Heterogeneous Particle Swarms in Dynamic Environments

B.J. Leonard, A.P. Engelbrecht, *Senior Member, IEEE* and A.B. van Wyk

***Abstract*—This paper investigates the performance of a dynamic heterogeneous particle swarm optimizer (dHPSO) on dynamic problems. The results are compared to that of charged and quantum particle swarms, specifically designed for optimization in dynamic environments. It is shown that dHPSO possesses the ability to manage the diversity of the swarm dynamically, allowing it to overcome the problem of diversity loss and to successfully track the moving optimum over time. Additionally, it is discovered that the partially maintained diversity implemented in charged and quantum swarms, while necessary for exploration, inhibit their ability to effectively rediversify in response to a change in the environment. Experiments that are conducted on a range of dynamic problems show that dHPSO consistently produces lower average errors than charged and quantum swarms over 2000 iterations.**

I. Introduction

Particle Swarm Optimization (PSO) is a stochastic, population-based search algorithm based on the social behavior of birds in a flock. PSO was first introduced in 1995 by Kennedy and Eberhart [6], [13] and is known to perform well on a wide range of static optimization problems [9]. However, many real-world problems are dynamic in the sense that the optimal settings for a solution to a problem changes over time. This situation is often modelled in benchmark problems by allowing the optima of a function to move around the search space [16]. The objective of the PSO algorithm is then to track the moving global optimum over time.

PSO was first applied to dynamic problems by Eberhart and Shi [7]. Consequent studies identified the problem of diversity loss to be detrimental to the performance of PSO in dynamic environments [5]. Two specialized versions of PSO, known as charged swarms and quantum swarms, were developed to address this problem by managing the diversity of the swarms throughout the search process [2], [5]. By allowing only some of the particles to converge, the swarms are able to detect optimum jumps in the environment.

The dynamic heterogeneous particle swarm optimizer, introduced by Engelbrecht [8], is investigated in this paper as a potentially well-suited algorithm for optimization in dynamic environments. The algorithm allows particles to change their behaviors in response to stagnation. This mechanic could potentially allow the swarm to manage its diversity dynamically, as required by the given environment. To date, the aglorithm has only been tested in static environments.

The rest of this paper is structured as follows: section II gives a brief overview of the PSO algorithm. In section III the problems that PSO faces in dynamic environments are discussed in detail, while section IV explains how charges and quantum swarms modify the PSO algorithm in order to solve the problem of diversity loss. Section V gives an overview of heterogeneous PSO and explains why the dynamic heterogeneous PSO shows potential to perform well in dynamic environments. Section VI explains the experiments used for this study, while the results are reported and discussed in section VII. The paper is concluded in section VIII with a summary of the findings.

B.J. Leonard is with the Department of Computer Science, University of Pretoria, Pretoria, 0002, South Africa (phone: +27 12 420 5242; email: bleonard@cs.up.ac.za).

A.P. Engelbrecht is with the Department of Computer Science, University of Pretoria, Pretoria, 0002, South Africa (phone: +27 12 420 3578; email: engel@cs.up.ac.za).

A.B. van Wyk is with the Department of Computer Science, University of Pretoria, Pretoria, 0002, South Africa (phone: +27 12 420 5242; email: avanwyk@cs.up.ac.za).

II. Particle swarm optimization

The PSO algorithm maintains a population, or a *swarm*, of candidate solutions called *particles*. Each particle i has a position $\mathbf{x}_i$ in the d-dimensional search space. The position is updated at each time step by adding to it a velocity $\mathbf{v}_i$. In addition to a particle's position and velocity, each particle also keeps track of the best position $\mathbf{y}_i$ it has found so far, known as the particle's *personal best position*. The best position $\hat{\mathbf{y}}$ found by the swarm is known as the *global best position*.

Each particle in the swarm is attracted to its own personal best position as well as the global best position. For the modified PSO presented in [17], the velocity of a particle i at time step $t+1$ is calculated as

$$\begin{aligned}\mathbf{v}_i(t+1) = \omega\mathbf{v}_i(t) + c_1\mathbf{r}_1(t)[\mathbf{y}_i(t) - \mathbf{x}_i(t)] \\ + c_2\mathbf{r}_2(t)[\hat{\mathbf{y}}(t) - \mathbf{x}_i(t)]\end{aligned} \quad (1)$$

where ω is the inertia weight, c_1, c_2 are acceleration constants, and $r_{1j}(t)$, $r_{2j}(t)$ are random values sampled from $U(0,1)$ in each dimension $j = 1,\ldots,d$. The second and third terms of equation (1) are referred to as the *cognitive component* and *social component*, respectively.

The inertia weight controls how strongly a particle's previous velocity $\mathbf{v}_i(t)$ influences the calculation of the new velocity $\mathbf{v}_i(t+1)$. Similarly, c_1 and c_2 affect the significance of the cognitive and social components, respectively. The vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ give the algorithm its stochastic properties.

After a particle's velocity has been calculated, its position is updated using

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1). \quad (2)$$

The behavior that emerges from this model is that particles stochastically return to regions of the search space that are known to be successful.

III. PSO in dynamic environments

A dynamic environment for the purpose of this paper is a function f whose optima may change during the search process. Changes may occur in two ways: firstly, the position $\mathbf{x}^*$ of an optimum may shift over time in any or all dimensions; secondly, the value $f(\mathbf{x}^*)$ of an optimum may increase or decrease over time. The objective of the PSO algorithm in dynamic environments is no longer to merely find the global optimum, but also to track its movement over time.

