The Fully Informed Particle Swarm: Simpler, Maybe Better

Rui Mendes, *Member, IEEE*, James Kennedy, and José Neves

***Abstract*—The canonical particle swarm algorithm is a new approach to optimization, drawing inspiration from group behavior and the establishment of social norms. It is gaining popularity, especially because of the speed of convergence and the fact that it is easy to use. However, we feel that each individual is not simply influenced by the best performer among his neighbors. We, thus, decided to make the individuals "fully informed." The results are very promising, as informed individuals seem to find better solutions in all the benchmark functions.**

***Index Terms*—Optimization, particle swarm optimization, social networks.**

I. Introduction

THE CANONICAL particle swarm algorithm works by searching iteratively in a region that is defined by each particle's best previous success, the best previous success of any of its neighbors, the particle's current position, and its previous velocity. The current paper proposes an alternative that is conceptually more concise and promises to perform more effectively than the traditional particle swarm algorithm. In this new version, the particle uses information from all its neighbors, rather than just the best one.

The standard algorithm is given in some form resembling the following:

$$\vec{v}_{t+1} = \alpha\vec{v}_t + \vec{\mathrm{U}}[0,\varphi_1] \otimes (\vec{P}_i - \vec{X}_t) + \vec{\mathrm{U}}[0,\varphi_2] \otimes (\vec{P}_g - \vec{X}_t) \quad (1)$$

$$\vec{X}_{t+1} = \vec{X}_t + \vec{v}_{t+1} \quad (2)$$

where $\otimes$ denotes point-wise vector multiplication, $\vec{\mathrm{U}}[\min,\max]$ is a function that returns a vector whose positions are randomly generated following the uniform distribution between $\min$ and $\max$, α is called the inertia weight and is less than 1, $\vec{v}_t$ and $\vec{X}_t$ represent the speed and position of the particle at time t, $\vec{P}_i$ refers to the best position found by the particle, and $\vec{P}_g$ refers to the position found by the member of its neighborhood that has had the best performance so far. The Type $1''$ constriction coefficient is often used [1]

$$\vec{v}_{t+1} = \chi\left(\vec{v}_t + \vec{\mathrm{U}}[0,\varphi_1] \otimes (\vec{P}_i - \vec{X}_t) + \vec{\mathrm{U}}[0,\varphi_2] \otimes (\vec{P}_g - \vec{X}_t)\right) \quad (3)$$

$$\vec{X}_{t+1} = \vec{X}_t + \vec{v}_{t+1}. \quad (4)$$

Manuscript received July 10, 2002; revised August 29, 2003. The work of R. Mendes was supported in part by PRAXIS XXI, ref. BD/3107/9 and POSI/ROBO/43904/2002.

R. Mendes and J. Neves are with the Departmento de Informática, Universidade do Minho, Braga 4710-057, Portugal (e-mail: azuki@di.uminho.pt; jneves@di.uminho.pt).

J. Kennedy is with the Bureau of Labor Statistics, Washington, DC 20212 USA (e-mail: Kennedy.Jim@bls.gov).

Digital Object Identifier 10.1109/TEVC.2004.826074

The two versions are equivalent, but are simply implemented differently. The second form is used in the present investigations. Other versions exist, but all are fairly close to the models given above.

A particle searches through its neighbors in order to identify the one with the best result so far, and uses information from that one source to bias its search in a promising direction. There is no assumption, however, that the best neighbor at time t actually found a better region than the second or third best neighbors. Important information about the search space may be neglected through overemphasis on the single best neighbor.

When constriction is implemented as in the second version above, lightening the right-hand side of the velocity formula, the constriction coefficient χ is calculated from the values of the acceleration coefficient limits φ_1 and φ_2, importantly, it is the sum of these two coefficients that determines what χ to use. This fact implies that the particle's velocity can be adjusted by any number of terms, as long as the acceleration coefficients sum to an appropriate value. For instance, the algorithm given above is often used with $\chi = 0.7298$ and $\varphi_1 = \varphi_2 = 2.05$. The φ coefficients must sum, for that value of χ to 4.1. Clerc's analysis was worked out using a condensed form of the formula

$$\vec{v}_{t+1} = \chi\left(\vec{v}_t + \varphi(\vec{P}_m - \vec{X}_t)\right) \quad (5)$$

$$\vec{X}_{t+1} = \vec{X}_t + \vec{v}_{t+1} \quad (6)$$

which was then expanded to partition the acceleration weight between the particle's own previous success and the neighborhood's, such that $\varphi = \varphi_1 + \varphi_2$. Note that in this deterministic model $\vec{P}_m$ is calculated as $\vec{P}_m = (\varphi_1 \cdot \vec{P}_i + \varphi_2 \cdot \vec{P}_g)/(\varphi_1 + \varphi_2)$.

