Comparing Particle Swarms for Tracking Extrema in Dynamic Environments

Xiaodong Li
School of Computer Science and
Information Technology
RMIT University
Melbourne VIC 3001, Australia
xiaodong@cs.rmit.edu.au

Khanh Hoa Dam
School of Computer Science and
Information Technology
RMIT University
Melbourne VIC 3001, Australia
kdam@cs.rmit.edu.au

Abstract- This paper presents a comparative study of particle swarm models on their abilities to track extrema in dynamic environments. A standard PSO, two randomized PSOs, and a fine-grained PSO are evaluated in non-trivial multimodal dynamic environments involving small constant step changes, different large step changes, many chaotic step changes of the extrema. DF1 proposed by Morrison and De Jong is used to generate these three types of dynamics [1]. Our results indicate that PSO and its variants are able to perform reasonably well in a 2-dimensional variable space, whereas perform well to a less extent in a 10-dimensional variable space. It is also found that the fine-grained PSO is able to outperform all other PSO variants in the 10-dimensional variable space, likely due to its ability to maintain better population diversity.

1 Introduction

Many real-world optimization problems are dynamic by nature. For example, traffic condition in a city changes continuously and dynamically over time. Traffic lights are switched on and off at all intersections. What might be regarded as an optimal traffic condition at one time might not be optimal in the next minute. In contrast to optimization towards a static objective, in dynamic environments the task has become to keep track of the changes occurred in the environment, then to follow as closely as possible the dynamically changed objective. Many real-world problems require systems that are adaptable to changes over time.

In recent years, there have been increasing interests in using evolutionary algorithms in dealing with dynamic optimization problems [2][3][4]. These EAs have shown promising performance as compared with traditional methods. Particle Swarm Optimization, another population-based optimization algorithm, which is in many aspects similar to EAs, has also been employed to solve dynamic optimization problems. Preliminary studies have shown that PSO has comparable performances to those of EAs on simple linear fitness lanscapes[8]. However most of real-world dynamic optimization problems are often non-linear and much more complex than what these simple linear functions can represent. This makes it very difficult to compare different EAs' abilities in handling complex dynamic environments. As the first step towards making it possible to carry out comparative studies over different dynamic optimization algorithms, Morrison and De Jong proposed a test function generator DF1 that allows a user to specify a wide range of dynamics ranging from simple to very complex dynamic environments [1]. Branke also independently proposed a moving peak benchmark function, which has a similar capability to DF1's [4].

In this paper, we carry out a comparative study over a number of PSO models on their abilities to track the extrema on a non-trivial multimodal fitness landscape, using a range of dynamics that can be produced by DF1, from those involving applying to the the highest peak with single small constant step changes, different large non-uniformed step changes, to even a sequence of chaotic step changes.

The paper is organized as follows: Section 2 provides the background information on dynamic optimization, including DF1. Section 3 gives an overview on PSO and its current approaches in dealing with dynamic environments. Section 4 describes the experimental setups, followed the results and analysis given in section 5. Finally section 6 concludes the paper.

2 Background

2.1 Dynamic Optimization Problems

In contrast to static optimization, the goal in dynamic optimization is not only to reach the optimum but also to track its trajectory as closely as possible in the search space over time. A dynamic optimization problem can be described as:

$$f(\vec{x}, t) \rightarrow min, \vec{x} \in \Re^N, t \in T \quad (1)$$

The objective function f depends on both the solution vector $\vec{x}$ and time t.

In [2], Angeline suggested a method of creating a number of dynamic environments from a base function (the parabolic function) of three dimensions:

$$f(\vec{x}) = \sum_{i=1}^{3} x_i \quad (2)$$

which has its minimum at (0,0,0). By specifying the amount of displacement of the base function along each dimension, Angeline was able to generate three different types of dynamics, $linear, circular$, and $random$ trajectories. The amount of displacement is also called $stepsize$; whereas the number of generations between each movement of the base function is the $updatefrequency$.

