

Adapting Particle Swarm Optimization to Dynamic Environments

Anthony Carlisle

*Department of Mathematical and Computer Sciences,
Huntingdon College*

Gerry Dozier

*Department of Computer Science and Software
Engineering, Auburn University*

Abstract

In this paper the authors propose a method for adapting the particle swarm optimizer for dynamic environments. The process consists of causing each particle to reset its record of its best position as the environment changes, to avoid making direction and velocity decisions on the basis of outdated information. Two methods for initiating this process are examined: periodic resetting, based on the iteration count, and triggered resetting, based on the magnitude of the change in the environment. These preliminary results suggest that these two modifications allow PSO to search in both static and dynamic environments.

KEYWORDS: Particle Swarm Optimization, Dynamic Environments

1. Introduction

The particle swarm optimization (PSO) was originally designed by Kennedy and Eberhart [5,11,15] and has been compared to genetic algorithms [6,1,13] for efficiently finding optimal or near-optimal solutions in large search spaces. The technique involves simulating social behavior among individuals (particles) “flying” through a multidimensional search space, each particle representing a single intersection of all search dimensions. The particles evaluate their positions relative to a goal (fitness) at every iteration, and particles in a local neighborhood share memories of their “best” positions, then use those memories to adjust their own velocities, and thus subsequent positions. The original formula developed by Kennedy and Eberhart was improved by Shi and Eberhart [14,15] with the introduction of an inertia parameter, ω , that increases the overall performance of PSO.

The original PSO formulae define each particle as a potential solution to a problem in D-dimensional space, with particle i represented $X_i=(x_{i1},x_{i2},\dots,x_{iD})$. Each particle also

maintains a memory of its previous best position, $P_i=(p_{i1},p_{i2},\dots,p_{iD})$, and a velocity along each dimension, represented as $V_i=(v_{i1},v_{i2},\dots,v_{iD})$. At each iteration, the P vector of the particle with the best fitness in the local neighborhood, designated g , and the P vector of the current particle are combined to adjust the velocity along each dimension, and that velocity is then used to compute a new position for the particle. The portion of the adjustment to the velocity influenced by the individual’s previous best position (P) is considered the *cognition* component, and the portion influenced by the best in the neighborhood is the *social* component [7,8,10,9,1,2].

With the addition of the inertia factor, ω , by Shi and Eberhart [3], these formulae are:

$$v_{id} = \omega * v_{id} + \eta_1 * \text{rand}() * (p_{id} - x_{id}) + \eta_2 * \text{rand}() * (p_{gd} - x_{id}) \quad (1a)$$

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

Constants η_1 and η_2 determine the relative influence of the social and cognitive components, and are usually both set the same to give each component equal weight as the cognitive and social learning rate [1].

In [7] Kennedy introduces four models of PSO, defined by omitting or restricting components of the velocity formula. The complete formula above defines the *Full Model*. Dropping the social component results in the *Cognition-Only Model*, whereas dropping the cognition component defines the *Social-Only Model*. A fourth model, *Selfless*, is the *Social-Only Model*, but the neighborhood best is chosen only from the neighbors, without considering the individual particle’s P vector, that is, $g \neq i$.

2. Experiment Design

For the purposes of these experiments, a PSO run is considered to be successful if any particle in the swarm “lands” within a specified radius (e) of a goal solution. This radius is dependent on the precision of the solution desired. If, during the execution of a PSO run, the goal value changes, the original PSO algorithm has no method for detecting this change, and the particles are still influenced by their memories of the original goal position.

If the change in the goal is small, this problem is self-correcting. Subsequent fitness evaluations will result in positions closer to the new goal location replacing earlier P vectors, and the swarm should follow, and eventually intersect, the moving goal. However, if the movement of the goal is more pronounced, it moves too far from the swarm for subsequent fitness evaluations to return values better than the current P vector, and the particles do not track the moving goal.