II. Variation and Partitioning of φ

The search of particle i converges on a point $\vec{P}_m$ in the search space. Variation is introduced in several ways.

- First, obviously, the term is weighted by a random number. This in itself, however, would not prevent the velocity from approaching a zero limit. For instance, if the $\vec{P}_m - \vec{X}_t$ difference equals zero, the velocity will still converge to zero.
- Thus, another important source of variation is the difference between $\vec{P}_m$ and $\vec{X}_t$. As long as the position of the particle differs from the previous best position, then there will be movement. In a constricted algorithm, however, this difference tends toward zero over time as $\vec{P}_m$ is updated.
- Of course, it is hoped in practice that $\vec{P}_m$ does not remain fixed, and a key source of variation is the updating of

$\vec{P}_m$ over time as new points are found in the search space which are better than those previous ones. It is not necessary for $\vec{P}_m$ to be i's own previous best point, in order for i's trajectory to converge to it. For convergence, it is only necessary for $\vec{P}_m$ to remain fixed.

- In the traditional particle swarm, the very most important source of variation is the difference between i's own previous best and the neighborhood's previous best, that is, between $\vec{P}_i$ and $\vec{P}_g$. Random weighting of the two terms keeps the particle searching between and beyond a region defined by the two points. While some investigators have looked at schemes for differentially weighting the two terms (e.g., [2]), the limits for the two uniform distributions are usually the same. That is, the total weight of φ is partitioned into two equal components.

Clerc's method, however, does not require that the velocity adjustments be shared between two terms. It is only necessary that the parts sum to a value that is appropriate for the constriction weight χ. The algorithm will behave properly, at least as far as its convergence and explosion characteristics, whether all of φ is allocated to one term, or it is divided into thirds, fourths, etc.

We propose an alternate form of calculating $\vec{P}_m$

$$\vec{\varphi}_k = \vec{\mathrm{U}}\left[0, \frac{\varphi_{\max}}{|\mathcal{N}|}\right] \ \forall k \in \mathcal{N} \tag{7}$$

$$\vec{P}_m = \frac{\sum_{k\in\mathcal{N}} \mathcal{W}(k)\vec{\varphi}_k \otimes \vec{P}_k}{\sum_{k\in\mathcal{N}} \mathcal{W}(k)\vec{\varphi}_k} \tag{8}$$

where $\mathcal{N}$ is the set of neighbors of the particle and $\vec{P}_k$ is the best position found by individual k.

The function $\mathcal{W}$ may describe any aspect of the particle that is hypothesized to be relevant; in the experiments reported below, we use the fitness of the best position found by the particle, and the distance from that particle to the current individual, or have $\mathcal{W}$ return a constant value. Because all the neighbors contribute to the velocity adjustment, we say that the particle is *fully informed*.

III. Sociometry in the Fully Informed Particle Swarm

In the traditional particle swarm, a particle with k neighbors selects one to be a source of influence and ignores the others. In that situation, neighborhood size means how many other particles you can choose among, and the more there are, the better the one you pick is likely to be. In the fully informed neighborhood, however, all neighbors are a source of influence. Thus, neighborhood size determines how diverse your influences will be and in an optimization algorithm diverse influences might mean that search is diluted rather than enhanced.

The rest of the paper will describe experiments with various neighborhoods, where all the neighbors' previous best values are used to modify the velocity of the particle. These arrangements of the neighborhoods can be thought of as social networks.

It should be appreciated that the topological structure of the population controls its exploration versus exploitation tendencies [3], [4].

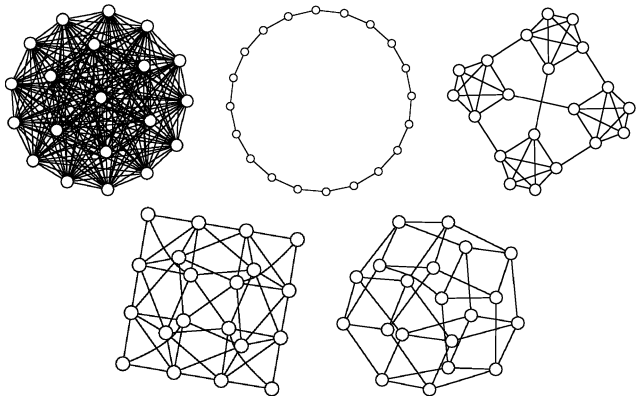

Fig. 1. Topologies used in the paper are presented in the following order: All, where all vertexes are connected to every other; Ring, where every vertex is connected to two others; Four clusters, with four cliques connected among themselves by gateways; Pyramid, a triangular wire-frame pyramid, and Square, which is a mesh where every vertex has four neighbors that wraps around on the edges as a torus.

