An Asynchronous and Steady State Update Strategy for the Particle Swarm Optimization Algorithm

C.M. Fernandes[1,2(✉)], J.J. Merelo[2], and A.C. Rosa[1]

[1] LARSyS: Laboratory for Robotics and Systems in Engineering and Science, University of Lisbon, Lisbon, Portugal
{cfernandes,acrosa}@laseeb.org
[2] Department of Computer Architecture, University of Granada, Granada, Spain
jmerelo@geneura.ugr.es

Abstract. This paper proposes an asynchronous and steady state update strategy for the Particle Swarm Optimization (PSO) inspired by the Bak-Sneppen model of co-evolution. The model consists of a set of fitness values (representing species) arranged in a network. By replacing iteratively the least fit species and its neighbors with random values (simulating extinction), the average fitness of the population tends to grow while the system is driven to a critical state. Based on these rules, we implement a PSO in which only the worst particle and its neighbors are updated and evaluated in each time-step. The other particles remain *steady* during one or more iterations, until they eventually meet the update criterion. The steady state PSO (SS-PSO) was tested on a set of benchmark functions, with three different population structures: *lbest* ring and lattice with von Neumann and Moore neighborhood. The experiments demonstrate that the strategy significantly improves the quality of results and convergence speed with Moore neighborhood. Further tests show that the major factor of enhancement is the selective pressure on the worst, since replacing the best or a random particle (and neighbors) yields inferior results.

1 Introduction

The Particle Swarm Optimization (PSO) is a population-based metaheuristics inspired by the social behavior of bird flocks and fish schools [8]. The search is carried out by a swarm of candidate solutions (called *particles*) that move around the fitness landscape of the target-problem, guided by mathematical rules that define their velocity at each time step. The most common configurations of PSO are synchronous: the fitness values of all particles are first computed and only then the particles update their velocity. Carlisle and Dozier [5] proposed a variant in which the velocity vector is updated immediately after computing the fitness of the corresponding particle. In this case, each particle is updated knowing the current best position found by half of its neighbors and the previous best found by the other half: the population of the asynchronous PSO (A-PSO) interacts with imperfect information about the global search. Asynchronous PSOs have been compared to the synchronous configuration (S-PSO) with contradictory results. While Carlisle and Dozier [5] suggested that A-PSO yields better results

J. Handl et al. (Eds.): PPSN XIV 2016, LNCS 9921, pp. 167–177, 2016.
DOI: 10.1007/978-3-319-45823-6_16

than S-PSO, Rada-Vilela *et al.* [13] reported that S-PSO is better than A-PSO in terms of the quality of the solutions and convergence speed.

One of the main motivations for investigating asynchronous update strategies for PSO is the possibility of parallelization [14]. Standard PSOs are easily parallelized (by assigning a particle or a set of particles to each processor, for instance) but due to load imbalances, synchronous update does not make an efficient use of the computational resource. For parallel PSO, asynchronicity is the logical approach. In addition, asynchronicity can also be useful in diversity maintenance and prevention of premature convergence [1], or to speed up convergence by skipping function evaluations [13]. In this paper, we follow an alternative approach. The goal is to design an asynchronous PSO that, unlike the standard A-PSO, significantly improves S-PSO in a wide range of problems. With that objective in mind, we propose a steady state PSO (SS-PSO). A system is said to be in steady state when some of its parts do not change for a period of time. In the SS-PSO, only a fraction of the population is updated and evaluated in each iteration.

The strategy is inspired by the Bak-Sneppen model of co-evolution between interacting species [3]. In order to investigate the dynamics of species extinction and coupled selection, Bak and Sneppen arrange a set of random fitness values (representing species) in a ring structure. Then, they replace the worst species and its neighbors by random values (extinction event), repeating the procedure during several iterations. After a long run, the system is driven to a critical state where most species have reached a fitness above a certain threshold and avalanches of extinction events produce non-equilibrium fluctuations in the configuration of the fitness values.

SS-PSO uses a similar scheme. However, here the worst particle and its neighbors are updated, instead of being replaced by random solutions. Like in the Bak-Sneppen model, the other particles remain steady until an update event hits them. For a proof of concept, the algorithm was tested on ten benchmark functions and compared to S-PSOs. The results show that SS-PSO significantly improves the performance of the S-PSO structured in a 2-dimensional square lattice with Moore neighborhood.