Assuming the environmental changes occur at known intervals, there are two problems that need to be overcome in order for PSO to track a moving optimum over time [4]. The first problem is that of *outdated memory* because of a recent change in the environment. The second, more severe, problem is that of *diversity loss* due to swarm convergence.

Outdated memory occurs when the environment changes and a particle's personal best position $\mathbf{y}_i$ and its corresponding fitness value $f(\mathbf{y}_i)$ is no longer valid. This problem is typically solved by re-initializing a portion of the particles in the swarm at random positions in the search space and/or re-evaluating f at some or all of the particles' personal best positions whenever the environment changes [7], [11].

Diversity loss occurs when the swarm converges on an optimum [1], [19]. If the environment changes after the swarm has mostly converged, there is a good chance that the shifted optimum is no longer within the space covered by the swarm. If so, the low velocities of the particles will limit their ability to re-diversify and locate the new optimum. In fact, Blackwell and Bentley [5] showed that swarms in this situation are prone to oscillation along a line perpendicular to the optimum. This phenomenon is known as *linear collapse*.

One approach to solving the problem of diversity loss is to allow the swarm to re-diversify when the environment changes. However, it has been noted [4] that the time it takes for the population to re-diversify and converge on the new optimum is often detrimental to the swarm's performance.

Another way to solve the problem of diversity loss is to adapt the PSO algorithm to ensure that swarm diversity is maintained throughout the search process. Two algorithms that have been developed for this purpose are discussed in section IV.

IV. PSO with maintained diversity

This section describes two PSO algorithms that were designed to address the problem of diversity loss explained in section III.

A. PSO with charged particles

Charged particles were introduced by Blackwell and Bentley [5] and further improved upon by Backwell and Branke [3].

A charged particle is a particle i that possesses an electrostatic charge $Q_i > 0$. A particle with a charge $Q_i = 0$ is called a *neutral particle*. A *charged swarm* consists of only charged particles, while a swarm made up of a combination of charged and neutral particles is known as an *atomic swarm*, because the structure of such a swarm closely resembles classical pictures of the atom.

The effect of charged particles in a swarm is that diversity is maintained by means of a repulsive force between the charged particles, even when the neutral particles converge. This solves the problem of diversity loss explained in section III.

To introduce the notion of charge, an additional Coulomb repulsion force $\mathbf{a}_i$ is added to the velocity update rule in equation (1). The repulsion force for a particle i at time step t is given by

$$\mathbf{a}_i(t) = \sum_{k \neq i} \frac{Q_i Q_k}{(\delta_{ik}(t))^3} \boldsymbol{\delta}_{ik}(t), p_{core} < \delta_{ik}(t) < p \tag{3}$$

where $\boldsymbol{\delta}_{ik}(t) = \mathbf{x}_i(t) - \mathbf{x}_k(t)$, and $\delta_{ik}(t) = \|\mathbf{x}_i(t) - \mathbf{x}_k(t)\|$. Two charged particles will only repel each other if the separation between them is less than p. The lower cut-off p_{core} is to prevent the repulsion force from exploding to a large value when the particles are very close to each other. Equation (1) then changes to

$$\begin{aligned} \mathbf{v}_i(t+1) = \omega \mathbf{v}_i(t) + c_1 \mathbf{r}_1(t)[\mathbf{y}_i(t) - \mathbf{x}_i(t)] \\ + c_2 \mathbf{r}_2(t)[\hat{\mathbf{y}}(t) - \mathbf{x}_i(t)] + \mathbf{a}_i(t). \end{aligned} \tag{4}$$

Thus, a charged particle i will experience the repulsive force from all other charged particles $k \neq i$ within the shell $p_{core} < \delta_{ik}(t) < p$. Neutral particles have no charge and do not contribute to the sum in equation (3).

One major drawback of maintaining diversity in a swarm by means of charged particles is that the complexity of calculating the repulsive force between all pairs of particles is of $O(N^2)$, where N is the number of particles in the swarm.

B. PSO with quantum particles

To address the issue of quadratic complexity in PSO with charged particles, Blackwell and Branke [2] proposed the use of atomic swarms made up of standard particles (see section II) and *quantum particles*, as opposed to neutral particles and charged particles.

At each time step $t+1$, a quantum particle i moves to a random position in the vicinity of the swarm's global best position $\hat{\mathbf{y}}(t)$. The new particle position is drawn from a uniform distribution such that $\mathbf{x}_i(t+1) \in B(\phi_{cloud})(t)$, where $B(\phi_{cloud})(t)$ is a d-dimensional ball with radius ϕ_{cloud} around $\hat{\mathbf{y}}(t)$.

In this model the quantum particles form a *quantum cloud* around the standard particles. This eliminates the need to calculate repulsive forces between particles, but achieves the same goal of maintained swarm diversity as PSO with charged particles.

V. Heterogeneous PSO

A *heterogeneous PSO* (HPSO) differs from the normal PSO in that particles may follow different behaviors from each other [8]. That is, particles may use different velocity and position update rules. The atomic swarm configurations discussed in sections IV-A and IV-B are both forms of

HPSO, because the charged and quantum particles follow different update rules from the neutral and standard particles, respectively.