The behavior of each particle is affected by its local neighborhood, which can be seen as a single region in the population topology. Thus, the topology affects search at a low level by defining neighborhoods. Particles that are acquainted to one another tend to explore the same region of the search space. It also affects search at a higher level, by defining the relationships between the local neighborhoods.

The current study tested five different social networks that had given good results in a previous study [3]. The networks are encoded in binary matrices for input into the program, and are depicted graphically in Fig. 1.

A social network can be characterized by a series of statistics that convey some information about its structure and the speed of communication flow. The most descriptive statistics are the graph's average distance, its diameter and the distribution sequence. The average distance measures the average number of edges between any two nodes. The diameter is the largest distance between two nodes in the graph. The distribution sequence is a descriptive statistic, of the form $\langle d_1, d_2, \ldots, d_n\rangle$ where d_i is the average number of nodes reachable from a vertex of the graph by traversing exactly i arcs, without cycles. Note that the first value of the distribution sequence, d_1, is the average degree of the graph.

Whenever a particle discovers a good region of the search space, it only directly influences its d_1 neighbors. Its d_2 second degree neighbors will only be influenced after those directly connected to it become highly successful themselves. Thus, there is a delay in the information spread through the graph. This delay can be characterized by the distribution sequence statistic. The average distance and the diameter of the graph are two simple statistics that represent the average and the maximum, respectively, number of cycles of influence needed to broadcast information throughout the graph.

By studying Table I, we can extract a number of conclusions about the topologies used. The *all* topology was the one used when the algorithm was developed and is still widely used by researchers. It represents a fully connected graph, and, based on all three statistics, we conjecture that information spreads

TABLE I
TOPOLOGIES USED IN THE STUDY AND THE ASSOCIATED GRAPH STATISTICS

Topology	Average Distance	Diameter	Distribution Sequence
All	1	1	⟨19⟩
Ring	5.26	10	⟨2, 2, 2, 2, 2, 2, 2, 2, 2, 1⟩
Four Clusters	2.26	3	⟨4.6, 4.8, 9.6⟩
Pyramid	2.04	4	⟨5.4, 7.8, 5.4, 0.4⟩
Square	2.32	4	⟨4, 7, 6, 2⟩

quickly. Sociologically, it could represent a small and closed community where decisions are taken in consensus.

The *ring* topology, the usual alternative to *all*, represents a regular graph with a minimum number of edges between its nodes. The graph statistics show that information travels slowly along the graph. This allows for different regions of the search space to be explored at the same time, as information of successful regions takes a long time to travel to the other side of the graph.

The *four clusters* topology represents four cliques connected among themselves by several gateways. Sociologically, it resembles four mostly isolated communities, where a few individuals have an acquaintance outside their group. This graph is characterized by the large number of individuals three hops away, despite the fact that its diameter is only 3.

The *pyramid* represents a three-dimensional wire-frame pyramid. It has the lowest average distance of all the graphs and the highest first and second degree neighbors. The *square* is a graph representing a rectangular lattice that folds like a torus. This structure, albeit artificial, is commonly used to represent neighborhoods in the Evolutionary Computation and Cellular Automata communities, and is referred to as the von Neumann neighborhood.

IV. DEPENDENT VARIABLES AND FREE LUNCH

The present experiments extracted three kinds of measures of performance on a standard suite of test functions. The functions were the sphere or parabolic function in 30 dimensions, Rastrigin's function in 30 dimensions, Griewank's function in 10 and 30 dimensions (the importance of the local minima is much higher in 10 dimensions, due to the product of co-sinuses, making it much harder to find the global minimum), Rosenbrock's function in 30 dimensions, and Schaffer's f6, which is in 2 dimensions. Formulas can be found in the literature, e.g., in [5].

It does not seem interesting to us to demonstrate that an algorithm is good on some functions and not on others. What we hope for is a problem-solver that can work well with a wide range of problems. This line of reasoning drives us head-on into the no free lunch (NFL) theorem [6], [7].

A. Free Lunch

NFL asserts that no algorithm can be better than any other, over all possible functions. This seems to be true because of two classes of functions: deceptive ones, and random ones. Deceptive functions lead a hill-climber away from the optimum, for instance there may be gradients that lead away from a discontinuous point that is the global optimum. Over all possible functions, it must be true that gradients lead away from the optimum at least as often as they lead the searcher toward it. The second class, random functions, contains very many more members than the first [8]. In fact, when all possible functions are considered, it seems certain—indeed it can be proven—that most of them are nonsense. Where gradients exist, they are unrelated to real solutions. On these very numerous function landscapes, a hill-climber will do no better than a hill-descender, no matter whether you are trying to maximize or minimize. It is like finding a needle in a haystack; no method of search can be any better than dumb luck. These two classes of functions explain why there is NFL.