The remaining of the paper is structured as follows. Section 2 gives a background review on synchronous and asynchronous update strategies for the PSO. Section 3 describes the Bak-Sneppen model of co-evolution and introduces the proposed update strategy. Section 4 describes the experiments and discusses the results. Finally, Sect. 5 concludes the paper and outlines futures lines of research.

2 Synchronous and Asynchronous Particle Swarms

PSO is a population-based algorithm in which a group of solutions travels through a fitness landscape according to a set of rules that drives it 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 $\vec{X}_i = (x_{i,1}, x_{i,2}, \ldots x_{1,D})$, where D is the dimension of the search space. Velocity is given

by $\vec{V}_i = (v_{i,1}, v_{i,2}, \ldots v_{1,D})$. 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:

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

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

where 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. Parameters r_1 and r_2 are vectors of random numbers uniformly distributed in the range $[0, 1]$ and c_1 and c_2 are acceleration coefficients that tune the relative influence of each term of the formula. In order to prevent particles from stepping out of the limits of the search space, positions $x_{i,d}(t)$ of the particles are limited by constants that in general correspond to the domain of the problem: $x_{i,d}(t) \in [-Xmax, Xmax]$. Velocity may also be limited within a range in order to prevent the *explosion* of the velocity vector: $v_{i,d}(t) \in [-Vmax, Vmax]$. Usually, $Xmax = Vmax$. Parameter ω is the inertia weight, proposed by Shi and Eberhart [17] to help fine-tuning the balance between local and global search, and it is widely used in PSO implementations.

The neighborhood of the particle defines the value of p_g and is a key factor in the performance of PSO. Most of the PSOs use one of two simple sociometric principles for defining the neighborhood network. One connects all the members of the swarm to one another, and it is called *gbest* (or *star*), where *g* stands for *global*. The degree of connectivity of *gbest* is $k = n$, where n is the number of particles. The other typical configuration, called *lbest* (where *l* stands for local), creates a neighborhood that comprises the particle itself and its k nearest neighbors. The most common *lbest* topology is the *ring* structure, in which the particles are arranged in a ring (resulting in a degree of connectivity $k = 3$, including the particle).

Between the $k = 3$ connectivity of *lbest* ring and $k = n$ of *gbest*, there are several possibilities. Two of the most used are the 2-dimensional square lattices with von Neumann and Moore neighborhood. In [9], Kennedy and Mendes tested several social structures and concluded that when they are ranked by the quality of solutions the structures with $k = 5$ (like the von Neumann lattice) perform better, but when ranked according to the number of iterations needed to meet the criteria, configurations with higher degree of connectivity (like Moore neighborhood, with $k = 9$) perform better. These results are consistent with the premise that low connectivity favors robustness, while higher connectivity favors convergence speed (at the expense of reliability).

In the standard PSO, all particles are evaluated before updating their velocity. Therefore, they move with complete information about the state of the search. In the asynchronous variant, each particle is evaluated immediately after being updated. Independently of the social structure (assuming it is regular), A-PSO particles use the current best position found by half of its neighbors and the previous best found by the other half: the particles are guided by partial or imperfect information.

A-PSO was first discussed by Carlisle and Dozier [5]. Several reports claim that A-PSO outperforms S-PSO. Luo and Zhang [12], for instance, compared the algorithms and concluded that A-PSO is more accurate and faster. However, they tested the

algorithms in only two functions and no statistical test is given. Perez and Basterrechea [15] tested the algorithm on six problems and concluded that A-PSO is faster and as accurate as S-PSO. Rada-Vilela et al. [16] compared S-PSO and A-PSO with ten functions, using a ring structure with number of neighbors k ranging from 2 to 30 and a population of 30 particles. They measured the quality of the solutions and speed of convergence and performed statistical tests on the results, concluding that S-PSO yields better results than A-PSO in unimodal functions. As for the multimodal, S-PSO yields similar or better results. These findings contradict the results of Carlisle and Dozier [5], Luo and Zhang [12] and Perez and Basterrechea [15].

