# A Radius-Free Quantum Particle Swarm Optimization Technique for Dynamic Optimization Problems

Kyle Robert Harrison
Department of Computer Science
University of Pretoria
Pretoria, South Africa

Beatrice M. Ombuki-Berman
Department of Computer Science
Brock University
St. Catharines, ON, Canada
bombuki@brocku.ca

Andries P. Engelbrecht
Department of Computer Science
University of Pretoria
Pretoria, South Africa
engel@cs.up.ac.za

***Abstract*—The quantum particle swarm optimization (QPSO) algorithm is a variant of the traditional particle swarm optimization (PSO) algorithm aimed at solving dynamic optimization problems. Some particles in the QPSO algorithm are selected as "quantum" particles and the positions of these particles are sampled, using some probability distribution, within a radius (i.e., a hypersphere) around the global best position while the remainder of particles follow standard PSO behaviour. The exploration and exploitation of the QPSO algorithm is heavily influenced by the probability distribution used as well as the size of the quantum radius. However, the best probability distribution and radius size are both problem and environment dependent. This work proposes using a parent centric crossover (PCX) operator to generate the positions of quantum particles, thereby removing the need for radius and probability distribution parameters completely. Two variants are proposed and results indicate that both variants are superior to QPSO, especially in environments exhibiting high temporal severity.**

## I. Introduction

Many real-world problems exhibit dynamic behaviour where the optimal solution changes over time. These dynamic optimization problems can exhibit changes in the location of the optima, the optimal fitness value, or both. Inherently, dynamic problems are more challenging for an optimizer as finding the optima is no longer sufficient. Rather, in a dynamic environment the optimizer must also adequately respond to an environmental change or its performance can be greatly hindered [1]. Moreover, optimizers which exhibit fast convergence may not have sufficient diversity to respond to an environmental change if the changes occur infrequently [1].

The particle swarm optimization (PSO) algorithm [2] is a stochastic, population-based optimization technique inspired by the flocking behaviour of birds. The agents, referred to as particles, in the PSO algorithm exhibit flight-like movements through a search space controlled by the iterative calculation and application of a velocity vector. Their movement direction is premised on two behaviours: movement towards their own personal best position and the best position found within their neighbourhood. While the traditional PSO algorithm has seen many successes in static environments (i.e., environments which do not change) [2], [3], [4], it suffers from two major drawbacks when faced with dynamic optimization problems. Firstly, outdated memory occurs when an environmental change occurs thereby (potentially) invalidating the personal best position of particles. The second major issue is the lack of diversity, which occurs when the particles have converged, and prevents an adequate response to an environmental change. Both aforementioned issues cause the performance of PSO to be greatly hindered in dynamic environments [5].

To address the limitations of PSO in dynamic environments, the quantum particle swarm optimization (QPSO) [6] was proposed as an inherently dynamic variant. In the QPSO algorithm, a number of particles are designated as "quantum" particles and forego the traditional velocity-vector position update. Rather, their positions are sampled probabilistically around the global best position. This quantum position update scheme prevents total convergence due to the base level of diversity provided by the probabilistic locations of the quantum particles. Traditionally, the particle positions have been sampled uniformly around the best position using a uniform distribution and are constrained by a radius. However, the performance of QPSO can be improved through the use of alternative probability distributions [7], [8].

Irrespective of the probability distribution, the size of the radius greatly influences the performance of the QPSO algorithm [5], [7], [8]. Furthermore, the probability distribution can influence the exploration and exploitation capabilities of the QPSO algorithm by governing the density of quantum particles which are positioned near the global best versus those which are positioned closer to the edge of the radius. However, this strategy is limited in that the diversity is implicitly controlled by the size of the radius; quantum particles can only be positioned within the radius-governed hypersphere around the global best position. Thus, the radius parameter needs to be appropriately tuned relative to the probability distribution to optimize performance of the QPSO algorithm. Suggestions have been proposed which claim that the radius size should be approximately half of the environmental change size to retain sufficient diversity [5], [7]. While such suggestions are beneficial in many cases, it becomes inapplicable to problems where the change severity is unknown *a priori*. Moreover, this suggestion is made with the assumption that the environmental changes are of constant magnitude.

