Bare Bones Particle Swarms

James Kennedy
US Bureau of Labor Statistics
Washington DC 20212
Kennedy_Jim@bls.gov

Abstract – The particle swarm algorithm is modified by eliminating the velocity formula. Variations are compared.

The particle swarm algorithm has just enough moving parts to make it hard to understand. The formula is very simple, it is even easy to describe the working of the algorithm verbally, yet it is very difficult to grasp in one's mind how the particles oscillate around centers that are constantly changing; how they influence one another; how the various parameters affect the trajectory of the particle; how the topology of the swarm affects its performance; and so on. This paper will strip away some traditional features of the particle swarm in the search for the properties that make it work. In the process some of the mysteries of the algorithm will be revealed, we will discover its similarity to other stochastic population-based problem solving methods, and new avenues of investigation will be suggested or implied.

I. INTRODUCTION TO THE ALGORITHM

Anyone reading this should be reasonably familiar with the particle swarm algorithm already, so this introduction will be brief.

The current canonical particle swarm algorithm loops through a pair of formulas, one for assigning the velocity and another for changing the particle's position:

$$\vec{v}_i = \chi(\vec{v}_i + U(0,\frac{\varphi}{2})(\vec{p}_i - \vec{x}_i) + U(0,\frac{\varphi}{2})(\vec{p}_g - \vec{x}_i))$$
$$\vec{x}_i = \vec{x}_i + \vec{v}_i$$

where $\vec{v}_i$ is the velocity vector for individual i, ? is a constriction coefficient that attenuates the velocity, $\vec{U}(0,\frac{\varphi}{2})$ is a function that generates uniformly distributed random numbers between zero and $\frac{\varphi}{2}$, $\vec{p}_i$ is the best position found so far by individual i, $\vec{x}_i$ is i's current position in the search space, and $\vec{p}_g$ is the best position found by any member of i's topological neighborhood. Following Clerc's analysis, φ is usually set at 4.1, so that $\frac{\varphi}{2}$ equals 2.05, and ? is calculated to equal approximately 0.729 (Clerc and Kennedy, 2002). Iteratively, the current position is evaluated for each population member, adjustments are made to the velocity, and the position for the next time-step is identified.

Kennedy and Mendes (2002) noted that a von Neumann or "square" topology seemed to perform better than the standard ones on a set of test functions; thus the square was used in all the experiments reported here. The population of N=20 particles is laid out as a 4×5 grid, and each individual is connected to the particles above, below, and to the left and right of itself. Rows and columns are wrapped; the population is a kind of torus. This topology will be referred to below as a form of *lbest*, as each particle communicates with its local topological neighbors only; this should not be confused with *lbest* neighborhoods defined by position on a ring lattice.

These tests were written in SAS, using macros for each algorithm version that could easily be plugged into the program. Each condition was run fifty times on each of five standard functions. Sphere, Griewank, Rosenbrock, and Rastrigin functions were run in 30 dimensions, and Shaffer's f6 is two-dimensional. All populations were initialized asymmetrically, as per Shi's (Shi and Eberhart, 2000) scheme, so that the global optimum was not in the middle of the initial population, but off to the side.

Data are presented in line graphs showing progression of the base-10 logarithm of the mean best over 3,000 iterations. Because the printed graphs may be hard to read, results at 3,000 iterations are given in a table at the end of the paper.

It has been noted (Kennedy, 1998; Clerc and Kennedy, 2002; Kennedy and Eberhart with Shi, 2001) that a particle's trajectory can be described as a cyclic path centered around a randomly-weighted mean, on each dimension, of the individual's and the best neighbor's previous best points. Two important aspects of the algorithm will be examined below; these are ways of defining the center of the oscillations, and ways of defining the magnitude of the variability around the center – the amplitude of the oscillation.

