# Particle Swarm with Self-Organized Criticality

Carlos M. Fernandes[1,2] cfernandes@laseeb.org
J.J. Merelo[2] jjmerelo@gmail.com
Francisco Fernández[3] fcofdez@unex.es
Agostinho C. Rosa[1] acrosa@laseeb.org

[1]Technical Univ. of Lisbon Av. Rovisco Pais, 1, TN 6.21, 1049-001, Lisbon, Portugal
[2]Univ. of Granada, Dept. of Computer Architecture and Technology, ETSIIT; 18071, Granada, Spain
[3]University of Extremadura, Dept. of Computer Architecture and Technology, Merida, Spain

## ABSTRACT
This paper introduces a strategy for controlling the parameters of the Particle Swarm Optimization (PSO) based on a Self-Organized Criticality (SOC) system known as the Bak-Sneppen model of co-evolution. An experimental setup compares the new algorithm with a state-of-the-art PSO with dynamic variation of the inertia weight value and perturbation of the particles' positions.

## Categories and Subject Descriptors
H.4 [Information Systems Applications]: Miscellaneous

## General Terms
Algorithms, Theory.

**Keywords:** Particle Swarm Optimization, Self-Organized Criticality, Parameter Control.

## 1. INTRODUCTION
PSO [3] is a population-based algorithm in which a group of solutions travels through the search space according to a set of rules that favor their movement towards optimal regions of the space. The algorithm is described by a simple set of equations that define the velocity and position of each particle. The position vector of the *i-th* particle is given by $\vec{X}_i = (x_{i,1}, x_{i,2}, \dots x_{1,D})$, where $D$ is the dimension of the search space. The velocity is given by $\vec{V}_i = (v_{i,1}, v_{i,2}, \dots v_{1,D})$. The particles are evaluated with a fitness function $f(\vec{X}_i)$ in each time step and then their positions and velocities are updated by:

$$v_{i,d}(t) = \omega . v_{i,d}(t-1) + c_1 r_1 \left(p_{i,d} - x_{i,d}(t-1)\right) + c_2 r_2 \left(p_{g,d} - x_{i,d}(t-1)\right) \quad (1)$$

$$x_{i,d}(t) = x_{i,d}(t-1) + v_{i,d}(t) \quad (2)$$

were $p_i$ is the best solution found so far by particle $i$ and $p_g$ is the best solution found so far by the neighborhood; $r_1$and $r_2$ are random numbers uniformly distributed in the range $[0,1]$; $c_1$ and $c_2$ are acceleration coefficients that tune the relative influence of each term of the formula. Although PSO may be very efficient on numerical optimization, it requires a proper balance between local and global search, and it often gets trapped in local optima. In order to achieve a balancing mechanism, Shi an Eberhart [5] introduced a inertia weight $\omega$ (see the first term of equation 1). This parameter may be hand-tuned in order to optimize the performance. Another possible strategy, introduced in [5], is the *time-varying inertia weight*: starting with an initial and pre-defined value, the parameter value decreases linearly with time, until it reaches the minimum. A more sophisticated approach is given by Suresh *et al.* in [6]. The authors use the Euclidean distance between the particle and the global best for computing $\omega$ in each time-step for each particle. In addition, the authors use a perturbation mechanism of the particles' positions, with a random value in the range $[1, \rho]$, where $\rho$ is a new parameter (see equation 3, which replaces equation 2). The authors report that the *Inertia-Adaptive PSO* (IA-PSO) outperforms other PSOs in a 12-function test. IA-PSO is included in the test set described in Section 2.

$$x_{i,d}(t) = (1+\rho).x_{i,d}(t-1) + v_{i,d}(t) \quad (3)$$

This paper proposes a method for tuning $\omega$ based on a Self-Organized Criticality (SOC) system known as the *Bak-Sneppen model of co-evolution of species* [1]. The resulting algorithm, called Bak-Sneppen PSO (BS-PSO), uses the fitness values of the population of *species*, which tend to *evolve* as a result of selection and random variation. The dynamics of the model's fitness values provides a promising basis for modeling PSO's working mechanisms. In fact, SOC has been used in the past in population-based metaheuristics (see [2] and [4]). In this paper, BS-PSO controls the inertia weight values by assigning a weight for each particle that is directly dependent on the fitness value of the species mapped to that particle. Furthermore, the exact same value is used for perturbing the particle's position, thus introducing a kind of mutation in the PSO equations, like in [6].

