# Particle Swarm Optimization

Yuhui Shi
Electronic Data Systems, Inc.
Kokomo, IN 46902, USA

***Abstract.*** **This paper surveys the research and development of PSO in five categories: algorithms, topology, parameters, hybrid PSO algorithms, and applications.**

## 1. Introduction

Particle swarm optimization (PSO) is one of the evolutionary computation techniques. Like the other evolutionary computation techniques, PSO is a population-based search algorithm and is initialized with a population of random solutions, called particles. Unlike in the other evolutionary computation techniques, each particle in PSO is also associated with a velocity. Particles fly through the search space with velocities which are dynamically adjusted according to their historical behaviors. Therefore, the particles have a tendency to fly towards the better and better search area over the course of search process. Since its introduction in 1995 (Kennedy and Eberhart 1995, Eberhart and Kennedy 1995), PSO has attracted a lot of attentions from the researchers around the world. A lot of research results have been reported in the literature. Special sessions have being organized in several conferences including the Congress on Evolutionary Computation since 1998. In 2003, the first IEEE Symposium on Swarm Intelligence was held in Indianapolis, Indiana, USA. The first book dedicated to PSO, Swarm Intelligence, coauthored by James Kennedy, Russell Eberhart with Yuhui Shi (Kennedy, Eberhart and Shi 2001) was published in 2001 by Morgan Kaufmann Publisher.

The researches on PSO generally can be categorized into five parts: algorithms, topology, parameters, hybrid PSO algorithms, and applications.

## 2. Algorithms

### *2.1. Original algorithm*

The original PSO algorithm is discovered through simplified social model simulation. It is related to the bird flocking, fishing schooling, and swarm theory. The PSO was first designed to simulate birds seeking food which is defined as a "cornfield vector." The bird would find food through social cooperation with other birds around it (within its neighborhood). It was then expanded to multidimensional search. The topological rather than Euclidean neighborhood was utilized (Kennedy and Eberhart 1995, Eberhart and Kennedy 1995, Eberhart, Simpson and Dobbins 1996). The original PSO algorithm is described as below:

$$v_{id} = v_{id} + c_1 rand()(p_{id}-x_{id}) + c_2 Rand()(p_{gd}-x_{id}) \quad (1a)$$

$$x_{id} = x_{id} + v_{id} \quad (1b)$$

where $c_1$ and $c_2$ are positive constants, and *rand*() and *Rand*() are two random functions in the range [0,1]; $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$ represents the ith particle; $P_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$ represents the best previous position (the position giving the best fitness value) of the ith particle; the symbol $g$ represents the index of the best particle among all the particles in the population; $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$ represents the rate of the position change (velocity) for particle $i$.

Equation (1) is the equation describing the flying trajectory of a population of particles. Equation (1a) describes how the velocity is dynamically updated and Equation (1b) the position update of the "flying" particles. Equation (1a) consists of three parts. The first part is the momentum part. The velocity can't be changed abruptly. It is changed from the current velocity. The second part is the "cognitive" part which represents private thinking of itself - learning from its own flying experience. The third part is the "social" part which represents the collaboration among particles - learning from group flying experience (Shi and Eberhart 1998b).

In Equation (1a), if the sum of the three parts on the right side exceeds a constant value specified by user, then the velocity on that dimension is assigned to be $\pm V_{max}$, that is, particles' velocities on each dimension is clamped to a maximum velocity $V_{max}$, which is an important parameter, and originally is the only parameter required to be adjusted by users. Big $V_{max}$ has particles have the potential to fly far past good solution areas while a small $V_{max}$ has particles have the potential to be trapped into local minima, therefore unable to fly into better solution areas. Usually a fixed constant value is used as the $V_{max}$, but a well-designed dynamically changing $V_{max}$ might improve the PSO's performance (Fan and Shi 2001).

The PSO algorithm is simple in concept, easy to implement and computational efficient. The original procedure for implementing PSO is as follows:

1. Initialize a population of particles with random positions and velocities on $D$ dimensions in the problem space.
2. For each particle, evaluate the desired optimization fitness function in $D$ variables.
3. Compare particle's fitness evaluation with its *pbest*. If current value is better than *pbest*, then set *pbest* equal to the current value, and $P_i$ equals to the current location $X_i$ in $D$-dimensional space.
4. Identify the particle in the neighborhood with the best success so far, and assign its index to the variable $g$.
5. Change the velocity and position of the particle according to Equation (1a) and (1b).
6. Loop to step 2) until a criterion is met, usually a sufficiently good fitness or a maximum number of iterations.

Like the other evolutionary algorithms, PSO algorithms is a population based search algorithm with random initialization, and there is interactions among population members. Unlike the other evolutionary algorithms, in PSO, each particle fly through the solution space, and has the ability to remember its previous best position, survives from generation to generation (Shi and Eberhart 2001b). Furthermore, compared with the other evolutionary algorithms, e.g. evolutionary programming, the original version of PSO is faster in initial convergence while slower in fine tuning (Angeline 1998a, 1998b).

