Extending Particle Swarm Optimisers with Self-Organized Criticality

Morten Løvbjerg
EVALife Group, Department of Computer Science
Ny Munkegade, Bldg. 540, University of Aarhus
DK-8000 Aarhus C, Denmark
www.evalife.dk lovbjerg@daimi.au.dk

Thiemo Krink
EVALife Group, Department of Computer Science
Ny Munkegade, Bldg. 540, University of Aarhus
DK-8000 Aarhus C, Denmark
www.evalife.dk krink@daimi.au.dk

Abstract - Particle Swarm Optimisers (PSOs) show potential in function optimisation, but still have room for improvement. Self-Organized Criticality (SOC) can help control the PSO and add diversity. Extending the PSO with SOC seems promising reaching faster convergence and better solutions.

I. Introduction

The Particle Swarm Optimisation (PSO) algorithm was originally introduced in [6] as an alternative to the standard Genetic Algorithm (GA). The PSO was inspired by insect swarms and has since its introduction turned out to be a competitor to the standard GA when it comes to function optimisation. Several researchers have analysed the performance of the PSO with different parameter and model settings, e.g. [7], [10], [12]. In [1] Angeline points out that the PSO performs well in the early iterations, but has problems reaching a near optimal solution in several real-valued function optimisation problems. This work was done to investigate one way of enhancing the PSO.

Complex system behaviour is neither linear (stable) nor uncorrelated (chaos), but formed by self-organisation in a long transient period and found at the border of stability and chaos - a state known as Self-Organized Criticality [2]. Typically, in complex systems the strength of an emergent property is inversely proportional to its frequency with the functional relationship of a power law: $x \rightarrow x^{-\tau}$. When plotted on a log/log scale, power law data can be described by a linear fit with a negative slope. Various complex systems follow these power laws e.g. species extinction rates in nature [2]. The main idea in SOC is that most state transitions in a component of a complex system only affects its neighbourhood, but once in a while entire avalanches of propagating state transitions can lead to a major reconfiguration of the system.

This study was motivated by the successful application of SOC to Evolutionary Algorithms. This was done with relation to mass extinction and mutation operator control by Krink et al. [8] and with relation to spatial mating control by Rickers et al. [9].

The next section describes the structure of the standard PSO and the SOC extension. Section III describes the experimental settings and the results are reported in Section IV. The results are discussed in Section V and finally Section VI summarises the study.

II. The PSO Models

A. The basic PSO model

The traditional PSO model, described in [6], consists of a number of particles moving around in the search space, each representing a possible solution to a numerical problem. Each particle has a position vector ($\vec{x}_i$), a velocity vector ($\vec{v}_i$), the position ($\vec{p}_i$) and fitness of the best point encountered by the particle, and the index (g) of the best particle in the swarm.

In each iteration the velocity of each particle is updated according to their best encountered position and the best position encountered by any particle, in the following way

$$\vec{v}_i = \chi(w\vec{v}_i + \vec{\varphi}_{1i}(\vec{p}_i - \vec{x}_i) + \vec{\varphi}_{2i}(\vec{p}_g - \vec{x}_i))$$

where χ is known as the *constriction coefficient* described in [3], w is the *inertia weight* described in [10] and $\vec{p}_g$ is the best position known for all particles. φ_1 and φ_2 are random values different for each particle and for each dimension. If the velocity is higher than a certain limit, called v_{max}, this limit will be used as the new velocity for this particle in this dimension.

The position of each particle is updated in each iteration. This is done by adding the velocity vector to the position vector, i.e. $\vec{x}_i = \vec{x}_i + \vec{v}_i$.

The particles have no neighbourhood restrictions, meaning that each particle can affect all other particles. This neighbourhood is of type *star* (fully connected network), which has been shown to be a good topology [7].

B. SOC extension

The structure of the SOC PSO model is illustrated in Fig. 1.

The main idea behind the SOC extension is to use SOC to add diversity. The only difference between a standard particle and a SOC particle is an additional variable for the current critical value (C). This variable also denoted as the criticality of the particle is initialised with the value zero. The closer swarm particles are to each other the less diverse the swarm is. If two particles are closer than a threshold distance (TD) their criticality increase by one each (*calculate criticality* in Fig. 1).