We attempt to rectify this problem by having the particles periodically replace their P vector with their current X vector, thus “forgetting” their experiences to that point. This differs from a restart, in that the particles, in retaining their current location, have retained the profits from their previous experiences, but are forced to redefine their relationship to the goal at that point.

To ascertain the effectiveness of this approach, we constructed a scenario in which a beacon (the goal) moves at some constant (possibly zero) velocity diagonally through the search space. Particles attempt to locate the beacon on the basis of the strength of the beacon’s signal. Thus, the fitness function f is the distance of the particle from the beacon, computed as:

$$f = \sqrt{\sum_{i=1}^D (g_i - x_i)^2}$$

where g_i is the position of the goal and x_i is the position of the particle on each i of the D dimensions.

3. Experiment Implementation

Two experiments were performed. In the first, the resetting of the P vector was done on a

regular frequency, based on the iteration count. In the second experiment, the resets were triggered [3,4] by detecting when the goal moved some specified distance from its original position.

The swarm design was patterned after the Shi and Eberhart model in [15]: twenty particles are randomly positioned ± 5.0 on each dimension, η_1 and η_2 are set to 2.0, and no Vmax is used. As discussed in [15], to compensate for the lack of a Vmax, ω is implemented as a linearly decreasing function, initially set to 0.65, decreasing to 0.15 over the first 1500 iterations, and remaining at 0.15 for the remainder of the run. (These values for ω were chosen empirically, and no attempt was made to tailor the parameters to the problem.) A run is considered successful if any particle lands within a threshold radius (e) of the goal, set to 0.002. Runs that do not locate the goal are terminated at 4000 generations.

The goal is initially placed at (5.0,5.0) and is moved in the (-x,-y) direction a constant distance each iteration. This distance is determined by the goal velocity (v_g), taken from a list of increments: (0.001, 0.002, 0.005, 0.01, 0.05, 0.1, 0.5, 1.0, and 2.0). Note that for low v_g (0.001, 0.002), the goal does not leave the initial swarm vicinity during the maximum 4000 iterations, but at high v_g , the goal exits the initial swarm region within a few iterations and continues in a straight line for a considerable distance. As a consequence, no Xmax is implemented, so the particles fly without bounds.

For each run we report three values: we compute the reliability of the algorithm in terms of solutions found, the efficiency in terms of average iterations per solution found, and the median iterations per run, which gives a quick indication of the distribution of the solutions.

4. Results

The algorithm was initially run without resetting the P vector, as a baseline. The results, in Table 1, show that the Full and Social-Only models perform well at low v_g , with a success rate of 100% through $v_g=0.002$, but performance falls off rapidly as v_g increases. The Social-Only Model consistently finds

solutions about 25% faster than the Full Model, the Selfless Model's performance is inferior to both the Full and Social-Only models, and the Cognition-Only Model has very poor performance.

In Experiment 1 we reset the P vector on periods of 1, 2, 4, ..., 2048 iterations. In general, performance improved for v_g greater than 0 when resets were implemented, but resetting the "best location" values too frequently (every generation or every other generation) resulted in more solutions being found at higher v_g , but with less reliability at the lower v_g . Likewise, resetting the values too infrequently (every 2048, 1024, or 512 iterations) reduced the likelihood of successes at the higher v_g . Midrange reset frequencies gave the best performance, with a reset frequency of every 16 iterations giving the best overall performance. In Table 2 we see the results of runs with periodic resets performed at 8, 16, and 32 iterations. At 16 iterations, the Full Model finds solutions 100% of the time for v_g up to 0.01. The Social-Only model consistently found solutions fastest, but as v_g increased, the reliability of the Social-Only model deteriorated more quickly than that of the Full Model, finding solutions only 96% and 98% of the time. As in the baseline run, the Selfless Model performance trailed that of both the Full and Social-Only models, and Cognition-Only never showed acceptable performance.

In Experiment 2, instead of resetting the P vector on a regular period, the reset was performed if the goal wandered beyond some fixed distance from its position at the last reset. This was implemented by selecting a random point in the search space and evaluating a fitness for that point. At each iteration, the fitness value of this point was reevaluated, and if the value differed by more than the allowable range (the *trigger value*), all particles P vectors were reset.