### *2.2. Binary PSO algorithms*

The original PSO is designed for the real-value problems. The algorithms now have been extended to tackle binary/discrete problems. To extend the real-value version of PSO to binary/discrete space, the most critical part is to understand the meaning of concepts such as trajectory,

velocity in the binary/discrete space.

Kennedy and Eberhart (Kennedy and Eberhart 1997) use velocity as a probability to determine whether $x_{id}$ (a bit) will be in one state or another (zero or one). They squashed vid using a logistic function $s(v) = 1/(1+\exp(-v))$ while the velocity is calculated using the same equation as the Equation (1a). If a randomly generated number within [0,1] is less than $s(v_{id})$, then $x_{id}$ is set to be 1, otherwise it is set to be 0. The version of binary PSO outperforms several versions of GAs in all tested problems except one (Kennedy and Spears 1998).

Agrafiotis and Cedeño (Agrafiotis and Cedeño 2002) adapted the real value PSO to the binary space and applied it to the problem of feature selection in which $x_{ij}$ can only take 0 or 1 and represents whether the $j$th feature in the $i$th particle is selected. The real value calculated by using Equation (1b) is treated as probabilities with which the roulette wheel is utilized to determine whether the new corresponding feature is selected or not in the next generation.

Mohan and Al-kazemi (Mohan and Al-kazemi 2001) proposed five binary variations of particle swarm optimization algorithms. The techniques they utilized for these five binary PSO are named as direct approach, quantum approach, regularization approach, bias vector approach, and mixed search approach, respectively. The direct approach binary PSO is a direct translation of the global version of PSO from continuous space to binary space. The positions of particles are the candidate solution and the Equation (1) is adopted. The results in the continuous space are converted to binary bit strings using some repair approach such as hard-decision decoding process. The quantum approach binary PSO is similar to the direct approach PSO except a different repair approach by which the resulted value is converted to either 0 or 1 with probability determined by the resulted value itself. The regularization approach is designed from a perspective that unifies PSO with other evolutionary algorithms by abstractly expressing PSO as a instance of Regularization (ref. 6 in Mohan and Al-kazemi 2001). In the bias vector approach, the particle in the new generation is randomly selected from the three parts in the right side of Equation (1a) with probabilities depending on the fitness values of the corresponding positions. The mixed search approach is similar to the direct approach except that the particles are divided into multiple groups in which each group can adopt local version or global version of binary PSO dynamically.

## 3. Topologies

The commonly used PSOs are either global version or local version of PSO. In the global version of PSO, each particle flies through the search space with a velocity that is dynamically adjusted according to the particle's personal best performance achieved so far and the best performance achieved so far by all the particle. While in the local version of PSO, each particle's velocity is adjusted according to its personal best and the best performance achieved as far within its neighborhood. The neighborhood of each particle is generally defined as topologically nearest particles to the particle at each side. The global version of PSO also can be considered as a local version of PSO with each particle's neighborhood to be the whole population. It has been suggested that the global version of PSO converges fast, but with potential to converge to the local minimum, while the local version of PSO might have more chances to find better solutions slowly (Kennedy 1999, Kennedy, Eberhart and Shi 2001). Since then, a lot of researchers have worked on improving its performance by designing or implementing different types of neighborhood structures in PSOs.

Kennedy (Kennedy 1999) claimed that PSO with small neighborhoods might perform better on complex problems while PSO with large neighborhood would perform better for simple problems.

Kennedy and Mendes (Kennedy and Mendes 2002) tested PSOs with regular shaped neighborhoods, such as global version, local version, pyramid structure, star structure, "small" structure, and von Neumann, and PSOs with randomly generated neighborhoods. The population size of their PSOs is fixed to be 20. They observed that among PSOs with randomly generally neighborhoods, those PSOs with average neighborhood size 5 have better performance based on their measurements. They further recommended that the PSO with von Neumann structured neighborhood may perform better than PSOs with other regular shaped neighborhoods including global version and local version.

Suganthan (Suganthan 1999) applied a combined version of PSO where a local version PSO is run first followed by a global version of PSO at the end of PSO running. Furthermore in the local version of PSO, the neighborhood of each particle is dynamically adjusted. The distances among particles are calculated and are then used as reference parameters to form new neighborhoods against a predefined criterion.

In (Hu and Eberhart 2002a), Hu and Eberhart introduced a dynamic neighborhood concept for their multi-objective optimization problems using PSO. The neighborhood of each particle is dynamically adjusted. The m closest particles in the performance space are selected to be its new neighborhood in each generation. The performance space is the space each coordinate of which is the variable representing the performance value of each objective function of the multi-objective optimization problem.

In (Mendes and Kennedy 2004), Mendes and Kennedy proposed a fully informed particle swarm optimization algorithm based on $\varphi$ coefficient analysis and their belief that there is no assumption that the best neighbor actually found a better region than the second or third-best neighbors. In this new algorithm , all the neighbors of a particle is involved in calculating the next movement instead of using the previous best positions in the original particle swarm optimization algorithm. The influence of each particle to its neighbors is weighted based on its fitness value and the neighborhood size.