The notion of SOC usually only applies to open systems. To prevent criticality from building up, each particle has its critical value C reduced by a certain percentage (CR) in every iteration (*reduce criticality* in Fig. 1). This can result in fractional critical values.

The SOC PSO has a globally set criticality limit (CL). If the critical value C of a particle exceeds this limit the particle responds by dispersing its criticality within a certain surrounding neighbourhood (SN) and then by relocating itself.

When a particle is to disperse its criticality (*disperse criticality* in Fig. 1) it increases the critical value C of the first CL particles within its neighbourhood (SN) by one. The dispersing particle then reduces its own critical value by CL. This happens even if there are less than CL particles in the given surrounding neighbourhood. In this case one might say that some of the criticality simply disappears. Dispersal keeps the current critical value of the dispersing particle non-negative. It also creates the possibility of other neighbouring particles exceeding the criticality limit.

```
begin
    initialise
    while (not terminate-condition) do
        begin
            evaluate
            calculate new velocity vectors
            move
            calculate criticality
            reduce criticality
            while (some agent has a critical value
                                    > criticality limit) do
                begin
                    disperse criticality
                    relocate agent
                end
        end
end
```

Fig. 1. Structure of the SOC PSO model.

After dispersal the critical particle relocates itself according to some relocation scheme (*relocate agent* in Fig. 1). Relocation of particles happens in order to increase diversity.

In our study, we tried out two relocation schemes to add diversity. The first scheme (relocation scheme 1) uses a random immigrant technique similar to the one described in [4], where the particles simply are re-initialised. This also includes that the particles “forget” their previously best found positions. This scheme adds some diversity, but in a pretty random fashion.

The second scheme (relocation scheme 2) does not remove the memory of the previously best found position, but instead pushes the particle a little further. The particle is pushed in the direction it is heading with a magnitude of the criticality limit CL added to the critical value C of the particle.

To improve the search further, we used SOC to control the particle inertia. This is a crucial parameter in the PSO model and changing it can have huge effects. In the standard PSO model the inertia is set to be linear decreasing from 0.7 to 0.4 over the course of the iterations. This scheme was introduced to ease local search [10]. In our model we exchanged the linear decreasing inertia with a scheme that uses the critical value C of each particle. In this scheme the inertia of a given particle is set to 0.2 plus a tenth of the current critical value of the particle. It follows that the particles now possibly have different inertia values that vary throughout the run.

III. Experimental Settings

In our experiments, we compare the performance of the standard PSO and the SOC PSO on four numerical benchmark problems (all minimisation). The first two are unimodal while the last two are multimodal with many local minima. All functions are designed such that their global minimum is at or near the origin of the search space. These four functions have been commonly used in other studies on particle swarm optimisers (e.g. [7], [11]).

Table I lists the testfunctions.

TABLE I

Sphere $f_1(x) = \sum_{i=1}^{n} x_i^2$
Rosenbrock $f_2(x) = \sum_{i=1}^{n-1}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$
Griewank $f_3(x) = \frac{1}{4000}\sum_{i=1}^{n}(x_i - 100)^2 - \prod_{i=1}^{n} cos(\frac{x_i - 100}{\sqrt{i}}) + 1$
Rastrigin $f_4(x) = \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i) + 10)$

The initial population is usually uniformly distributed over the entire search space. According to Angeline ([1]) this can give false indications of relative performance, especially if the search space is symmetric around the origin where many test functions have their global optimum. To prevent this, and to ease comparison with other models, the asymmetric initialisation method used by Angeline ([1]) was used. Search space and initialisation ranges for the experiments are listed in Table II.

TABLE II

Function	Search space	Initialisation
f_1	$-100 \leq x_i \leq 100$	$50 \leq x_i \leq 100$
f_2	$-100 \leq x_i \leq 100$	$15 \leq x_i \leq 30$
f_3	$-600 \leq x_i \leq 600$	$300 \leq x_i \leq 600$
f_4	$-10 \leq x_i \leq 10$	$2.56 \leq x_i \leq 5.12$

The testfunctions used were all 30 dimensional and each test ran for 3000 iterations. To weed out stochastic differences tests with the standard PSO were run 500 times each and tests with the SOC PSO were run 100 times each.

