Encyclopedia of Machine Learning pp 760–766

Particle Swarm Optimization

James Kennedy

The Canonical Particle Swarm

The particle swarm is a population-based stochastic algorithm for optimization which is based on social–psychological principles. Unlike evolutionary algorithms, the particle swarm does not use selection; typically, all population members survive from the beginning of a trial until the end. Their interactions result in iterative improvement of the quality of problem solutions over time.

A numerical vector of D dimensions, usually randomly initialized in a search space, is conceptualized as a point in a high-dimensional Cartesian coordinate system. Because it moves around the space testing new parameter values, the point is well described as a particle. Because a number of them (usually $10 < N < 100$) perform this behavior simultaneously, and because they tend to cluster together in optimal regions of the search space, they are referred to as a *particle swarm*.

Besides moving in a (usually) Euclidean problem space, particles are typically enmeshed in a topological network that defines their communication pattern. Each particle is assigned a number of neighbors to which it is linked bidirectionally.

The most common type of implementation defines the particles' behaviors in two formulas. The first adjusts the velocity or step size of the particle, and the second moves the particle by adding the velocity to its previous position.

On each dimension *d*:

$$v_{id}^{(t+1)} \leftarrow \alpha v_{id}^{(t)} + U(0,\beta)\left(p_{id} - x_{id}^{(t)}\right) + U(0,\beta)\left(p_{gd} - x_{id}^{(t)}\right)$$

(1)

$$x_{id}^{(t+1)} \leftarrow x_{id}^{(t)} + v_{id}^{(t+1)}$$

(2)

where *i* is the target particle's index, *d* is the dimension, $\vec{x}_i$ is the particle's position, $\vec{v}_i$ is the velocity, $\vec{p}_i$ is the best position found so far by *i*, *g* is the index of *i*'s best neighbor, α and *β* are constants, and *U*(0, *β*) is a uniform random number generator.

Though there is variety in the implementations of the particle swarm, the most standard version uses α = 0. 7298 and *β* = *ψ* ⁄ 2, where *ψ* = 2. 9922, following an analysis published in Clerc and Kennedy (2002). The constant α is called an *inertia weight* or *constriction coefficient*, and *β* is known as the *acceleration constant*.

The program evaluates the parameter vector of particle *i* in a function $f(\vec{x})$ and compares the result to the best result attained by *i* thus far, called $pbest_i$. If the current result is *i*'s best so far, the vector $\vec{p}_i$ is updated with the current position $\vec{x}_i$, and the previous best function result $pbest_i$ is updated with the current result.

When the system is run, each particle cycles around a region centered on the centroid of the previous bests $\vec{p}_i$ and $\vec{p}_g$; as these variables are updated, the particle's trajectory shifts to new regions of the search space, the particles begin to cluster around optima, and improved function results are obtained.

The Social–Psychological Metaphor

Classical social psychology theorists considered the pursuit of *cognitive consistency* to be an important motivation for human behavior (Heider, 1958; Festinger, 1957; Abelson et al., 1968). Cognitive elements might have emotional or logical aspects to them which could be consistent or inconsistent with one another; several theorists identified frameworks for describing the degree of consistency and described the kinds of processes that an individual might use to increase consistency or balance, or decrease inconsistency or cognitive dissonance.

Contemporary social and cognitive psychologists frequently cast these same concepts in terms of connectionist principles. Cognitive elements are conceptualized as a network with positive and negative vertices among a set of nodes. In some models, the elements are given and the task is to reduce error by adjusting the signs and values of the connections between them, and in other models the connections are given and the goal of optimization is to find activation values that maximize coherence (Thagard, 2000), harmony (Smolensky, 1986), or some other measure of consistency. Typically, this optimization is performed by gradient-descent programs which psychologically model processes that are private to the individual and are perfectly rational, that is, the individual always decreases error or increases consistency among elements. The particle swarm simulates the optimization of these kinds of structures through social interaction; it is commonly observed, not only in the laboratory but in everyday life, that a person faced with a problem typically solves it by talking with other people.

A direct precursor of the particle swarm is seen in Nowak, Szamrej, and Latané's (1990) cellular automaton simulation of social impact theory's predictions about interaction in human social populations. Social impact theory predicted that an individual was influenced to hold an attitude or belief in proportion to the Strength, Immediacy, and Number of sources of influence holding that position, where Strength was a measure of the persuasiveness or prestige of an individual, Immediacy was their proximity, and Number was literally the number of sources of influence holding a particular attitude or belief. In the simulation, individuals iteratively interacted, taking on the prevalent state of a binary attitude in their neighborhood, until the system reached equilibrium.