II. ALTERNATIVES TO THE BEST NEIGHBOR

The standard particle swarm uses the best information in its neighborhood. Typically, the topological neighbors are queried, the best one identified, and that best neighbor's best success is used, with the target individual's, to define the center of the particle's oscillations. A paradoxical result has been noted, however: when the neighborhood is expanded to include all members of the population, so all particles are influenced by the best success found by any member of the swarm (the so-

0-7803-7914-4/03/$10.00©2003 IEEE

called *gbest* topology), performance suffers on some multimodal functions. Using the best available information, in other words, can be detrimental. This is the same concern that underlies simulated annealing, and the use of roulette-wheel and tournament selection in evolutionary algorithms; all these methods allow poorer solutions to propagate.

It follows that it might not be necessary to identify the best neighbor, even in a localized sociometry. Thus a version was coded where g was randomly chosen from the neighborhood each time.

Further, it is possible to distribute the weight of φ across as many terms as we like, as long as the sum of the weights is less than 4.1 (Clerc and Kennedy, 2002). Thus a version was coded where all neighbors were considered simultaneously. The fully informed particle swarm (FIPS) has performed comparably to the canonical particle swarm on many tests (Kennedy, Mendes, and Neves, in review). The individual does not influence itself in this version.

Note that these manipulations affect the centering of the particle's oscillations, but not its amplitude. The center of the cycle was defined using the neighborhood best, population best, a random neighbor's previous best, or the center of all neighbors' bests. The amplitude of the cycle was determined, as usual, by the previous iteration's velocity and the distance from the center.

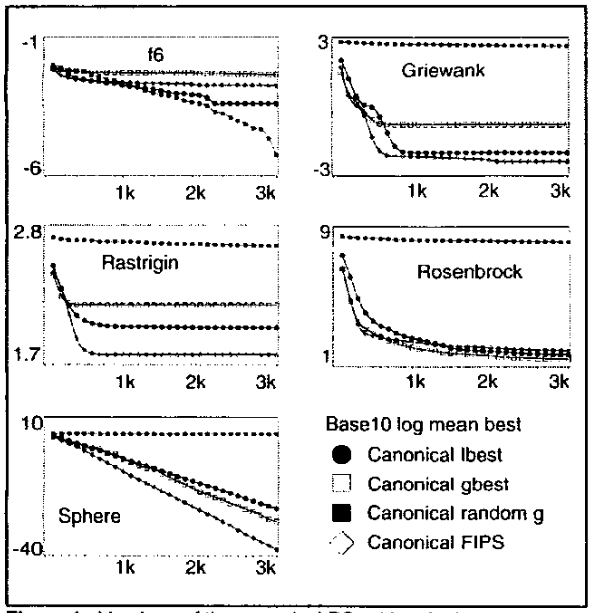

Figure 1. Versions of the canonical PS, with velocity.

As seen in Figure 1, the random-neighbor algorithm performed very well on the highly multimodal f6 function, and poorly on the rest. The performance of the FIPS algorithm was a little better than the others on Griewank, Rastrigin, and sphere, and the canonical particle swarm performed slightly better than the others on Rosenbrock.

The question that is addressed in this section has to do with the importance of centering the particle's trajectory around a really good center. *Gbest* is of course the very best center known, but as has been previously noted it tends to misdirect the population toward premature convergence on a local optimum, as seen in f6, Griewank, and Rastrigin results in Figure 1. On the other hand there were small differences when the center of the oscillation was defined by the mean of the individual and its best local neighbor, compared with the entire neighborhood excluding the individual. Both standard *lbest* square and FIPS square performed well in these tests. In sum, the center does matter; in some cases it may be helpful to use the "best best," while in other cases a more modest center should be chosen.