2.2 DF1 Function Generator

The parabolic function only reflects a linear and unimodal fitness landscape [2][8], however real world problems are often much more complex, with highly nonlinear and multimodal fitness landscapes. These systems also exhibit a wide variety of dynamic behaviours, from simple to complex or even chaotic. Morrison and De Jong proposed a test function generator DF1 [1], which is capable of generating fitness landscapes with some of these features. For a 2-dimensional problem, the static evaluation function in DF1 is defined as the following:

$$f(X,Y) = max_{i=1,N}[H_i - R_i * \sqrt{(X-X_i)^2 + (Y-Y_i)^2)}\,] \quad (3)$$

where N denotes the number of cones in the environment. Each cone is independently specified by its location (X_i, Y_i), where $X_i \in$ [-1.0,1.0] and $Y_i \in$ [-1.0,1.0], its height H_i, where $H_i \in [Hbase, Hbase + Hrange]$, and its slope R_i, where $R_i \in [Rbase, Rbase + Rrange]$. These cones are blended using the max function. Note that function f can be specified for any number of dimensions. A variety of fitness landscape shapes (morphology) with varying complexity can be generated by specifying N, the number of peaks; $Rbase$, the minimum value of the slope control variable; $Rrange$, the allowed range of the slope variable; $Hbase$, the minimal cone height; and $Hrange$, the range of allowed cone heights. A range of dynamic changes on the landscape can be produced by using a one-dimensional, nonlinear logistics function that has simple bifurcation transitions to progressively more complex behaviour:

$$Y_i = A * Y_{i-1} * (1 - Y_{i-1}) \quad (4)$$

where A is a constant, and Y_i is the value at time step i. The Y values produced at each iteration by the above logistics function can be then used as step sizes to generate the types of dynamics desired. By specifying different values of A, one can generate a range of different Y values, eg., a single small constant Y value, multiple constant Y values, to a chaotic sequence of Y values. The Y values can be then used with equation (3) to produce a wide range of dynamics. The reader can refer to [1] for more information on DF1.

3 Particle Swarm Models

Particle Swarm Optimization is an optimization technique simulating the social behaviour of insects or animals. PSO is also like EA as it is population-based, yet it differs itself from EAs in the way it manipulates each individual in the population. Instead of using evolutionary operators such as crossover and mutation, PSO modifies each individual through its current position in the search space, $\vec{x_i} = (x_{i1}, x_{i2}, \ldots, x_{iN})$, and its velocity (ie., the rate of position change), $\vec{v_i} = (v_{i1}, v_{i2}, \ldots, v_{iN})$. The velocity v_i is modified at each iteration step by the i-th particle's previous best position, ie., the position giving the best fitness value, $\vec{pbest_i} = (p_{i1}, p_{i2}, \ldots, p_{iN})$, and the population's best particle's position $\vec{gbest_i}$. At each iteration, the velocity and position of each particle in the swarm are updated according to the following two equations [7]:

$$\begin{aligned} \vec{v_i} &= K * (\vec{v_i} + c_1 * rand() * (\vec{pbest_i} - \vec{x_i}) + \\ &\quad c_2 * rand() * (\vec{gbest_i} - \vec{x_i})) \quad (5) \\ \vec{x_i} &= \vec{x_i} + \vec{v_i} \quad (6) \end{aligned}$$

where c_1 and c_2 are two positive constants. $rand()$ is a random function returning values in the range [0, 1]. K is called the constriction coefficient and it is computed according to:

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \quad (7)$$

where $\varphi = c_1 + c_2, \varphi > 4$.

3.1 Existing Approaches to Handle Changes

An optimization method when handling changes in a dynamic system needs to achieve two goals: detecting that a change in the environment has actually occurred, and reacting appropriately to the change so that the optimum can still be tracked. Carlisle and Dozier used a *sentry* particle which is randomly chosen at each iteration [6]. The sentry particle gets evaluated before each iteration and compares its fitness with its previous fitness value. If the two values are different, indicating the environment has changed, ie., the optimum has moved, the whole swarm gets alerted and several possible responses can then be performed. Hu and Eberhart proposed to re-evaluate the global best $\vec{gbest}$ and a second-best $\vec{gbest}$ to detect the movement of the optimum [10].