In both standard and SOC PSO models the upper limits for φ_1 and φ_2 were set to 2.0. A linearly decreasing inertia weight starting at 0.7 and ending at 0.4 was used for the standard PSO tests and for the SOC PSO tests that did not include the SOC inertia scheme. The constriction coefficient χ was set to 1. The maximum velocity (v_{max}) of each particle was set to half the length of the search space for each dimension (e.g. $v_{max} = 100$ for f_1 and f_2).

Previous research by Shi ([10]) regarding scalability of the standard PSO showed that the performance of the standard PSO is not sensitive to the population size. In all experiments we used a fixed population size of 20 particles.

A. Settings for parameters unique to the SOC PSO

The size of the threshold distance *TD* (wherein particles affect the criticality of eachother) is set to decrease over time. It decreases linear from 1 percent of the length of the searchspace in one dimension to zero. This was done to ease local search towards the end of the run, as was the case for the inertia in the standard PSO [10].

Further, we set the global criticality limit *CL* to *popsize*/5. Since population size was kept fixed at 20 the fixed criticality limit was 4.

As with the criticality limit, we set the criticality removal rate *CR* to be correlated to the population size by default 2/*popsize* resulting in a removal rate of 0.1 corresponding to 10 percent. This setting seemed to work out well, but to further investigate the effects of the criticality removal rate, we also tested a different parameter setting of 0.2/*popsize* corresponding to a removal rate of 1 percent.

To further investigate the effects of SOC inertia we used 0.2 plus a fourth of the current critical value *C* for a given particle as an alternative setting.

As mentioned in Section II criticality simply disappears from the system when a critical particle does not have enough particles to disperse criticality to. To narrow the possibility of this happening, the size of surrounding neighbourhood *SN*, wherein criticality is set to be dispersed, is set to be double the size of the current threshold distance *TD*. Hence the surrounding neighbourhood *SN* also decreases over time.

IV. Experimental Results

The results of the experiments are shown in Fig. 2 to Fig. 12. All graphs are averages over the number of runs performed by each algorithm. Graphs are denoted *STD* for the standard PSO, *SOC* for the SOC PSO. Relocation scheme 1 is denoted *rel1* and relocation scheme 2 likewise. No relocation is denoted *norel* and *iner* denotes that the SOC PSO uses SOC inertia.

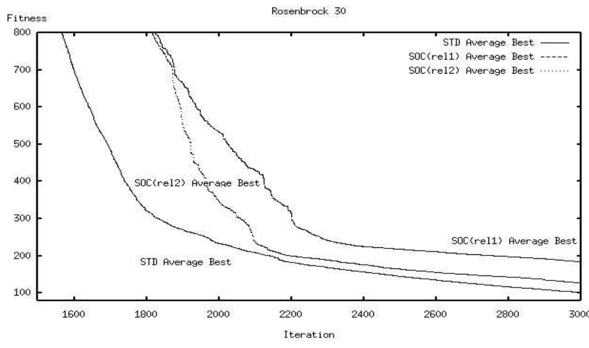

Fig. 2. Standard PSO versus two SOC PSOs each with their relocation scheme. Rosenbrock function.

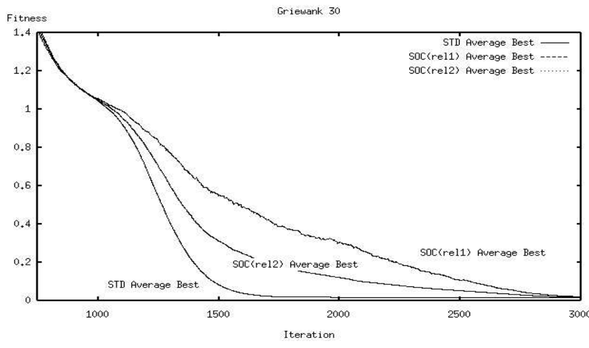

Fig. 3. Standard PSO versus two SOC PSOs each with their relocation scheme. Griewank function.

A. Testing relocation schemes

The achieved fitness's of the standard PSO and two SOC PSOs each with their relocation scheme are shown in Fig. 2 to Fig. 4 for comparison.