Algorithm 1 (Bak-Sneppen Model)
1. Set $mutations = 0$; max $\_mutations = 2 \times population\_size$
2. Find the index $j$ of the species with lowest bak-sneppen fitness
3. Set $minFit = bs\_fitness(\vec{X_j})$
4. Replace the fitness of individuals with indices $j$, $j-1$, and $j+1$ by random values in the range $[0, 1.0]$
5. Increment mutations: $++mutations$
6. Find the index $j$ of the species with lowest fitness
7. If $bs\_fitness(\vec{X_j}) < minFit$ or $mutations = max\_mutations$, return to 4; else, end Alg. 1

Algorithm 2 (BS-PSO)
1. Initialize velocity and position of each particle.
2. Evaluate each particle: $fitness\ (\vec{X_i}) = f(\vec{X_i})$
3. Initialize bak-sneppen fitness: $bs\_fitness(\vec{X_i}) = random[0, 1.0]$
4. For each particle $i$:
   5. Set $\omega = \rho = 1 - bs\_fitness(\vec{X_i})$
   6. Update velocity and position:
7. Update Bak-Sneppen Model (Algorithm 1).
8. If (stop criteria not met) return to 4; else, end.

In the Bak-Sneppen model, a population of individuals (species) is placed in a ring topology and a random real number in the range $[0,1]$ is assigned to each individual. In the BS-PSO, the size of this ecosystem (number of species) is equal to the size of the


*GECCO'12 Companion,* July 7–11, 2012, Philadelphia, PA, USA.
ACM 978-1-4503-1178-6/12/07.

swarm. Therefore, the algorithm may be implemented just by assigning a second fitness value, called *bak-sneppen fitness value* (*bs_fitness*) $f_{bs}(\vec{X})$ to each individual, i.e, an individual is both the particle of PSO and the species of the co-evolutionary model, with two independent fitness values: the quality measure fitness value $f_{bs}(\vec{X})$, computed by the objective function, and the bak-sneppen fitness value $f_{bs}(\vec{X})$, which is modified according to Algorithm 1.

$$\omega_i(t) = 1 - \ bs\_fitness(\overrightarrow{X_i}) \tag{4}$$

$$x_{i,d}(t) = (1 + \rho_i(t)).x_{i,d}(t-1) + v_{i,d}(t) \tag{5}$$

The main body of BS-PSO is very similar to the basic PSO. The differences are: Algorithm 1 is called in each time-step, modifying three or more bak-sneppen fitness values; the inertia weigh of each particle is defined in each time-step and for each particle $i$ using the bak-sneppen fitness values (see equation 4); the position are updated using equation 5, where:

$$\rho_i(t) = random\left[0, \left(1 - bs\_fitness(\overrightarrow{X_i})\right)\right] \tag{6}$$

Algorithm 1 is executed in each time-step. At $t = 0$, the bak-sneppen fitness values are randomly set. Then, at each $t$, the algorithm finds the worst species (lowest *bs_fitness*), stores its fitness value (*minFit*) and mutates it by replacing its *bs_fitness* with a random value in the range $[0, 1.0]$. The neighbors of the worst species are also mutated. Then, the algorithm searches again for the worst species. If its fitness is lower than *minFit*, the process repeats: the species and its neighbors are mutated. This cycle proceeds while the worst fitness in the population is bellow *minFit* and the number of mutations is below a pre-defined limit. When the worst fitness is above *minFit*, the algorithm proceeds to the PSO's standard procedures (see Algorithms 1 and 2).

**Table 1: Numerical results (with standard deviation).**

| | $\rho = 0$ | | $\rho = 0.25$ | | *Bak – Sneppen* $\rho$ | |
|---|---|---|---|---|---|---|
| | **IA-PSO** | **BS-PSO** | **IA-PSO** | **BS-PSO** | **IA-PSO** | **BS-PSO** |
| **lbest** | | | | | | |
| $f_1$ | 5.19e-02 (2.61e-02 | **1.38e-15 (3.21e-15)** | 6.56e-03 (5.34e-03) | **0.00e00 (0.00e00)** | 2.60e-02 (1.70e-02) | **0.00e00 (0.00e00)** |
| $f_2$ | 3.10e+02 (7.25e+02) | **1.64e+02 (4.30e+02)** | 2.90e+01 (9.81e-01) | **2.68e+01 (2.53e-01)** | 2.94e+01 (8.42e-01) | **2.67e+01 (2.13e-01)** |
| $f_3$ | 1.27e+07 (3.36e+07) | **1.17e+02 (3.05e+01)** | 3.26e+01 (4.05e+01) | **3.07e-01 (1.44e+00)** | 2.68e+01 (3.16e-01) | **5.39e00 (1.33e+01)** |
| $f_4$ | 1.84e00 (1.27e+01) | **1.25e-02 (1.26e-02)** | 1.11e-02 (7.74e-03) | **4.74e-03 (2.36e-03)** | 1.30e-02 (7.08e-03) | **5.24e-03 (2.38e-03)** |
| **gbest** | | | | | | |
| $f_1$ | 1.22e+04 (1.11e+04) | **9.41e+03 (8.08e+03** | 1.80e-01 (1.17e-01) | **0.00e00 (0.00e00)** | 3.34e-01 (3.76e-01) | **0.00e00 (0.00e00)** |
| $f_2$ | **1.12e+07 (2.80e+07)** | 1.27e+07 (3.36e+07) | 3.35e+01 (9.48e+00) | **3.26e+01 (4.05e+01)** | 2.94e+01 (1.86e+02) | **2.68e+01 (3.16e-01)** |
| $f_3$ | **1.18e+02 (5.05e+01)** | 1.99e+02 (6.72e+01) | 4.33e+01 (4.18e+01) | **3.68e+01 (4.36+01)** | **5.89e+01 (4.02e+01**) | 8.06+01 (5.41+01) |
| $f_4$ | **9.76e+01 (8.72e+01)** | 9.58e+01 (8.69e+01) | **1.36e-01 (1.71e-01)** | 3.02e-02 (6.89e-02) | **2.70e-01 (2.56e-01)** | 1.65e-02 (2.30e-02) |