Each neighborhood structure has its strength and weakness. It works better in one kind of problems, but worse on the other kind of problems. When using PSO to solve problem, not only the problem needs to be specified, but the neighborhood structure of the PSO utilized should also be clearly specified.

## 4. Parameters

Velocity changes of a PSO consist of three parts, the "social" part, the "cognitive" part, and the momentum part. The balance among these parts determines the balance of the global and local search ability, therefore the performance of a PSO.

The first new parameter added into the original PSO algorithm is the inertia weight (Shi and Eberhart 1998a, 1998b). The dynamic equation of PSO with inertia weight is modified to be:

$$v_{id} = wv_{id} + c_1 rand()(p_{id}-x_{id}) + c_2 Rand()(p_{gd}-x_{id}) \quad (2a)$$

$$x_{id} = x_{id} + v_{id} \quad (2b)$$

Equation (2) is the same as the Equation (1) except a new parameter, inertia weight $w$. The inertia weight is introduced to balance between the global and local search abilities. The large inertia weight facilitates global search while the small inertia weight facilitates local search. The introduction of the inertia weight also eliminates the requirement of carefully setting the maximum velocity $V_{max}$ each time the PSO algorithm is used. The $V_{max}$ can be simply set

to the value of the dynamic range of each variable and the PSO algorithm still performs well enough if not better.

Another parameter called constriction coefficient is introduced with the hope that it can insure a PSO to converge (Clerc 1999, Clerc and Kennedy 2002). A simplified method of incorporating it appears in Equation (3), where $k$ is a function of $c_1$ and $c_2$ as seen in Equation (4).

$$v_{id} = k[v_{id} + c_1 rand()(p_{id}-x_{id}) + c_2 Rand()(p_{gd}-x_{id})] \quad (3a)$$

$$x_{id} = x_{id} + v_{id} \quad (3b)$$

with

$$k = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \quad (4)$$

where $\varphi = c_1 + c_2$, $\varphi > 4$

Mathematically, Equation (2) and (3) are equivalent by setting inertia weight $w$ to be $k$, and $c_1$ and $c_2$ meet the condition $\varphi = c_1 + c_2$, $\varphi > 4$. The PSO algorithm with the constriction factor can be considered as a special case of the PSO algorithm with inertia weight while the three parameters are connected through Equation (4). A better approach to use as a rule of thumb is to utilize the PSO with constriction factor while limiting $V_{max}$ to $X_{max}$, the dynamic range of each variable on each dimension, or utilize the PSO with inertia weight while selecting $w$, $c_1$ and $c_2$ according to Equation (4) (Eberhart and Shi 2000).

When Clerc's constriction method is used, $\varphi$ is commonly set to 4.1 and the constant multiplier $k$ is approximately 0.729. This is equivalent to the PSO with inertia weight when $w \approx 0.729$ and $c_1=c_2=1.49445$. Since the search process of a PSO algorithm is nonlinear and complicated. A PSO with well-selected parameter set can have good performance, but much better performance could be obtained if a dynamically changing parameter is well designed. Intuitively, the PSO should favor global search ability at the beginning of PSO running while it should favor local search ability at the end of PSO running.

Shi and Eberhart (Shi and Eberhart 1998a, 1999) first introduced a linearly decreasing inertia weight to the PSO over the course of PSO, then they further designed fuzzy systems to nonlinearly changing the inertia weight (Shi and Eberhart 2001a, 2001b). The fuzzy systems have some measurements of the PSO performance as the input and the new inertia weight as the output of the fuzzy systems. In more recent study, an inertia weight with a random component [0.5 + (*rand*()/2.0)] rather than time-decreasing is utilized. This produces a randomly varying number between 0.5 and 1.0, with a mean of 0.75 which is similar to Clerc's constriction factor described above (Eberhart and Shi 2001b).

In (Ratnaweera, Halgamuge and Watson 2004), Ratnaweera *et al*. introduced into the PSO the time varying acceleration coefficients in addition to the time varying inertia weight. Furthermore, a PSO called "Self-Organizing Hierarchical Particle Swarm Optimizer" is proposed, in which only the "social" part and the "cognitive" part are kept in the algorithm as in (He *et al*. 1998) while the momentum part is only used for reinitializing particles when the particles have stagnated in the search space, that is inertia weight is set to be 0 except at the time of re-initialization. In (Fan and Shi 2001), a linearly decreasing $V_{max}$ is introduced as mentioned before.