For Sphere (not shown) the best found optima for the standard and the SOC PSOs with the two relocation

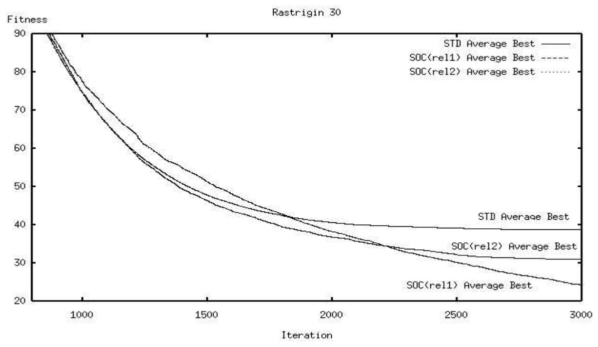

Fig. 4. Standard PSO versus two SOC PSOs each with their relocation scheme. Rastrigin function.

schemes are quite equal, with the standard PSO reaching a slightly worse result. The SOC PSO though suffers for both Rosenbrock (Fig. 2) and Griewank (Fig. 3) compared to the standard PSO in both cases with a slower convergence but for Griewank to roughly the same optimal value. For Rastrigin (Fig. 4) the standard PSO is clearly outperformed at the end especially by the SOC PSO with relocation scheme 1.

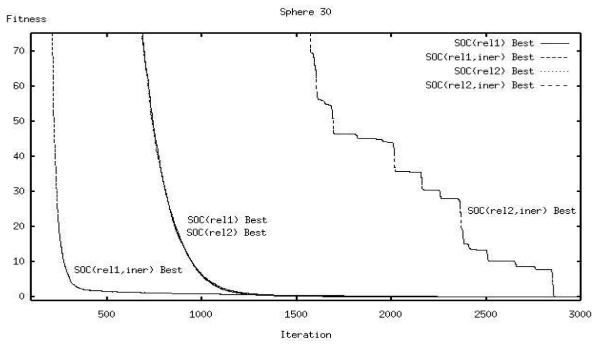

Fig. 5. SOC PSOs with each relocation scheme versus SOC PSOs with additional SOC inertia. Sphere function.

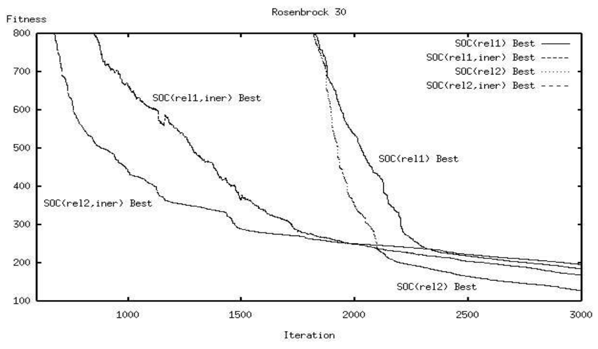

Fig. 6. SOC PSOs with each relocation scheme versus SOC PSOs with additional SOC inertia. Rosenbrock function.

B. Testing SOC inertia

The graphs in Fig. 5 to Fig. 8 show SOC PSOs with each relocation scheme compared to SOC PSOs with additional SOC inertia for the four testfunctions. In all cases regarding relocation scheme 1 the SOC PSO has a faster convergence with SOC inertia than without. Additionally it converges to roughly the same value as the SOC PSO without SOC inertia. This is also the case for the SOC PSO with relocation scheme 2 for the functions Rosenbrock (Fig. 6) and Rastrigin (Fig. 8). Results for Sphere (Fig. 5) and Griewank (Fig. 7) however show slower convergence for the SOC PSO with relocation scheme 2 and SOC inertia, but to a similar value.

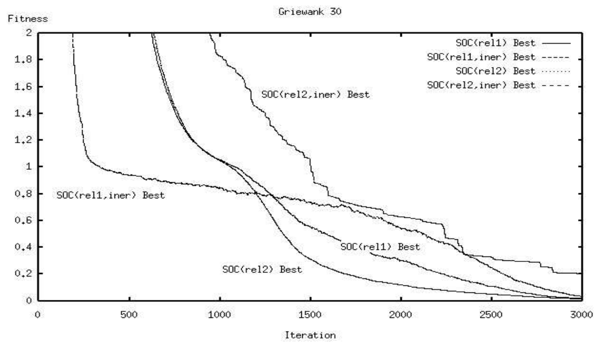

Fig. 7. SOC PSOs with each relocation scheme versus SOC PSOs with additional SOC inertia. Griewank function.