The next goal for the optimization method is to provide an effective mechanism to respond to these changes. This is also our main focus in this study. At the time this paper is written, there have been three significant studies [8], [10], [6] relating to PSO. Those approaches can be grouped into one of the following categories: **1) Recalculate** the fitness

value of current personal best vector $\vec{pbest}$ of each particle in the swarm when a change is detected[6]. If the new $\vec{pbest}$ value of a particle is less fit than its current position $\vec{x}$, then that value is replaced by the current position $\vec{x}$. The motivation of this approach resides in the fact that when the environment changes, i.e. the optimum location moves, it is more likely better for particles to "forget" their previous experiences. However, instead of blindly replacing all $\vec{pbest}$ vectors by their associated current position vector $\vec{x}$, the above comparison is performed so that valid experience can be still persevered. Therefore, particles can still partially benefit from their memory and consequently more accurate results may be produced. **2) Re-randomize** a portion of the swarm, ranging from 10% to 100% of the particles [8],[10]. The motivation of this approach is two fold. Firstly, re-randomizing is necessary because it avoids the situation where the whole swarm has already stuck to a small area and as a result it seems not to be easy to jump out to keep tracking the changes. Secondly, a portion of the population is preserved since it might be a better approach to use the old swarm in the case when the system change is relatively small. [10] has in fact proposed and tested different re-randomization methods.

Despite of the success of the above proposed approaches [6][8][10], most current works are restricted to experimentation with simple dynamic systems. A typical example is the parabolic function, which is linear and unimodal[2]. In addition, these experiments were only concerned with uniformed changes (ie., constant step sizes) in a simple environment. More complex dynamic environments with non-uniformed changes are yet to be explored. For example, changes controlled by using step sizes, from single small constant,to many different and even chaotic step sizes. In the real world, dynamic systems are often non-linear, change non-uniformly, or even chaotically in a complex multimodal search space. The abilities of particle swarm models to handle this type of dynamic systems are of great importance and warrants further studies.

3.2 Comparing Particle Swarm Models

To investigate PSO's ability to track the location of a moving optimum, the performance of four PSO models are compared over a range of different dynamic environments generated by using DF1. Apart from a standard PSO model, two randomized PSO variants are included, one variant PSO-R20 with 20% of particles reinitialized when a change occurs; another one PSO-R50 with 50% of the particles reinitialized when a change occurs in the system. The fourth PSO variant is a fine-grained PSO (FGPSO), based on the suggestions by Sarma and De Jong and more recently by Kennedy and Mendes [13] [12]. In a FGPSO, each particle is mapped onto a grid point on a 2-dimensional lattice. Each point on the lattice has a Von Neumann neighbourhood, which is defined to include its 4 neighbours (ie., the grid points to the east, west, north and south). The fine-grained PSO restricts a particle's interactions to its neighbourhood. The overlapping neighbourhoods introduce an implicit mechanism for the interaction among particles throughout the grid. As a result, a particle's influence can only propagate through the swarm population gradually, resulting in the population diversity being better maintained than otherwise in a standard PSO. Kennedy and Mendes found that population topology has a profound effect on the PSO's performance [12]. After conducting numerous experiments, they concluded that the Von Neumann neighbourhood gave consistently better results. In our implementation of the fine-grained PSO model, each particle chooses a local best from its neighbourhood (including its four neighbours and itself), to act as the global best $\vec{gbest}$ in equation (5) for updating the current velocity $\vec{v_i}$ of the particle. Therefore, at each iteration, the position and velocity vectors, $\vec{x_i}$ and $\vec{v_i}$, of each particle are recalculated based on its personal best position so far and one of its four neighbours in the Von Neumann neighbourhood.