III. THE DISTRIBUTION OF SAMPLES

An iterative algorithm like the particle swarm or an evolutionary algorithm operates by sampling points on the problem space and using discovered knowledge to guide exploration. There can obviously be very many strategies for selecting the next point to test. Figure 2 shows a histogram of points that were tested in one million iterations of a standard particle swarm where p_i and p_g were held constant at ±10. The figure is a tidy bell curve centered midway between the two previous best points and extending symmetrically beyond them – this is the famous "overflying" effect that is known to be important in exploring new regions.

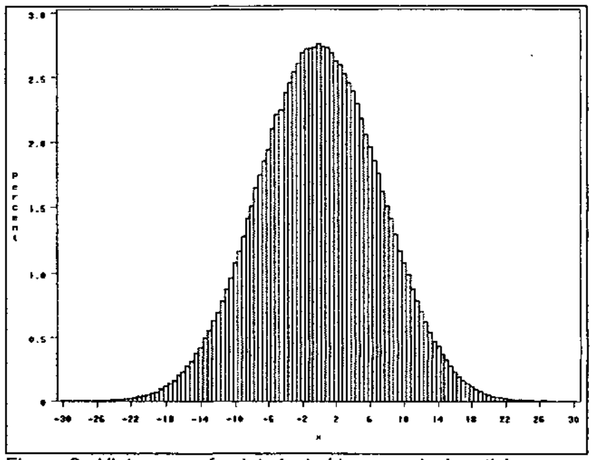

Figure 2. Histogram of points tested by canonical particle swarm with previous bests held constant, $p_i = -10$ and $p_g = +10$.

It is clear from Figure 2 that the difference between the individual and neighborhood previous best points is an important parameter for scaling the amplitude of particles' trajectories: *step size is a function of consensus.*

When neighbors' best points are all in the same region of the search space, smaller steps will be taken. This is because the stochastic mean, which wanders due to the random coefficients, always lies between the two bests, never outside their range.

Consensus in the particle swarm then serves the same function as the strategy parameters in evolution strategies (ES) and evolutionary programming (EP), but, it appears, more elegantly. Not only is it not necessary in the particle swarm to optimize another D-dimensional vector along with the problem parameters, but there is no lag between when an adaptation is needed and when it occurs. In order to evolve strategy parameters, some must be proposed and rejected over the course of generations, using selection; in the particle swarm the "strategy parameters" emerge from the nature of the search itself. Another approach, differential evolution (DE; Price, 1999), also capitalizes on the scaling effect of consensus, though search is implemented in a different way there.

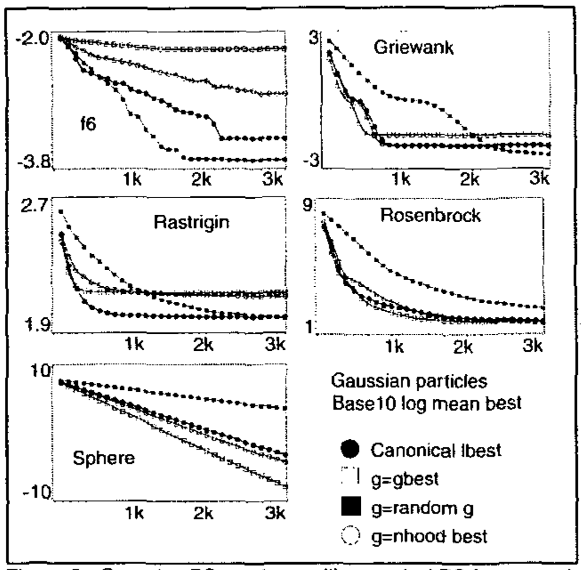

Figure 3. Gaussian PS versions, with canonical PS for comparison.

Figure 2 suggests that we should try dropping the whole velocity part of the particle swarm formula, and simply generate normally distributed random numbers around the mean $\frac{p_{id}+p_{gd}}{2}$ on each dimension d, using some consensus information from the neighborhood to scale the standard deviation of the distribution. Versions were coded where the center was the mean just mentioned, and the standard deviation of the gaussian random number distribution was $|p_{id}-p_{gd}|$, the difference on each dimension between the individual's previous best and the best neighbor's previous best position. The versions differed in how g was defined: *gbest*, neighborhood best, or random neighbor. In Figure 3, results from the canonical (square) *lbest* particle swarm are included for comparison.