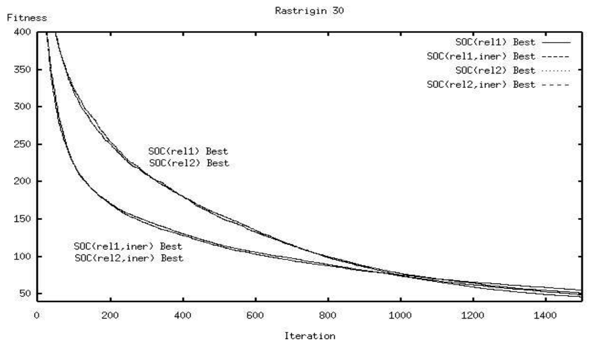

Fig. 8. SOC PSOs with each relocation scheme versus SOC PSOs with additional SOC inertia. Rastrigin function.

In order to further investigate the relation between relocation of particles and SOC inertia, we ran additional experiments using the Rastrigin function. One result in Fig. 9 shows an average of a series of runs with a SOC PSO using SOC inertia, but without any relocation of particles. The figure shows that the PSO has a pretty fast convergence, but to a suboptimal value.

The mean value plots in Fig. 10 show the relation between means for experiments with the Rastrigin function. The means are similar for the other functions. For the SOC PSO without SOC inertia the means are higher and more fluctuating than for the standard PSO. The means for the SOC PSO with SOC inertia are lower than for the standard PSO in the beginning, but end up higher. Both with and without SOC inertia the SOC PSO with relo-

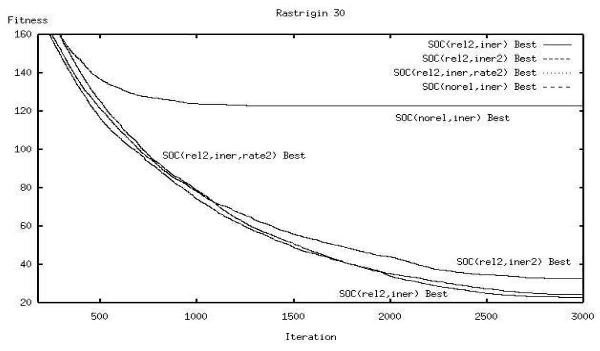

Fig. 9. SOC PSO with different relocation and SOC inertia schemes. Rastrigin function.

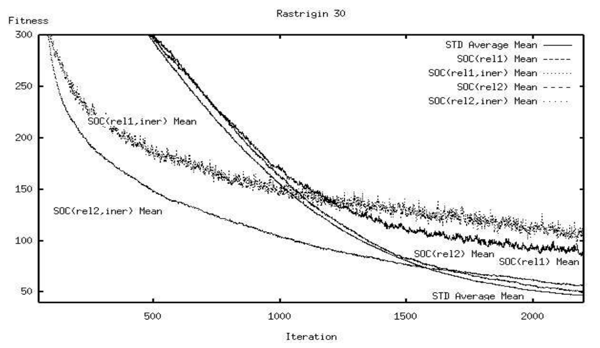

Fig. 10. Average mean values for the standard PSO versus SOC PSOs with each relocation scheme with and without SOC inertia. Rastrigin function.

cation scheme 1 has a higher mean value than the SOC PSO with relocation scheme 2. With a low best value found, a higher mean indicates a higher swarm diversity.

C. Other experiments

The SOC extension introduces a couple of new parameters. We investigated the effects of these parameters using the Rastrigin function and a SOC PSO with relocation scheme 2. The results of the other SOC inertia scheme and the other removal rate setting mentioned in Section III are shown in Fig. 9. The best found optimum for the SOC PSO with the original SOC inertia setting outperforms the SOC PSO with the second SOC inertia setting. The figure also shows no change bordering a slight performance decrease as a result of the second removal rate setting.