Various other forms of HPSO have also been proposed [14], [18], [20], [21]. In all cases, particles follow carefully crafted behaviors, either for the entire duration or at specific times of the search process. Consider for example the predator-prey PSO [18] and the division of labor PSO [21]:

In the predator-prey PSO model *predator particles* are attracted only to the global best position $\hat{\mathbf{y}}$. The other particles are *prey particles* whose velocities are updated using equation (1), but with an additional term to repel them from any nearby predator particles. A particle never changes its behavior at any stage in the search process. The behaviors of predator and prey particles are designed to promote exploration throughout the entire run of the algorithm.

In the division of labor PSO model particles may switch to a local search method near the end of the search process. Therefore the behavior of particles may change, but the new behavior is again very specific and is designed to promote exploitation in the final stages of the algorithm.

The problem with most HPSO configurations is that the behaviors assigned to particles are based on assumptions about the problem space that don't necessarily hold for a wide range of problems.

Engelbrecht [8] proposed a new form of HPSO that discards this notion of pre-determined behaviors. Instead, particles are able to adapt to their current situation by allowing any particle to select a new behavior from a pool of available behaviors when the particle stagnates. In this way, a particle that stagnates can literally attempt to escape stagnation by adopting a different behavior. This configuration is known as *dynamic HPSO* (dHPSO).

The ability of particles to switch from exploitative to exploratory behaviors and *vice versa* should allow the swarm to re-diversify and re-converge if an optimum shifts at some point during the search process. Recall from section III that re-diversification is required in order to solve the problem of tracking a moving optimum over time. The fact that re-diversification is an inherent feature of dHPSO makes it an appealing candidate for optimization in dynamic environments.

In this study a dHPSO algorithm is tested on a variety of dynamic functions. The experiments are explained in section VI and the results reported in section VII.

VI. Experimental Procedure

In this section, detail is given on the configurations of the three algorithms used in this study. In addition, the measurements used to report the results are described.

Unless stated otherwise, the inertia weight ω was assigned a value of 0.729844 and the acceleration constants c_1 and c_2 were set to 1.496180. All algorithms executed for 2000 iterations and a population size of 50 was used in all cases.

A. Charged PSO setup

A Charged PSO (CPSO) for the purpose of this paper is an atomic swarm consisting of charged particles and neutral particles. In all experiments a charge ratio of 0.5 was used, meaning that 50% of the particles were charged. The remaining particles were neutral. The charge magnitude was set to 16 to correspond with the charged swarms presented in [5]. The lower cut-off value p_{core} was set to 1, while p had a value of 30.

B. Quantum PSO setup

A quantum PSO (QPSO) for the purpose of this paper is an atomic swarm consisting of quantum particles and standard particles. Again, 50% of the particles in the swarm were quantum particles, while the rest were standard. The radius ϕ_{cloud} was set to 30.

C. Dynamic HPSO setup

In all the experiments done for this study the behavior pool was populated with the same five behaviours used in [8]. They are listed here again together with a short description and the parameters that were used for each of them:

- **Standard PSO behavior:** where a particle uses equations (1) and (2) to update its velocity and position, respectively. The cognitive acceleration constant c_1 was initially set to a value of 2.5 and linearly decreased to 0.5 throughout the search process. The initial value of the social acceleration constant c_2 was 0.5 and it was linearly increased to 2.5 throughout the search process. This behavior promotes exploration near the beginning of the search process and exploitation near the end of the search process.
- **Social PSO behavior:** where the cognitive component of equation (1) is discarded, so that a particle is only attracted to the global best position $\hat{\mathbf{y}}$. Equation (2) is still used for position updates. The social acceleration constant c_2 was set to a constant value of 2.5. This behavior facilitates exploitation, as all particles that follow this behavior at a time step t effectively become a single stochastic hill-climber.
- **Cognitive PSO behavior:** where the social component of equation (1) is discarded, so that a particle is only attracted to its own personal best position $\mathbf{y}_i$. Equation (2) is still used for position updates. The cognitive acceleration constant c_1 was set to a constant value of 2.5. This behavior aids exploration as all particles that follow this behavior become independent hill-climbers.
- **Bare bones PSO behavior:** The bare bones PSO was introduced by Kennedy [12]. The velocity update equation is replaced with a Gaussian distribution such that

$$v_{ij}(t+1) \sim N\left(\frac{y_{ij}(t)+\hat{y}_j(t)}{2}, \sigma^2\right) \quad (5)$$

where the variance $\sigma^2 = |y_{ij}(t) - \hat{y}_j(t)|$. Furthermore, the position update equation changes to

$$\mathbf{x}_i(t+1) = \mathbf{v}_i(t+1). \quad (6)$$

Note that the velocity of a particle is no longer added to the particle's current position. Rather, the velocity is the particle's new position. This behavior results in exploitation of the point in the middle of a particle's personal best position and the global best position. Initially, this behavior will enhance exploration as the particles' personal best positions are distributed throughout the search space. As the personal best positions converge towards the global best, the focus shifts towards exploitation.

- **Modified Bare bones PSO behavior:** Kennedy [12] also proposed a modified version of the bare bones PSO where equation (5) changes to

$$v_{ij}(t+1) = \begin{cases} y_{ij}(t) & \text{if } U(0,1) < 0.5 \\ \varphi & \text{otherwise.} \end{cases} \quad (7)$$

where

$$\varphi \sim N\left(\frac{y_{ij}(t)+\hat{y}_j(t)}{2}, \sigma^2\right). \quad (8)$$

This behavior exhibits better exploration and exploitation than the original bare bones PSO as particles focus on their personal best positions 50% of the time.