The trigger values used were computed in terms of the threshold radius e : 0.5%, 1%, 2.5%, 5%, 10%, 25%, 50%, 100%, and 200% of e . This method of initiating resets did not suffer the loss of reliability encountered in the previous experiment. Every trigger resulted in 100% solutions at lower v_g for the Full Model. Table 3 shows the results of this experiment for trigger values of 10%, 25%, 50%, and 75%, the

range of values bracketing in the best performance. As in Experiment 1, the Social-Only Model is more efficient at finding solutions, but overall the Full Model is more reliable, finding solutions 100% of the time for v_g up to 0.01 for some runs. The Social-Only Model rarely produced such a success rate, more often finding solutions about 96% of the time. In general, the Selfless Model was less efficient than the Social-Only Model and less reliable than the Full Model, and the Cognition-Only Model again never performed well.

Not unexpectedly, no model was able to consistently find any solutions for v_g higher than $v_g=0.1$. In these cases, the goal simply "outruns" the swarm.

5. Discussion

This is a first look at adapting PSO to dynamic environments, a sort of *proof of concept* experiment. As such, these experiments have many obvious weaknesses. For one, the choice of evaluation function, geometric distance, is simple. Another simplification was made in the mechanism used in Experiment 2 to trigger the reset. This method relies on the entire environment changing at the same rate. In an environment with localized fluctuations, this method would be insufficient. Finally, in these implementations, the P vector reset is done simultaneously for all particles, whereas a more gradual reset throughout the population might provide better convergence.

6. Conclusions

The particle swarm optimization technique can be modified to work with dynamic goals, if the rate of change of the goal does not exceed the maximum velocity of the swarm. Periodically resetting the particle memories to the current positions allows the swarm to track a changing goal with minimum overhead. Likewise, the reset can be triggered by detecting when the goal has moved too far, but with the added expense of an additional function evaluation in each generation.

7. Further Research

| Target Speed: | Full Model | | | Cognition-Only | | | Social-Only | | | Selfless | | |
|---------------|------------|---------|--------|----------------|---------|--------|-------------|---------|--------|----------|---------|--------|
| | % Solved | Ave/Sol | Median | % Solved | Ave/Sol | Median | % Solved | Ave/Sol | Median | % Solved | Ave/Sol | Median |
| 0.000 | 100.00 | 39.34 | 39.0 | 10.00 | 27.20 | 4000.0 | 100.00 | 29.18 | 30.0 | 100.00 | 56.24 | 40.0 |
| 0.001 | 100.00 | 40.44 | 42.0 | 18.00 | 681.67 | 4000.0 | 100.00 | 30.96 | 31.0 | 94.00 | 59.81 | 39.0 |
| 0.002 | 100.00 | 41.70 | 40.0 | 18.00 | 116.67 | 4000.0 | 100.00 | 29.20 | 30.0 | 94.00 | 58.43 | 41.0 |
| 0.005 | 94.00 | 45.62 | 43.0 | 30.00 | 209.00 | 4000.0 | 88.00 | 31.36 | 30.0 | 88.00 | 37.64 | 39.0 |
| 0.010 | 62.00 | 37.03 | 48.0 | 20.00 | 133.60 | 4000.0 | 76.00 | 30.63 | 33.0 | 58.00 | 45.45 | 63.0 |
| 0.050 | 14.00 | 27.00 | 4000.0 | 0.00 | | 4000.0 | 18.00 | 22.00 | 4000.0 | 10.00 | 28.80 | 4000.0 |
| 0.100 | 6.00 | 13.33 | 4000.0 | 2.00 | 18.00 | 4000.0 | 8.00 | 18.00 | 4000.0 | 6.00 | 18.00 | 4000.0 |

Table 1: No resets are performed (baseline runs).