The canonical version performed competitively but not outstandingly; it converged fastest on Rastrigin, and was otherwise not the best. This rather surprising result suggests that the entire velocity vector and all that it implies, including complicated stuff with inertia weights and constriction coefficients, and all the interesting things having to do with overlapping cycles (Kennedy, 1998), surfing the sine waves (Ozcan and Mohan, 1999), constriction and explosion (Clerc and Kennedy, 2002), etc., are simply unnecessary.

As seen in Figure 3, the *gbest*-gaussian version was the fastest to converge on the sphere function, and the random-neighbor gaussian version was the slowest. The random-neighbor version also performed excellently on most functions, especially f6, and caught up with the canonical version on the Rastrigin function near 3,000 iterations.

All in all, the gaussian algorithms, with the random numbers centered halfway between p_{id} and p_{gd} and the standard deviation equal to the absolute value of their difference, turned out to be very competitive optimizers for these five difficult testbed functions.

IV. NOTE ON SCALING THE STEP-SIZE

It seems preposterous that the standard deviation of the random numbers should be exactly the difference between i's and g's previous best points. Weighting the difference by coefficients above or below 1.0, however, did not seem to improve performance. Thus, in the absence of contrary evidence, we will suppose there is something magical about the simple difference that makes it a perfect standard deviation.

V. GAUSSIAN FIPS

The standard FIPS algorithm (with velocity) has been shown to perform about as well as the canonical particle swarm. Versions of the gaussian algorithm were coded using some FIPS features. The center of the normal distribution was the mean of the entire neighborhood, as in FIPS, but there was no clear way to determine the standard deviation. Several versions were tried.

In these versions, the standard deviation of the normal distribution was determined, as before, to be $|p_{id}-p_{gd}|$, where g is variously defined as the population best, the neighborhood best, a random neighbor, or the center of the FIPS neighborhood. As can be seen in Figure 4, the canonical particle swarm performed best by far on the Rastrigin function, and was mediocre otherwise. It is interesting to note the results on the sphere function, where the version using the difference between the individual's previous best and the population best (*gbest*) climbs the hill the fastest, followed by the neighborhood best, and then the random neighbor. The locations of these points were not used in calculations, only the difference between them and the target particle's previous success. Their distance from *i* was an important consideration in scaling the length of steps through the search space.

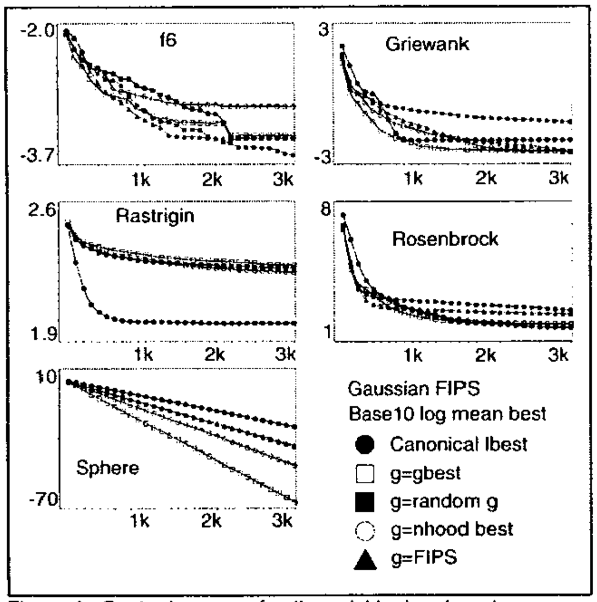

Figure 4. Center is mean of entire neighborhood, various measures of s.d.=distance from *i*.