The particle swarm extends this model by supposing that various states can be evaluated, for instance, that different patterns of cognitive elements may be more or less dissonant; it assumes that individuals hold more than one attitude or belief, and that they are not necessarily binary; and Strength is replaced with a measure of self-presented success. One feature usually found in particle swarms and not in the paper by Nowak et al. is the phenomenon of persistence or momentum, the tendency of an individual to keep changing or moving in the same direction from one time-step to the next.

Thus, the particle swarm metaphorically represents the interactions of a number of individuals, none knowing what the goal is, each knowing its immediate state and its best performance in the past, each presenting its neighbors with its best success-so-far at solving a problem, each functioning as both source and target of influence in the dynamically evolving system. As individuals emulate the successes of their neighbors, the population begins to cluster in optimal regions of a search space, reliably discovering good solutions to difficult problems featuring, for instance, nonlinearity, high dimension, deceptive gradients, local optima, etc.

The Population Topology

Several kinds of topologies have been most widely used in particle swarm research; the topic is a current focus of much research. In the *gbest* topology, the population is conceptually fully connected; every particle is linked to every other. In practice, with the best neighbor canonical version, this is simpler to implement than it sounds, as it only means that every particle receives influence from the best performing member of the population.

The *lbest* topology of degree K_i comprises a ring lattice, with the particle linked to its K_i nearest neighbors on both sides in the wrapped population array.

Another structure commonly used in particle swarm research is the von Neumann or “square” topology. In this arrangement, the population is laid out in rows and columns, and each individual is connected to the neighbors above, below, and on each side of it in the toroidally wrapped population. Numerous other topologies have been used, including random (Suganthan, 1999), hierarchical (Janson & Middendorf, 2005), and adaptive ones (Clerc, 2006).

The most important effect of the population topology is to control the spread of proposed problem solutions through the population. As a particle finds a good region of the search space, it may become the best neighbor to one of the particles it is connected to. That particle then will tend to explore in the vicinity of the first particle’s success, and may eventually find a good solution there, too; it could then become the best neighbor to one of its other neighbors. In this way, information about good regions of the search space migrates through the population.

When connections are parallel, e.g., when the mean degree of particles is relatively high, then information can spread quickly through the population. On unimodal problems this may be acceptable, but where there are local optima there may be a tendency for the population to converge too soon on a suboptimal solution. The *gbest* topology has repeatedly been shown to be vulnerable to the lure of locally optimal attractors.

On the other hand, where the topology is sparse, as in the *lbest* model, problem solutions spread slowly, and subpopulations may search diverse regions of the search space in parallel. This increases the probability that the population will end up near the global optimum. It also means that convergence will be slower.

Vmax and Convergence

The particle swarm has evolved very much since it was first reported by Kennedy and Eberhart (1995) and Eberhart and Kennedy (1995). Early versions required a system constant *Vmax* to limit the velocity. Without this limit, the particles' trajectories would swing wildly out of control.

Following presentation of graphical representations of a deterministic form of the particle swarm by Kennedy (1998), early analyses by Ozcan and Mohan (1999) led to some understanding of the nature of the particle's trajectory. Analytical breakthroughs by Clerc (reported in Clerc and kennedy (2002)), and empirical discoveries by Shi and Eberhart (1998), resulted in the application of the α constant in concert with appropriate values of the acceleration constant β. These parameters brought the particle under control, allowed convergence under appropriate conditions, and made *Vmax* unnecessary. It is still used sometimes, set to very liberal values such as a half or third of the initialization range of a variable for more efficient swarm behavior, but it is not necessary.

Step Size and Consensus

Step size in the particle swarm is inherently scaled to consensus among the particles. A particle goes in one direction on each dimension until the sign of its velocity is reversed by the accumulation of $(p - x)$ differences; then it turns around and goes the other way. As it searches back and forth, its oscillation on each dimension is centered on the mean of the previous bests $(p_{id} + p_{gd})/2$, and the standard deviation of the distribution of points that are tested is scaled to the difference between them. In fact this function is a very simple one: the standard deviation of a particle's search, when p_{id} and p_{gd} are constants, is approximately $|(p_{id} - p_{gd})|$. This means that when the particles' previous best points are far from one another in the search space, the particles will take big steps, and when they are nearer the particles will take little steps.

Over time, this usually means that exploring behavior is seen in early iterations and exploiting behavior later on as particles come to a state of consensus. If it happens, however, that a particle that has begun to converge in one part of the search space receives information about a good region somewhere else, it can return to the exploratory mode of behaving.