Stagnation detection was implemented by monitoring the personal best positions of all particles. Whenever a particle's personal best position did not change for ten consecutive iterations, the particle was assumed to have stagnated and forced to select a new behavior from the behavior pool. For the purpose of this paper selection was done randomly.

D. Dynamic Environments setup

The environments used for this paper were generated using the DF1 function generator, presented by Morrison and De Jong [16]. The generator can simulate environments with any number of static and dynamic optima. The positions and heights of optima change at pre-defined intervals with adjustable severity.

Each of the three algorithms were tested on 30 unimodal and 30 multimodal environments over 30 samples. In each case 15 environments had high severity changes, while the remaining 15 environments changed with low severity. The interval of change was fixed at 100 iterations. At each change the algorithm responded by randomly selecting and re-initializing 50% of the particles. In addition, the personal best positions of all the particles were re-evaluated to reflect their true values.

The function domain for all functions was [-50, 50]^I, where the dimensionality $I = 10$. For multimodal functions, ten optima were used. All optima were dynamic and could move around the entire domain. Peak heights oscillated independently in the range [10, 50].

For each of the 60 environments, the random seed for the generator was fixed so that the three algorithms could be tested in identical circumstances.

E. Measurements

The following measurements were recorded at each iteration of the experiments:

- **Diversity**: taken as the average euclidean distance around the center of the swarm [15]:

$$D = \frac{1}{|S|}\sum_{i=1}^{|S|}\sqrt{\sum_{j=1}^{I}(x_{ij} - \bar{x}_j)^2} \quad (9)$$

  where $|S|$ is the population size and $\bar{\mathbf{x}}$ is swarm center.
- **Error**: taken as the difference between the global best fitness $f(\hat{\mathbf{y}})$ and the global maximum value (assuming a maximization problem) at the current iteration.
- **Collective mean error**: taken as the average of the error measurements over all iterations performed so far.

VII. Results and discussion

dHPSO obtained lower collective mean errors than both CPSO and QPSO in almost all test cases. The lower error of dHPSO is attributed to its ability to manage the diversity of the swarm as required by the environment. Figures 1 and 2 show the diversity measurements for typical unimodal environments with low and high change severity, respectively.

In environments with low change severity CPSO and QPSO exhibited very similar behaviors regarding the diversity of the swarms. However, dHPSO showed a much higher spike in its diversity whenever a change in the environment occurred. This is an unexpected result that deserves close examination. The partially maintained diversity in CPSO and QPSO are features designed to promote exploration in a radius around the global best position. While this is necessary, the results presented here would suggest that it in fact hampers the ability of the swarms to effectively re-diversify in response to a change in the environment.

Consider an atomic swarm with 50 particles in which the exploring particles are scattered around a highly converged nucleus. The scattered particles cover a pre-defined radius around the nucleus, and thus only a subset (say 30%) of the search space. Now, when a change in the environment occurs, the algorithm responds by uniformly selecting 25 (50%) of the particles for re-initialization. Since the selection process is uniform, it is reasonable to assume that roughly 12 of the selected particles will have formed part of the nucleus, while the rest of the selected particles were exploring the space around the nucleus. When the selected particles are then positioned randomly throughout the search space, it is not taken into account that the space around the nucleus is already covered by the exploring particles that were not selected for re-initialization. The result is that a portion of the selected particles are now placed back into the radius around the nucleus, leaving only the remaining particles to fully explore the new environment.

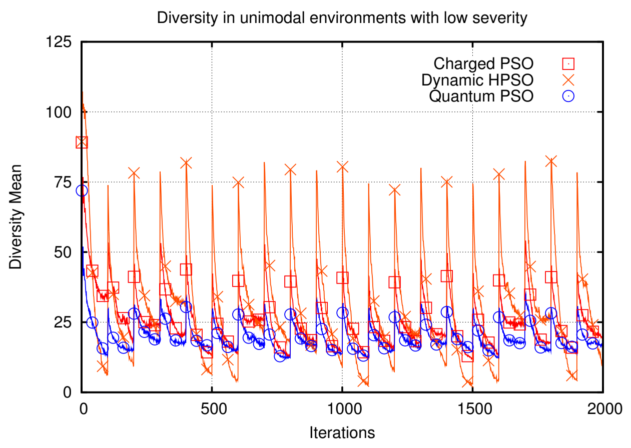

Fig. 1. Diversity graph for CPSO, dHPSO and QPSO in unimodal environments with low change severity.

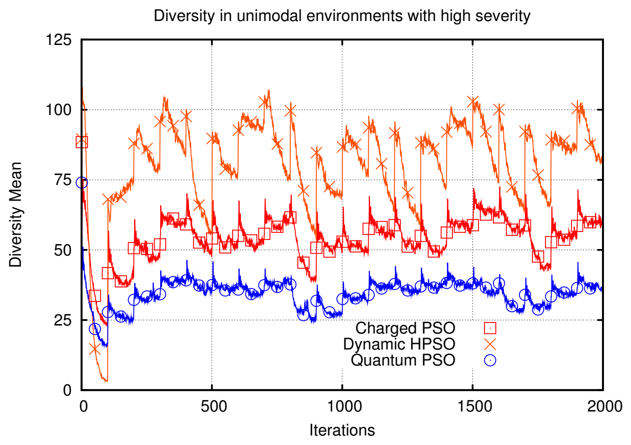

Fig. 2. Diversity graph for CPSO, dHPSO and QPSO in unimodal environments with high change severity.