As stated above, parallelization is one of the main motivations for investigating asynchronous PSOs, mainly because synchronous parallel implementations do not make an efficient use of computational resources when load imbalance exists. In this line of work, Koh *et al.* [10] compared parallel asynchronous and synchronous PSOs in homogeneous and heterogeneous environments. They concluded that the parallel performance of the asynchronous version is significantly better than that of asynchronous PSO for heterogeneous environments or heterogeneous computational tasks. Venter and Sobieszczanski-Sobieski [18] also studied and compared parallel synchronous and asynchronous PSOs. Their results indicate that the asynchronous PSO significantly outperforms the synchronous in terms of parallel efficiency.

In this paper, we are not yet concerned with parallelization. We follow a different approach from the works based on Carlisle and Dozier's seminal proposal. Our main objective is to evaluate the numerical results of the algorithm and validate it as an alternative to the standard synchronous approach. The strategy is based on a model of co-evolution that is described in the next section.

3 From a Model of Co-evolution to the Steady State PSO

Natural species in the same eco-system are related through several features (like food chains or symbiosis, for instance) and the extinction of one species affects the species that are related to them, in a chain reaction that can reach large proportions. Fossil records suggest that the size of extinctions events is in power-law proportion to its frequency. It is also known that the biological history of life on Earth is punctuated by catastrophic extinction events. The Bak-Sneppen model [3] was conceived with the objective of understanding the mechanisms underlying these mass extinctions. It consists of a number of species, each one with a fitness value assigned and connected to other species (neighbors). Every time step, the least fit species and its neighbors are eliminated from the system and replaced by individuals with random fitness.

This description may be translated to a mathematical model. The system is defined by n^d fitness numbers f_i arranged on a d-dimensional lattice (ecosystem) with n cells. At each time step, the smallest f value and its $2 \times d$ neighbours are replaced by uncorrelated random values drawn from a uniform distribution. With this simple rule applied iteratively, the system is driven to a critical state where most species have a fitness above a certain threshold. Complex behavior is observed even in the 1-dimensional case, where species are arranged in a ring and each one has two neighbors.

The Bak-Sneppen model is an example of a system with self-organized criticality (SOC) [2], a critical state formed by self-organization in a long transient period at the border of order and chaos. While *order* means that the system is working in a predictable regime where small disturbances have only local impact, *chaos* is an unpredictable state very sensitive to initial conditions or small disturbances. In complex adaptive systems, complexity and self-organization usually arise at that transition region between order and chaos, or *edge of chaos*, as it is sometimes stated. SOC systems are dynamical with a critical point at the region between order and chaos as an attractor. However, and unlike many physical systems, which have a parameter that needs to be tuned in order to obtain the critical state, SOC systems are able to self-tune to the critical point.

SOC and the Bak-Sneppen model inspired, for instance, a metaheuristic called *extremal optimization* (EO) [4]. In EO, a single solution to a problem is modified by local search. The algorithm removes the worst components of the solution and replaces them with randomly generated material. By plotting the fitness of the solution, distinct stages of evolution are observed, where improvement is disturbed by brief periods of dramatic decrease in the quality of the solution. Chen *et al.* [6] used EO to enhance the search abilities of PSO and prevent premature convergence to local optima. They tested the hybrid algorithm on a set of benchmark functions and compared it favorably with other metaheuristics.

Løvbjerg and Krink [11] applied SOC to PSO in order to control the convergence of the algorithm and maintain diversity. The authors introduce a *critical value* associated with each particle and define a rule that increments it when two particles are closer than a *threshold distance*. When the critical value of a particle exceeds a globally set *criticality limit*, the algorithm responds by dispersing the criticality of the particle within a certain surrounding neighborhood. In addition, the algorithm uses the critical value to control the inertia weight. The authors claim that their method is faster and attains better solutions than the standard PSO. However, the algorithm introduces five parameters that must be tuned or set to constant *ad hoc* values.

More recently, Fernandes et al. [7] used the Bak-Sneppen model to control the inertia weight and acceleration coefficients of each particle. An experimental setup demonstrates the validity of the algorithm and shows that the incorporation of each control mechanism improves its performance or at least reduces the tuning effort.