This paper proposes a QPSO variant which removes the need for a radius parameter completely by generating the positions of quantum particles using a parent centric crossover (PCX) operator. While the radius parameter has been removed from QPSO, replacing the position update with the PCX operator comes with the caveat that two new parameters, namely the deviations used in the PCX operation (see Section II-E), have been introduced. However, the replacement of the

position update with the PCX operator also removes the need for a probability distribution to be specified. If one views the probability distribution as a parameter, which it rightly should be, then the total number of parameters remains the same. Additionally, the probability distribution employed may require further parametrization, causing the number of parameters in the proposed variant to actually be less than the original QPSO algorithm. Although previous works have employed PCX within PSO [9], [10], [11], these works entirely replaced the particle position update with the PCX operation. This paper uses the PCX operator primarily for diversity maintenance, thus only quantum particles make use of the PCX operation. Moreover, previous works were limited to static environments while this paper focuses on dynamic environments.

The remainder of this paper is structured as follows: Section II provides the necessary background information for the proposed algorithm introduced in Section III. The experimental procedure is outlined in Section IV while the results are presented in Section V. Finally, concluding remarks are given in Section VI.

## II. Background

This section provides the necessary background information for the remainder of the paper.

### A. Dynamic Environments

A dynamic environment is, generally speaking, an environment which changes over time. While this definition is quite vague, a great deal of research effort has been devoted to both describe and classify dynamic environments based on how frequently and severely the environment changes [12], [13], [14], [15]. However, many of the classification schemes only consider certain types of dynamism. Duhain and Engelbrecht [15] point out three benefits to having a more complete classification of dynamic environments. Firstly, such a classification would allow for better inferences about algorithmic strengths and weakness to be made based on their performance. Secondly, this would allow for reasoning about algorithmic performance on an unseen problem once the environmental type of the problem is known. Finally, algorithms can exhibit observable and predictable characteristics when faced with certain types of dynamism and this can be indicative of the environment type. Therefore, more complete classification facilitates the gathering of additional information about the environment based on previous experiences.

Recently, a generalized framework for classifying dynamic environments based on four environment classes, namely *quasi-static*, *progressive*, *abrupt*, and *chaotic*, was proposed [15]. Each class differs in the relative temporal and spatial severities. On one extreme, quasi-static environments observe infrequent, small changes while the other extreme, chaotic environments, observe frequent and large changes. Note that the boundaries between the environments are fuzzy and can not be rigorously defined [15].

### B. Moving Peaks Benchmark

The moving peaks benchmark (MPB) [16] is a dynamic environment generator extensively used in the literature (see, for example, [5], [7], [8], [15]) due to its flexibility. MPB landscapes are constructed by defining a number of peaks consisting of a position, height, and width. Environmental changes occur by varying the peak parameters over time. The temporal severity of the problem is thus defined by the change interval relative to the number of evaluations while the spatial severity is determined by the user-supplied parameters.

When an environmental change occurs, a velocity vector $\mathbf{v}_i$, with fixed length $s$ (a user-supplied parameter), is determined for each peak $i$ in the landscape by

$$\mathbf{v}_i = \frac{s}{|\mathbf{r} + \mathbf{v}_i|} \left((1-\lambda)\mathbf{r} + \lambda \mathbf{v}_i\right) \tag{1}$$

where $\mathbf{r}$ is a vector of uniformly random variables normalized to length $s$ and $\lambda \in [0, 1]$ is a correlation coefficient for the peak movements. When $\lambda = 0$, each peak moves in a random direction (i.e., successive movements are not correlated) and when $\lambda = 1$, the movement direction of each peak is highly correlated with it's previous movement direction. Additionally, user-supplied height and width severity parameters define scaling coefficients for the peak height and width, respectively. If a velocity vector happens to take a peak outside of the problem domain, components of the velocity vector are negated as necessary to ensure the peak stays within the bounds.

### C. Particle Swarm Optimization

Introduced by Kennedy and Eberhart [2], the PSO algorithm was inspired by a simple model of the social dynamics of a flock of birds. The PSO algorithm is a population-based, stochastic optimization algorithm targeted towards real-valued, static problems. The movement of each agent (referred to as a particle) is premised on two simple behaviours: move towards the best position in a neighbourhood and move towards its own personal best position.

