4 we plot each fitness improvement found by the two approaches. For PSO, whilst there are many early improvements, these cease after a while. The number and ability to continue to locate further improvements is again a function of the problem and the chosen parameters. CriPS continues to locate improvements with continuing iterations.

Table 1 quantifies these results. Whilst fewer improvements are made by CriPS as it runs, it still continues to locate improvements, whilst PSO ceases to find better solutions.

Benchmark comparison Assessment of metaheuristic algorithms used to solve optimisation problems requires comparison across a number of problems. In recent years this is often achieved via competitions that present a range of benchmark problem functions with different characteristics. Such benchmarks may or may not capture the nature of real world problems, but seems a reasonable approach in the absence of alternatives.
Figure 2: Different $\varepsilon$ values result in differing mixtures of exploration and exploitation behaviours. Here we set $\sigma$ to $\frac{1}{\sqrt{5}}$ of the problem space’s size. Top, supercritical, $\varepsilon = 0.5$. Middle, subcritical, $\varepsilon = 0.075$. Bottom, critical, $\varepsilon = 0.15$.

We adopt the protocol of the 2013 IEEE Congress of Evolutionary Computation (CEC2013)’s Real-Parameter Single Objective Optimization competition. Algorithms were run against 28 problem functions (with translations and rotations applied), 51 times in each of 10, 30 and 50 dimensions, with a maximum functional evaluation budget. A rank-sum approach evaluated the 21 competing algorithms. We have executed CriPS using the same protocol. For our comparisons the mean result we achieve against each function:dimension pair is compared to the matching values for the competition’s algorithms that finished in positions 1, 5, 10, 15, 17, and 21. For each function:dimension pair we rank these results. The average rank across all pairs is used to rank the algorithms as shown in table 2. We estimate that we would have come roughly 16th in the 21 algorithms. This seems a reasonable result given that we are only using the standard PSO mechanisms to locate improvements.

Table 1: Comparison of CriPS algorithm with our implementation of PSO. Each algorithm is run 30 times, for 50000 iterations. The mean and standard deviation (in parentheses) of the number of times an improved fitness is located was counted within each 10000 iteration period. In all cases the problems are defined as 30 dimensional, there are 30 particles in the swarm. Parameter $\omega$ was initialised to 0.815, $\alpha_1$ and $\alpha_2$ to 1. Additionally $\varepsilon$ was set to 1 and $\sigma$ set to the size of the problem space.

<table>
<thead>
<tr>
<th>Iteration range</th>
<th>PSO</th>
<th>CriPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-10000</td>
<td>2255.1 (342.1)</td>
<td>2499.6 (366.5)</td>
</tr>
<tr>
<td>10001-20000</td>
<td>4.1 (9.5)</td>
<td>1864.3 (921.4)</td>
</tr>
<tr>
<td>20001-30000</td>
<td>1.1 (0.55)</td>
<td>217.7 (449.7)</td>
</tr>
<tr>
<td>30001-40000</td>
<td>10.1 (45.8)</td>
<td>140.4 (540.2)</td>
</tr>
<tr>
<td>40001-50000</td>
<td>0 (0)</td>
<td>20.3 (82.4)</td>
</tr>
</tbody>
</table>

Figure 3: Distribution of swarm size changes for CriPS and Griewank problem space. Parameter $\sigma$ is set to the problem space size, $\varepsilon = 1$, all PSO parameters set to arbitrary initial values.
In particular CriPS performed particularly poorly on the simpler problems. This is to be expected as even when the best course of action is simply to do gradient descent the dynamics of our algorithm will expend effort by causing expansions in the swarm. We note however that against one function (number 8: Rotated Ackleys Function) we ranked first. In a number of other problems we ranked in the top ten.

**Discussion**

Our CriPS swarm showed some limited evidence for criticality. Whilst the log-log plots appear to show straight-line sections, the range over which they manifest is too limited to draw concrete conclusions. However the algorithm does appear to achieve a balance of exploitation and exploration such that all problem space may potentially be visited. This dynamic allows the swarm to avoid stagnation. In addition no special parameter values need tuning. Values are either arbitrary or are set using a measure of the problem. In future it may be that feedback from the discovered manifold of the problem space (rather than the average velocity norm metric) may help to tune the algorithm to locate better solutions.

CriPS varies the PSO parameters synchronously. As we begin with non optimal parameter values there are large portions of the $\omega, \alpha_1, \alpha_2$ parameter space that the algorithm can’t use. We found that a CriPS variant that stochastically picks one of the three parameters to vary on each iteration appears to outperform our standard approach. We speculate that this is because the algorithm can exploit the full PSO parameter space.

We achieve reasonable results when compared with the CEC2013 competition. Performing in the top ten for a number of the objective functions specified (numbers 8, 9, 16, 19), notably winning on function 8 in all dimensionalities. However, our chief aim is to explore the possibility of engineering a critical dynamic in the PSO swarm to avoid the need to tune parameters and to avoid stagnation. CriPS uses standard PSO to guide it to locate better results. This is a limited factoring in achieving better results. Recent high performing metaheuristics e.g. CMA-ES, are more effective at guiding swarms to better solutions. These approaches still require decisions to be made regarding when the swarm should be restarted, and what population size to choose. We are combining a CriPS-like mechanism with covariance matrix update to allow CMA-ES to avoid the need to adopt a restart strategy.

**Acknowledgements**

This work was supported by the Engineering and Physical Sciences Research Council, grant number EP/K503034/1.

---

References