Like the Bak-Sneppen model, the population of PSO is structured by a network. With this likeness in mind, we devised an asynchronous and steady state update strategy for PSO in which only the least fit particle and its neighbors are updated and evaluated in each time step. The neighborhood is defined by the network: if the particles are connected by *lbest* with $k = 3$, only the worst particle and its two nearest neighbors are updated and evaluated; if a lattice with Moore neighborhood is used ($k = 9$), the least fit and its eight nearest neighbors are updated. Please note that local synchronicity is used here: the fitness values of the worst and its neighbors are first computed and only then their velocity is updated. For the remaining working mechanisms and parameters, the algorithm is exactly as standard PSO. Since part of the population remains steady in each time step, we named the algorithm steady state PSO (SS-PSO). SS-PSO is summarized in Algorithm 1.

1.Initialize velocity and position of each particle.
2.For (each particle j):Compute fitness.
3. For (each particle j):Compute $p_i$and p_g.
4. For (each particle j):if jis the least fit particle, update velocity and position of jand neighbors
5. Compute fitness of particles jand neighbors.
6. If (stop criteria not met) return to 3; else, end.

Algorithm 1: SS-PSO

4 Experiments and Results

The experimental setup was constructed with ten benchmark problems (Table 1). Functions f_1, -f_3 are unimodal; f_4, -f_8 are multimodal; f_9 is the shifted f_2 with noise and f_{10} is the rotated f_5 (f_9 global optimum and f_{10} matrix were taken from the CEC2005 benchmark). The dimension of the search space is $D = 30$ (except f_6, with $D = 2$). In order to construct square lattices with von Neumann and Moore neighborhood, population size μ is set to 49, a value that lies within the typical range [8]. Following [16], c_1 and c_2 were set to 1.4962 and ω to 0.7298. *Xmax* is defined as usual by the domain's upper limit and $Vmax = Xmax$. A total of 50 runs for each test were performed. Asymmetrical initialization is used (initialization ranges are in Table 1).

In order to assess the quality of solutions and convergence speed of the algorithms, two sets of experiments were conducted. First, the algorithms were run for a limited amount of iterations (3000 for f_1, f_3 and f_6, 20000 for the remaining) and the fitness of the best solutions found were recorded over the 50 runs. In the second set of experiments the algorithms were all run for 20000 iterations or until reaching a function-specific stop criterion (given in Table 1). The number of iterations required to meet the criterion was recorded and statistical measures were taken over the 50 runs. A success measure was defined as the number of runs in which an algorithm attains the stop criterion. The experimental setup is similar to those in [9, 16].

SS-PSO and S-PSO were implemented with three topologies: *lbest* with $k = 3$ and 2-dimensional square lattices with von Neumann ($k = 5$) and Moore neighborhood ($k = 9$). *Gbest* was not tested for two reasons. Firstly, it is fast but converges often to local optima. We have performed some tests with *gbest* and the success rates were very poor. Furthermore, SS-PSO uses the neighborhood structure to decide which particles to update, i.e., in the von Neumann ($k = 5$), five particles are updated. Since *gbest* has $k = n$, the proposed strategy would update the entire population and be equivalent to the S-PSO. Hence, we have restricted the tests to *lbest*, von Neumann and Moore. Please note that at this point of the research we are not primarily concerned in comparing the update strategy with state of the art PSOs. First, it is necessary to investigate in which situations the proposed algorithm is able to improve the convergence speed and accuracy of standard PSO and understand its underlying mechanisms. Only after the proof of concept we can compare it against other PSOs.

With *lbest*, the steady state strategy could not improve the standard synchronous update: $\text{SS-PSO}_{\text{lbest}}$ yields worse results than $\text{S-PSO}_{\text{lbest}}$ in most of the functions. As for the von Neumann neighborhood, the results are dual: $\text{SS-PSO}_{\text{VN}}$ yields better results in multimodal functions but it is outperformed in the unimodal by the S-PSO_{VN}.

Due to space restrictions, we omit the numerical results of the PSOs with *lbest* and von Neumann network and proceed to the PSOs with Moore neighborhood.

S-PSO$_{\text{Moore}}$ attained the best results in most of the functions when compared to *lbest* and it is faster (considering both mean and median values of the evaluations required to meet the criteria) in every function. When compared to S-PSO$_{\text{VN}}$, S-PSO$_{\text{Moore}}$ is also faster in every function and attains better mean and median fitness values in unimodal functions. These results are consistent with the ones in Kennedy and Mendes [9]. Therefore, we believe that the Moore neighborhood structure is well suited for assessing the validity and relevance of our proposal.

