



## Faculty Publications

---

2004-07-01

# Choosing a Starting Configuration for Particle Swarm Optimization

Mark Richards  
markedup@gmail.com

Dan A. Ventura  
ventura@cs.byu.edu

Follow this and additional works at: <https://scholarsarchive.byu.edu/facpub>



Part of the [Computer Sciences Commons](#)

## Original Publication Citation

Mark Richards and Dan Ventura, "Choosing a Starting Configuration for Particle Swarm Optimization", Proceedings of the International Joint Conference on Neural Networks, pp. 239-2312, July 24.

---

## BYU ScholarsArchive Citation

Richards, Mark and Ventura, Dan A., "Choosing a Starting Configuration for Particle Swarm Optimization" (2004). *Faculty Publications*. 431.  
<https://scholarsarchive.byu.edu/facpub/431>

This Peer-Reviewed Article is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Faculty Publications by an authorized administrator of BYU ScholarsArchive. For more information, please contact [ellen\\_amatangelo@byu.edu](mailto:ellen_amatangelo@byu.edu).

# Choosing a Starting Configuration for Particle Swarm Optimization

Mark Richards  
Department of Computer Science  
Brigham Young University  
Provo, UT 84604  
E-mail: mdr@cs.byu.edu

Dan Ventura  
Department of Computer Science  
Brigham Young University  
Provo, UT 84604  
E-mail: ventura@cs.byu.edu

**Abstract**—The performance of Particle Swarm Optimization can be improved by strategically selecting the starting positions of the particles. This work suggests the use of generators from centroidal Voronoi tessellations as the starting points for the swarm. The performance of swarms initialized with this method is compared with the standard PSO algorithm on several standard test functions. Results suggest that CVT initialization improves PSO performance in high-dimensional spaces.

## I. INTRODUCTION

Particle Swarm Optimization (PSO) was introduced in 1995 by James Kennedy and Russell Eberhart as an optimizer for unconstrained continuous functions [7][8]. PSO models the search through the problem space as the flight of a swarm of particles (points). The goal is to have the particles converge on the global minimum or maximum.

Like many other optimization algorithms, PSO includes some stochastic behavior. At each iteration, a particle adjusts its position  $x$  and velocity  $v$  along each dimension  $d$ , based on the best position  $b_d$  it has encountered so far in its flight and the best position  $n_d$  found by any other particle in its neighborhood:

$$v_d^t = \chi [v_d^{t-1} + \phi_1 r_1 (b_d - x_d^{t-1}) + \phi_2 r_2 (n_d - x_d^{t-1})]$$
$$x_d^t = x_d^{t-1} + v_d^t$$

where  $\chi$ ,  $\phi_1$ , and  $\phi_2$  are constants and  $r_1$  and  $r_2$  are uniform random numbers on  $[0, 1]$ . A particle's neighborhood is comprised of the subset of other particles in the swarm with which it has direct communication. (This network of communication links in the swarm that determines the particles' neighborhoods, known as the *sociometry* of the swarm, is the subject of ongoing research [5][6][9][10].) Because the magnitudes of the attraction to a particle's own best position and the attraction to the best position of its neighbors are determined by two random numbers, the performance of the algorithm may vary considerably from run to run, even when all of the user-defined parameters (swarm size, sociometry, trust coefficients, inertia weight, constriction coefficient) are held constant. This inconsistency is a necessary evil, as the randomness is critical to the success of the algorithm. However, the algorithm would be more attractive if its performance was more predictable, so the variability should be reduced as much as possible.

Additional variability in PSO performance results from the use of random starting configurations. The initial position for a particle in the swarm is selected by drawing a uniform random number along each dimension of the problem space. To ensure broad coverage of the search space, the particles should be initialized so that they are distributed as evenly as possible throughout the space. The standard method of initialization fails to accomplish this goal, especially in high-dimensional spaces.

Using a starting configuration based on centroidal Voronoi tessellations (CVT) can lead to improved performance. Section II explains what centroidal Voronoi tessellations are and how they can be computed. Section III discusses experimental results comparing PSO performance with and without CVT initialization. Conclusions and possibilities for future work are presented in Section IV.

## II. CENTROIDAL VORONOI TESSELLATIONS

Voronoi tessellations can be thought of as a way to partition a space into compartments. A group of points in the space is designated to be the set of *generators*. The space is then partitioned into subsets, based on each point's proximity to the generators. Each subset is associated with one of the generators and consists of all of the points that are closer to that generator than to any of the other generators, with respect to some given distance function (e.g., the  $L_2$  norm). Figure 1 shows a graphical representation of Voronoi tessellations using 50 generators. The generators are depicted as dots. The lines mark the boundaries of the Voronoi cells. In this example, the generators were selected the same way initial points would be chosen for Particle Swarm Optimization. Note that the generators are not very evenly distributed throughout the space. By chance, some of the generators are at almost exactly the same point in the space. Note also the region near the lower righthand corner where there is a fairly large area with no generators. The result of this poor distribution is that some of the Voronoi cells are much larger than others.