The Fully Informed Particle Swarm (FIPS)

Mendes (2004) reported a version of swarm that featured an alternative to the best neighbor strategy. While the canonical particle is influenced by its own previous success and the previous success of its best neighbor, the fully informed particle swarm (FIPS) allowed influence by all of a particle's neighbors. The acceleration constants were set to $\beta = \psi / 2$ in the traditional version; it was defined in this way because what mattered was their sum, which could be distributed among any number of difference terms. In the standard algorithm there were two of them, and thus the sum was divided by 2. In FIPS a particle of K_i degree has coefficients $\beta = \psi / K_i$.

The FIPS particle swarm removed two aspects that were considered standard features of the algorithm. First of all, the particle i no longer influenced itself directly, e.g., there is no $\vec{p}_i$ in the formula. Second, the best neighbor is now averaged in with the others; it was not necessary to compare the successes of all neighbors to find the best one.

Mendes found that the FIPS swarm was more sensitive than the canonical versions to the differences in topology. For instance, while in the standard versions the fully connected *gbest* topology meant influence by the best solution known to the entire population, in FIPS *gbest* meant that the particle was influenced by a stochastic average of the best solutions found by all members of the population; the result tended to be near-random search.

The lesson to be learned is that the *meaning* of a topology depends on the mode of interaction. Topological structure (and Mendes tested more than 1,340 of them) affects performance, but the way it affects the swarm's performance depends on how information is propagated from one particle to another.

Generalizing the Notation

Equation 2 above shows that the position is derived from the previous iteration's position plus the current iteration's velocity. By rearranging the terms, it can be shown that the current iteration's velocity $\vec{v}_i^{(t+1)}$ is the difference between the new position and the previous one: $\vec{v}_i^{(t+1)} = \vec{x}_i^{(t+1)} - \vec{x}_i^{(t)}$. Since this happened on the previous time-step as well, it can be shown that $\vec{v}_i^{(t)} = \vec{x}_i^{(t)} - \vec{x}_i^{(t-1)}$; this fact makes it possible to combine the two formulas into one:

$$\begin{aligned} x_{id}^{(t+1)} \quad &\leftarrow x_{id}^{(t)} + \alpha \left(x_{id}^{(t)} - x_{id}^{(t-1)} \right) \\ &+ \sum U\left(0, \frac{\psi}{K_i}\right) \left(p_{kd} - x_{id}^{(t)} \right) \end{aligned}$$

(3)

where K_i is the degree of node *i*, *k* is the index of *i*'s *k*th neighbor, and adapting Clerc's (Clerc & Kennedy, 2002) scheme α = 0. 7298 and ψ = 2. 9922.

In the canonical best neighbor particle swarm, $K_i = 2, \quad \forall\ i : i = 1, 2, \ldots, N$ and $k \in (i, g)$, that is, *k* takes the values of the particle's own index and its best neighbor's index. In FIPS, K_i may vary, depending on the topology, and *k* takes on the indexes of each of *i*'s neighbors. Thus, Eq.3 is a generalized formula for the trajectories of the particles in the particle swarm.

This notation can be interpreted verbally as:

$$\begin{aligned} \textit{NEW POSITION} \quad &= \textit{CURRENT POSITION} \\ &+ \textit{PERSISTENCE} \\ &+ \textit{SOCIAL INFLUENCE} \end{aligned}$$

(4)

That is, on every iteration, every particle on every dimension starts at the point it last arrived at, persists some weighted amount in the direction it was previously going, then makes some adjustments based on the differences between the best previous positions of its sources of influence and its own current position in the search space.

The Evolving Paradigm

The particle swarm paradigm is young, and investigators are still devising new ways to understand, explain, and improve the method. A divergence or bifurcation of approaches is observed: some researchers seek ways to simplify the algorithm (Peña, Upegui, & Eduardo Sanchez, 2006; Owen & Harvey, 2007), to find its essence, while others improve performance by adding features to it, e.g., (Clerc, 2006). The result is a rich unfolding research tradition with innovations appearing on many fronts.

Although the entire algorithm is summarized in one simple formula, it is difficult to understand how it operates or why it works. For instance, while the *Social Influence* terms point the particle in the direction of the mean of the influencers' successes, the *Persistence* term offsets that movement, causing the particle to bypass what seems to be a reasonable target. The result is a spiral-like trajectory that goes past the target and returns to pass it again, with the spiral tightening as the neighbors come to consensus on the location of the optimum.

