Standard Particle Swarm Optimisation

Maurice Clerc

Standard Particle Swarm Optimisation

From 2006 to 2011

Maurice.Clerc@WriteMe.com

2012-09-23 version

1 Introduction

Since 2006, three successive standard PSO versions have been put on line on the Particle Swarm Central [10], namely SPSO 2006, 2007, and 2011. The basic principles of all three versions can be informally described the same way, and in general, this statement holds for almost all PSO variants. However, the exact formulae are slightly different, because they took advantage of latest theoretical analysis available at the time they were designed.

2 Informal description

You have a *search space*. On each point of this search space, you know how to evaluate a *fitness*, which is a numerical value. Now, you are looking for the best point, i.e. the one that has the best fitness (say the smallest one). This point is called the *global optimum point* (or simply *optimum point*, in short). In order to do that, SPSO makes use of "agents" called *particles*, which move step by step. A step is called an *iteration* (or sometimes a *time step*). A particle is made of

- a position inside the search space
- the fitness value at this position
- a velocity (in fact a displacement), which will be used to compute the next position
- a memory, that contains the best position (called the *previous best*) found so far by the particle.
- the fitness value of this previous best

可以在这里加上<形态学>

The set of particles is called the *swarm*. Inside the swarm a topology is defined: it is a set of links between particles, saying "who informs whom". When a particle is informed by another one, it means that the particle knows the previous best

(position and fitness) of the "informing" particle. The set of particles that informs a particle is called its *neighbourhood*. In SPSO, the neighbourhood contains the particle itself, but is not necessarily symmetrical. The search is performed in two phases: initialisation of the swarm, and then a cycle of iterations.

2.1 Initialisation of the swarm

For each particle:

- pick a random position inside the search space. Compute the fitness. Begin with the previous best set to this initial position.
- pick a random velocity

2.2 Iteration

- compute the new velocity (displacement), by combining the following elements:
  - the current position
  - the current velocity
  - the previous best
  - the best previous best in the neighbourhood
- move, by applying this new velocity to the current position
- apply a confinement method, to ensure that the new position is still inside the search space, and compute the new fitness. One may use the "let them fly" method: no confinement, and if the new position is outside the search space, no re-evaluation
- if the new fitness is better than that of the previous best position, replace the previous best by a copy of the new position, and its fitness by a copy of the new fitness

For the iterations, there are two stop criteria:

- when the fitness value on the optimum point is known, a maximum admissible error. As soon as the absolute difference between this known fitness on the optimum point and the best one that has been found is smaller than this error, the algorithm stops
- a maximum number of fitness evaluations, given in advance. As in standard PSO the swarm size is constant, it is equivalent to a maximum number of iterations

3 Formal description

Let us define some notations

D is the dimension of the search space

E is the search space, a hyperparallelepid defined as the Euclidean product of D real intervals (for a discrete search space, see section 4).

$$E = \bigotimes_{d=1}^{D} [min_d, max_d] \tag{3.1}$$

f is the fitness function defined on E. We suppose here that we are looking for its minimum

t is the number of the current time step

$x_i(t)$ is the position of the particle i at time t. It has D coordinates.

$v_i(t)$ is the velocity at time t. It has D components.

$p_i(t)$ is the previous best position, at time t. It has D coordinates.

$l_i(t)$ is the best previous best position found in the neighbourhood. It has D coordinates.

3.1 The swarm size

The swarm size is S and the particles are numbered from 0 to S-1. However S is defined differently in the three SPSO versions:

- SPSO 2006. Automatically computed by a formula depending on the dimension D

$$S = 10 + \left[2\sqrt{D}\right] \tag{3.2}$$

in which $[u]$ is the integer part of the real number u.

- SPSO 2007. The same as above

- SPSO 2011. User defined. The suggested value is 40. Optionally, for a sequence of runs, the swarm size can be randomly chosen "around" a given value. The reason for this "regression" is that quite often, formula 3.2 gives result that is far from the best swarm size (see figure 3.1). Clearly, there is still a lack of theoretical analysis about how an adaptive swarm size could be defined.