VI. INTERACTION PROBABILITY

So far all the methods have changed every element of *i*'s problem solution vector. But since $\vec{p}_i$ is *i*'s best, and probably not a poor point at all, retaining some previous best vector elements might speed up convergence.

The idea here is that influence will be accepted on a vector element only if a probability test is passed. Given the void of knowledge about how to implement this, the experiments reported here set the interaction probability *IP*=0.50. Approximately half the time, vector element $x_{id} = p_{id}$ will be tested. The other half of the time, the vector element will be derived from a gaussian distribution with mean $\bar{x}_{id} = \frac{p_{id} - p_{gd}}{2}$ and standard deviation $s.d.(x_{id}) = \left|p_{id} - p_{gd}\right|$.

The strengths and weaknesses of the various algorithms can be seen in Figure 5. For instance, none of the gaussian versions with *IP*=0.50 performed especially well on f6. This function is notorious for its local optima, and this result suggests that the speed-up effect of *IP*<1.0 might introduce a vulnerability for local optima.

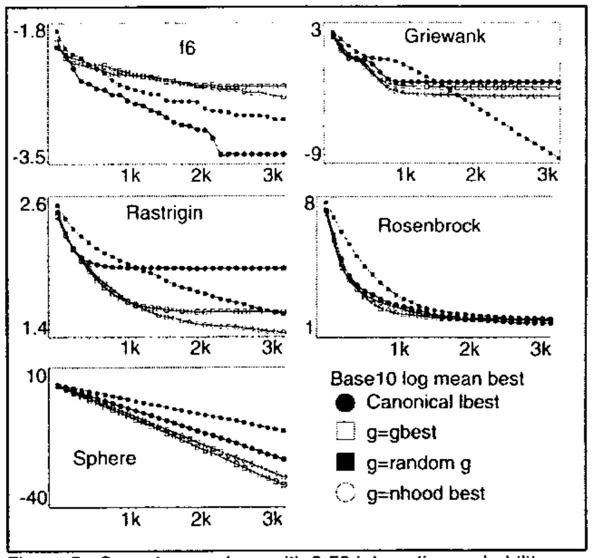

Figure 5. Gaussian versions with 0.50 interaction probability.

The performance of the random-neighbor version is interesting, as well. This version continued to improve throughout the course of the iterations, on every function (most notable on Griewank). Thus, it appears that the slowing-down effect of the random neighbor counteracts the speeding-up effect of *IP*<1.0; if you are not in a hurry, this might be the most reliable algorithm presented here.

(*A Note on Sphere*: The logarithm of performance over time on the sphere function tells much about an important characteristic of a search algorithm. It will be noted that in every graph in this paper, the sphere log plots are straight lines. What this means is that the algorithm is a well-scaled hill-climber. When it needs to take big steps, it takes big steps, and it takes little ones when the scale of the problem requires it, e.g., as it nears the minimum of the function. Each algorithm has a characteristic slope that describes its ability to climb smooth hills regardless of scale. It is obvious from the results presented here that hill-climbing ability is not a predictor of excellence on multimodal and otherwise complicated functions.)

VII. COMBINED RESULTS

It is possible to compare the various algorithms by combining the results from all function trials. First, function data are standardized; the data are linearly transformed so that each function has a mean of 0.00 and standard deviation of 1.00. Then for each algorithm variation, the mean standardized result can be calculated. This statistic indicates how well an algorithm performs relative to the others in this study. As all functions were minimized, a negative value is better. Table 1 shows the standardized results for each algorithm.

Table 1. Mean standardized function results after 3,000 iterations. Values indicate the distance from the population average, in standard deviation units (negative is better).