Further, while authors often talk about the particle's velocity carrying it "toward the previous bests," in fact the velocity counterintuitively carries it *away from* the previous bests as often as toward them. It is more accurate to say the particle "explores around" the previous bests, and it is hard to describe this against-the-grain movement as "gradient descent," as some writers would like.

It is very difficult to visualize the effect of ever-changing sources of influence on a particle. A different neighbor may be best from one iteration to the next; the balance of the random numbers may favor one or another or some compromise of sources; the best neighbor could remain the same one, but may have found a better $\vec{p}_i$ since the last turn; and so on. The result is that the particle is pulled and pushed around in a complex way, with many details changing over time.

The paradoxical finding is that it is best not to give the particle information that is too good, especially early in the search trial. Premature convergence is the result of amplified consensus resulting from too much communication or overreliance on best neighbors, especially the population best. Various researchers have proposed ways to slow the convergence or clustering of particles in the search space, such as occasional reinitialization or randomization of particles, repelling forces among them, etc., and these techniques typically have the desired effect. In many cases, however, implicit methods work as well and more parsimoniously; the effect of topology on convergence rate has been mentioned here, for instance.

Binary Particle Swarms

A binary particle swarm is easily created by treating the velocity as a probability threshold (Kennedy & Eberhart, 1997). Velocity vector elements are squashed in a sigmoid or other function, for instance $S(\upsilon) = 1/(1 + exp(-\upsilon))$, producing a result in (0..1). A random number is generated and compared to $S(\upsilon_{id})$ to determine whether x_{id} will be a 0 or a 1. Though discrete systems of higher cardinality have been proposed, it is difficult to define such concepts as distance and direction in a meaningful way within nominal data.

Alternative Probability Distributions

As was noted above, the particle's search is centered around the mean of the previous bests that influence it, and its variance is scaled to the differences among them. This has suggested to several researchers that perhaps the trajectory formula can be replaced, wholly or partly, by some type of random number generator that directly samples the search space in a desirable way.

Kennedy (2003) suggested simple Gaussian sampling, using a random number generator (RNG) $G(mean, s.d.)$ with the mean centered between $\vec{p}_i$ and $\vec{p}_g$, and with the standard deviation defined on each dimension as $s.d. = |(p_{id} - p_{gd})|$. This "bare bones" particle swarm eliminated the velocity component; it performed rather well on a set of test functions, but not as well as the usual version.

Krohling (2004) simply substituted the absolute values of Gaussian-distributed random numbers for the uniformly distributed values in the canonical particle swarm. He and his colleagues have had success on a range of problems using this approach. Richer and Blackwell (2006) replaced the Gaussian distribution of bare bones with a Lévy distribution. The Lévy distribution is bell-shaped like the Gaussian but with fatter tails. It has a parameter α which allows interpolation between the Cauchy distribution (α = 1) and Gaussian (α = 2) and can be used to control the fatness of the tails. In a series of trials, Richer and Blackwell (2006) were able to emulate the performance of a canonical particle swarm using α = 1. 4. Kennedy (2005) used a Gaussian RNG for the social influence term of the usual formula, keeping the "persistence" term found in the standard particle swarm. Variations on this format produced results that were competitive with the canonical version.

Numerous other researchers have begun exploring ways to replicate the overall behavior of the particle swarm by replacing the traditional formulas with alternative probability distributions. Such experiments help theorists understand what is essential to the swarm's behavior and how it is able to improve its performance on a test function over time.

Simulation of the canonical trajectory behavior with RNGs is a topic that is receiving a great deal of attention at this time, and it is impossible to predict where the research is leading. As numerous versions have been published showing that the trajectory formulas can be replaced by alternative strategies for selecting a series of points to sample, it becomes apparent that the essence of the paradigm is not to be found in the details of the movements of the particles, but in the nature of their interactions over time, the structure of the social network in which they are embedded, and the function landscape with which they interact, with all these factors working together gives the population the ability to find problem solutions.

Recommended Reading

Abelson, R. P., Aronson, E., McGuire, W. J., Newcomb, T. M., Rosenberg, M. J., & Tannenbaum, R. H. (Eds.), (1968). *Theories of cognitive consistency: A sourcebook*. Chicago: Rand McNally.

Clerc, M. (2006). *Particle swarm optimization*. London: Hermes Science Publications.

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

Eberhart, R.C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the 6th international symposium on micro machine and human science, (Nagoya, Japan)* (pp. 39–43). Piscataway, NJ: IEEE Service Center.