退化现象

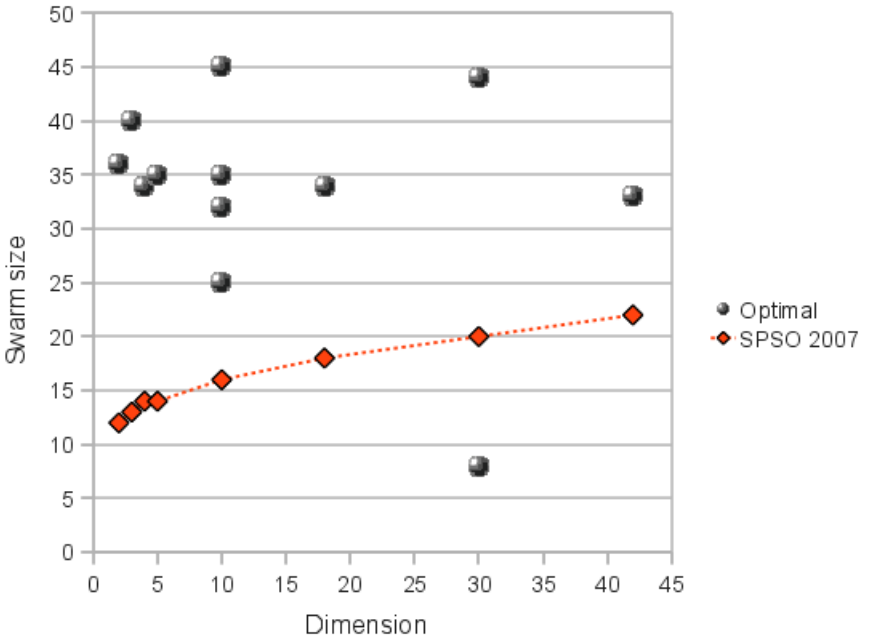

Fig. 3.1: On a set of 12 problems of different dimensions, the formula for the swarm size is far from optimal

3.2 The neighbourhood (information links)

3.2.1 The ring topology

This topology has been used for years in different PSO variants because of its simplicity. For that reason, we give a brief description of this topology here. However, a more robust one, partly random, has been used for SPSO since 2006. The neighbourhood of the particle i is

$$i-1\,mod(S), i, i+1\,mod(S)$$

For instance, if $S = 20$ the neighbourhood of the particle 0 is $\{19, 0, 1\}$, and the neighbourhood of the particle 19 is $\{18, 19, 0\}$. .

3.2.2 The adaptive random topology

This topology has been defined in [2], and is used in the three SPSO versions we are considering. Note that it is a particular case of the "stochastic star" proposed later in [8]. At the very beginning, and after each unsuccessful iteration (no improvement of the best known fitness value), the graph of the information links is modified: each particle informs at random K particles (the same particle may be chosen several times), and informs itself. The parameter K is usually set to 3. It means that each particle informs at less one particle (itself), and at most $K+1$ particles (including itself). It also means that each particle can be informed by any number of particles between 1 and S. However, the distribution of the possible number of "informants" is not uniform. On average, a particle is often informed by about K others, but it may be informed by a much larger number of particles with a small probability, as we can see from figure 3.2.

3.3 Initialisations

3.3.1 SPSO 2006 and 2007

Let $N_i(t)$ be the set of neighbours of the particle i at time t. A first random graph of links is generated, as explained above, so that all $N_i(0)$ are defined. Then we initialise the other elements as follows:

$$\begin{cases} x_i(0) &= U(min_d max_d) \\ v_i(0) &= \frac{U(min_d, max_d) - x_i(0)}{2} \\ p_i(0) &= x_i(0) \\ l_i(0) &= argmin_{j \in N_i(0)}(f(p_j(0))) \end{cases} \tag{3.3}$$

where $U(min_d, max_d)$ is a random number in $[min_d, max_d]$ drawn according to the uniform distribution. 除了笛卡尔，这世界上没人了?