With centroidal Voronoi tessellations, the generators lie at the center of their corresponding cells. Computation of CVTs can be expensive. Two of the most well-known algorithms for computing CVTs are MacQueen's method [3] and Lloyd's method [1]. MacQueen's method is probabilistic and requires

many iterations to converge, but each iteration is relatively simple. Conversely, Lloyd's algorithm is deterministic and requires only a few iterations, but each one is computationally expensive. Ju, Du, and Gunzburger have developed an algorithm that combines elements of both methods [4]. Their algorithm produces computationally feasible approximations of CVTs. The only motivation for using CVTs in the present research is to have the particles' initial positions more evenly distributed in the search space. Thus, it is unnecessary to require that the CVTs are completely accurate. A reasonable approximation will do. Pseudo-code for the Ju-Du-Gunzburger (JDG) algorithm is shown below.

```

function CVT(integer  $k$ ) returns set of  $k$  CVT generators
Choose  $k$  random points  $\{g_i\}_{i=1}^k$  as initial generators
do
  Choose  $q$  random points
  for  $i = 1$  to  $k$ 
    Gather into  $G_i$  the subset of the  $q$  points that are closer
      to  $g_i$  than to any of the other generators
    Compute the average  $a_i$  of the points in  $G_i$ 
    Move  $g_i$  some percentage of the distance closer to  $a_i$ 
  next  $i$ 
until stopping criterion met
return  $\{g_i\}_{i=1}^k$ 

```

When the algorithm starts,  $k$  random points are chosen as the generators. These generators are incrementally updated to improve the spacing between them. At each iteration, a set of  $q$  random points is chosen. Each of these random points is assigned to the current generator that it is closest to. Then, each generator's position is updated so that it moves closer to the average location of the random points that are associated with it. Figure 2 shows an example of generators found after five iterations of the algorithm. Note that although the Voronoi cells are still not exactly the same size, the generator points are much more evenly distributed than in Figure 1.

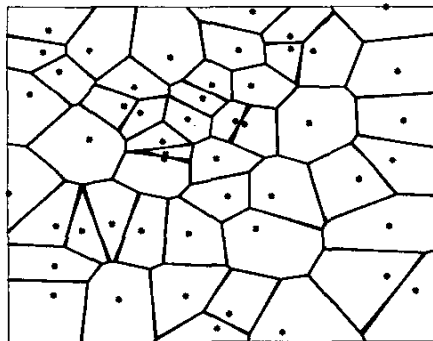


Fig. 1. Voronoi tessellations for 50 random points. The generators were created by drawing uniform random numbers along each dimension.

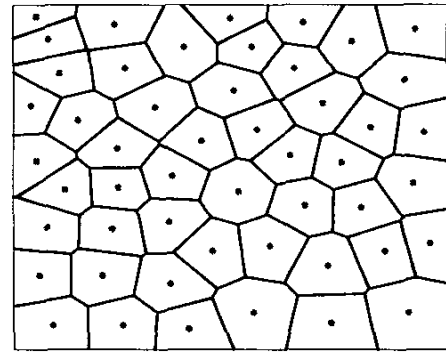


Fig. 2. Approximate centroidal Voronoi tessellations for 50 points. The generators were found by the JDG algorithm.

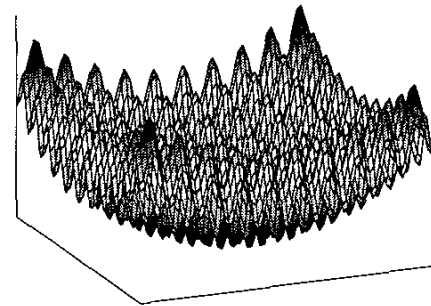


Fig. 3. The Rastrigin function.

### III. EXPERIMENTAL RESULTS

The use of perfect CVT generators as the initial points for the particles in PSO would minimize the maximum distance that the global optimum could be from one of the starting points, which should improve the algorithm's performance. The JDG algorithm does not produce perfect CVTs, but it does create a good approximation in a feasible amount of computation time.

PSO with CVT initialization was compared against the standard PSO algorithm on several well-known test functions (Table I). In all of the CVT experiments, the JDG algorithm was run for 5 iterations, using 50,000 sample points at each time step. Adding more sample points or allowing the algorithm to run for more iterations did not appear to improve the CVTs appreciably. On a modern desktop computer, the generation of the CVTs takes only a few seconds or less. If a large number of generators was required, the algorithm could of course take much longer, but PSO is usually implemented with only a few particles, perhaps 20–50.