But there is a third class of functions. These are functions where regularities on the fitness landscape do provide clues as to the location of a problem solution. Speaking of dumb luck, it is lucky for us that this third class contains most of the kinds of functions that we call *problems*. Problems are a special subclass of functions; they are special because somebody thinks there may be a solution to them, and wants to find it.

It is interesting to consider whether this third class of functions is actually more common in the world, perhaps because of correlations forced by physical laws, or whether they are merely more salient because of some idiosyncrasy of human attention. As we cannot count up instances of real function landscapes—like the set of "all possible functions" it is innumerable and meaningless—we will never be able to satisfy our curiosity regarding this question.

How do we know if a function has a solution or not? Of course, we have known since Turing that we cannot tell with certainty whether an algorithm will ever reach finality [9], that is in this case, whether a problem can be solved. But even though there is no certainty, there are clues. For instance, if it is believed that a cause and effect relationship exists among variables in the function, then we may expect to find some exploitable regularities in the fitness landscape. Even if the causal relationship is noisy, or if the relationship involves variables not mentioned in the function (e.g., the "third variable problem" in correlational research [10]), it is often possible to find useful features on the function landscape.

Another clue that a function might be solvable is when it is compressible. The easiest-to-spot form of this clue exists when the problem is given as a mathematical formula, rather than a lookup table. If the formula is shorter than the table of all possible input–output matches, then we have been given a hint that it might be useful to watch for regularities. The evidence of this is seen in the difficulty of the search for functions that produce random outputs [11]; it is not easy to produce an unpredictable series out of a mathematical formula, e.g., a good random number generator, even though random functions are known to comprise the larger share of the universe of all possible functions.

In some hard cases, we may have only hypotheses and intuitions to provide ideas for how to search for patterns that will reveal the problem solution. Sometimes we are wrong, and a problem is reassigned to one of the first two classes of functions.

We reiterate the important point that a function is only a problem if someone thinks it is a problem. That means that

a function, let us say Schaffer's f6, may exist as a curiosity without anyone ever trying to find its minimum. All of science can be viewed as a progression of things that have always existed suddenly becoming problems to be solved and explained. The argument presented here is the pragmatist's response to NFL. If somebody is trying to solve it, it is a problem; even if it does turn out to be deceptive or random, and they give up on it, and it stops being a problem, it is a problem during the time they are trying to solve it. It may remain a problem if something about it gives the researcher hope of solving it.

This is all a way of saying that the NFL theorem, eyeball-popping as it is, is not especially relevant to the task of problem-solving. NFL does not say that the search for a general problem-solver is futile; it does say that the search for a general function optimizer is futile. As researchers, it is our aim to minimize the amount of time we devote to searching for optima on deceptive and random function spaces.

Thus, in the current exercises we combined results from all the test functions, all of which are commonly used in experimentation with optimization algorithms, with the goal in mind of finding versions of the particle swarm algorithm that perform well on all of them. If we are successful in this, then we will naturally extend the range of problems until we have widened the applicability of the particle swarm to its broadest extent.

B. Performance

The first dependent variable is simply the best function result after some arbitrary number of iterations, here, we use 1,000. Basically, this is a measure of sloppy speed. It does not necessarily indicate whether the algorithm is close to the global optimum; a relatively high score can be obtained on some of these multimodal functions simply by finding the best part of a locally optimal region.

It is not possible to combine raw results from different functions, as they are all scaled differently. For instance, almost any decent algorithm will find a function result less than 0.01 on the sphere function, but a result of 40.0 on Rosenbrock is considered good. In order to combine the function outputs, we standardized the results of each function to a mean of 0.0 and standard deviation of 1.0. All results of all trials for a single function are standardized to the same scale; as all of these problems involve minimization, a lower result is better, and after standardization that means that a negative result is better than average. After standardizing each function separately, we can combine them and find the average for a single condition.

One comment about combining data from different functions: when a very good performance is combined with a very bad one, the result is a moderate average. On the other hand, a very good average can only be attained through combining very good scores. In this paper, we are interested in discovering very good performers and will neglect the confusion found in the middle.

C. Iterations to Criteria

The second dependent variable is the number of iterations required to reach a criterion. Function criteria are given in Table II. This is also a measure of speed, but in this case the criteria are intended to indicate that the searcher has arrived in the region of the global optimum.