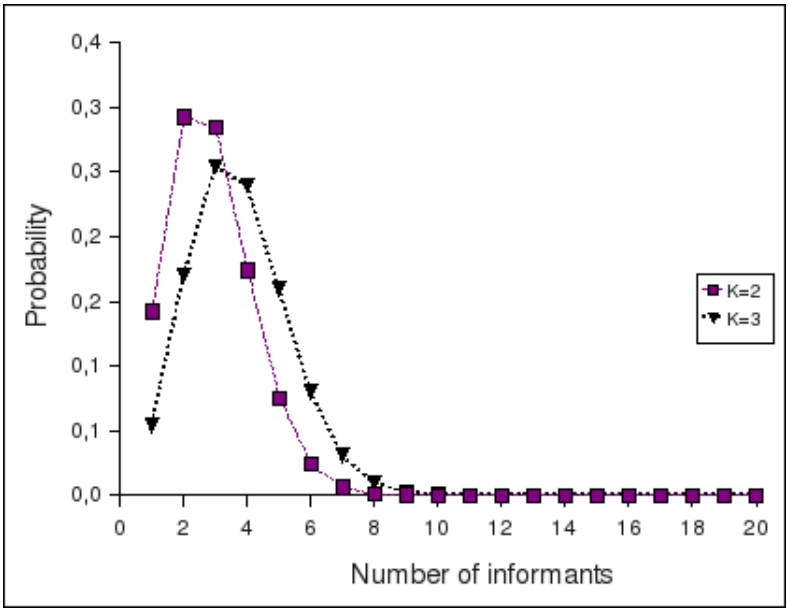

Fig. 3.2: Adaptive random topology. Distribution of the number of informants of a particle.

3.3.2 SPSO 2011

The initialisation is the same except for the velocity. Indeed, it has been proved in 2008 [5] that with the above formula most particles immediately leave the search space when the dimension D is high. In the new formula each component d is computed as follows

$$v_i(0) = U(min_d - x_{i,d}(0), max_d - x_{i,d}(0)) \tag{3.4}$$

There is an option that normalises the search space to transform it into a hypercube if necessary. This may be useful because the specific velocity update method (see 3.4.2) makes use of hyperspheres and seems to work better in a hypercube. Of course, in that case, the fitness is evaluated in the real search space.

3.4 The velocity update equations

The fundamental velocity update equation is a combination of three vectors

$$v_i(t+1) = \mathcal{C}(v_i(t), p_i(t) - x_i(t), l_i(t) - x_i(t)) \tag{3.5}$$

which is immediately followed by the move itself

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{3.6}$$

The combination $\mathcal{C}$ is defined differently in the successive SPSO versions, but is always partly random.

3.4.1 SPSO 2006 and 2007

The velocity update equation is applied dimension by dimension, as follows:

$$\begin{cases} v_{i,d}(t+1) = & wv_{i,d}(t) + \\ & U(0,c)(p_i(t) - x_i(t)) + U(0,c)(l_i(t) - x_i(t)) \end{cases} \tag{3.7}$$

The parameter values are

$$\begin{cases} w = \frac{1}{2\ln(2)} \simeq 0.721 \\ c = \frac{1}{2} + \ln(2) \simeq 1.193 \end{cases} \tag{3.8}$$

These values are coming from [3].

The next position is given by equation 3.6, which applies the new velocity to the current position. Equivalently we can also define two intermediate points (t is omitted for simplicity):

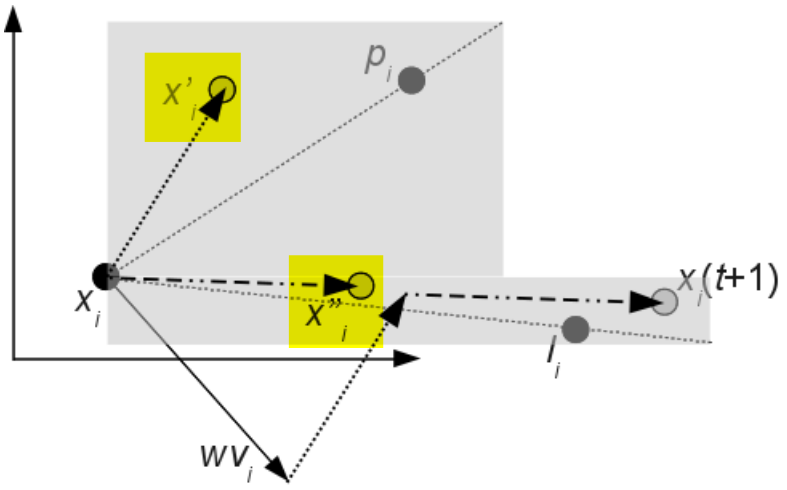

Fig. 3.3: SPSO 2006 and 2007. Construction of the next position. The points x_i' and x_i'' are chosen at random inside two hyperparallelepids parallel to the axis's

x_i', chosen at random (uniform distribution) inside the hyperparallelepid

$$\bigotimes_{d=1}^{D} [x_{i,d}, x_{i,d} + c\,(p_{i,d} - x_{i,d})]$$

x_i'', chosen at random (uniform distribution) inside the hyperparallelepid

$$\bigotimes_{d=1}^{D} [x_{i,d}, x_{i,d} + c\,(l_{i,d} - x_{i,d})]$$

and say that the new position is

$$x_i\,(t+1) \quad = \quad wv_i + (x_i' - x_i) + (x_i'' - x_i) \tag{3.9}$$

This construction is illustrated in figure 3.3. It is clear that the new position depends on the system of coordinates. If we consider the distribution of all possible next positions (DNPP), its support is a D-rectangle, and it is *not* uniform (more dense near to the centre). In 2D, it looks like a "Maya pyramid".