The level of criticality and the effect of the second removal rate setting are illustrated in Fig. 11. As a measure, we used the average criticality per iteration per particle. For the SOC PSO with relocation scheme 2, standard SOC inertia and standard SOC removal rate the criticality increases over time from ~0.3 to ~0.65. The tested second setting for the SOC removal rate results in a higher criticality level as can be seen in Fig. 11. This

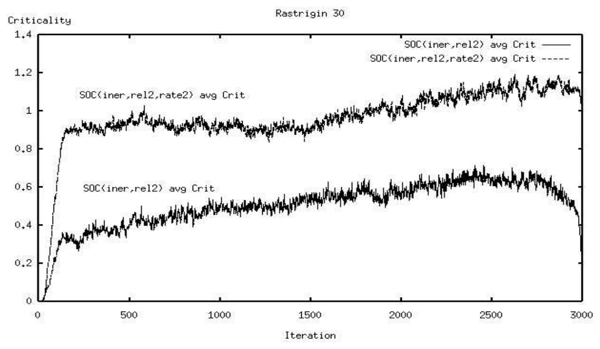

Fig. 11. Criticality for SOC PSO with relocation scheme 2 and SOC inertia versus criticality for SOC PSO with additional second removal rate setting. Rastrigin function.

level increases over time from ~0.9 to ~1.1. The criticality level of the SOC PSO with relocation scheme 2 and the second SOC inertia setting is not shown, but has a somewhat lower level than the original setting increasing over time from ~0.25 to ~0.6.

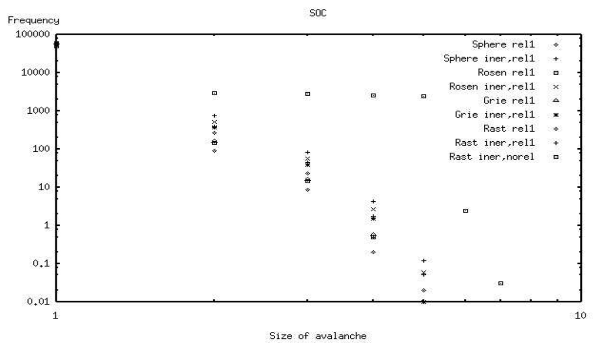

Fig. 12. SOC PSO relations between the frequency and size of avalanches on a log/log scale for Sphere, Rosenbrock, Griewank and Rastrigin functions.

As mentioned previously a particle disperses its criticality when it becomes critical. This can lead to other neighbouring particles also exceeding the criticality limit, which then propagate the criticality further and so on and so on (resulting in occasional avalanches). To verify the SOC power law relation between the number and size of avalanches we plotted the propagation frequency by a log/log scale shown in Fig. 12. The figure shows the avalanches for the functions Sphere, Rosenbrock, Griewank and Rastrigin for a SOC PSO with relocation scheme 1 both with and without SOC inertia. The points approximate a straight line, as is the case for a SOC PSO with relocation scheme 2, and thus indeed resemble a SOC power law. The only exception is the Rastrigin plot for testing the SOC PSO with SOC inertia but without relocation. Avalanches here are more frequent and do not follow a SOC power law relation.

V. Discussion

As pointed out in the previous section extending the PSO with SOC increases the PSO diversity. This is indicated in Fig. 10 showing an increase in mean values for particles. As expected relocation scheme 1 (the random immigrant scheme) introduced more diversity than scheme 2 (the “pushing” scheme). Only for the Rastrigin function (Fig. 4) a higher diversity did seem to pay off adequately by a better optimum at the end of the run. Relocation scheme 2, which pushes particles, but does not make them forget their previously best known position, did not seem to help them reach a better optimum. For Rosenbrock (Fig. 2) the result was even a bit worse.

Adding SOC inertia turned out to be a lot more promising regarding the convergence speed. The SOC PSO with relocation scheme 1 and the additional SOC inertia outperforms the original SOC PSO by far regarding all the tested functions (Fig. 5 to Fig. 8). Further, the faster convergence is achieved without compromising an as good overall best solution. The same is the case for the SOC PSO with relocation scheme 2 for the Rosenbrock and Rastrigin functions (Fig. 6 and Fig. 8). For Sphere and Griewank (Fig. 5 and Fig. 7) the convergence is slower but still achieves the same optimal value.