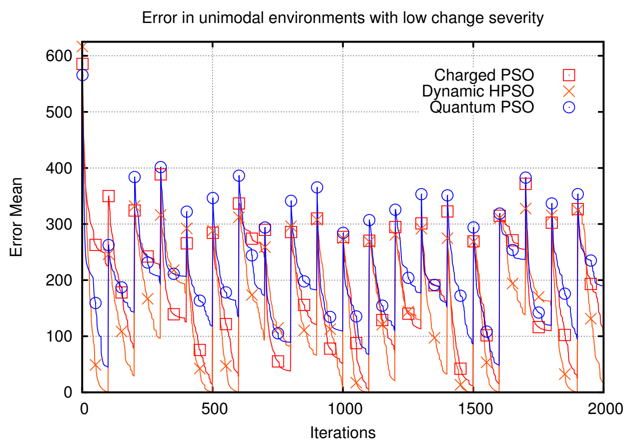

Fig. 3. Error graph for CPSO, dHPSO and QPSO in unimodal environments with low change severity.

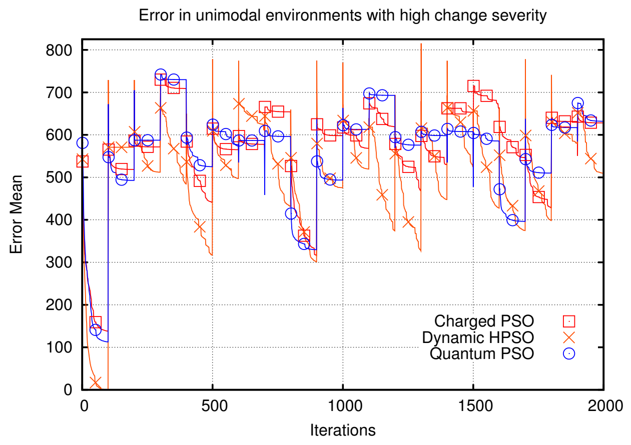

Fig. 4. Error graph for CPSO, dHPSO and QPSO in unimodal environments with high change severity.

In dHPSO, all the particles in the swarm converge. Therefore, all the particles that are selected for re-initialization are taken from more or less the same position (depending on the degree of convergence achieved) and placed at random positions in the search space. The probability that any particle is placed back into the same region that it was taken from is significantly lower for dHPSO than for CPSO of QPSO. This explains the higher spike in the diversity of dHPSO.

To illustrate the problem, the swarms are visualized in figures 5, 6, 7 and 8. In each figure the state of the swarm is visualized by a set of lines that pass through specific points on each of ten vertical lines. The non-vertical lines are visual representations of the particles' positions in the search space. The vertical lines are scaled segments of the ten dimensions (axes) of the search domain. By plotting a particle's position x_j in each of the ten dimensions on the corresponding vertical line and connecting those points, the particle's position in the swarm can be accurately visualised. The figures were generated using FluxViz, developed by Franken [10].

Figures 5 and 6 show the swarm states at iteration 100 of dHPSO and QPSO, respectively. Visualisations of CPSO are not presented as they are similar to QPSO. In dHPSO all the particles are highly converged in all dimensions. The small scale of the vertical lines should be noted as it may give the illusion of a partially diverged swarm. In QPSO the nucleus of the swarm is clearly visible as the converged particles form a thick black line more or less in the center of the swarm. The quantum particles are scattered in a radius around the nucleus as expected. Figures 7 and 8 show the same swarms one iteration later, after they have responded to a change in the environment. Note how the area around the nucleus of QPSO is highly populated and that the rest of the search space is covered by only a small number of particles. However, the re-initialized particles of dHPSO are much more evenly distributed throughout the search space. It should be noted that the converged particles in dHPSO are so close together that they are represented by a single line at this scale.

The fact that the particles in dHPSO are more evenly

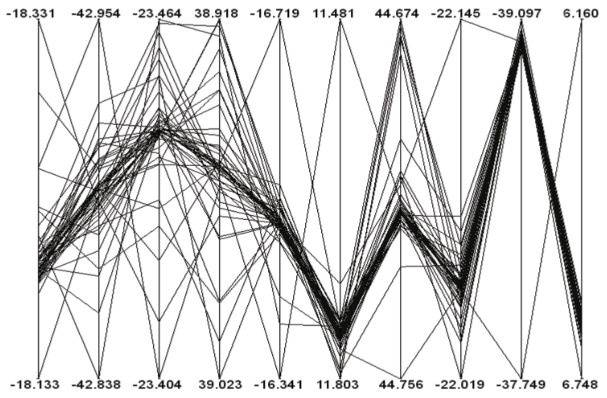

Fig. 5. Particle positions in a highly converged dHPSO. Note the small scale of the vertical line segments.

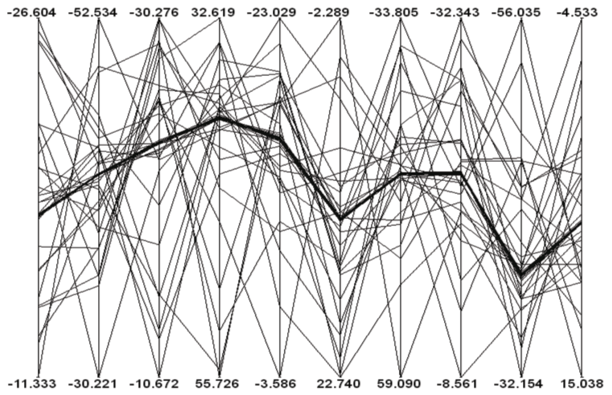

Fig. 6. Particle positions in an atomic swarm with a highly converged nucleus.