3.4.2 SPSO 2011

It was well known for years ([9]) that the above dimension by dimension method is biased: when the optimum point lies on a axis, or on a diagonal, or worse, on the centre of the system of coordinates, it is "too easy" for PSO to find it. A more complete analysis of this phenomenon has been presented in 2010 [11]. That is why in SPSO 2011 the velocity is modified in a "geometrical" way that does not depend on the system of coordinates.

Let $G_i(t)$ be the centre of gravity of three points: the current position, a point a bit "beyond" the best previous position (relative to $x_i(t)$), and a point a bit "beyond" the best previous position in the neighbourhood. Formally, it is defined by the following formula, in which t has been removed for simplicity

$$G_i = \frac{x_i+(x_i+c(p_i-x_i))+(x_i+c(l_i-x_i))}{3} = x_i + c\frac{p_i+l_i-2x_i}{3} \quad (3.10)$$

We define a random point x_i' (not necessarily according to an uniform distribution) in the hypersphere

$$\mathcal{H}_i(G_i, \|G_i - x_i\|)$$

of centre G_i and of radius $\|G_i - x_i\|$. Then the velocity update equation is

$$v_i(t+1) = wv_i(t) + x_i'(t) - x_i(t) \quad (3.11)$$

It means that the new position is simply

$$x_i(t+1) = wv_i(t) + x_i'(t)$$

Note that with this method it is easy to rigorously define "exploitation" and "exploration". There is *exploitation* when $x_i(t+1)$ is inside at least one hypersphere $\mathcal{H}_j$, and *exploration* otherwise.

The source code contains some options (like hyperspherical Gaussian distribution) that are not described here. The default method is "uniform random direction, and uniform radius, i.e. $r = U(0, r_{max})$. The support of the resulting DNPP is a D-sphere, and the distribution itself is *not* uniform. It would be only with$r = (U(0, r_{max}))^{1/D}$. It is more dense near to the centre, like with SPSO 2006 and 2007.

3.4.3 When local best=previous best

We may sometimes have $l_i(t) = p_i(t)$ when the particle i is precisely the one that has the best previous best in its neighbourhood. Depending on the SPSO versions, the rule that is applied is slightly different:

- SPSO 2006. Nothing special.

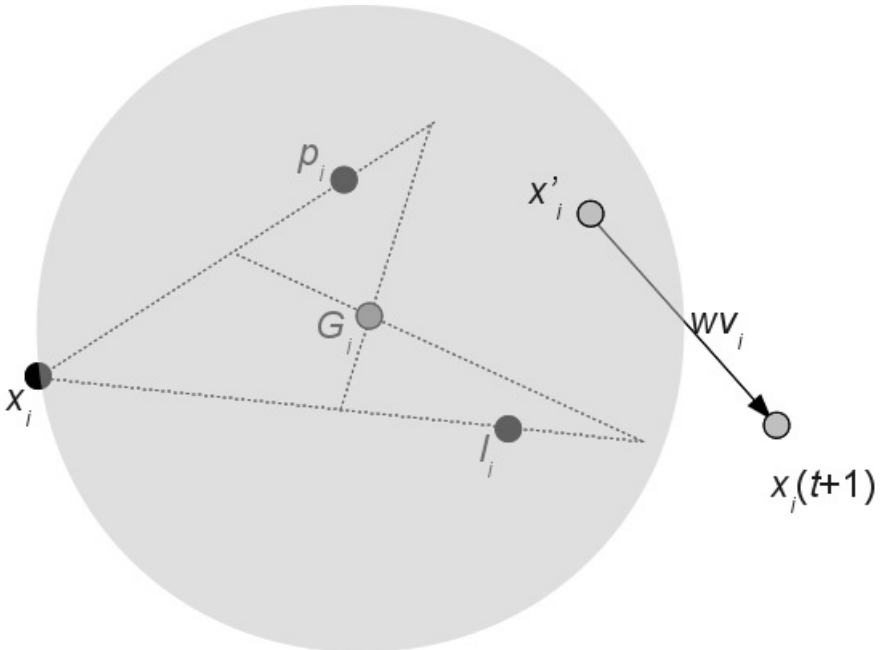

Fig. 3.4: SPSO 2011. Construction of the next position. The point x_i' is chosen at random inside the hypersphere $\mathcal{H}_i\left(G_i, \|G_i - x_i\|\right)$