### 5. Hybrid PSO Algorithms

Another research trend is to merge or combine the PSO with the other techniques, especially the other evolutionary computation techniques. Evolutionary operators like selection, crossover and mutation have been applied into the PSO. By applying selection operation in PSO, the particles with the best performance are copied into the next generation, therefore, PSO can always keep the best performed particles (Angeline 1998). By applying crossover operation, information can be swapped between two individuals to have the ability to "fly" to the new search area as that in evolutionary programming and genetic algorithms (Løvbjerg, Rasmussen and Krink 2001). Among the three evolutionary operators, the mutation operators are the most commonly applied evolutionary operators in PSO. The purpose of applying mutation to PSO is to increase the diversity of the population and the ability to have the PSO to escape the local minima (Miranda and Fonseca 2002, Løvbjerg and Krink 2002, Blackwell and Bentley 2002, Krink, Vesterstrøm and Riget 2002, Ratnaweera, Halgamuge and Watson 2004). One approach is to mutate parameters such as $\chi$, $c_1$ and $c_2$, the position of the neighborhood best (Miranda and Fonseca 2002), as well as the inertia weight (Løvbjerg and Krink 2002). Another approach is to prevent particles from moving too close to each other so that the diversity could be maintained and therefore escape from being trapped into local minima. In (Løvbjerg and Krink 2002), the particles are relocated when they are too close to each other. In (Blackwell and Bentley 2002, Krink, Vesterstrøm and Riget 2002), collision-avoiding mechanisms are designed to prevent particle from colliding with each other and therefore increase the diversity of the population.

In addition to incorporating evolutionary operations into PSO, different approaches to combine PSO with the other evolutionary algorithms have been reported. Robinson *et al*. (Robinson, Sinton and Rahmat-Samii 2002) obtained better results by applying PSO first followed by applying GA in their profiled corrugated horn antenna optimization problem. In (Krink and Løvbjerg 2002), either particle swarm optimization algorithm, genetic algorithm, or hill-climbing search algorithm can be applied to a different sub-population of individuals which each individual is dynamically assigned to according to some predesigned rules. In (Hendtlass and Randall 2001), ant colony optimization is combined with PSO. A list of best positions found so far is recorded and the neighborhood best is randomly selected from the list instead of the current neighborhood best. In (Hendtlass 2001), differential evolution is combined with PSO. Particles fly according to Equation (2), but occasionally differential evolution is applied to replace one poorly performed particle with a better one while retaining its velocity.

Also, non-evolutionary techniques have been incorporated into PSO. In (van den Bergh and Engelbrecht 2004), a Cooperative Particle Swarm Optimizer (CPSO) is implemented. The CPSO employs cooperative behavior to significantly improve the performance of the original PSO algorithm through using multiple swarms to optimize different components of the solution vector cooperatively. The search space is partitioned by splitting the solutions vectors into smaller vector. For example, a swarm with n-dimensional vector is partitioned into n swarms of one-dimensional vectors with each swarm attempting to optimize a single component of the solution vector. A credit assignment mechanism needs to be designed to evaluate each particle in each swarm. In (Løvbjerg, Rasmussen and Krink 2001), the population of particles is divided into subpopulations which would breed within their own sub-population or with a member of another with some probability so that the diversity of the population can be increased. In (Parsopoulos and Vrahatis 2004), deflection and stretching techniques as well as a repulsion technique

are incorporated into the original particle swarm optimization to avoid particles moving toward the already found global minima so that the PSO can have more chances to find as many global minima as possible. In (Xie and Zhang and Yang 2002), a "dissipative particle swarm" is designed by adding negative entropy into the PSO to discourage premature convergence.

The above hybrid PSO algorithms incorporate other techniques into PSO to improve the PSO's performance. On the other hand, the concept of PSO could also be "borrowed" and applied into other evolutionary algorithms to improve their performance. Wei *et al*. (Wei, He, Zhang and Pei 2002) applied PSO's velocity concept into the evolutionary programming to guide its mutation operations in order to have a fast evolutionary programming algorithm.

In general, there is a trend that the distinction among evolutionary algorithms is becoming more and more blur. More and more hybrid algorithms are being designed and implemented in the hope to have better performance. Certainly, physical meanings beyond the original algorithms are becoming vague too.

## 6. Applications

PSO is simple in concept. It has few parameters to adjust and is easy to implement. It has found applications in a lot of areas. In general, all the application areas that the other evolutionary application techniques are good at are the good application areas for PSO.

### *6.1. Constrained optimization problems*

Constrained optimization is one of the most common application areas for PSO. One of the major issues for solving constrained optimization problems is how to handle the constraints. A straight forward approach is to convert the constrained optimization problem into a non-constrained optimization problem by adding penalty for violation of constraints (Parsopoulos and Vrahatis 2002a). Another approach is to preserve feasible solutions and repair the infeasible solutions (Hu and Eberhart 2002b). The third approach, called hybrid algorithms, usually employs some information decoding strategy. For example, in (Ray and Liew2001), the constraints are handled by a constraint matrix. The constraint matrix is used to generated better performer list (BPL) which is used to set the search direction of the rest of the particles.

### *6.2. Min-max problems*

PSO has been also applied to solve min-max problems or the problems which can be converted to min-max problems (Shi and Krohling 2002, Krohling, Knidel and Shi 2002, Laskari, Parsopoulos and Vrahatis 2002a). One straight forward approach is to treat the min-max problem as a minimization problem in the hope that the obtained solutions can meet the requirements of the min-max problem by embedding the maximum part in the calculation of the fitness values (Laskari, Parsopoulos and Vrahatis 2002). Another approach is to use multi-PSO strategy. The min-max problem is first converted into two optimization problems: one is a maximum problem; the other is a minimum problem. Two PSO are used to solve these two optimization problems, respectively, and are run independently. Each PSO is treated as a changing environment of the other PSO. Therefore, the two PSO cooperate through the fitness calculation (Shi and Krohling 2002).