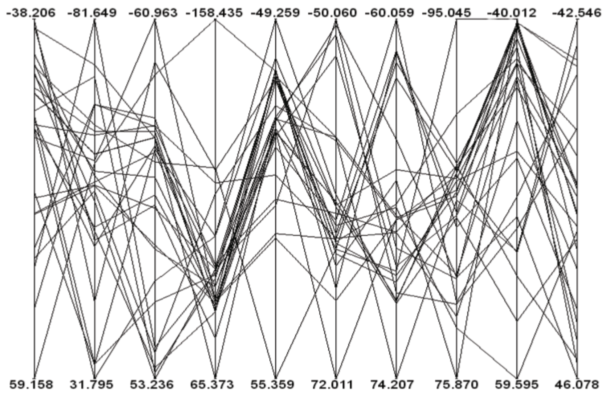

Fig. 7. Particle positions in a dHPSO immediately after the swarm responded to a change in the environment.

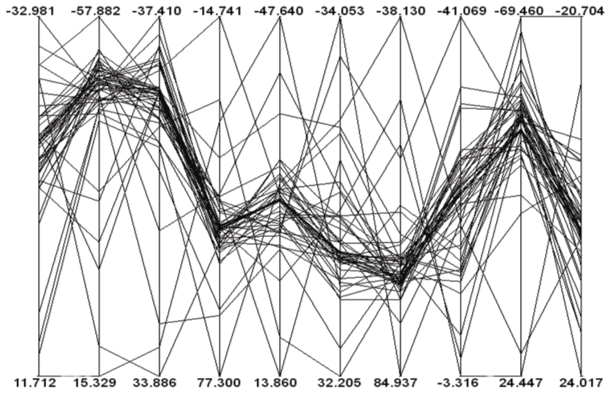

Fig. 8. Particle positions in an atomic swarm immediately after the swarm responded to a change in the environment.

dispersed throughout the search space improves the swarm's ability to explore the changed environment and find the shifted optimum. Figure 3 shows how dHPSO was the only algorithm to reach the optimum on several occasions.

Interestingly, dHPSO converged at a much faster rate (in many cases to a lower average diversity) than the other two algorithms after the re-diversification process. The reason for this is that a portion of the particles in CPSO or QPSO is unable to converge due to repulsion (CPSO) or random positioning (QPSO), slowing down the rate of convergence and preventing the average diversity from dropping below a certain value. With dHPSO, as noted before, all the particles eventually converge to $\hat{\mathbf{y}}$ resulting in a smaller swarm diameter and thus a lower diversity before a change in the environment occurs.

In the case of environments with high change severity, presented in figure 2, all three algorithms maintained a higher average diversity throughout the search process. The highest average diversity was maintained by dHPSO. Especially note the behavior of dHPSO between iterations 0 and 200.

Initially dHPSO converged at a much faster rate than the other two algorithms. When the first change in the environment occurred, the swarm re-diversified as expected. But then, contrary to the convergent behavior of CPSO and QPSO, the diversity of dHPSO continued to increase. This behavior is elegantly explained by the ability of particles to change their behaviors in response to stagnation. Recall that roughly 50% of the particles in dHPSO is converged at this point. In an environment with low change severity, these particles could simply follow the optimum while staying converged, presumably following *social* behaviors. However, in an environment with high change severity, the optimum may move too far for the particles to simply follow. As a result, the particles stagnate and begin to change their behaviors periodically. In this situation *cognitive* behaviors will produce better results, causing the diversification observed in figure 2. Throughout the remainder of the search process a high diversity was maintained. This behavior is justified by the results presented in figure 4 during the same iterations. Note how the partially converged states of the swarms prevented them from significantly improving their error measurements after the first change in the environment occurred. dHPSO was the only algorithm that showed the ability to appropriately respond to this situation.

It is important to mention at this stage that the diversity of charged and quantum swarms are dependent on tunable parameters (p_{core}, p and Q for CPSO and ϕ_{cloud} for QPSO). However, dHPSO is able to adapt to its environment without the need for external parameter tuning. This statement is supported by results that showed how charged and quantum particles were forced out of the search domain when the domain was lowered to $[-1,1]^{10}$ without changing the parameter settings of the algorithms. dHPSO particles were able to adapt to this change and stay inside the search domain.

The diversity graphs for multimodal environments are similar to those of the unimodal environments and are not presented here. The two major differences are however discussed.

Firstly, in multimodal environments with low change severity, dHPSO did not converge to a lower diversity than CPSO or QPSO. This again shows the ability of dHPSO to maintain diversity in the swarm when needed.

Secondly, dHPSO failed to reach the global optimum in multimodal environments with low change severity as it did on several occasions in the unimodal environments. This may indicate that particles get trapped in local optima and fail to respond to the error before a change in the environment occurs. A longer period between environmental changes may allow particles enough time to escape a local optimum. However, this scenario was not tested.

Tables I and II report the collective mean errors (ξ) of all three algorithms for each of the 60 environments at the end of the 2000 iterations. The standard deviation (σ) on each of the respective collective mean errors is also reported. In each case, the error value of the best performing algorithm is highlighted in boldface.

The standard deviation on the collective mean error of dHPSO was significantly lower than for QPSO and CPSO in all cases, indicating that dHPSO is much more consistent in its ability to track a moving optimum. The different ways in which the three algorithms maintain swarm diversity could give a possible explanation for this result.