- SPSO 2007. The last term of equation 3.7 is removed

$$v_{i,d}(t+1) = wv_{i,d}(t) + U(0,c)(p_i(t) - x_i(t)) \tag{3.12}$$

- SPSO 2011. Same idea, $l_i(t)$ is ignored. Equation 3.10 that defines the gravity centre G_i becomes

$$G_i = \frac{x_i + (x_i + c(p_i - x_i))}{2} = x_i + c\frac{p_i - x_i}{2} \tag{3.13}$$

3.5 Confinement

The default method is the same for SPSO 2006 and 2007: for each dimension d the particle must stay inside $[min_d, max_d]$, and each edge is seen as a "wall". So the position and the velocity computed by 3.6, 3.7, and 3.11 may be modified

$$\begin{cases} if\ (x_{i,d}(t+1) < min_d) \quad then \quad \begin{cases} x_{i,d}(t+1) = min_d \\ v_{i,d}(t+1) = 0 \end{cases} \\ if\ (x_{i,d}(t+1) > max_d) \quad then \quad \begin{cases} x_{i,d}(t+1) = max_d \\ v_{i,d}(t+1) = 0 \end{cases} \end{cases} \tag{3.14}$$

For SPSO 2011, the formulae are quite similar, except that the velocity is not set to zero:

$$v_{i,d}(t+1) = -0.5v_{i,d}(t+1) \tag{3.15}$$

The source codes of the three SPSO contain some other options (like the "random back" or the "let them fly" methods) that are not described here. For a detailed comparison of various confinement methods, see [4], or [1].

3.6 Sequential versus parallel

According to the informal description given in section 2, it is clear that the moves can be computed in a parallel (synchronous) way.

- SPSO 2006. A pure simple loop "for i=1 to S do ..." inside each iteration. It means that for a given t the behaviour of particle j may depends on one of the particles i if $i < j$. It happens when the particle i improves its previous best and becomes the best neighbour of the particle j.

- SPSO 2007. To prevent any premature convergence that may be induced by moving the particles always in the same order, their numbering is randomly permuted before each iteration (time step). The Matlab version is synchronous: there is a separate loop to update all the previous best positions, after the moves.

- SPSO 2011. The same.

Note that the sequential/asynchronous method seems to be slightly better, experimentally. On the other hand, for big problems, it may be necessary to run the algorithm on parallel machines, and so, of course, the synchronous method has to be used. Also, the synchronous method would be more consistent with the informal description.

4 SPSO and discrete or heterogeneous search space

Along any dimension d the search space may be discrete, according to a "granularity" q_d. In that case, the corresponding coordinate of the new position is clamped to the nearest acceptable one

$$x_{i,d}(t+1) \quad \Leftarrow \quad q_d \left[0.5 + x_{i,d}(t+1)/q_d\right] \tag{4.1}$$

5 SPSO and random number generation

Theoretically, SPSO should use pure random numbers. In practice, the three versions make use of pseudo-random number generators (RNG):

- SPSO 2006. The RNG is the one provided by the implementation language (C in the version available on the Particle Swarm Central)
- SPSO 2007. The RNG is KISS [7], a pretty good one, at least far better than the C one. However the Matlab version available on the PSC does not explicitly use it
- SPSO 2011. The same. For fair comparisons and reproducible results, the suggestion is to always use the same seed (1294404794) before any sequence of runs. Also, the RNG is "warmed up" before to be really used in the algorithm, as suggested in [6]. Again, this is not implemented in the Matlab version.

It is worth noting that on some problems the performance is *very sensitive* to the RNG. Moreover, the number of runs needed for a correct estimation of an average performance may be far bigger than what is usually done in practice, which is typically about 30. See figure 5.1 for an example.