Canonical versions		Mean
	g=gbest	0.013571
	g=lbest	-0.31304
	g=random neighbor	2.612338
	g=FIPS	-0.31274
Gaussian versions		
	g=gbest	0.093788
	g=lbest	-0.18433
	g=random neighbor	-0.32638
FIPS gaussian		
	g=gbest	-0.13053
	g=lbest	-0.12825
	g=random neighbor	-0.15086
	g=FIPS	-0.14526
IP = 0.50		
	g=gbest	-0.2886
	g=lbest	-0.35497
	g=random neighbor	-0.38473

As can be seen in Table 1, the best-performing algorithm was the gaussian version with *IP*=0.50, where *g* was a randomly selected neighbor. The worst, by far, was the canonical algorithm (e.g., with velocity) where *g* was random. The standard *lbest* particle swarm was not bad, but not the best.

VIII. VARIATION, SELECTION, SELF-ORGANIZATION

In the experiments above I have treated the particle swarm's velocity formula as if it were an arbitrary kind of gaussian random number generator (RNG). But of course, though the terms of the velocity formula have a stochastic component, its result is not even quasirandom, as the points in the series are not independent. For instance, because of the perpetuation of the previous iteration's directional tendency, a particle tends to stay on the same side of its previous best for more consecutive times than would be expected if the pattern were generated by an RNG.

There may be an advantage to this, and it may be that even more effective strategies can be found for generating points within the particle swarm framework. As the algorithm iterates in some high-dimensional space, it will be important to test combinations of new ideas or patterns of variables, and their direction may be as important as the magnitude. That is, it may be important for a variable at time *t* to try a value greater than its previous best at the same time that another variable tries a value lower that its previous best, no matter whether the new values are very near the previous best or far from it. Thus one thing we look for in an algorithm is the ability to invent new patterns of variables to test, and it may just be that the particle swarm's velocity formula does a good job of that.

If the algorithm is to escape from local optima, then it will be important for it to generate, at least occasionally, values that fall far from the tree, and further, in keeping with the previous point, it may be important to generate a number of these simultaneously. Outliers should come often enough to keep the spirit of exploration alive, but rarely enough to allow investigation of good regions.

A. Variation

Gaussian random numbers are used in other techniques, notably some evolutionary algorithms, and the presentation given above certainly seems to connect the particle swarm method to those other approaches. It would seem that the particle swarm and the various EAs can be subsumed into one family of optimizers or searchers that use variation, selection, and self-organization to some degree.

One of the mysteries of human cognition is the process by which people are able to generate new ideas. Campbell (1960) suggested that we might use a darwinian-like process of blind variation and selective retention to create and mold new thoughts. But, plausible as the theory was, Campbell had no advice on how "blind variation" was supposed to work. It may be that creative ideas are generated by a random process analogous to mutation, perhaps an epileptiform neuronal misfiring that produces random thoughts – but there is no empirical evidence for any "random idea generator" in human psychology.

The gaussian distribution is observed widely in nature; a great number of traits are distributed normally. Thus it seems a defensible model of variation in the current programs. The process generating normality in nature, though, remains unexplained.

The gaussian distribution (or any bell-shaped approximation) has some characteristics that make it good for problem-solving. First, the central peak of the bell describes its conservative aspect: most of the points generated will be close to previous values. Second, the infinite tails describe the ability of the process to generate outliers. This is necessary for the discovery of new solutions.

A gaussian distribution is defined by two parameters, its mean and its standard deviation. The studies reported here have largely explored some different ways to set these two parameters, using information from the swarm's structure and search history.

The dispersion of the distribution is handled elegantly in the particle swarm approach. The clustering of individuals in a section of the search space is an indication that the region is good – they have all found their best success there, so it must be relatively good. Having found a good region, then, individuals should focus their search there. An extremely important feature of the particle swarm is the effect of consensus on step-size.

When neighborhood consensus is used to guide the step size, the effect is instantaneous. The individual looks to its neighborhood to see how much variance exists, and uses that variance to set the variance of its own investigations of the search space. There is no lag. Note that this happens whether the velocity formula is used, or one of the gaussian variations presented above; it is a core feature of the particle swarm algorithm.