**Table 2: Kolmogorov-Smirnov statistical tests. A '+' sign means that BS-PSO is significantly better.**

| | *lbest* | | | *gbest* | | |
|---|---|---|---|---|---|---|
| | $\rho = 0$ | $\rho = 0.25$ | *BakSneppen* | $\rho = 0$ | $\rho = 0.25$ | *BakSneppen* |
| $f_1$ | + | + | + | + | + | + |
| $f_2$ | ~ | + | + | ~ | ~ | + |
| $f_3$ | + | + | + | – | ~ | ~ |
| $f_4$ | + | + | + | ~ | + | + |

# 2. RESULTS AND DISCUSSION

Four benchmark functions were used for testing BS-PSO and compare it to IA-PSO: Sphere (1), Rosenbrock (2), Rastrigin (3), Griewank (4). The dimension of the search space is set to $D = 30$. The population size is set to 20 and $c_1 = c_2 = 2$. The maximum number of generations is 3000. A total of 50 runs for each experiment were conducted. The PSOs were tested with *lbest* ($k = 2$ in a ring topology) and *gbest* topologies. Assymetrical initialization within the ranges defined in [6] is used. In order to compare the efficiency of each perturbation scheme, $\rho$ was set to 0, 0.25, and "bak-sneppen controlled". In the $lbest$ topology, BS-PSO attains better results than IA-PSO (Table 1), independently of the perturbation strategy. The superiority of the proposed algorithm is confirmed by the statistical tests in Table 3. As for the *gbest* topology (Table 2), the results are more balanced. However, IA-PSO is only significantly better than BS-PSO in one scenario. When comparing the results with different $\rho$ we see that the performance is more similar when $\rho = 0$. When the parameter is controlled by the Bak-Sneppen model, BS-PSO is clearly better than IA-PSO, except with $f_3$ and $gbest$ topology, in which the two algorithm's results are statistically equivalent.

# ACKNOWLEDGEMENTS

Authors wish to thank FCT (Ministério da Ciência e Tecnologia) Fellowship SFRH/BPD/66876/2009; FCT (ISR/IST plurianual funding) through the PIDDAC Program funds; projects TIN2011-28627-C04-02, by Spanish Ministry of Science an Innovation and P08-TIC-03903, by the Andalusian Regional Government.

# REFERENCES

[1] Bak, P., and Sneppen, K. 1993. Punctuated Equilibrium and Criticality in a Simple Model of Evolution. *Physical Review Letters*, Vol. 71(24), 4083-4086.

[2] Fernandes, C.M., Merelo, J.J., Ramos, V., Rosa, A.C. 2008. A Self-Organized Criticality Mutation Operator for Dynamic Optimization Problems. In *Proceedings of the 2008 Genetic and Evolutionary Computation Conference*, ACM, 937-944.

[3] Kennedy, J.; Eberhart, R. 1995. Particle Swarm Optimization. In *Proceedings of IEEE International Conference on Neural Networks*, Vol.4, 1942–1948.

[4] Løvbjerg, M., Krink, T. 2002. Extending particle swarm optimizers with self-organized criticality. In *Proc. of the 2002 IEEE Congress on Evolutionary Computation*, Vol. 2, IEEE Computer Society, 1588–1593.

[5] Shi, Y. Eberhart, R.C. 1998. A Modified Particle Swarm Optimizer. In *Pro. of IEEE 1998 International Conference on Evolutionary Computation*, IEEE Press, 69–73.

[6] Suresh, K., Ghosh, S., Kundu, D., Sen, A., Das, S., Abraham, A. 2008. Inertia-Adaptive Particle Swarm Optimizer for Improved Global Search. In *Proceedings of the 8th Inter. Conference on Intelligent Systems Design and Applications*, Vol. 2. IEEE, Washington, DC, USA, 253-258.