Table 2 shows mean, standard deviation and statistical measures of the empirical distributions of best fitness values attained by S-PSO$_{\text{Moore}}$ and SS-PSO$_{\text{Moore}}$. The later yields better results in most of the functions: it attains lower mean and median fitness values in every unimodal function and in the multimodal f_4, f_6 and f_7. Mann-Whitney U tests were performed to compare the distribution of fitness values of each algorithm in each function. Results of the tests are significant at $p \leq 0.05$ for $f_1, f_2, f_3, f_4, f_6, f_7, f_9$, i.e., the null hypothesis that the two samples come from the same population is rejected. For the remaining functions (f_5, f_8, f_{10}), the null hypothesis is not rejected.

In terms of function evaluations (Table 3), SS-PSO$_{\text{Moore}}$ is faster in the entire set of unimodal problems. In the multimodal problems, SS-PSO$_{\text{Moore}}$ needs less evaluations in f_5, f_6, f_6 and f_8. Results of Mann-Whitney U tests are significant at $p \leq 0.05$ for functions $f_1, f_2, f_3, f_5, f_7, f_8, f_{10}$. Although S-PSO$_{\text{Moore}}$ requires less evaluations in f_4, the result of the statistical test is not significant. Finally, the success rates (Table 3) are similar, except for f_7, in which SS-PSO clearly outperforms the synchronous version, and f_9. In conclusion: the empirical results, together with the statistical tests, show that SS-PSO$_{\text{Moore}}$ outperforms S-PSO$_{\text{Moore}}$ in most of the functions according to accuracy, speed and reliability, while not being outperformed in any case.

The previous tests demonstrate that the steady state update strategy in a PSO structured with Moore neighborhood significantly improves its performance. However, at this point, a question arises: what is the major factor for the performance enhancement, the steady state approach, or the set of particles that are updated? In order to shed light on this issue, a final test was conducted. Two variants of SS-PSO were implemented: one updates the best particle and its neighbors (*replace-best*); the second updates a randomly selected particle and its neighbors (*replace-random*). The algorithms were tested on the set of benchmark functions (see Table 4) and compared to results of the proposed SS-PSO$_{\text{Moore}}$ (or *replace-worst*) given in Tables 2 and 3.

Replace-best update strategy is clearly inferior to *replace-worst*. With the exception of f_1 and f_3, the quality of solutions is degraded when compared to the proposed SS-PSO and even to S-PSO. Success rates are considerably lower in most functions. As for *replace-random*, it improves S-PSO in some functions, but in general it is not better than *replace-worst*: *replace-random* strategy is less accurate and slower in most of the functions. The test shows that selective pressure on the least fit individuals is a major factor in the performance SS-PSO.

Table 1. Benchmark functions.