These experiments dealt with a goal whose movement was constant in both direction and velocity. We need to determine the effect of varying either or both parameters on both experiments. We also need to examine the effect distributing the reset operation over some number of iterations, rather than abruptly cleansing the memories of all particles simultaneously. We need to determine a method of detecting a change in the environment at the particle level, rather than using an outside trigger. Finally, we need to evaluate the effect of the changing environment on the choice of the inertia parameter.

References

- [1] Angeline, P. (1998). Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Difference, The 7th Annual Conference on Evolutionary Programming, San Diego, USA.
- [2] Angeline, P. (1998). Using Selection to Improve Particle Swarm Optimization, IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, USA.
- [3] Cobb, H. G. (1990). An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments, NRL Memorandum Report 6760.
- [4] Cobb, H. G. and Grefenstette, J. J. (1993). Genetic Algorithms for Tracking Changing Environments, Fifth International Conference on Genetic Algorithms.
- [5] Eberhart, R. and Kennedy, J. (1995). A New Optimizer Using Particle Swarm Theory, Proc. Sixth International Symposium on Micro Machine and Human Science (Nagoya, Japan), IEEE Service Center, Piscataway, NJ, 39-43.
- [6] Eberhart, R. and Shi, Y. (1998). Comparison between Genetic Algorithms and Particle Swarm Optimization, The 7th Annual Conference on Evolutionary Programming, San Diego, USA.
- [7] Kennedy, J. (1997). The Particle Swarm: Social Adaptation of Knowledge, IEEE International Conference on Evolutionary Computation (Indianapolis, Indiana), IEEE Service Center, Piscataway, NJ, 303-308.