TABLE II
PARAMETERS AND CRITERIA FOR THE TEST FUNCTIONS

Function	Dimensions	Symmetric Initialization	Asymmetric Initialization	Criterion
Sphere	30	±100	[50, 100]	0.01
Rastrigin	30	±5:12	[2.56, 5.12]	100
Griewank10	10	±600	[300, 600]	0.05
Griewank30	30	±600	[300, 600]	0.05
Rosenbrock	30	±30	[15, 30]	100
Schaffer f6	2	±100	[50, 100]	0.00001

There is, however, a problem with this measure, too. That is, some trials might never reach the criteria. Many hours have been lost waiting, trying to give each version a fair chance to find the global optimum, often in vain. Trials where the criteria are not met after a reasonable time—here, we use 10 000 iterations—must be coded as infinite, which means among other things that the mean is meaningless.

The proper measure of central tendency for such a data set is the median. If the majority of trials are coded as infinite, then the median is represented as infinity, shown in the results tables with the lemniscus. In order to combine iteration data, we used the mean of the medians, with the caveat that if any median were infinite, the mean would be infinite, too.

Note that the first measure, performance, considers data after a short run of 1000 iterations, and is a speed measure. The trials were run for as many as 10 000 iterations, however, to determine whether the criterion would be met at all. Thus, one measure was taken at 1000 iterations, and then if the criterion had not been met, the trial ran for as many iterations as were necessary. If the criterion was not met by 10 000 iterations, the trial was treated as if the criterion would never be met. In most cases this is true, as failure after 10 000 iterations suggests that the population has converged in an area of the search space that is not globally optimal. The first measure determines whether the algorithm can get a good solution fast, e.g., after only 1000 iterations, while the second and third measures determine how long it takes to find the global optimum if left to run, or whether it can find it at all.

D. Proportion Reaching Criteria

The third dependent measure is perhaps the most important one. This is a simple binary code indicating whether the criteria were met within 10 000 iterations or not. Averaged over all function trials, this gives the proportion of trials that successfully found the global optimum. There is no trick to this one; the mean of the ones and zeroes, where one indicates success and zero failure, gives the proportion of successes.

V. METHOD

The experiment manipulated neighborhood topologies, initialization strategies, and algorithm details. The types of topologies have been described and are shown in Fig. 1. Two kinds of initialization strategies were used, which we called, after Shi and Eberhart [12] "symmetrical" and "asymmetrical." Symmetrical initialization is performed over the entire spectrum of valid solutions, while asymmetrical initialization

TABLE III
STANDARDIZED PERFORMANCE OF THE TOPOLOGIES AND ALGORITHMS. NEGATIVE VALUES ARE BELOW THE MEAN WHILE POSITIVE VALUES ARE ABOVE. AS THE TASKS INVOLVE MINIMIZATION, THE BEST PERFORMANCES ARE THE MOST NEGATIVE. IN BOLD ARE THE BEST RESULTS FOR EACH ALGORITHM/INITIALIZATION PAIR

	Square	Ring	FourClusters	Pyramid	All	USquare	URing	UPyramid	UAll
Canon	-0.324	-0.307	-0.329	-0.324	-0.317	**-0.339**	-0.316	-0.334	-0.330
FIPS	-0.398	-0.302	-0.368	-0.383	-0.246	**-0.415**	-0.161	-0.407	-0.286
wFIPS	-0.398	-0.337	-0.384	-0.388	-0.249	**-0.412**	-0.156	-0.402	-0.275
wdFIPS	3.377	3.490	-0.387	3.469	3.457	**-0.412**	-0.162	-0.396	-0.333
Self	-0.389	-0.284	-0.363	-0.371	-0.340	**-0.401**	-0.287	-0.400	-0.355
wSelf	-0.396	-0.331	-0.384	-0.385	-0.340	**-0.407**	-0.305	-0.399	-0.363
Canonasym	-0.271	-0.229	-0.292	-0.272	-0.244	**-0.308**	-0.246	-0.289	-0.263
FIPSasym	0.287	-0.218	-0.219	0.402	1.767	**-0.344**	-0.150	-0.086	1.491
wFIPSasym	0.282	-0.247	-0.278	0.377	1.772	**-0.354**	-0.155	-0.119	1.489

started particles with an offset, so they were off-center. This eliminated any advantage that might be gained when function optima were located near the center of the parameter space.

There were five kinds of algorithm types:

Canonical: the traditional particle swarm, with Type 1″ constriction;

FIPS: the fully informed particle swarm with $\mathcal{W}$ returning a constant, i.e., where all contributions have the same value;

wFIPS: a fully informed swarm, where the contribution of each neighbor was weighted by the goodness of its previous best;

wdFIPS: also fully informed, with the contribution of each neighbor weighted by its distance in the search space from the target particle;

Self: a fully informed model, where the particle's own previous best received half the weight;

wSelf: a fully informed model, where the particle's own previous best received half the weight and the contribution of each neighbor was weighted by the goodness of its previous best.