B. Selection

I myself have said that there is no selection in particle swarms, that this is the feature that distinguishes this method from the evolutionary pack. But of course there is a kind of selection, in the replacement of the previous best $\vec{p}_i$ whenever a better solution is found. The difference is that the identity of the individual is maintained over time; this is change and not replacement. A fundamentalist darwinian might insist that there is competition between the individual and its offspring at every iteration, but it seems just as good to say that the individual improves over time. In an evolutionary system, it is not possible to say that the individual improves.

C. Self-organization

As a particle swarm (whether a canonical version with velocity or one of these newfangled ones) iterates over time, the population identifies promising areas of the problem space and searches them more intensively as a result of consensus among individuals. The higher-level view is that the population becomes organized into clusters, in the case of a multimodal space, or that it converges on an optimum. The lower-level view is that each individual moves toward its neighbors' successes.

The fixed topology of the swarm causes individuals to interact iteratively with particular neighbors, or, in the case of *gbest*, with neighbors that meet a certain criterion. Some (i.e., Clerc, 1999) have experimented with adaptive topologies, and this will be an important realm of particle swarm research, but the concept of a more-or-less permanent social topology seems fundamental to the paradigm.

Mendes and I have been researching the effects of various population topologies, and find that this higher-level structure can make the difference between a population that *never* finds the solution to any of the standard testbed problems, and one that finds the solution more than 98 per cent of the time. The studies reported here used one topological structure, but that is another moving part that makes understanding the particle swarm as a whole difficult.

IX. IN SUM

The gaussian versions presented above may seem to be something other than particle swarms. After all, we often define the paradigm in terms of particles "flying" through the problem space; these appear more to splatter than fly. Yet the fundamentals of particle swarm optimization are retained. The essence of the paradigm is not to be found in its flying trajectories, but in the influence that individuals have upon one another, in the persistence of individual identities over time, in the stability of sociometric interaction patterns.

The trajectories have been a focus of research and analysis, but it seems to me that Figure 2 says what needs to be said about them, that they generate a kind of bell-curve of test points centered between the individual's previous best point and some other point derived from social experience. It is entirely possible that other strategies for selecting points in the search space will have even better characteristics than these first investigations.

The standard particle swarm's velocity adjustment comes from its origin in bird-flocking. In models such as those of Reynolds (1987) and Heppner and Grenander (1990), birds (or "boids") adjust their velocity in response to neighbors' change of velocity, as well as some other environmental information. Early flocking simulations were modified so that the birds would be able to find food (Kennedy and Eberhart, 1995), which was originally a two-dimensional point on the pixel plane, and quickly became a set of neural network weights. In the early days of investigation, flocking-specific variables were torn out, for instance, topological neighbors were substituted for euclidean ones, collisions were allowed, etc. – but some things stayed, including the velocity adjustment. The current paper suggests that even this is an arbitrary piece of baggage from the past.

REFERENCES

Campbell, D.T. 1960. Blind variation and selective retention in creative thought as in other knowledge processes. *Psychological Review 67*, 380—400.

Clerc, M. (1999). The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. *Proceedings of the IEEE Congress on Evolutionary Computation* (CEC 1999), pp.1951-1957, *1999*

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

Heppner, F. and U. Grenander (1990). A stochastic nonlinear model for coordinated bird flocks. In S. Krasner, Ed., *The Ubiquity of Chaos*. AAAS Publications, Washington, DC.

Kennedy, J. (1998). The behavior of particles. *Proceedings of the 1998 Evolutionary Programming Conference.*

Kennedy, J., and Eberhart, R. C., with Shi, Y. (2001). *Swarm Intelligence*. San Francisco: Morgan Kaufmann/ Academic Press.