### *6.3. Multiobjective optimization problems*

In multi-objective optimization problems, multiple objectives need to be optimized simultaneously. In most cases, no single optimal solution usually can be found to be optimal to all the objective functions. Instead, there exist a group of alternative solutions, known as a Pareto optimal set or Pareto front. The solutions in this group are equivalent in the absence of any preference among all objectives.

Multi-objective optimization problem (MOO) has been one of the most studied application areas of PSO algorithms. Number of approaches have been utilized and/or designed to tackle MOO problems using PSO. A straight forward approach is to convert MOO to a single objective optimization problem. One simple implementation of the conversion is the so-called weighted aggregation approach which sums all the objectives to form a weighted combination. The weights can be fixed and dynamic changing during the optimization process (Parsopoulos and Vrahatis 2002b).

The second approach to tackle MOO problem by PSO is to record a set of better performing particles and then move towards particles randomly selected from the set instead of the neighborhood best in the original PSO to maintain a diversity of population and therefore maintain a well distribution along the Pareto front (Ray and Liew 2002, Coello Coello and Lechuga 2002, Coello Coello, Pulido and Lechuga 2004). In (Ray and Liew 2002), Ray and Liew combine Pareto dominance with PSO. Better performing particles are recorded into a set of leaders (SOL) based on non-dominated rank for unconstrained MOO problems and multi-level sieve implementation for constrained MOO problems. The remaining particles move towards a leader randomly selected from the SOL. Leaders with fewer individuals around them have a higher probability of being selected. In (Coello Coello and Lechuga 2002, Coello Coello, Pulido and Lechuga 2004), the authors also incorporate Pareto dominance into PSO. The algorithm stores the non-dominated vectors found so far in a second population of particles from which the neighborhood best will be later randomly selected by the primary population of particles to update their velocities. An adaptive grid is also introduced to generate well-distributed Pareto fronts, and special mutation operators are designed to mutate both the particles and their dynamic ranges to enhance the exploratory capabilities of the proposed PSO (Coello Coello, Pulido and Lechuga 2004).

Another approach proposed by Hu and Eberhart (Hu and Eberhart 2002a) optimizes one objective at a time. They introduce dynamic neighborhood strategy and particle memory updating into PSO. In each generation, according to a particle's distance to the other particles, it determines its new neighbors from which it selects its new neighborhood best. The approach is then further improved by adding a secondary population, called extended memory, to store global Pareto optimal solutions to reduce computation time (Hu, Eberhart and Shi 2003).

Li (Li 2003) proposed an approach, called Non-dominated Sorting Particle Swarm Optimizer (NSPSO), for MOO problems. It adopts a more effective non-domination comparison by comparing all particles' personal bests and their offspring in the entire population instead of a single comparison between a particle's personal best and its offspring.

### *6.4. Dynamic tracking*

The dynamic tracking problems are difficult problems for all evolutionary algorithms, including PSO, to handle since the landscapes of the optimization functions are changing over the time, therefore the currently found good solutions may not be good any more even in the near future (Shi and Eberhart 2001a, Hu and Eberhart 2001). A straight forward approach to handle dynamic tracking problems is to directly apply PSO without any change based on the belief that PSOs have the ability to converge

fast (Parsoupolis and Vrahatis 2001). The second approach is to re-evaluate and reset the previous best when a change in the environment was found (Carlisle and Dozier 2000). This is suitable for problems with slow-changing environment. The third approach is to implement some mechanism to detect and respond to dynamic functions, and re-randomize particles when change is detected (Hu and Eberhart 2002c).

### *6.5 Other applications*

In addition to the above four main application areas, PSO has been successfully applied to solve many other problems including a lot of practical application problems. It has been applied to evolve weights and structure of neural networks (Eberhart and Shi 1998a, Kennedy, Eberhart and Shi 2001), analyze human tremor (Eberhart and Hu 1999), register 3D-to-3D biomedical image (Wachowiak et al., 2004), play games (Messerschmidt and Engelbrecht 2004), control reactive power and voltage (Yoshida et al., 2000), etc.. Generally speaking, PSO can be applied to solve most optimization problems and problems that can be converted to optimization problems.

## 7. Summaries

This paper surveys the research and development of PSO in five categories: algorithms, topology, parameters, hybrid PSO algorithms, and applications. There are certainly other research works on PSO which are not included in this paper due to the space limitation. In general, the search process of a PSO algorithm should be a process consisted of both contraction and expansion so that it could have the ability to escape from local minima, and eventually find good enough solutions. A mathematical foundation of PSO is in need to have a deep understanding of the dynamic process of PSO. There is also a need of an unique representation of the PSO topology and a need of standard set of benchmark functions so that researchers can duplicate each other's work and compare their work with the others'.

## References:


Agrafiotis, D. K., and Cedeño, W.. (2002). Feature selection for structure-activity correlation using binary particle swarms. Journal of Medicinal Chemistry, 2002, 45, 1098-1107.