CVT initialization did not appear to produce any improvement in performance for problems with few dimensions. A visual analysis helps explain why this is the case. Figure 3 shows the well-known Rastrigin test function in two dimensions. Figure 4 shows the 30 initial starting points and the locations of the function evaluations for the first 10 iterations of a typical

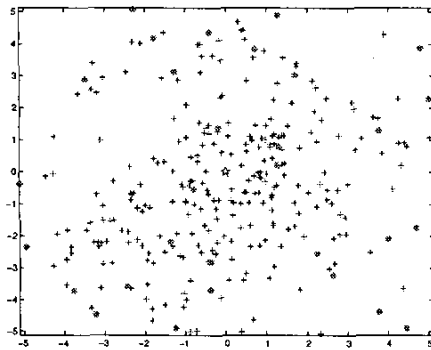


Fig. 4. Ten iterations of PSO on the two-dimensional Rastrigin function, without CVT initialization. The circles show the starting positions. Subsequent function evaluations are designated by +.

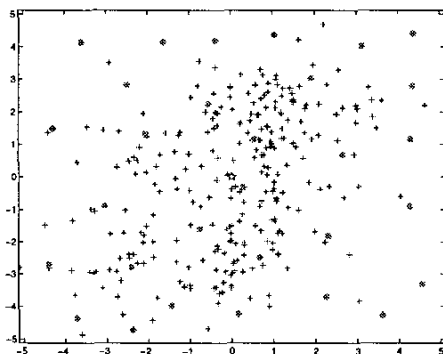


Fig. 5. Ten iterations of PSO on the two-dimensional Rastrigin function, with CVT initialization. The circles show the starting positions. Subsequent function evaluations are designated by +.

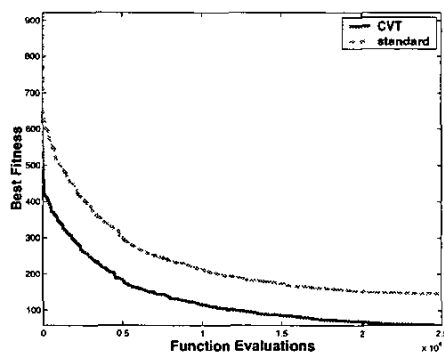


Fig. 6. Progress of the PSO algorithm over 25,000 function evaluations, with and without CVT initialization.

run of standard PSO on the Rastrigin function. Figure 5 shows similar information for PSO with CVT initialization. There does not appear to be much difference in the coverage of the space. For all of the test problems studied, there was no significant difference in the mean best solution found by the two versions of PSO over 30 runs of each algorithm. At least for the two-dimensional case, the randomness of PSO seems to produce good coverage of the space, even when the particles' starting positions are not carefully selected.

As the number of dimensions in the search space increases, the number of particles in the swarm quickly becomes sparse with respect to the total size of the space. (The size of the swarm may be increased, but large swarms are often inefficient with respect to the number of function evaluations needed for the optimization.) Because of this sparseness, the choice of starting positions also becomes more important. Figure 6 shows a comparison of the two versions of PSO on the Rastrigin function in 50 dimensions. The graph shows the intermediate best fitness of the swarm over 25,000 function evaluations. The lines trace the median best fitness of 30 runs at each time step. On average, the swarms with CVT initialization find good points in the space much earlier than their standard PSO counterparts. The gap does not narrow as the optimization continues.

Table II shows the mean performance for both algorithms on the test functions in 50 dimensions. CVT initialization leads to significantly better results in all cases. Table III reports the results of significance tests comparing the algorithms on each problem in 20, 30, and 50 dimensions.

The one-sided  $p$ -values shown were obtained using random permutation tests [2]. For each test, the results for 30 runs of each algorithm were compiled and averaged. The difference between the two means was recorded. Then the 60 values were randomly shuffled and reassigned to two groups. The new difference in group means was compared to the actual difference. This process was repeated one million times. The  $p$ -value reported in the table is the proportion of the one million trials that produced a difference in group means that was more extreme than the one that was actually observed. This gives an estimate of the likelihood that a difference that extreme would be observed by chance, if there were in fact no difference between the two algorithms. A  $p$ -value of zero in the table means that none of the one million trials produced a difference more extreme than the observed one. This is strong evidence that there is a difference in the performance of the two algorithms.

The permutation tests are attractive because they make it possible to obtain a highly accurate  $p$ -value without having to use a two-sample  $t$ -test. The  $t$ -test assumes that the two populations being sampled are normally distributed and have equal spreads. These assumptions are often egregiously violated for PSO tests.

Table III shows how the importance of initial position increases with the dimensionality of the problem. In 20 dimensions, the CVT-initialized swarms perform significantly better (at the  $\alpha = .05$  level) on only four of the eight problems. In

30 dimensions, the differences are significant for six problems. In 50 dimensions, there is a highly significant difference for all eight problems.