Further, we found that the effect of SOC inertia depends on the relocation scheme. Apparently some relocation is needed in order to keep diversity and to prevent premature convergence. The convergence without relocation is shown in Fig. 9.

Changing the criticality influence on the SOC inertia from a tenth to a fourth did not make much of a performance difference. In fact Fig. 9 shows a small efficiency decrease. A larger influence of criticality results in higher inertia values, which imply more particle movement meaning less crowding or bumping into other particles. Less bumping results in less criticality build-up.

Reducing the criticality removal rate from 10 to 1 percent increases the average criticality (Fig. 11) as expected. An increased criticality level could lead to more particles being relocated thus to a higher swarm diversity. A higher criticality level would also mean a higher particle SOC inertia. Even though the increase in criticality is substantial (Fig. 11) we do not find significant changes in the performance (Fig. 9).

Figure 12 shows how critical particles follow a power law both with and without SOC inertia. This relationship only seems to hold in conjunction with a relocation scheme. Removing relocation results in too frequent big avalanches making the particles converge prematurely (Fig. 9). Relocation is in some sense necessary to make the SOC PSO work.

VI. Conclusions and Future Work

Our results show that the extension of PSOs with SOC yields superior performance compared to simple PSOs for four numerical benchmark problems. Clearly the SOC PSO inertia weight resulted in the largest improvements, but SOC also helps by adding diversity. The level of diversity highly depends on a number of parameters especially the type of relocation scheme. Future work could include tests with other relocation schemes, other settings for the criticality limit or the criticality removal rate. Having a criticality history attached to each particle would help investigate the effect of SOC further and could help develop other dispersal or relocation schemes.

References

[1] P. J. Angeline, ”Evolutionary Optimization Versus Particle Swarm Optimization: Philosophy and Performance Differences”, Evolutionary Programming VII (1998), Lecture Notes in Computer Science 1447, 601–610. Springer.

[2] P. Bak, ”How Nature Works”, 1996, Copernicus, Springer-Verlag, 1. edition.

[3] M. Clerc, ”The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization”, Proceedings of the 1999 Congress of Evolutionary Computation, vol. 3, 1951–1957. IEEE Press.

[4] J. Grefenstette, ”Genetic Algorithms for changing environments”,Proceedings of Parallel Problem Solving From Nature (PPSN-2) (1992), pp.137-144.

[5] R. C. Eberhart and Y. Shi, ”Comparison between Genetic Algorithms and Particle Swarm Optimization”, Evolutionary Programming VII (1998), Lecture Notes in Computer Science 1447, 611–616. Springer.

[6] J. Kennedy and R. C. Eberhart, ”Particle swarm optimization”, Proceedings of the 1995 IEEE International Conference on Neural Networks, vol. 4, 1942–1948. IEEE Press.

[7] J. Kennedy, ”Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance”, Proceedings of the 1999 Congress of Evolutionary Computation, vol. 3, 1931–1938. IEEE Press.

[8] T. Krink, R. Thomsen, and P. Rickers, ”Applying Self-Organised Criticality to Evolutionary Algorithms”, Parallel Problem Solving from Nature - PPSN VI (2000), vol. 1, p. 375-384

[9] P. Rickers, R. Thomsen and T. Krink, ”Applying Self-Organized Criticality to the Diffusion Model”, Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference, vol. 1, p. 325-330

[10] Y. Shi and R. C. Eberhart, ”Parameter Selection in Particle Swarm Optimization”, Evolutionary Programming VII (1998), Lecture Notes in Computer Science 1447, 591–600. Springer.

[11] Y. Shi and R. C. Eberhart, ”Empirical Study of Particle Swarm Optimization”, Proceedings of the 1999 Congress of Evolutionary Computation, vol. 3, 1945–1950. IEEE Press.

[12] P. N. Suganthan, ”Particle Swarm Optimiser with Neighbourhood Operator”, Proceedings of the 1999 Congress of Evolutionary Computation, vol. 3, 1958–1962. IEEE Press.