Kennedy, J., and Mendes, R. (2002). Population Structure and Particle Swarm Performance. *Proceedings of the 2002 World Congress on Computational Intelligence.*

Ozcan, E., and Mohan, C. (1999). Particle swarm optimization: surfing the waves. *Proc. 1999 Congress on Evolutionary Computation,* 1939-1944. Piscataway, NJ: IEEE Service Center.

Price, Kenneth V. (1999). An Introduction to Differential Evolution. In: David Corne, Marco Dorigo and Fred Glover (editors) (1999). *New Ideas in Optimization.* McGraw-Hill, London (UK), pp. 79–108.

Reynolds, C. W. (1987). Flocks, herds and schools: a distributed behavioral model. *Computer Graphics, 21*(4):25-34.

Kennedy, J., and Eberhart, R. C. (1995). Particle swarm optimization. *Proceedings of the 1995 IEEE International Conference on Neural Networks* (Perth, Australia), IEEE Service Center, Piscataway, NJ, IV: 1942-1948.

Shi, Y., and Eberhart, R., (2000). Experimental study of particle swarm optimization. *Proc. SCI2000 Conference, Orlando, FL.*

APPENDIX

Mean best function results after 3,000 iterations. Standard deviations in parentheses.

	Sphere	Griewank	Rosenbrock	f6	Rastrigin
Canonical versions					
g=gbest	1.01E-28	0.158421	20.68675	0.004469	146.7558
	(3.69E-28)	(0.404761)	(22.75011)	(0.004892)	(39.59975)
g=lbest	5.54E-24	0.008613	38.43955	0.000389	95.77445
	(1.11E-23)	(0.010564)	(30.67907)	(0.001923)	(26.15445)
g=random neighbor	61114.31	534.0548	1.8E+08	4.71E-06	446.6093
	(9016.474)	(88.52225)	(48305856)	(9.68E-06)	(31.30839)
FIPS	3.13E-39	0.003534	67.03906	0.001675	57.8667
	(8.25E-39)	(0.009502)	(57.48949)	(0.003485)	(16.66862)
Gaussian versions					
g=gbest	2.85E-36	0.02884	35.54519	0.006218	137.3038
	(7.48E-36)	(0.05784)	(29.42198)	(0.004711)	(39.827)
g=lbest	1.15E-26	0.009498	43.18685	0.001555	128.995
	(4.65E-26)	(0.009919)	(33.54539)	(0.003598)	(38.69897)
g=random neighbor	1.94E-06	0.003452	262.5542	0.000194	94.50759
	(2.65E-06)	(0.006785)	(259.7229)	(0.001374)	(29.58299)
Gaussian FIPS					
g=gbest	2.21E-68	0.002626	58.6977	0.00042	190.9091
	(5.00E-68)	(0.011845)	(29.60341)	(0.001929)	(19.11126)
g=lbest	1.63E-46	0.002595	59.25127	0.000972	175.7602
	(3.27E-46)	(0.011195)	(35.43223)	(0.002944)	(16.35434)
g=random neighbor	6.21E-36	0.052859	304.77	0.00024	185.4823
	(8.70E-36)	(0.079125)	(1192.061)	(0.000891)	(11.97994)
g=FIPS center	8.05E-35	0.002693	189.2309	0.000389	184.0702
	(1.54E-34)	(0.016166)	(682.6363)	(0.001923)	(12.96109)
IP=0.50 versions					
g=gbest	1.60E-33	0.002962	54.69925	0.00272	39.61918
	(3.55E-33)	(0.007018)	(40.05709)	(0.004407)	(15.20883)
g=lbest	1.46E-30	0.000458	60.26698	0.001997	26.11007
	(2.26E-30)	(0.001773)	(40.98154)	(0.003908)	(10.29736)
g=random neighbor	1.32E-13	1.66E-09	69.38136	0.001048	38.53877
	(1.30E-13)	(6.30E-09)	(42.70819)	(0.00296)	(11.18241)