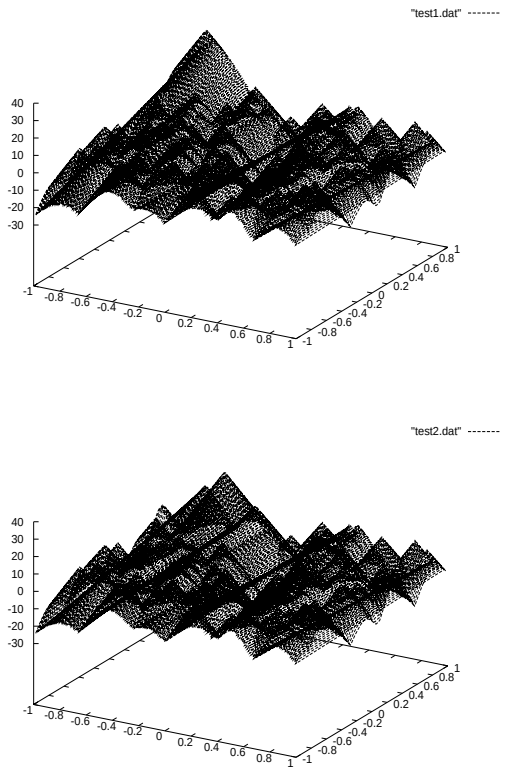
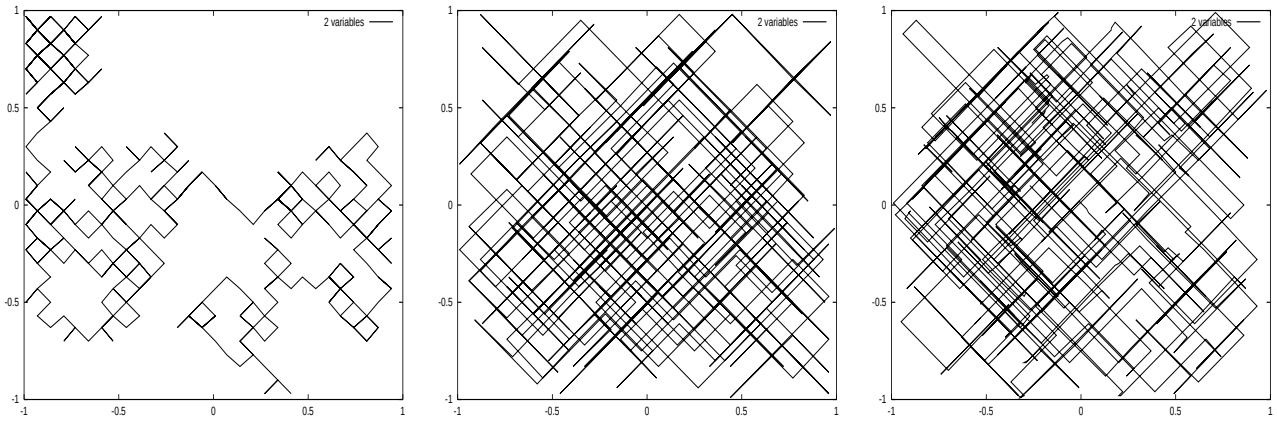


Figure 1: A landscape a) before and b) after its highest peak has moved



a) b) c)

Figure 2: The trajectory of the highest peak moves over 500 generations in each of the three dynamic environments respectively - a) A = 1.2; b) A = 3.3; c) A = 3.99.

4 Experimental setup

We use DF1 to generate 100 peaks randomly, and then pick a peak randomly and make it 3 times the height of the second highest peak. The highest peak moves in different sets of step sizes to represent different dynamic environments, while all other peaks remain static. The PSO variants are tested over such landscapes. Figure 1 shows an example of landscapes generated by DF1 for a 2-dimensional variable space, one being the initial landscape, and the second one showing after the highest peak has moved. These landscapes have non-linear and multimodal surface, therefore they are more complex and realistic than those used in previous PSO studies [8].

4.1 Dynamics Specified using Step Sizes

Dynamics are generated by DF1 through controlling a variety of step sizes. Three types of step sizes are specified and applied to the highest peak, which moves in a randomly chosen direction at each generation, to create three different dynamic environments: **1)** with a single constant small step size ($A = 1.2$); **2)** with one of the two large different step sizes ($A = 3.3$); **3)** with a step size randomly chosen from a sequence of chaotically changing step sizes ($A = 3.99$).

All peaks are generated within the variable range [-1.0, 1.0]. With A = 1.2, DF1 generates a constant Y value of 0.12?? (using equation 4). If $A = 3.3$ is chosen, a pair of Y values $\{0.8236, 0.4794\}$ is generated. $A = 3.99$ produces a sequence of chaotic Y values. The Y values produced are then scaled by a certain percentage to create step sizes required for different dynamic environments. The direction of a step change is set randomly, however if the peak reaches a boundary of the allowed range, then the step change direction is reversed. Figure 2 shows for a 2-dimensional example, the trajectory of the highest peak's movement over 500 generations, for A = 1.2, 3.3 and 3.99 respectively.

4.2 Update Frequency

The highest peak makes a step change at different frequencies, representing a dynamically changing environment at different rates. Three update frequencies $\Delta g \in \{1, 10, 100\}$ are used, that is, applying a step move to the highest peak at every generation, every 10, or 100 generations respectively. A smaller Δg means the swarm population has less time to adapt to the changes, whereas a larger Δg, ie., update occurring over a longer interval would allow more time for the population to converge during the Δg generations before the next update occurs.