In CPSO the electrostatic repulsion force between particles is completely decoupled from the problem at hand. That is, the repulsion force is an external force brought in to ensure maintained diversity regardless of the given environment. This may result in different swarms behaving inconsistently in the same environment as a result of being initialized differently (which was the case for each of the 30 samples run on each problem).

Consider, for example, two charged particles that are moving towards the global optimum from different directions after a recent change in the environment. When the particles get close to the optimum they will start to repel each other, preventing either of them from reaching the optimum. In a different charged swarm in the same environment, one of these particles may be absent, because it was initialized at a different position in the search space. The remaining particle is now able to reach the optimum.

A similar argument can be made for QPSO. The fact that a very important problem-solving mechanic in CPSO and QPSO pays no attention to the actual problem can potentially hamper their ability to produce consistent results.

In the case of dHPSO, the swarm responds solely to events and situations in the given environment. Therefore every action (movement or change of behavior) a particle takes, addresses a time- and problem-specific issue, ensuring consistent results produced by different swarms in the same environment.

VIII. Conclusion and future work

In this paper a dHPSO algorithm was tested on a variety of dynamic problems. The performance of the algorithm was compared to that of CPSO, as well as QPSO. The latter two algorithms were developed specifically for optimization in dynamic environments.

The dHPSO was shown to produce a lower collective mean error than both CPSO and QPSO on the vast majority of test problems over 2000 iterations. It was also observed that dHPSO re-diversifies better than CPSO and QPSO in response to a change in the environment; a result that was attributed to the discovery that maintained diversity in CPSO and QPSO, while assisting exploration, interferes with the ability of the swarms to effectively re-diversify. Additionally, it was observed that the dHPSO is able to re-diversify not only in response to, but also in between changes in the environment as a result of the particles being able to change their behaviors when they repeatedly fail to find better solutions. The inherent re-diversification properties of dHPSO allowed the swarm to overcome the problem of diversity loss and to successfully track a moving optimum over time.

Future work will include the testing of dHPSO on other dynamic benchmark environments. An adaptive dHPSO will also be investigated, where the different behaviors will be rewarded when they produce good results. A particle will then probabilistically, rather than randomly, select behaviors from the behavior pool, favoring those behaviors that have performed well in past iterations.

References

[1] M. Blackwell. Particle swarms and population diversity. *Soft Comput.*, 9(11):793–802, 2005.

[2] T. Blackwell and J. Branke. Multi-swarm optimization in dynamic environments. *Applications of Evolutionary Computing*, pages 489–500, 2004.

[3] T. Blackwell and J. Branke. Multiswarms, exclusion, and anti-convergence in dynamic environments. *Evolutionary Computation, IEEE Transactions on*, 10(4):459 –472, aug. 2006.

[4] Tim Blackwell. Particle swarm optimization in dynamic environments. In Shengxiang Yang, Yew-Soon Ong, and Yaochu Jin, editors, *Evolutionary Computation in Dynamic and Uncertain Environments*, volume 51 of *Studies in Computational Intelligence*, pages 29–49. Springer Berlin / Heidelberg, 2007.

[5] TM Blackwell and P.J. Bentley. Dynamic search with charged swarms. In *Proceedings of the genetic and evolutionary computation conference*, pages 19–26. Citeseer, 2002.

[6] R.C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science*, volume 43. New York, NY, USA: IEEE, 1995.

[7] R.C. Eberhart and Yuhui Shi. Tracking and optimizing dynamic systems with particle swarms. volume 1, pages 94 –100 vol. 1, 2001.

[8] Andries Engelbrecht. Heterogeneous particle swarm optimization. In Marco Dorigo, Mauro Birattari, Gianni Di Caro, Ren Doursat, Andries Engelbrecht, Dario Floreano, Luca Gambardella, Roderich Gro, Erol Sahin, Hiroki Sayama, and Thomas Sttzle, editors, *Swarm Intelligence*, volume 6234 of *Lecture Notes in Computer Science*, pages 191–202. Springer Berlin / Heidelberg, 2010.

[9] Andries P. Engelbrecht. *Computational Intelligence: An Introduction*. Wiley Publishing, 2007.

TABLE I
RESULTS FOR UNIMODAL ENVIRONMENTS

Low Change Severity							High Change Severity						
	CPSO		QPSO		dHPSO			CPSO		QPSO		dHPSO	
Env	ξ	σ	ξ	σ	ξ	σ	**Env**	ξ	σ	ξ	σ	ξ	σ
1	141.94	99.45	198.15	120.44	**112.27**	28.13	1	553.60	32.76	552.16	27.19	**453.43**	31.72
2	132.23	30.90	215.66	96.99	**110.35**	16.33	2	588.74	36.12	581.29	29.63	**562.07**	37.45
3	110.91	63.94	153.46	123.87	**94.83**	15.62	3	528.92	26.79	536.42	25.77	**455.09**	23.41
4	401.54	105.54	484.73	119.60	**194.32**	29.05	4	575.62	16.60	564.24	9.21	**553.96**	38.56
5	165.48	26.56	189.20	48.67	**133.13**	11.60	5	563.19	20.11	552.47	20.60	**499.91**	28.91
6	324.84	112.05	399.63	68.25	**176.36**	28.58	6	535.07	32.26	**513.77**	5.66	536.10	30.69
7	175.74	120.65	186.21	145.20	**109.04**	18.08	7	559.20	31.86	560.72	30.43	**492.43**	33.67
8	473.10	115.48	502.76	123.06	**202.32**	32.06	8	544.24	24.78	549.30	29.24	**440.65**	25.89
9	379.69	108.05	504.49	82.55	**212.48**	41.17	9	580.20	23.67	568.85	25.89	**522.52**	28.27
10	373.20	102.66	454.27	81.89	**191.45**	40.01	10	583.15	26.47	570.16	17.80	**549.66**	46.26
11	129.22	68.53	211.65	163.79	**116.75**	22.70	11	561.11	30.81	547.68	26.69	**542.75**	39.73
12	306.82	66.64	387.18	99.66	**185.63**	30.75	12	533.99	30.15	**521.52**	16.27	532.34	29.23
13	**90.27**	38.64	165.02	113.09	99.34	20.06	13	567.50	25.80	558.34	28.19	**541.35**	33.10
14	380.05	113.43	448.10	100.62	**196.93**	47.61	14	559.68	26.06	554.70	26.95	**512.14**	30.40
15	359.02	107.83	468.60	86.02	**214.46**	33.57	15	597.23	31.45	594.04	30.21	**541.75**	28.72