| Target Speed: | Full Model | | | Cognition-Only | | | Social-Only | | | Selfless | | |
|--------------------------------------|------------|---------|--------|----------------|---------|--------|-------------|---------|--------|----------|---------|--------|
| | % Solved | Ave/Sol | Median | % Solved | Ave/Sol | Median | % Solved | Ave/Sol | Median | % Solved | Ave/Sol | Median |
| Recalc period is every 8 iterations | | | | | | | | | | | | |
| 0.000 | 98.00 | 44.61 | 43.0 | 8.00 | 23.75 | 4000.0 | 94.00 | 36.83 | 37.0 | 94.00 | 55.36 | 44.0 |
| 0.001 | 100.00 | 43.68 | 44.0 | 18.00 | 1159.67 | 4000.0 | 94.00 | 33.60 | 34.0 | 84.00 | 49.76 | 53.0 |
| 0.002 | 100.00 | 43.72 | 44.0 | 10.00 | 1723.40 | 4000.0 | 98.00 | 31.55 | 31.0 | 92.00 | 45.37 | 43.0 |
| 0.005 | 100.00 | 47.02 | 42.0 | 12.00 | 495.17 | 4000.0 | 96.00 | 32.35 | 32.0 | 84.00 | 50.79 | 50.0 |
| 0.010 | 98.00 | 53.67 | 49.0 | 12.00 | 209.83 | 4000.0 | 94.00 | 34.23 | 36.0 | 68.00 | 46.15 | 53.0 |
| 0.050 | 70.00 | 82.06 | 103.0 | 0.00 | | 4000.0 | 36.00 | 32.61 | 4000.0 | 20.00 | 40.90 | 4000.0 |
| 0.100 | 14.00 | 84.43 | 4000.0 | 2.00 | 12.00 | 4000.0 | 10.00 | 20.60 | 4000.0 | 6.00 | 16.00 | 4000.0 |
| 0.500 | 2.00 | 17.00 | 4000.0 | 0.00 | | 4000.0 | 2.00 | 32.00 | 4000.0 | 0.00 | | 4000.0 |
| 1.000 | 2.00 | 10.00 | 4000.0 | 0.00 | | 4000.0 | 0.00 | | 4000.0 | 0.00 | | 4000.0 |
| Recalc period is every 16 iterations | | | | | | | | | | | | |
| 0.000 | 100.00 | 41.74 | 41.0 | 8.00 | 18.75 | 4000.0 | 96.00 | 31.38 | 32.0 | 90.00 | 44.62 | 39.0 |
| 0.001 | 100.00 | 40.04 | 38.0 | 14.00 | 131.86 | 4000.0 | 96.00 | 32.33 | 30.0 | 94.00 | 51.21 | 44.0 |
| 0.002 | 100.00 | 43.62 | 42.0 | 28.00 | 318.36 | 4000.0 | 96.00 | 31.60 | 32.0 | 98.00 | 44.94 | 45.0 |
| 0.005 | 100.00 | 43.34 | 42.0 | 12.00 | 623.17 | 4000.0 | 90.00 | 32.16 | 31.0 | 86.00 | 48.07 | 41.0 |
| 0.010 | 100.00 | 47.96 | 46.0 | 16.00 | 115.25 | 4000.0 | 94.00 | 33.49 | 31.0 | 82.00 | 43.83 | 47.0 |
| 0.050 | 54.00 | 74.41 | 174.0 | 2.00 | 24.00 | 4000.0 | 20.00 | 32.60 | 4000.0 | 24.00 | 39.58 | 4000.0 |
| 0.100 | 28.00 | 87.93 | 4000.0 | 0.00 | | 4000.0 | 8.00 | 27.50 | 4000.0 | 2.00 | 21.00 | 4000.0 |
| 0.500 | 2.00 | 116.00 | 4000.0 | 0.00 | | 4000.0 | 0.00 | | 4000.0 | 0.00 | | 4000.0 |
| Recalc period is every 32 iterations | | | | | | | | | | | | |
| 0.000 | 100.00 | 38.82 | 40.0 | 14.00 | 21.57 | 4000.0 | 98.00 | 30.65 | 31.0 | 96.00 | 46.62 | 42.0 |
| 0.001 | 100.00 | 38.26 | 36.0 | 30.00 | 428.80 | 4000.0 | 94.00 | 31.47 | 32.0 | 100.00 | 44.58 | 42.0 |
| 0.002 | 100.00 | 39.66 | 41.0 | 26.00 | 302.38 | 4000.0 | 98.00 | 29.04 | 28.0 | 98.00 | 49.67 | 43.0 |
| 0.005 | 100.00 | 46.50 | 43.0 | 24.00 | 185.08 | 4000.0 | 98.00 | 32.04 | 33.0 | 94.00 | 53.68 | 43.0 |
| 0.010 | 98.00 | 45.96 | 46.0 | 12.00 | 140.00 | 4000.0 | 78.00 | 32.18 | 35.0 | 78.00 | 59.41 | 66.0 |
| 0.050 | 44.00 | 65.14 | 4000.0 | 2.00 | 46.00 | 4000.0 | 26.00 | 30.62 | 4000.0 | 12.00 | 31.83 | 4000.0 |
| 0.100 | 20.00 | 117.50 | 4000.0 | 2.00 | 11.00 | 4000.0 | 16.00 | 23.12 | 4000.0 | 6.00 | 15.33 | 4000.0 |
| 0.500 | 2.00 | 46.00 | 4000.0 | 2.00 | 9.00 | 4000.0 | 0.00 | | 4000.0 | 0.00 | | 4000.0 |

Table 2: Resets are performed periodically based on the iteration count.