	Mathematical representation	Range of search/initialization	Stop criterion
Sphere f1	$f_1(\vec{x}) = \sum_{i=1}^{D} x_i^2$	$(-100, 100)^D$ $(50, 100)^D$	0.01
Quadric f2	$f_2(\vec{x}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$	$(-100, 100)^D$ $(50, 100)^D$	0.01
Hyper Ellipsoid f3	$f_1(\vec{x}) = \sum_{i=1}^{D} i x_i^2$	$(-100, 100)^D$ $(50, 100)^D$	0.01
Rastrigin f4	$f_4(\vec{x}) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10 \right)$	$(-10, 10)^D$ $(2.56, 5.12)^D$	100
Griewank f5	$f_5(\vec{x}) = 1 + \frac{1}{4000} \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}} \right)$	$(-600, 600)^D$ $(300, 600)^D$	0.05
Schaffer f6	$f_6(\vec{x}) = 0.5 + \frac{\left(\sin\sqrt{x^2 + y^2} \right)^2 - 0.5}{(1.0 + 0.001(x^2 + y^2))^2}$	$(-100, 100)^2$ $(15, 30)^2$	0.00001
Weierstrass f7	$f_7(\vec{x}) = \sum_{i=1}^{D} \left(\sum_{k=0}^{kmax} \left[a^k cos\left(2\pi b^k (x_i + 0.5) \right) \right] \right)$ $-D \sum_{k=0}^{kmax} \left[a^k cos\left(2\pi b^k \cdot 0.5 \right) \right],$ $a = 0.5,\ b = 3,\ kmax = 20$	$(-0.5, 0.5)^D$ $(-0.5, 0.2)^D$	0.01
Ackley f8	$f_8(\vec{x}) = -20exp\left(-0.2\sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2} \right)$ $-exp\left(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi x_i) \right) + 20 + e$	$(-32.768, 32.768)^D$ $(2.56, 5.12)^D$	0.01
Shifted Quadric with noise f9	$f_9(\vec{z}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2 * (1 + 0.4\lvert N(0.1) \rvert),$ $\vec{z} = \vec{x} - \vec{o},$ $\vec{o} = [o_1, ..o_D]:$ *shifted global optimum*	$(-100, 100)^D$ $(50, 100)^D$	0.01
Rotated Griewank f10	$f_{10}(\vec{z}) = 1 + \frac{1}{4000} \sum_{i=1}^{D} z_i^2 - \prod_{i=1}^{D} \cos\left(\frac{z_i}{\sqrt{i}} \right),$ $\vec{z} = \boldsymbol{M}\vec{x}$, **M**:ortoghonal matrix	$(-600, 600)^D$ $(300, 600)^D$	0.05

Table 2. Best fitness: mean, standard deviation, median, minimum and maximum.

	S-PSO$_{Moore}$					SS-PSO$_{Moore}$				
	Mean	St.dev	Median	Min	Max	Mean	St.dev.	Median	Min	Max
f_1	1.31e−11	1.12e−11	1.04e−11	1.39e−12	4.93e−11	**1.14e−14**	1.16e−14	**7.83e−15**	**2.06e−16**	**6.33e−14**
f_2	1.38e−29	6.36e−29	5.88e−31	4.71e−34	4.42e−28	**1.40e−46**	9.90e−46	**0.00e00**	**0.00e00**	**7.00e−45**
f_3	3.61e−11	3.50e−11	2.76e−11	2.64e−12	1.65e−10	**3.57e−14**	5.32e−14	**1.58e−14**	**9.43e−16**	**2.56e−13**
f_4	6.36e+01	1.73e+01	6.17e+01	3.78e+01	1.13e+02	**5.25e+01**	1.45e+01	**5.12e+01**	**2.19e+01**	**1.04e+02**
f_5	**5.86e−03**	7.22e−03	**1.08e−19**	**0.00e00**	2.95e−02	1.28e−02	2.06e−02	9.86e−03	**0.00e00**	1.20e−01
f_6	1.94e−04	1.37e−03	**0.00e00**	**0.00e00**	9.72e−03	**0.00e00**	0.00e00	**0.00e00**	**0.00e00**	**0.00e00**
f_7	1.94e−01	5.27e−01	2.27e−02	**2.86e−05**	3.16e00	**1.73e−02**	8.17e−02	**2.86e−05**	**2.86e−05**	**5.19e−01**
f_8	**1.01e−15**	2.01e−16	**8.88e−16**	**8.88e−16**	1.33e−15	1.07e−15	2.20e−16	**8.88e−16**	**8.88e−16**	**1.33e−15**
f_9	3.60e+01	2.32e+02	9.8e−05	6.44e−07	1.64e+03	**7.20e−05**	1.54e−04	**1.01e−05**	1.73e−08	7.11e−04
f_{10}	**6.70e−03**	9.20e−03	**1.08e−19**	**0.00e00**	3.68e−02	8.37e−03	1.01e−02	**1.08e−19**	0.00e00	3.70e−02

Table 3. Evaluations: mean, standard deviation, median, minimum and maximum.