6 Conclusion

From SPSO 2006 to SPSO 2011, the tendency is clear: to design an algorithm whose behaviour is more and more consistent with the informal basic principles which remain the same since 1995, and less and less dependent on the way they are coded. From this point of view, replacing the "dimension by dimension" coding by a geometrical one (in SPSO 2011), which is non-sensitive to the system of

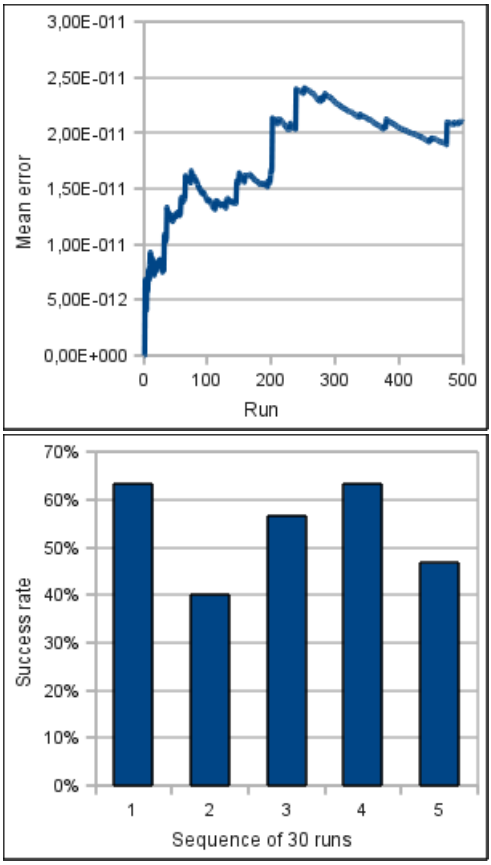

Fig. 5.1: On the classical Gear train problem, you need at least 200 runs for a correct estimation of the performance (runs with SPSO 2011). Also, depending on the seed of the RNG, over just 30 runs the success rate may be very different.

coordinates, can be seen as an improvement. However, from this point of view, some further improvements are still possible:

- the asynchronous method could be replaced by the synchronous one
- the swarm size does not need to be constant. An adaptive swarm size would still be consistent with the informal description
- the topology may be modified in a different way

Note that the purpose of a well defined version of SPSO is to be a kind of milestone. As written in the comments of all versions

This PSO version does not intend to be the best one on the market

It means that if your brand new algorithm does not significantly "beat" SPSO (on a difficult enough non-biased benchmark) you have to improve it.

References

[1] Maurice Clerc. Confinements and biases in particle swarm optimisation. Technical report, Open archive HAL http://hal.archives-ouvertes.fr/, ref. hal-00122799, 2006.

[2] Maurice Clerc. *Particle Swarm Optimization*. ISTE (International Scientific and Technical Encyclopedia), 2006.

[3] Maurice Clerc. Stagnation analysis in particle swarm optimization or what happens when nothing happens, http://hal.archives-ouvertes.fr/hal-00122031. Technical report, 2006.

[4] Sabine Helwig and Rolf Wanka. Particle Swarm Optimization in High-Dimensional Bounded Search Spaces. In *Proceedings of the 2007 IEEE Swarm Intelligence Symposium*, pages 198–205, Honolulu, Hawaii, USA, April 2007. IEEE Press.

[5] Sabine Helwig and Rolf Wanka. Theoretical analysis of initial particle swarm behavior. In *10th International Conference on Parallel Problem Solving from Nature (PPSN 2008).*, pages 889–898, Dortmund, Germany, September 13-17 2008. PPSN X, Springer.

[6] David Jones. Good practice in (pseudo) random number generation for bioinformatics applications. Technical report, UCL Bioinformatics Group, 2010.

[7] G. Marsaglia and A. Zaman. The KISS generator. Technical report, Dept. of Statistics, U. of Florida, 1993.

[8] Vladimiro Miranda, Hrvoje Keko, and Alvaro Jaramillo Duque. Stochastic Star Communication Topology in Evolutionary Particle Swarms (EPSO). *International Journal of Computational Intelligent Research*, 2008.

[9] Christopher K. Monson and Kevin D. Seppi. Exposing Origin-Seeking Bias in PSO. In *GECCO'05*, pages 241–248, Washington, DC, USA, 2005.

[10] PSC. Particle Swarm Central, http://www.particleswarm.info.

[11] William M. Spears, Derek T. Green, and Diana F. Spears. Biases in particle swarm optimization. *International Journal of Swarm Intelligence Research*, 1(2):34–57, 2010.