| Target Speed: | Full Model | | | Cognition-Only | | | Social-Only | | | Selfless | | |
|---------------|---|---------|--------|----------------|---------|--------|-------------|---------|--------|----------|---------|--------|
| | % Solved | Ave/Sol | Median | % Solved | Ave/Sol | Median | % Solved | Ave/Sol | Median | % Solved | Ave/Sol | Median |
| 0.000 | 100.00 | 34.82 | 34.0 | 10.00 | 30.40 | 4000.0 | 100.00 | 30.08 | 31.0 | 100.00 | 43.24 | 37.0 |
| 0.001 | 100.00 | 41.68 | 41.0 | 16.00 | 560.50 | 4000.0 | 96.00 | 29.35 | 29.0 | 98.00 | 62.76 | 42.0 |
| 0.002 | 100.00 | 41.76 | 37.0 | 24.00 | 738.58 | 4000.0 | 100.00 | 30.04 | 29.0 | 94.00 | 45.06 | 42.0 |
| 0.005 | 100.00 | 44.54 | 41.0 | 16.00 | 197.00 | 4000.0 | 96.00 | 30.71 | 31.0 | 96.00 | 45.35 | 40.0 |
| 0.010 | 94.00 | 49.09 | 50.0 | 4.00 | 253.50 | 4000.0 | 88.00 | 36.64 | 38.0 | 80.00 | 47.85 | 50.0 |
| 0.050 | 62.00 | 59.23 | 98.0 | 6.00 | 106.33 | 4000.0 | 36.00 | 34.56 | 4000.0 | 18.00 | 35.33 | 4000.0 |
| 0.100 | 12.00 | 35.83 | 4000.0 | 2.00 | 2.00 | 4000.0 | 8.00 | 30.25 | 4000.0 | 4.00 | 18.50 | 4000.0 |
| 0.500 | 6.00 | 28.67 | 4000.0 | 2.00 | 27.00 | 4000.0 | 4.00 | 10.00 | 4000.0 | 0.00 | | 4000.0 |
| | Recalc is done when target moves outside 10.00% of the threshold radius | | | | | | | | | | | |
| 0.000 | 100.00 | 38.74 | 39.0 | 8.00 | 20.75 | 4000.0 | 100.00 | 29.76 | 30.0 | 100.00 | 68.52 | 43.0 |
| 0.001 | 100.00 | 36.78 | 37.0 | 26.00 | 153.23 | 4000.0 | 96.00 | 31.73 | 33.0 | 98.00 | 49.33 | 42.0 |
| 0.002 | 100.00 | 40.40 | 39.0 | 14.00 | 311.86 | 4000.0 | 96.00 | 30.31 | 29.0 | 98.00 | 42.08 | 36.0 |
| 0.005 | 100.00 | 40.32 | 40.0 | 32.00 | 193.12 | 4000.0 | 92.00 | 29.33 | 29.0 | 96.00 | 51.67 | 45.0 |
| 0.010 | 100.00 | 43.20 | 42.0 | 14.00 | 237.14 | 4000.0 | 92.00 | 33.67 | 33.0 | 78.00 | 45.28 | 46.0 |
| 0.050 | 58.00 | 72.79 | 111.0 | 4.00 | 14.50 | 4000.0 | 36.00 | 39.83 | 4000.0 | 16.00 | 42.38 | 4000.0 |
| 0.100 | 30.00 | 69.07 | 4000.0 | 2.00 | 46.00 | 4000.0 | 22.00 | 30.09 | 4000.0 | 12.00 | 27.00 | 4000.0 |
| 0.500 | 2.00 | 13.00 | 4000.0 | 0.00 | 4000.0 | 4000.0 | 0.00 | | 4000.0 | 0.00 | | 4000.0 |
| | Recalc is done when target moves outside 50.00% of the threshold radius | | | | | | | | | | | |
| 0.000 | 100.00 | 37.26 | 37.0 | 20.00 | 21.60 | 4000.0 | 94.00 | 30.94 | 30.0 | 100.00 | 60.04 | 41.0 |
| 0.001 | 100.00 | 43.18 | 43.0 | 16.00 | 366.38 | 4000.0 | 98.00 | 28.45 | 29.0 | 96.00 | 51.52 | 44.0 |
| 0.002 | 100.00 | 41.32 | 42.0 | 18.00 | 31.44 | 4000.0 | 100.00 | 28.70 | 28.0 | 92.00 | 48.83 | 48.0 |
| 0.005 | 100.00 | 38.16 | 35.0 | 20.00 | 121.40 | 4000.0 | 90.00 | 31.02 | 31.0 | 90.00 | 52.11 | 42.0 |
| 0.010 | 100.00 | 49.54 | 46.0 | 16.00 | 105.88 | 4000.0 | 76.00 | 30.08 | 34.0 | 84.00 | 51.60 | 56.0 |
| 0.050 | 58.00 | 69.38 | 115.0 | 6.00 | 25.33 | 4000.0 | 34.00 | 27.59 | 4000.0 | 36.00 | 36.33 | 4000.0 |
| 0.100 | 22.00 | 89.27 | 4000.0 | 0.00 | 4000.0 | 4000.0 | 16.00 | 28.25 | 4000.0 | 4.00 | 32.50 | 4000.0 |
| 0.500 | 4.00 | 31.00 | 4000.0 | 0.00 | 4000.0 | 4000.0 | 0.00 | | 4000.0 | 0.00 | | 4000.0 |
| | Recalc is done when target moves outside 75.00% of the threshold radius | | | | | | | | | | | |
| 0.000 | 100.00 | 43.16 | 38.0 | 16.00 | 23.25 | 4000.0 | 98.00 | 30.88 | 29.0 | 100.00 | 40.34 | 37.0 |
| 0.001 | 100.00 | 36.08 | 36.0 | 20.00 | 454.10 | 4000.0 | 98.00 | 29.98 | 30.0 | 100.00 | 45.90 | 43.0 |
| 0.002 | 100.00 | 38.22 | 38.0 | 24.00 | 190.50 | 4000.0 | 96.00 | 31.06 | 31.0 | 98.00 | 48.37 | 44.0 |
| 0.005 | 98.00 | 51.12 | 47.0 | 10.00 | 165.80 | 4000.0 | 86.00 | 29.16 | 29.0 | 94.00 | 49.62 | 42.0 |
| 0.010 | 98.00 | 55.71 | 45.0 | 26.00 | 203.38 | 4000.0 | 78.00 | 31.36 | 32.0 | 88.00 | 51.20 | 47.0 |
| 0.050 | 60.00 | 87.23 | 134.0 | 6.00 | 99.33 | 4000.0 | 28.00 | 27.57 | 4000.0 | 16.00 | 43.50 | 4000.0 |
| 0.100 | 26.00 | 52.62 | 4000.0 | 2.00 | 71.00 | 4000.0 | 16.00 | 25.12 | 4000.0 | 12.00 | 27.33 | 4000.0 |
| 0.500 | 4.00 | 38.50 | 4000.0 | 0.00 | 4000.0 | 4000.0 | 0.00 | | 4000.0 | 0.00 | | 4000.0 |