	S-PSO$_{Moore}$						SS-PSO$_{Moore}$					
	Mean	St.dev.	Median	Min	Max	SR	Mean	St.dev.	Median	Min	Max	SR
f_1	20434.0	840.8	20433. 0	18326	22099	50	**17241.3**	716.2	**17320.5**	**14526**	**18594**	50
f_2	168599.0	12721.1	168119.0	140630	193501	50	**133140.6**	16854.2	**135828.0**	**105399**	**171702**	50
f_3	22987.9	1075.4	22956.5	20972	26019	50	**19519.6**	788.0	**19561.5**	**18045**	**21600**	50
f_4	**15635.0**	7771.5	**13524.0**	**7448**	49392	49	15902.8	8047.7	14256.0	7659	58248	49
f_5	18671.0	986.8	18595.5	16366	21952	50	**16419.2**	1300.7	**16060.5**	**14607**	**19683**	50
f_6	11443.0	9439.1	7105.0	3822	39788	49	**8049.0**	4852.6	**6381.0**	**2727**	**21744**	50
f_7	37272.7	1590.1	36970.5	34790	41846	24	**33192.0**	1184.8	**33340.5**	**30645**	**35685**	46
f_8	21029.8	1164.7	20923.0	19012	24794	50	**17723.6**	957.0	**17752.5**	**15750**	**19809**	50
f_9	704144.6	96262.6	706972	453201	922327	47	**653808.8**	95860.3	**671175**	**425655**	852786	50
f_{10}	18876.8	901.7	18963	**16954**	**20727**	**50**	**16140.8**	1122.9	**15975**	**13995**	**18612**	**50**

Table 4. Results of SS-PSO variants: median, min, max and success rates (SR)

	SS-PSO$_{Moore}$ (replace-best)							SS-PSO$_{Moore}$ (replace-random)						
	Fitness			Evaluations				Fitness			Evaluations			
	Median	Min	Max	Median	Min	Max	SR	Median	Min	Max	Median	Min	Max	SR
f_1	4.09e−29	2.50e−33	2.00e +04	9468	6714	24669	45	6.04e−14	7.86e−14	6.59e−12	18972	16425	20781	50
f_2	1.50e+04	0.00e00	4.50e +04	66717	65844	79443	3	8.33e−32	4.59e−34	5.00e+03	170091	136062	195498	47
f_3	3.01e−27	9.54e−34	1.00e +05	11718	8208	36000	35	1.66e−12	1.30e−13	2.25e−11	21118	19548	23283	50
f_4	1.30e+02	7.46e+01	2.00e +02	15192	8964	108495	9	5.62e+01	2.39e+01	8.76e+01	11052	5679	23571	50
f_5	4.17e−02	1.08e−19	9.05e +01	8014.5	6570	21186	28	7.40e−03	0.00e00	4.18e−02	17190	15570	19989	50
f_6	3.59e−04	0.00e00	9.72e −03	39811.5	1242	140247	38	0.00e00	0.00e00	9.72e−03	8460	3276	62091	50
f_7	7.35e00	2.51e00	1.38e +01	-	-	-	0	7.57e−04	2.86e−05	2.02e00	34168,5	31041	42507	32
f_8	2.28e00	8.86e−16	3.84e00	20898	13158	28764	6	1.11e−15	8,86e−16	1.33e−15	19822.5	18252	25416	50
f_9	1.06e−01	1.98e−03	1.53e +04	902407	812736	949590	12	1.64e−04	1.44e−06	6.01e+01	736713	546858	891432	49
f_{10}	4.17e−02	1.08e−19	9.05e +01	8014.5	6570	21186	27	7.40e−03	0.00e00	4.18e−02	17190 (50)	15570	19989	50

5 Conclusions and Future Work

This paper proposes an asynchronous and steady state update strategy for the PSO based on a model of co-evolution. Instead of the whole population, like in standard particle swarms (either synchronous or asynchronous), only the worst solution and its neighbors are updated and evaluated in each time step. The remaining particles are kept in a steady state. Accordingly, we have named it steady state PSO (SS-PSO).

The strategy was implemented with three social network structures (*lbest* and square lattices with von Neumann and Moore neighborhood) and tested on a set of ten unimodal, multimodal, shifted, noisy and rotated benchmark problems. Quality of solutions, convergence speed and success rates were compared. SS-PSO significantly improved the performance of S-PSO on a lattice with Moore neighborhood in every function. Since S-PSO$_{\text{Moore}}$ has been found to be the more accurate and faster POS in the set of benchmark functions, we believe that these results validate the proposal.