4.3 PSO Settings

For all four PSO variants, we use an initial swarm population of 225, which are randomly generated for each variable in the range [-50, 50]. DF1 produces cones only in the range [-1.0, 1.0], and using equation (3) guarantees that particles with positions outside [-1.0, 1.0] very unfit. This makes all particles fly quickly towards this region at the start of a run. For the fine-grained PSO variant (FGPSO), each of the 225 randomly generated particles is mapped onto a node of a 15x15 two-dimensional lattice. Similar to [12], we set both c_1 and c_2 to 2.01, which produces constrictor factor K of 0.729844 (equation 7).

4.4 Performance Measurement

The ability of a PSO variant to track extrema in specified dynamic environments is measured by the distance between the best particle in the population and the highest peak, at each generation. The smaller this distance is, the fitter the particle is considered to be, and the better the PSO variant's ability in tracking the highest peak. Each PSO variant is run for 1000 generations, and this distance value is averaged over 50 runs.

When the highest peak moves, to signal a change in the environment, the best previous position $\vec{p_i}$ of a particle is reset to its current position (see section 3.1). The reason for doing so is to force each particle of the swarm to *forget* its own experience, starting afresh with respect to the new position of the goal. To allow for a fair comparison, this modification is implemented in all four PSO variants.

5 Results

Figure 3 and 4 show for all four PSO variants the average of the distance between the best particle and the highest peak at each generation over 50 runs (section 4.4). Each figure plots the results of using three different sets of step sizes determined by three different A values, $A \in \{1.2, 3.3, 3.99\}$, in the top, middle, and bottom row respectively. The setting of different update frequencies $\Delta g \in \{1, 10, 100\}$ are shown increasingly from left to right. Plots in the left column show results for $\Delta g = 1$; plots in the middle column show results for $\Delta g = 10$; and plots in the right column show results for $\Delta g = 100$.

For the 2-dimensional problem, when the update frequency is set to every generation ($\Delta g = 1$), all plots in the left column of figure 3 show that all PSO variants are able to track the goal reasonably well for all different step sizes, with the exception of FGPSO, which has increasingly deteriorating performance after around generation 70 when $A = 1.2$. Because the highest peak moves at every generation (very frequent), a PSO variant has to readjust itself to the moving peak at every generation. All four PSO variants have reached a minimum distance of around 1 to the highest peak. When Δg is increased to 10, all four PSO variants are able to track the goal even better, minimizing the distance further to a value around 0.1. It is interesting to see that FGPSO this time outperforms the other 3 variants for $A = 3.99$ and $\Delta g = 10$ (the middle plot in the bottom row). When an even larger $\Delta g = 100$ is used, all four PSO variants are able to track the highest peak significantly better than those with $\Delta g \in \{1, 10\}$ for $A = 1.2$ (ie., a small constant step size applied to the highest peak). Basically all PSOs are able to converge significantly closer to the highest peak during the Δg generations. These result are in agreement with those by [8]. The two restart versions: PSO-R20 and PSO-R50 continue to perform well for $A \in \{2.2, 3.99\}$, however in these two cases, the standard PSO and FGPSO gradually lose its ability to effectively track the goal. Overall figure 3 shows the restart PSOs consistently perform well for all three types of dynamic environments, however the standard PSO and FGPSO could lose their ability to track the moving goal.

The initial idea of using FGPSO to maintain a better population diversity, thereby leading to a better performance is not evident in the results, especially when $A \in \{3.3, 3.99\}$, and $\Delta g = 100$ as shown in figure 3. One possible explanation for this is that the majority of particles are stuck on the 99 sub-optimal peaks across the landscape, and for those few particles left within the vicinity of the highest peak, they are not able to converge towards the highest peak, because of the use of the neighbourhood best as $\vec{gbest}$. The restart PSO-R20 and PSO-R50 obviously have no such problems.