#### IV. CONCLUSIONS AND FUTURE WORK

The starting configuration of PSO can be critical to its performance, especially on high-dimensional problems. By using the generators of centroidal Voronoi tessellations as the starting points, the particles can be more evenly distributed throughout the problem space. Perfect CVTs can be computationally expensive to compute, but the JDG algorithm can be used to generate approximate CVTs in a reasonable amount of time.

The present research involved trials of PSO on several popular test functions, but all of the functions have their global optimum at the origin and do not have complex interactions among their variables. Future research will address more challenging optimization problems.

Future work will also address additional issues related to the use of PSO in high-dimensional spaces, including the selection of swarm size.

TABLE I  
TEST FUNCTION DEFINITIONS

Function	Formula	Range
Ackley	$f(x) = -20 \exp \left[ -0.02 \sqrt{n^{-1} \sum_{i=1}^n x_i^2} \right] - \exp \left[ n^{-1} \sum_{i=1}^n \cos(2\pi x_i) \right] + 20 + e$	$\pm 30$
DeJong	$f(x) = \sum_{i=1}^n i \cdot x_i^4$	$\pm 20$
Griewank	$f(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos \left[ \frac{x_i}{\sqrt{i}} \right] + 1$	$\pm 300$
Rastrigin	$f(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$	$\pm 5$
Rosenbrock	$f(x) = \sum_{i=1}^n [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$\pm 10$
Schaffer	$f(x) = \sum_{i=1}^{n-1} \left[ 0.5 + \frac{(\sin \sqrt{x_i^2 + x_{i+1}^2} - 0.5)^2}{(1 + .001(x_i^2 + x_{i+1}^2))^2} \right]$	$\pm 100$
Schaffer2	$f(x) = \left( \sum_{i=1}^n x_i^2 \right)^{.25} \left[ 1 + \left( 50 \left( \sum_{i=1}^n x_i^2 \right)^{.1} \right)^2 \right]$	$\pm 100$
Sphere	$f(x) = \sum_{i=1}^n x_i^2$	$\pm 50$

TABLE II  
AVERAGE PERFORMANCE OVER 30 RUNS FOR STANDARD PSO VS. PSO WITH CVT INITIALIZATION FOR FUNCTIONS IN 50 DIMENSIONS

Function	Standard	CVT	Diff
Ackleys	1.67	1.37	0.30
DeJong	1.07	.095	.975
Griewank	.241	.107	.134
Rastrigin	144	58.5	85.5
Rosenbrock	234	144	90
Schaffer	18.4	14	4.4
Schaffer2	2250	1250	1000
Sphere	.227	.055	.172

TABLE III  
P-VALUES COMPARING AVERAGES OF 30 RUNS FOR BOTH STANDARD AND CVT-INITIALIZED PSO

Function	20d	30d	50d
Ackleys	.44	.28	0
DeJong	.0022	0	0
Griewank	.20	.30	0
Rastrigin	0	0	0
Rosenbrock	.45	.03	0
Schaffer	0	0	0
Schaffer2	.002	2.5e-6	0
Sphere	.06	8.0e-6	0

#### REFERENCES

- [1] Q. Du, M. Faber, M. Gunzburger, "Centroidal Voronoi tessellations: Applications and algorithms," *SIAM Review*, vol. 41, 1999, pp. 637-676.
- [2] P. Good, *Permutation Tests: A Practical Guide to Resampling Methods for Testing Hypothesis*. New York: Springer-Verlag, 1994.
- [3] J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, University of California, 1967, pp. 281-297.
- [4] L. Ju, Q. Du, M. Gunzburger, "Probabilistic methods for centroidal Voronoi tessellations and their parallel implementations," Penn State Department of Mathematics Report No. AM250, 2001.
- [5] J. Kennedy, "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance," *Proceedings of the Congress of Evolutionary Computation*, Washington D.C., 1999, pp. 1931-1938.
- [6] J. Kennedy, "Stereotyping: improving particle swarm performance with cluster analysis," *Proceedings of the IEEE Congress on Evolutionary Computation*, San Diego, CA, 2000, pp. 1507-1512.
- [7] J. Kennedy, R. Eberhart, "Particle Swarm Optimization," *Proceedings of the IEEE Conference on Neural Networks*, Perth, Australia, 1995, pp. 1942-1948.
- [8] J. Kennedy and R. Eberhart, *Swarm Intelligence*, New York: Academic Press, 2001.
- [9] J. Kennedy, R. Mendes, "Population structure and particle swarm performance," *Proceedings of the Congress on Evolutionary Computation*, vol. 2, 2002, pp. 1671-1676.
- [10] M. Richards, D. Ventura, "Dynamic sociometry in particle swarm optimization," *Proceedings of the Sixth International Conference on Computational Intelligence and Natural Computing*, North Carolina, 2003, pp. 1557-1560.