Angeline, P. J.. (1998a). Using selection to improve particle swarm optimization. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1998), Anchorage, Alaska, USA. 1998.

Angeline, P. J.. (1998b). Evolutionary optimization versus particle swarm optimization: philosophy and performance differences. Evolutionary Programming VII: Proceedings of the Seventh Annual Conference on Evolutionary Programming, 1998.

Blackwell, T., and Bentley, P. J.. (2002). Don't push me! collision-avoiding swarms. IEEE Congress on Evolutionary Computation, 2002 Honolulu, Hawaii USA.

Carlisle, A., and Dozier, G.. (2000). Adapting particle swarm optimization to dynamic environments. Proceedings of International Conference on Artificial Intelligence, 2000 pp. 429-434. Las Vegas, Nevada, USA.

Clerc, M.. (1999). The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. Proc. 1999 Congress on Evolutionary Computation, Washington, DC, pp 1951-1957. Piscataway, NJ: IEEE Service Center.

Clerc, M., and Kennedy, J.. (2002). The particle swarm: explosion, stability, and convergence in a multi-dimensional complex space. IEEE Transactions on Evolutionary Computation, vol. 6, p. 58-73.

Coello Coello, C. A., and Lechuga, M. S.. (2002). MOPSO: A proposal for multiple objective particle swarm optimization. IEEE Congress on Evolutionary Computation, 2002 Honolulu, Hawaii USA.

Coello Coello, C. A., Pulido, G. T., and Lechuga, M. S.. (2004). Handling multiple objectives with particle swarm optimization. IEEE Transactions on Evolutionary Computation (accepted for special issue on PSO).

Eberhart, R. C., and Kennedy, J.. (1995). A new optimizer using particle swarm theory. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 39-43. Piscataway, NJ: IEEE Service Center.

Eberhart, R. C., and Shi, Y.. (1998 a). Evolving artificial neural networks. Proc. 1998 Int'l. Conf. on Neural Networks and Brain, Beijing, P.R.C., PL5-PL13.

Eberhart, R. C. and Shi, Y.. (1998 b). Comparison between genetic algorithms and particle swarm optimization. In V. W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben, Eds. Evolutionary Programming VII: Proc. 7th Ann. Conf. on Evolutionary Programming Conf., San Diego, CA. Berlin: Springer-Verlag.

Eberhart, R. C., and Hu, X.. (1999). Human tremor analysis using particle swarm optimization. Proc. Congress on Evolutionary Computation 1999, Washington, DC, pp 1927-1930. Piscataway, NJ: IEEE Service Center.

Eberhart, R. C., and Shi, Y.. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. Proc. CEC 2000, San Diego, CA, pp 84-88.

Eberhart, R. C., and Shi, Y.. (2001 a). Tracking and optimizing dynamic systems with particle swarms. Proc. Congress on Evolutionary Computation 2001, Seoul, Korea. Piscataway, NJ: IEEE Service Center.

Eberhart, R. C., and Shi, Y.. (2001 b). Particle swarm optimization: developments, applications and resources. Proc. Congress on Evolutionary Computation 2001, Seoul, Korea. Piscataway, NJ: IEEE Service Center.

Eberhart, R. C., Simpson, P. K., and Dobbins, R. W.. (1996). Computational Intelligence PC Tools. Boston, MA: Academic Press Professional.

Fan, H., and Shi, Y.. (2001). Study on Vmax of particle swarm optimization. Proceedings of the Workshop on Particle Swarm Optimization. Indianapolis, IN: Purdue School of Engineering and Technology, IUPUI . April, 2001.

He, Z.,Wei, C., Yang, L., Gao, X., Yao, S., Eberhart, R., and Shi, Y.. (1998). Extracting Rules from Fuzzy Neural Network by Particle Swarm Optimization, IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, USA.

Hendtlass, T and Randall, M.. (2001). A survey of ant colony and particle swarm meta-heuristics and their application to discrete optimization problems. Proceedings of The Inaugural Workshop on Artificial Life (AL'01), pp. 15-25.

Hu, X., and Eberhart, R. C.. (2001). Tracking dynamic systems with PSO: where's the cheese? Proceedings of the workshop on particle swarm optimization Purdue school of engineering and technology, Indianapolis, IN.

Hu, X., and Eberhart, R. C.. (2002a). Multi-objective optimization using dynamic neighborhood particle swarm optimization, Proceeding of the 2002 Congress on Evolutionary Computation, Honolulu, Hawaii, May 12-17, 2002.

Hu, X., and Eberhart, R. C.. (2002 b). Solving constrained nonlinear optimization problems with particle swarm optimization. 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002) Orlando, USA, 2002.

Hu, X., and Eberhart, R. C.. (2002 c). Adaptive particle swarm optimization: detection and response to dynamic systems. IEEE Congress on Evolutionary Computation, 2002 Honolulu, Hawaii USA.

Hu, X., and Eberhart, R. C., and Shi, Y.. (2003). Particle swarm with extended memory for multiobjective optimization. Proc. of 2003 IEEE Swarm Intelligence Symposium, pages 193-197. Indianapolis, Indiana, USA, April 2003. IEEE Service Center.