Canonical, *FIPS*, and *wFIPS* were tested with both symmetrical and asymmetrical initializing.

The five types of topologies shown in Fig. 1 were tested. As some were tested with and without including the target particle in the neighborhood, there were nine topology conditions: Square, Ring, Pyramid, and All were tested both ways, and FourClusters was only tested with the self excluded. Conditions without the self are written with a "U" prefix, e.g., USquare is the Square topology, with the reference to the particle's own index removed from the neighborhood.

VI. RESULTS

We present the results on the three dependent measures separately. Following that we look at patterns across the measures, and finally we discuss the implications of the results.

A. Performance

Table III shows the pattern of standardized averages across the topologies and algorithms. Recalling that positive values indicate bad performance and negative ones good for the minimization problem, we notice some patterns immediately. For instance, four of the nine wdFIPS algorithm conditions are quite bad (more than three standard deviations worse than the mean); one cell of the All topology was more than 3 s.d., and two cells more than one s.d. worse than the mean; and two of the UAll topology conditions were farther than one s.d. worse than the mean. Two other cells in the Square, and two in the Pyramid topology, were less than one s.d. worse than the mean. These account for all of the worse-than-average cells in the design.

Looking for excellence, we note that of the eight conditions resulting in a performance -0.4 standard deviations or farther below the mean, five of them occurred when the neighborhood was the unselfed square. The other three appear in selfless pyramid conditions. The best performance of all occurred in the selfless-square FIPS configuration.

In light of the results presented below, it is noteworthy that problem solving using the URing topology was rather slow, relative to the others, while the USquare was rather fast.

The performance measure tells us how well a problem-solver is able to do within a limited amount of time. Many times in real-world applications it is "good enough" to find a good point on a local optimum; this first dependent variable tells us how high an algorithm is able to get on a fitness peak, but says nothing about whether it is the globally best peak.

B. Iterations to Criteria

How quickly does an algorithm reach a criterion that presumably reflects the presence of a global optimum? In Table IV, we see that some algorithm conditions cannot reach the criterion, even after 10 000 iterations. In particular, the wdFIPS tends not to reach it, especially with topologies that showed badly on the performance measure, as well; the All and UAll measures also failed in all cases with the FIPS variations, though they displayed about average success on the canonical algorithms. A few other topologies had trouble with the asymmetrical initializations.

Again, the URing was relatively slow and the USquare relatively faster than others. The canonical versions were moderately slow. The configurations that converged the fastest were the UPyramid on both FIPS and wFIPS, and the Four-Cluster topology on wFIPS.

Medians are used in this measure to account for failures to meet the criterion at all. A cell may have as many as half its trials fail to meet the standard, but if the remaining trials went quickly, the median iterations will suggest erroneously that something good has happened. Fast convergence of a configuration to the performance criterion on a large percentage of trials would suggest good problem solving qualities; fast convergence on half

TABLE IV
MEDIAN NUMBER OF ITERATIONS TO CRITERIA. THESE REPRESENT THE NUMBER OF ITERATIONS THE ALGORITHM TOOK TO REACH THE CRITERIA. AN INFINITE VALUE MEANS THAT AT LEAST HALF THE EXPERIMENTS WERE UNSUCCESSFUL. IN BOLD ARE THE QUICKEST RESULT FOR EVERY ALGORITHM/INITIALIZATION PAIR

	Square	Ring	FourClusters	Pyramid	All	USquare	URing	UPyramid	UAll
Canon	531	690	585	503	**444**	598.5	790	563.5	452.5
FIPS	672	486	339	1372.5	∞	367.5	2262.5	**290**	∞
wFIPS	381	372.5	279	869.5	∞	314.5	2065	**245**	∞
wdFIPS	∞	∞	301	∞	∞	339	2177.5	**232.5**	∞
Self	2722	934.5	834	7701	∞	578	**506**	981	∞
wSelf	681	604	612.5	6868	∞	396	**374**	583	∞
Canonasym	593	800	642	**552.5**	766	631	864	614	599
FIPSasym	∞	598.5	550.5	∞	∞	**371.5**	2260.5	471	∞
wFIPSasym	∞	411.5	352.5	∞	∞	**327**	2078.5	409.5	∞

TABLE V
PROPORTION OF EXPERIMENTS REACHING CRITERIA. THEY REPRESENT FOR EACH CONFIGURATION THE PROPORTION OF RUNS THAT WERE ABLE TO REACH THE REGION SPECIFIED BY THE CRITERIA. IN BOLD ARE THE BEST RESULTS FOR EACH ALGORITHM/INITIALIZATION PAIR