Festinger, L. (1957). *A theory of cognitive dissonance*. Stanford, CA: Stanford University Press.

Heider, F. (1958). *The psychology of interpersonal relations*. New York: Wiley.

Janson, S., & Middendorf, M. (2005). A hierarchical particle swarm optimizer and its adaptive variant. *IEEE Transactions on Systems, Man, and Cybernatics – Part B: Cybernatics*, *35*(6), 1272–1282.

Kennedy, J. (1998). The behavior of particles. In V. W. Porto, N. Saravanan, D. Waagen, & A. E. Eiben (Eds.), *Evolutionary programming VII. Proceedings* *of the 7th annual conference on evolutionary programming*.

Kennedy, J. (2003). Bare bones particle swarms. In *Proceedings of the IEEE swarm intelligence symposium* (pp. 80–87). Indianapolis, IN.

Kennedy, J. (2005). Dynamic-probabilistic particle swarms. In *Proceedings of the genetic and evolutionary computation conference (GECCO-2005)* (pp. 201–207). Washington, DC.

Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In *Proceedings of the 1995 IEEE international conference on neural networks (Perth, Australia)* (pp. 1942–1948). Piscataway, NJ: IEEE Service Center.

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. In *Proceedings of the 1997 conference on systems, man, and cybernetics* (pp. 4104–4109). Piscataway, NJ: IEEE Service Center.

Krohling, R. A. (2004). Gaussian Swarm. A novel particle swarm optimization algorithm. *Proceedings of the 2004 IEEE conference on cybernetics and intelligent systems* (vol. 1, pp. 372–376).

Mendes, R. (2004). *Population topologies and their influence in particle swarm performance*. Doctoral thesis, Escola de Engenharia, Universidade do Minho, Portugal.

Nowak, A., Szamrej, J., & Latané, B. (1990). From private attitude to public opinion: A dynamic theory of social impact. *Psychological Review, 97*, 362–376.

Owen, A., & Harvey, I. (2007). Adapting particle swarm optimisation for fitness landscapes with neutrality. In *Proceedings of the 2007 IEEE swarm intelligence symposium* (pp. 258–265). Honolulu, HI: IEEE Press.

Ozcan, E., & Mohan, C. K. (1999). Particle swarm optimization: Surfing the waves. In *Proceedings of the congress on evolutionary computation, Mayflower hotel, Washington D.C.* (pp. 1939–1944). Piscataway, NJ: IEEE

Service Center.

Peña, J., Upegui, A., & Eduardo Sanchez, E. (2006). Particle swarm optimization with discrete recombination: An online optimizer for evolvable hardware. In *Proceedings of the 1st NASA/ESA conference on adaptive hardware and systems (AHS-2006), Istanbul, Turkey* (pp. 163–170). Piscataway, NJ: IEEE Service Center.

Richer, T. J., & Blackwell, T. M. (2006). The Levy particle swarm. In *Proceedings of the 2006 congress on evolutionary computation (CEC-2006)*. Piscataway, NJ: IEEE Service Center.

Shi, Y., & Eberhart, R. C. (1998). Parameter selection in particle swarm optimization. In *Evolutionary Programming VII: Proc. EP98* (pp. 591–600). New York: Springer.

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group, (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition. Vol. 1, Foundations* (pp. 194–281). Cambridge, MA: MIT Press.

Suganthan, P. N. (1999). Particle swarm optimisation with a neighbourhood operator. In *Proceedings of congress on evolutionary computation.* Washington DC, USA.

Thagard, P. (2000). *Coherence in thought and action*. Cambridge, MA: MIT Press.

Author information

Authors and Affiliations

U.S. Bureau of Labor Statistics, Washington, DC, USA

James Kennedy

Editor information

Editors and Affiliations

School of Computer Science and Engineering, University of New South Wales, Sydney, Australia, 2052

Claude Sammut

Faculty of Information Technology, Clayton School of Information Technology, Monash University, P.O. Box 63, Victoria, Australia, 3800

Geoffrey I. Webb

Rights and permissions

Copyright information

About this entry

Cite this entry

Kennedy, J. (2011). Particle Swarm Optimization. In: Sammut, C., Webb, G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30164-8_630

.RIS .ENW .BIB

DOI
https://doi.org/10.1007/978-0-387-30164-8_630

Publisher Name
Springer, Boston, MA

Print ISBN
978-0-387-30768-8

Online ISBN
978-0-387-30164-8

eBook Packages
Computer Science
Reference Module
Computer Science and Engineering

Share this entry