Kennedy, J., and Eberhart, R. C.. (1995). Particle swarm optimization, Proc. of IEEE International Conference on Neural Networks (ICNN), Vol.IV, pp.1942-1948, Perth, Australia, 1995.

Kennedy, J.. (1999). Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance, Proceeding of the 1999 Conference on Evolutionary Computation, 1931-1938.

Kennedy, J., and Eberhart, R. C.. (1997). A discrete binary version of the particle swarm algorithm. Proc. 1997 Conf. on Systems, Man, and Cybernetics, 4104-4109. Piscataway, NJ: IEEE Service Center.

Kennedy, J., Eberhart, R. C., with Shi Y.. (2001). Swarm Intelligence, Morgan Kaufmann, 2001

Kennedy, J., and Mendes, R.. (2002).

Population structure and particle swarm performance, Proceeding of the 2002 Congress on Evolutionary Computation, Honolulu, Hawaii, May 12-17, 2002.

Kennedy, J., and Spears, W. M.. (1998). Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. Proc. Intl. Conf. on Evolutionary Computation, 78-83. Piscataway, NJ: IEEE Service Center.

Krink, T., and Løvbjerg, M.. (2002) The LifeCycle model: combining particle swarm optimisation, genetic algorithms and hill-climbers. Proceedings of Parallel Problem Solving from Nature (PPSN), Granada, Spain, September, 2002.

Krink, T., Vesterstrøm, J., S. and Riget, J.. (2002). Particle swarm optimization with spatial particle extension. Proceedings of the Fourth Congress on Evolutionary Computation (CEC-2002).

Krohling, R. A., Knidel, H., and Shi, Y. . (2002). Solving numerical equations of hydraulic problems using particle swarm optimization. IEEE Congress on Evolutionary Computation, 2002 Honolulu, Hawaii USA.

Laskari, E.C., Parsopoulos, K.E., Vrahatis, M.N.. (2002). Particle swarm optimization for min-max problems, Proceedings of the IEEE 2002 Congress on Evolutionary Computation.

Li, X.. (2003). A non-dominated sorting particle swarm optimizer for multi-objective optimization. Lecture Notes in Computer Science (LNCS) No. 2723: Proceedings of the Genetic and Evolutionary Computation Conference 2003 (GECCO 2003), Chicago, IL, USA. pp. 37-48, 2003

Løvbjerg, M., and Krink, T.. (2002). Extending particle swarms with self-organized criticality. Proceedings of the Fourth Congress on Evolutionary Computation (CEC-2002).

Løvbjerg, M., Rasmussen, T., and Krink, T.. (2001). Hybrid particle swarm optimiser with breeding and subpopulations. Proceedings of the third Genetic and Evolutionary Computation Conference (GECCO-2001), vol. 1, p. 469-476

Mendes, R., Kennedy, J., and Neves, J.. (2004). The fully informed particle swarm: simpler, maybe better. IEEE Transactions on Evolutionary Computation (accepted for special issue on PSO).

Messerschmidt, L., Engelbrecht, A. P.. (2004). Learning to play games using a PSO-based competitive learning approach. IEEE Transactions on Evolutionary Computation (accepted for special issue on PSO).

Miranda, V., and Fonseca, N.. (2002). New evolutionary particle swarm algorithm (epso) applied to voltage/VAR control. The 14th Power Systems Computation Conference (PSCC'02), Seville, Spain, June, 2002.

Mohan, C. K., and Al-kazemi, B.. (2001) Discrete particle swarm optimization. Proceedings of the Workshop on Particle Swarm Optimization 2001, Indianapolis, IN. 2001

Parsopoulos, K.E., Vrahatis, M.N.. (2001). Particle swarm optimizer in noisy and continuously changing environments, M.H. Hamza (ed.), Artificial Intelligence and Soft Computing, pp. 289-294, IASTED/ACTA Press (Anaheim, CA, USA).

Parsopoulos, K.E., Vrahatis, M.N.. (2002a). Particle swarm optimization method for constrained optimization problems. Proceedings of the Euro-International Symposium on Computational Intelligence (E-ISCI 2002).

Parsopoulos, K.E., Vrahatis, M.N.. (2002b). Particle swarm optimization method in multiobjective problems. Proceedings of the 2002 ACM Symposium on Applied Computing (SAC 2002), pp. 603-607.

Parsopoulos, K. E., and Vrahatis, M.. (2004). On the computation of all global minimizers through particle swarm optimization. IEEE Transactions on Evolutionary Computation (accepted for special issue on PSO).

Ratnaweera, A., Halgamuge, S. K., and Watson, H. C.. (2004). Self-organizing hierarchical particle swarm optimizer with time varying accelerating Coefficients. IEEE Transactions on Evolutionary Computation (accepted for special issue on PSO).

Ray, T., and Liew, K. M.. (2001). A swarm with an effective information sharing mechanism for unconstrained and constrained single objective optimization problem. Proc. congress on evolutionary computation 2001 pp. 75-80. IEEE service center, Piscataway, NJ., Seoul, Korea., 2001.