Figure 4 shows the results of applying three different sets of step sizes over 10-dimensions (ie., 10 variables). The PSO variants are still able to track the highest peak, but perhaps to a less extent. In contrast to figure 3, FGPSO outperforms the other 3 PSO variants in all cases with the only exception of $A = 3.99$ and $\Delta g = 100$, where PSO-R50 beats FGPSO. Another observation of figure 4 is that when using $A = 1.2$, FGPSO is able to reach to a relatively small distance from the goal, and then stay at such distance almost unchanged over the remaining generations. This indicates that FGPSO's ability to maintain a better diversity in the swarm population is effectively harnessed here for the 10-dimensional problem space, whereas other PSOs, including PSO-R20 and PSO-R50 could not do as well as FGPSO in these cases.

One surprising observation is the standard PSO's ability in tracking extrema in higher dimensional problem space. The standard PSO outperforms the PSO-R20 for all cases where $A \in \{1.2, 3.3, 3.99\}$ and $\Delta g \in \{1, 10\}$, and even PSO-R50 occasionally. This indicates that simply randomizing the swarm population whenever a change is detected does not necessarily always warrant better performance than otherwise.

6 Conclusions

This paper has presented a comparative study of a number of PSO variants over their abilities to track extrema in three different dynamic environments, specified by using a variation of step sizes. Fitness landscapes produced by using DF1 are used in this study to represent non-trivial and more complex multimodal fitness landscapes than the typical time-varying sphere model used in many previous studies.

Our results shows that even in a noisy environment with numerous small sub-optimal peaks, PSO and its variants are

able to track reasonably well in a 2-dimensional problem space, and to a less extent in a higher dimensional space. These results not only agree with the findings from previous studies on PSO's ability to track extrema in simple dynamic environments [8] [5] [6] [10], but also provide some interesting insights on PSO's ability to track extrema in more complex multimodal dynamic environments.

Our findings also show that the fine-grained PSO seems to be more effective in tracking extrema in a higher dimensional variable spaces, but less effective over a 2-dimensional variable space. Further studies to identify under exactly what conditions FGPSO works better will be carried out.

Bibliography

[1] Morrison, R. W. and DeJong, K. A.: A test problem generator for non-stationary environments. In Congress on Evolutionary Computation, volume 3, pages 2047-2053. IEEE, 1999

[2] Angeline, J. P.: Tracking Extrema in Dynamic Environments. Proceeding of the Evolutionary Programming VI, 1997, Indianapolis, IN, Berlin: Springer-Verlag, p. 335-345.

[3] Back, T: On the behaviour of evolutionary algorithms in dynamic environments. Proceeding of the International Conference on Evolutionary Computation, Anchorage, AK, Piscataway, 1998 NJ: IEEE Press, p. 446-551

[4] Branke, J.: Evolutionary Optimisation in Dynamic Environments. Volume 3 of the Book Series on Genetic Algorithms and Evolutionary Computation, Kluwer Academic Publishers, 2002.

[5] Carlisle, A. and Dozier, G.: Adapting particle swarm optimization to dynamic environments. Proceedings of International Conference on Artificial Intelligence (ICAI 2000). p.429-434, Las Vegas, Nevada, USA, 2000

[6] Carlisle, A. and Dozier, G.: Tracking Changing Extrema with Adaptive Particle Swarm Optimizer. ISSCI, 2002 World Automation Congress, Orlando FL USA, June, 2002

[7] Clerc, M. and Kenedy, M.: The Particle Swarm: Explosion, Stability, and Convergence in a Multi-Dimensional Complex Space. IEEE Transactions on Evolutionary Computation, vol. 6, p. 58-73, 2002.

[8] Eberhart, R. C. and Shi, Y.: Tracking and optimizing dynamic systems with particle swarms. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2001). pp.94-97, Seoul, Korea., May 2001

[9] Eberhart, R. C., and Shi, Y.: Comparing inertia weights and constriction factors in particle swarm optimization. Proceeding Congress on Evolutionary Computation 2000, San Diego, CA, pp 84-88, 2000

[10] Hu, X. and Eberhart, R. C: Adaptive particle swarm optimization: detection and response to dynamic systems. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002). Honolulu, Hawaii USA, May 2002

[11] Kennedy, J. and Eberhart, R.: Particle swarm optimization. Proceedings of 1995 ICEC, Perth, Australia, 1985

[12] Kennedy, J. and Mendes, R.: Population structure and particle swarm performance. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002), Honolulu, Hawaii USA, May 2002

[13] Sarma, J. and DeJong, K. A.:The behaviour of spatially distributed evolutionary algorithms in non-stationary environments. In Proceedings of the First Genetic and Evolutionary Computation Conference (GECCO 1999), San Francisco, CA, 572-578. Morgan Kaufmann, 1999