Table 3: Resets are triggered by the magnitude of the change in the goal location.

- [8] Kennedy, J. (1997). Minds and cultures: Particle swarm implications. Socially Intelligent Agents: Papers from the 1997 AAAI Fall Symposium. Technical Report FS-97-02, 67-72. Menlo Park, CA: AAAI Press.
- [9] Kennedy, J. (1998). The Behavior of Particles, 7th Annual Conference on Evolutionary Programming, San Diego, USA.
- [10] Kennedy, J. (1998). Thinking is social: Experiments with the adaptive culture model. Journal of Conflict Resolution, 42, 56-76.
- [11] Kennedy, J. and Eberhart, R. (1995). Particle Swarm Optimization, IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ, IV: 1942-1948.
- [12] Kennedy, J. and Eberhart, R. (1997). A Discrete Binary Version of the Particle Swarm Algorithm, 1997 IEEE Conference on Systems, Man, and Cybernetics (Orlando, FL), pp. 4104-4109
- [13] Kennedy, J. and Spears, W. (1998). Matching Algorithms to Problems: An Experimental Test of the Particle Swarm and some Genetic Algorithms on the Multimodal Problem Generator, IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, USA.
- [14] Shi, Y. H., Eberhart, R. C., (1998), A Modified Particle Swarm Optimizer, IEEE International Conference on Evolutionary Computation, Anchorage, Alaska.
- [15] Shi, Y. H., Eberhart, R. C., (1998). Parameter Selection in Particle Swarm Optimization, The 7th Annual Conference on Evolutionary Programming, San Diego, USA.