Ray, T., and Liew, K. M.. (2002). A swarm metaphor for multi-objective design optimization. Engineering Optimization. 34(2): 141-153, March 2002.

Robinson, J., Sinton, S., and Rahmat-Samii, Y.. (2002). Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna. IEEE International Symposium on Antennas & Propagation. San Antonio, Texas. June, 2002

Shi, Y., and Eberhart, R. C.. (1998a). Parameter selection in particle swarm optimization, Proceedings of the 1998 Annual Conference on Evolutionary Computation, March 1998

Shi, Y., and Eberhart, R. C.. (1998b). A modified particle swarm optimizer. Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, 69-73. Piscataway, NJ: IEEE Press. May 1998.

Shi, Y., and Eberhart, R. C.. (1999). Empirical study of particle swarm optimization. Proceedings of the 1999 Congress on Evolutionary Computation, 1945-1950. Piscataway, NJ: IEEE Service Center.

Shi, Y., and Eberhart, R. C.. (2000). Experimental study of particle swarm optimization. Proc. SCI'2000 Conference, Orlando, FL.

Shi, Y., and Eberhart, R. C.. (2001a). Fuzzy adaptive particle swarm optimization, Proc. Congress on Evolutionary Computation 2001, Seoul, Korea. Piscataway, NJ: IEEE Service Center.

Shi, Y., and Eberhart, R. C.. (2001b). Particle swarm optimization with fuzzy adaptive inertia weight. Proceedings of the Workshop on Particle Swarm Optimization 2001, Indianapolis, IN. 2001

Shi, Y., and Krohling, R. A.. (2002). Co-evolutionary particle swarm optimization to solve min-max problems. IEEE Congress on Evolutionary Computation, Honolulu, Hawaii USA. 2002.

Suganthan, P.N.. (1999). Particle swarm optimizer with neighborhood operator. Proceeding of the 1999 Conference on Evolutionary Computation. 1958-1962. Piscataway, NJ: IEEE Service Center.

van den Bergh, F., Engelbrecht, A. P.. (2004). A cooperative approach to particle swarm optimization. IEEE Transactions on Evolutionary Computation (accepted for special issue on PSO).

Wachowiak, M. P., Smolíková, R., Zheng, Y., Zurada, J. M., and Elmaghraby, A. S.. (2004). An approach to multimodal biomedical image registration utilizing particle swarm optimization. IEEE Transactions on Evolutionary Computation (accepted for special issue on PSO).

Wei, C., He, Z., Zhang, Y., and Pei, W.. (2002). Swarm directions embedded in fast evolutionary programming. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002), Honolulu, Hawaii USA. 2002

Xie, X., Zhang, W., and Yang, Z.. (2002). A dissipative particle swarm optimization. IEEE Congress on Evolutionary Computation, 2002 Honolulu, Hawaii USA

Yoshida, H., Kawata, K., Fukuyama, Y., Takayama, S., and Nakanishi, Y.. (2000). A particle swarm optimization for reactive power and voltage control considering voltage security assessment. IEEE Transactions on Power Systems, Vol. 15, No. 4, pp. 1232-1239.

**Yuhui Shi** received his Ph.D degree in digital signal processing from Southeast University, China in 1992. His expertise is in the areas of computational intelligence, biomedical engineering, embedded systems, and mobile multimedia systems. Dr. Shi served as the General Chair of the 2003 IEEE Swarm Intelligence Symposium, the Technical Chair of the 2001 Particle Swarm Optimization Workshop, and a member of the program committees of numerous conferences. He is the Proceedings Chair of the 2004 Congress on Evolutionary Computation (CEC2004). He was invited to give tutorial on the introduction to computational intelligence at 1998 World Congress on Computational Intelligence, and tutorial on evolutionary computation and fuzzy systems at 1998 Conference of Artificial Neural Networks in Engineering. Dr. Shi is the co-chair of the Task Force on Swarm Intelligence, Evolutionary Computation Technical Committee, IEEE Neural Networks Society, and an Associate Editor of IEEE Transactions on Evolutionary Computation. He co-authored a book on swarm intelligence together with Dr. James Kennedy and Dr. Russell C. Eberhart, and a book on computational intelligence, which is to be published by Morgan Kaufmann Publishers, together with Dr. Russell C. Eberhart. He has published more than 40 technical papers in the areas of his interests.

Particle Swarm Optimization (PSO) is a well established algorithm and is often cited in the literature and reported to have been applied to solve efficiently numerous problems which arise in real life. From: Introduction to Nature-Inspired Optimization, 2017. Related terms The pyswarm package is a gradient-free, evolutionary optimization package for python that supports constraints. Table of ContentsÂ¶. OverviewÂ¶. The package currently includes a single function for performing PSO: pso. It is both Python2 and Python3 compatible. RequirementsÂ¶. NumPy.Â  ReferencesÂ¶. Particle swarm optimization on Wikipedia. Navigation. index. Â© Copyright 2013â€“2014, Abraham Lee. Created using Sphinx 1.2.2.