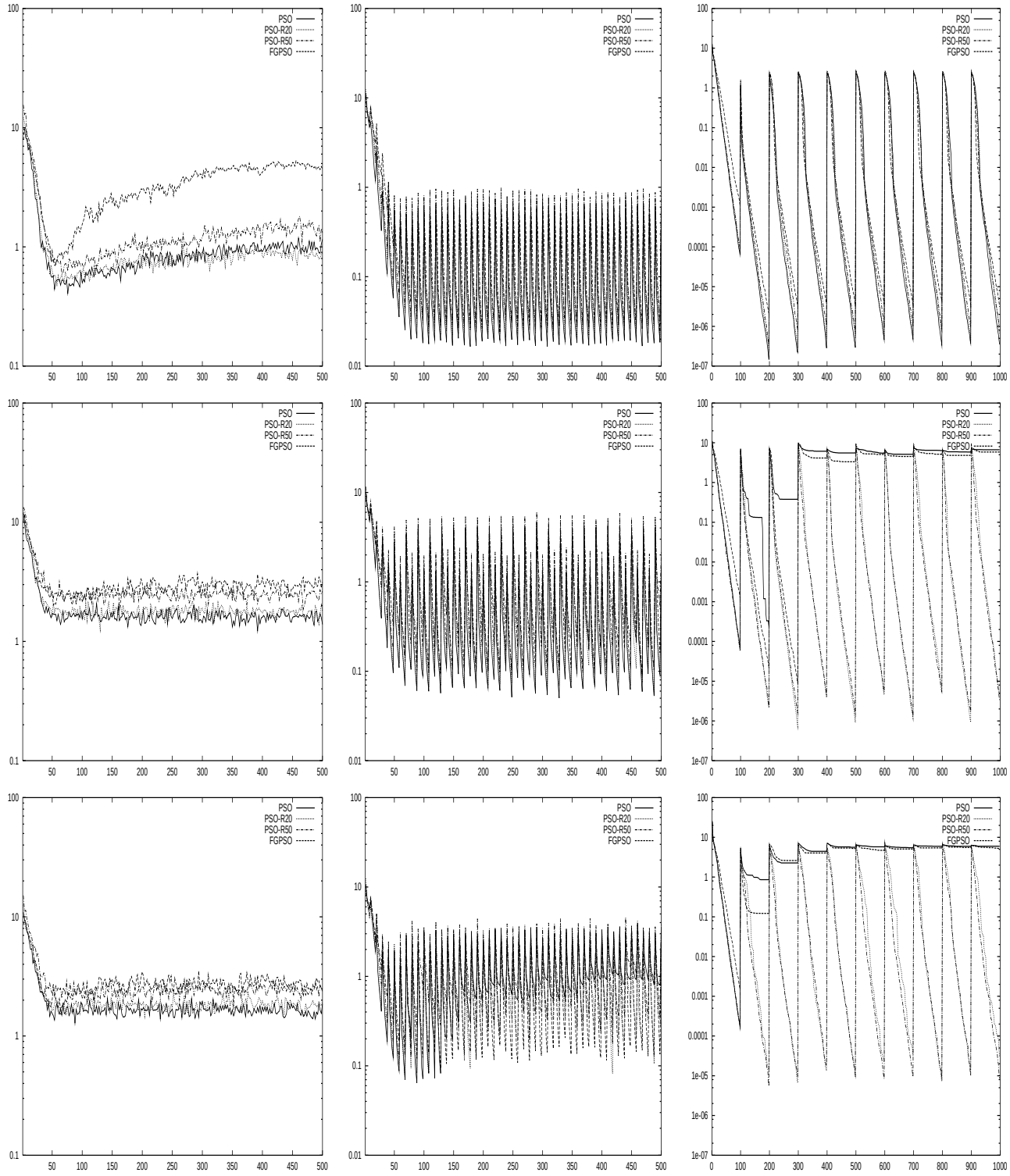


Figure 3: Results of applying various step sizes to the highest peak over 2-dimensions. The y-axis is the best particle's distance to goal, and the x-axis is the number of generations. The plots in the top, middle and bottom row show the results using $A \in \{1.2, 3.3, 3.99\}$ respectively. Plots on the left, middle, and right column show $\Delta g \in \{1, 10, 100\}$ respectively.