TABLE II
RESULTS FOR MULTIMODAL ENVIRONMENTS

Low Change Severity							High Change Severity						
	CPSO		QPSO		dHPSO			CPSO		QPSO		dHPSO	
Env	ξ	σ	ξ	σ	ξ	σ	**Env**	ξ	σ	ξ	σ	ξ	σ
1	354.14	137.14	428.64	180.80	**281.37**	77.91	1	705.02	46.10	719.47	46.13	**634.15**	34.83
2	372.96	136.40	468.81	200.88	**289.57**	82.39	2	734.83	52.60	737.02	61.12	**622.78**	29.78
3	190.52	68.08	230.48	105.55	**149.34**	30.59	3	363.80	31.67	368.03	31.28	**317.44**	20.50
4	171.69	60.52	226.40	87.70	**146.08**	30.37	4	348.82	26.83	343.02	35.80	**319.06**	14.14
5	150.33	76.94	205.57	97.13	**147.35**	37.35	5	360.98	23.87	363.26	38.71	**311.78**	19.12
6	198.09	59.14	221.08	89.29	**166.00**	34.29	6	368.23	21.00	382.25	27.77	**340.58**	18.57
7	179.68	82.61	240.66	85.75	**149.43**	35.71	7	354.90	21.00	361.06	27.11	**309.13**	18.65
8	167.54	80.56	239.13	88.38	**147.87**	33.45	8	349.92	21.73	354.93	26.62	**310.61**	16.43
9	215.36	66.96	258.10	98.84	**156.27**	45.75	9	363.38	22.31	369.06	20.21	**313.12**	18.16
10	172.38	61.11	212.28	83.57	**156.28**	39.17	10	372.69	22.94	392.24	20.97	**343.54**	16.07
11	143.87	57.59	198.45	85.71	**127.03**	25.31	11	376.21	33.24	396.41	36.56	**344.65**	23.66
12	149.55	60.66	169.53	69.47	**124.26**	29.18	12	357.28	23.43	371.94	31.08	**324.52**	18.60
13	**131.92**	58.06	167.24	79.34	139.37	32.59	13	387.41	29.05	397.83	23.38	**337.12**	17.68
14	169.41	60.34	236.25	96.76	**151.59**	36.77	14	385.18	16.91	386.68	33.71	**341.67**	21.00
15	183.76	71.28	226.86	77.40	**170.89**	35.34	15	361.89	21.16	376.92	29.98	**328.98**	18.49

[10] N. Franken. Visual exploration of algorithm parameter space. pages 389 –398, may. 2009.

[11] Xiaohui Hu and Russell C. Eberhart. Adaptive particle swarm optimization: Detection and response to dynamic systems. In *In Proceedings of the IEEE Congress on Evolutionary Computation, CEC2002. IEEE*, pages 1666–1670. Press, 2002.

[12] J. Kennedy. Bare bones particle swarms. pages 80 – 87, apr. 2003.

[13] J. Kennedy, R.C. Eberhart, et al. Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks*, volume 4, pages 1942–1948. Perth, Australia, 1995.

[14] T. Krink and M. Løvbjerg. The lifecycle model: combining particle swarm optimisation, genetic algorithms and hillclimbers. *Parallel Problem Solving from NaturePPSN VII*, pages 621–630, 2002.

[15] T. Krink, J.S. Vesterstrom, and J. Riget. Particle swarm optimisation with spatial particle extension. volume 2, pages 1474 –1479, 2002.

[16] R.W. Morrison and K.A. De Jong. A test problem generator for non-stationary environments. volume 3, page 2053 Vol. 3, 1999.

[17] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 69–73. IEEE, 2002.

[18] A. Silva, A. Neves, and E. Costa. An empirical comparison of particle swarm and predator prey optimisation. *Artificial Intelligence and Cognitive Science*, pages 1–45, 2002.

[19] F. van den Bergh. *An Analysis of Particle Swarm Optimizers*. PhD thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

[20] F. van den Bergh and AP Engelbrecht. A new locally convergent particle swarm optimizer. In *Proceedings of the IEEE international conference on systems, man and cybernetics*, volume 3, pages 6–12, 2002.

[21] J.S. Vesterstrøm, J. Riget, and T. Krink. Division of labor in particle swarm optimisation. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002), Honolulu, Hawaii*. Citeseer, 2002.