	Square	Ring	FourClusters	Pyramid	All	USquare	URing	UPyramid	UAll
Canon	0.875	0.913	0.921	0.854	0.754	**0.925**	0.908	0.871	0.754
FIPS	0.813	**0.988**	0.913	0.704	0.138	**0.988**	0.967	0.917	0.167
wFIPS	0.867	0.983	0.917	0.783	0.183	0.963	**1.000**	0.900	0.221
wdFips	0.000	0.000	0.858	0.000	0.000	0.942	**0.983**	0.863	0.325
Self	0.750	0.867	0.867	0.600	0.338	0.871	**0.988**	0.808	0.367
wSelf	0.796	0.875	0.833	0.646	0.388	0.867	**0.983**	0.833	0.392
Canonasym	0.758	0.767	**0.850**	0.783	0.596	0.842	0.792	0.783	0.650
FIPSasym	0.446	**0.942**	0.775	0.300	0.000	0.929	0.975	0.658	0.000
wFIPSasym	0.463	0.958	0.838	0.363	0.000	0.958	**0.988**	0.671	0.000

the trials would not. The next dependent variable tells us how often the criteria were met.

C. Proportion of Trials Reaching Criteria

For us, the third dependent measure is the most important. With today's computer speeds, the difference of a few thousand iterations may be a matter of seconds, and slight speed advantages are not usually crucial. The proportion measure tells, though, in black and white, whether the given algorithm/topology configuration can solve the problems (Table V).

The first result that jumps out is that the URing topology with the wFIPS algorithm found the global optimum (as measured by meeting the criteria) on 100% of its trials, that is, 40 trials each on 6 functions, amounting to 240 total trials. This is obviously a remarkable performance. We note also that 24 algorithm/topology combinations, out of 81, met the criterion 90% of the time or more. The canonical algorithm harbored performances greater than 0.90 on five of nine topologies, and the wFIPS on five of nine. wFIPS beat the 90% mark in three of the asymmetric initialization conditions, while the canonical algorithm never did. Unweighted FIPS was above 0.90 four times in the symmetric and three times in the asymmetric initialization conditions, and the weighted and unweighted Self algorithm broke the 0.90 standard one time each.

Looking at topologies, we see that none of the Square, Pyramid, All, or UAll conditions met the criterion 90% of the time. The Ring did it five times; the Four-Clusters thrice; USquare six times; URing eight times; and UPyramid twice. It appears that the USquare and URing topologies were the most successful vehicles for the communication among particles, at least across the algorithms tested here.

D. Combining the Measures

Our prejudice is that the most weight should be given to the last measure, the proportion of successes, though the other measures should be taken into account. By this rule of thumb, we could recommend always using the URing, which never failed when implemented with the wFIPS algorithm. We remember, however, that it was relatively slow by both the first two measures—plus, weighting the FIPS adds some computational cost.

If speed is a requirement, and the URing's relative slowness may create problems, then we would suggest the USquare with the unweighted FIPS algorithm. This combination succeeded approximately 98.9% of the time, meaning that it failed three times out of 240. The USquare/FIPS also had the best score at 1000 iterations and was the ninth fastest to reach the criteria.

VII. CONCLUSION

The canonical particle swarm algorithm showed itself to be a journeyman problem solver, finding the global optimum a respectable proportion of the time, depending on topology, and getting there in a respectably fast time. It was outperformed, though, by the FIPS versions, on every dependent measure.

It should be mentioned that the FIPS versions were the only ones able to consistently find the minimum to the Griewank function in ten dimensions. The best result obtained by a canonical configuration was the USquare with a proportion of 72.5% followed by much lower proportions using the other topologies (lower than 60%). Both FIPS algorithms with all the social topologies except the All versions were able to find the minimum 100% of the time.

The effect of weighting neighbors' contributions by their fitness is not clearly settled. We have to note that the one condition that met the criteria 100% of the time was a weighted condition. It could be argued that the extra computational expense of weighting is not justified; the unweighted versions performed quite well, maybe well enough.

A word needs to be said about the particle's contribution to its own trajectory. The U-versions, for instance USquare, performed better in many cases than versions where the self was included in the neighborhood. This goes contrary to particle swarm lore, which describes the algorithm in terms of the combination of "cognitive" and "social" experience. FIPS versions where half the weight was given to the self did not perform outstandingly, and it does not appear that the individual's own previous best needs to be part of the formula.

But it must be noted that the particle's own experience does contribute information to its trajectory, in the form that Miranda and Fonseca [13] call its "habit." For instance, the current velocity is created out of the velocities of the previous time steps as the particle maintains a cyclic trip through the search space. The particle's current position is featured in every comparison $\vec{P} - \vec{X}$ term. Finally, the position in the next step is a function of the position in the current one. Thus, the habit of the particle affects everything about its trajectory. It is just that, in the FIPS versions without self, the explicit memory of the previous best point it has sampled is not part of the particle's immediate decision.