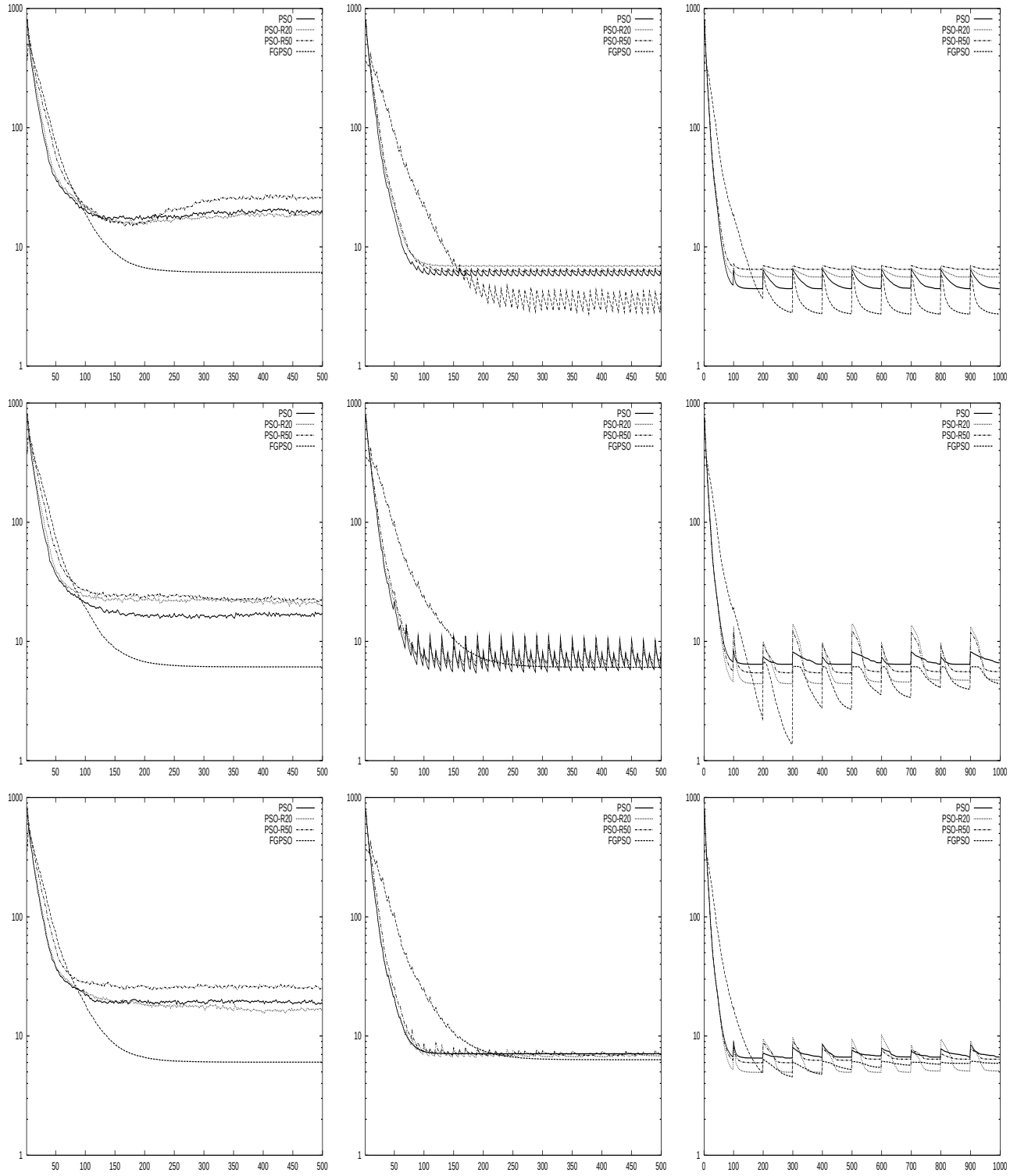


Figure 4: Results of applying various step sizes to the highest peak over 10-dimensions. The y-axis is the best particle's distance to goal, and the x-axis is the number of generations. The plots in the top, middle and bottom row show the results using $A \in \{1.2, 3.3, 3.99\}$ respectively. Plots on the left, middle, and right column show $\Delta g \in \{1, 10, 100\}$ respectively.