The strategy was tested with standard PSOs. In the future, and in order to assess the contribution of our proposal to the state of the art, we intend to test it with (efficient variants of the standard PSO and even compare it to other metaheuristics. Scalability of the steady state PSO regarding population size and problem dimension will also be studied. Finally, the emergent patterns of the algorithm (extension of events, stasis, critical values) will be compared to those of the Bak-Sneppen model.

Acknowledgements. First author wishes to thank FCT, *Ministério da Ciência e Tecnologia*, his Research Fellowship SFRH/BPD/66876/2009). This work was supported by FCT PROJECT [UID/EEA/50009/2013], EPHEMECH (TIN2014-56494-C4-3-P, Spanish Ministry of Economy and Competitivity), PROY-PP2015-06 (Plan Propio 2015 UGR), project CEI2015-MP-V17 of the Microprojects program 2015 from CEI BioTIC Granada.

References

1. Aziz, N.A.B., Mubin, M., Mohamad, M.S., Aziz, K.A.: A synchronous-asynchronous particle swarm optimisation algorithm. Sci. World J. **2014**, 1–17 (2014). Article ID 123019
2. Bak, P., Tang, C., Wiesenfeld, K.: Self-organized criticality: an explanation of 1/f noise. Phys. Rev. Lett. **59**(4), 381–384 (1987)
3. Bak, P., Sneppen, K.: Punctuated equilibrium and criticality in a simple model of evolution. Phys. Rev. Lett. **71**(24), 4083–4086 (1993)
4. Boettcher, S., Percus, A.G.: Optimization with extremal dynamics. Complexity **8**(2), 57–62 (2003)
5. Carlisle, A., Dozier, G.: An off-the-shelf PSO. In: Workshop on Particle Swarm Optimization (2001)
6. Chen, M.-R., Li, X., Lu, Y.-Z.: A novel particle swarm optimizer hybridized with extremal optimization. App. Soft Comput. **10**(2), 367–373 (2010)
7. Fernandes, C.M., Merelo, J.J., Rosa, A.C.: Controlling the parameters of the particle swarm optimization with a self-organized criticality model. In: Coello, C.A., Cutello, V., Deb, K., Forrest, S., Nicosia, G., Pavone, M. (eds.) PPSN 2012, Part II. LNCS, vol. 7492, pp. 153–163. Springer, Heidelberg (2012)

8. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
9. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proceedings of the IEEE World Congress Evolutionary Computation, pp. 1671–1676 (2002)
10. Koh, B.-I., George, A.D., Haftka, R.T., Fregly, B.J.: Parallel asynchronous particle swarm optimization. Int. J. Numer. Meth. Eng. **67**(4), 578–595 (2006)
11. Løvbjerg, M., Krink, T.: Extending particle swarm optimizers with self-organized criticality. In: Proceedings of the 2002 IEEE Congress on Evolutionary Computation, vol. 2, pp. 1588–1593. IEEE Computer Society (2002)
12. Luo, J., Zhang, Z.: Research on the parallel simulation of asynchronous pattern of particle swarm optimization. Comput. Simul. **22**(6), 78–170 (2006)
13. Majercik, S.: GREEN-PSO: conserving function evaluations in particle swarm optimization. In: Proceedings of the IJCCI 2013, pp. 160–167 (2013)
14. McNabb, A.: Serial PSO results are irrelevant in a multi-core parallel world. In: Proceedings of the 2014 IEEE Congress on Evolutionary Computation, pp. 3143–3150 (2014)
15. Perez, R., Basterrechea, J.: Particle swarm optimization and its application to antenna farfield-pattern prediction from planar scanning. Microw. Opt. Technol. Lett. **44**(5), 398–403 (2005)
16. Rada-Vilela, J., Zhang, M., Seah, W.: A performance study on synchronous and asynchrounous updates in particle swarm. Soft. Comput. **17**(6), 1019–1030 (2013)
17. Shi, Y., Eberhart, R.C.: A modified particle swarm optimizer. In: Proceedings of IEEE International Conference on Evolutionary Computation, pp. 69–73. IEEE (1998)
18. Venter, G., Sobieszczanski-Sobieski, J.: A parallel particle swarm optimization algorithm accelerated by asynchronous evaluations. J. Aerosp. Comput. Inf. Commun. **3**(3), 123–137 (2006)