As expected, increasing the size of the neighborhood seems to deteriorate the performance of the swarm. The very worse FIPS conditions in the study were the UAll and All topologies, where the particle is truly fully informed, gathering information from every single member of the population. The best were the Ring and Square versions, where the particle has three and five neighbors (counting itself), respectively, plus their U-versions, which subtract one.

We note that, though asymmetric initialization radically hurt performance in many conditions, it had nearly no effect on the USquare and URing conditions in FIPS and wFIPS. The unweighted FIPS with URing actually found the global optimum a greater proportion of the time with asymmetrical initialization, though we do not insist that this was a significant difference—but asymmetry clearly does not impair the algorithm.

The fully informed particle swarm is not a radical departure from previous versions. The standard two-term PSO is simply seen to be a special case of the FIPS, one that includes the selection of one particular neighbor to influence the target particle. The FIPS representation of the particle swarm algorithm has the potential for freeing investigators to look at other important features of the algorithm.

REFERENCES

[1] M. Clerc and J. Kennedy, "The particle swarm: Explosion, stability, and convergence in a multi-dimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, pp. 58–73, Feb. 2002.

[2] A. Carlisle and G. Dozier, "An off-the-shelf PSO," in *Proc. Workshop on Particle Swarm Optimization*. Indianapolis, IN: Purdue School of Eng. Technol., IUPUI, Apr. 2001.

[3] J. Kennedy and R. Mendes, "Topological structure and particle swarm performance," in *Proc. 4th Congr. Evolutionary Computation (CEC-2002)*, D. B. Fogel, X. Yao, G. Greenwood, H. Iba, P. Marrow, and M. Shackleton, Eds., Honolulu, HI, May 2002, pp. 1671–1676.

[4] J. Kennedy, "Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance," in *Proc. 1999 Conf. Evolutionary Computation* Washington, DC, 1999, pp. 1931–1938.

[5] R. G. Reynolds and C. Chung, "Knowledge-based self-adaptation in evolutionary programming using cultural algorithms," in *Proc. IEEE Int. Conf. Evolutionary Computation (ICEC'97)*, 1997, pp. 71–76.

[6] D. H. Wolpert and W. G. Macready. (1995) No free lunch theorems for search. Tech. Rep. SFI-TR-95-02-010, Santa Fe Inst., Sante Fe, New Mexico. [Online]. Available: citeseer.nj.nec.com/wolpert95no.html

[7] ——, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, pp. 67–82, Apr. 1997.

[8] T. M. English, "Optimization is easy and learning is hard in the typical function," in *Proc. Congr. Evolutionary Computation CEC00* La Jolla, CA, 6–9, 2000, pp. 924–931.

[9] A. Turing, "On computable numbers with an application to the Entscheidungsproblem," in *Proc. London Mathematical Society*, 1936, pp. 230–265.

[10] T. Cook and D. Campbell, *Quasiexperimentation: Designs and Analysis Issues for Field Settings*. Skokie, IL: Rand McNally, 1979.

[11] W. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 1992.

[12] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming VII: Proc. EP98*, Springer-Verlag, pp. 591–600.

[13] V. Miranda and N. Fonseca, "EPSO—best-of-two-worlds meta-heuristic applied to power system problems," in *Proc. 4th Congr. Evolutionary Computation (CEC-2002)*, D. B. Fogel, X. Yao, G. Greenwood, H. Iba, P. Marrow, and M. Shackleton, Eds. Honolulu, HI, May 2002, pp. 1080–1085.

Rui Mendes (M'03) received the B.S. degree in mathematics and computer science and the Ph.D. degree in computer engineering from the University of Minho, Braga, Portugal, in 1994 and 2004, respectively.

His thesis was called "Population Topologies and Their Influence in Particle Swarm Performance." He is a Computer Scientist, who has been working with the particle swarm algorithm since 2001. His research interests are swarm intelligence and evolutionary computation.

James Kennedy received the Ph.D. degree from the University of North Carolina, Chapel Hill, in 1992.

He is with the U.S. Department of Labor, Washington, DC. He is a Social Psychologist who has been working with the particle swarm algorithm since 1994. He has published dozens of articles and chapters on particle swarms and related topics, in computer science and social science journals and proceedings. He is a coauthor of *Swarm Intelligence* (San Mateo, CA: Morgan Kaufmann, 2001), with R. C. Eberhart and Y. Shi, now in its third printing.

José Neves is a Full Professor in the Informatics Department, University of Minho, Braga, Portugal. He is the Head of the Artificial Intelligence Group and coordinates several projects with applications in the areas of law and medicine. His research interests are knowledge representation and computational logic.