The movement of each particle through the search space is governed by the iterative process of calculating and applying a velocity vector to the current position of the particle. The velocity for a particle is calculated as

$$\begin{aligned} v_{ij}(t+1) = \omega v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] \\ + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \end{aligned} \tag{2}$$

where $v_{ij}(t)$ is the velocity of particle $i$ in dimension $j$ at time $t$. The position of the particle in dimension $j$ is given by $x_{ij}$, $\omega$ denotes the inertia coefficient, while $c_1$ and $c_2$ represent the cognitive and social coefficients, respectively. A stochastic component is added via the random constants, $r_{1j}$ and $r_{2j} \sim U(0, 1)$. Finally, $y_{ij}(t)$ and $\hat{y}_j(t)$ denote the personal and neighbourhood best positions in dimension $j$, respectively. Particle positions are then updated according to

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1). \tag{3}$$

Originally proposed for static environments, the PSO algorithm is known to have two major drawbacks when employed in dynamic environments, namely outdated memory and diversity loss [5]. The outdated memory issue refers to when a particle's fitness and personal best position may be misleading, or incorrect, due to an environmental change. The other issue, diversity loss, occurs when the particles have exhibited convergence and all particles have converged to the same region. In a dynamic environment, lack of diversity may

prevent an adequate response to an environmental change as the low velocities of the particles may inhibit the optimum tracking capabilities of the PSO algorithm.

### D. Quantum Particle Swarm Optimization

Quantum PSO [6] was introduced as a dynamic variant of PSO and was inspired by a quantum model of an atom. In this model, electron positions orbit a nucleus in a non-deterministic fashion. Thus, a proportion of particles in the QPSO algorithm are designated as "quantum" particles, and these particles do not update their positions using a velocity vector. Rather, the positions of quantum particles are sampled from a probability distribution centred around the global best position,

$$\mathbf{x}_i(t+1) = d(\hat{\mathbf{y}}(t), r_{cloud}) \tag{4}$$

where $d$ is a probability distribution and $r_{cloud}$ is the quantum radius. The remainder of particles, referred to as neutral particles, update their positions using the standard position update as given in Equation (3).

### E. Parent Centric Crossover

The PCX operator [17] was proposed as a multi-parent crossover which biases the offspring towards the parents, specifically one parent which is selected for mutation. Offspring are generated by selecting $n_\mu \geq 3$ parents and computing their centre of mass, $\overline{\mathbf{x}}(t)$. One of the $n_\mu$ parents are selected for mutation, denoted $\mathbf{x}_i$, and a direction vector is calculated as

$$\mathbf{d}_i(t) = \mathbf{x}_i(t) - \overline{\mathbf{x}}(t)$$

where $\overline{\mathbf{x}}(t)$ is the centre of mass,

$$\overline{\mathbf{x}}(t) = \sum_{l=1}^{n_\mu} \mathbf{x}_l(t),$$

of the parents. For each of the remaining $n_\mu - 1$ parents, perpendicular distances to the line $\mathbf{d}_i(t)$ are calculated, $\delta_l$, and their average, $\overline{\delta}$, is taken. Offspring are then generated according to

$$\tilde{\mathbf{x}}_i(t) = \mathbf{x}_i(t) + N(0,\sigma_1^2)|\mathbf{d}_i(t)| + \sum_{l=1, l\neq i}^{n_\mu - 1} N(0,\sigma_2^2)\overline{\delta}\mathbf{e}_l(t) \tag{5}$$

where $\overline{\delta}\mathbf{e}_l(t)$ are the $n_\mu - 1$ orthonormal basis vectors that space the subspace perpendicular to $\mathbf{d}_i(t)$ and $\sigma_1$ and $\sigma_2$ are deviations of two Gaussian distributions. Thus, offspring are generated by mutating the selected parent based on the distance of the selected parent to the centre of mass and the distance that each other parent is from the the direction vector, $\mathbf{d}_i(t)$.

## III. Quantum Particle Swarm Optimization using Parent Centric Crossover

To alleviate the issue of having to determine an appropriate probability distribution and corresponding radius size, this paper proposes a QPSO algorithm that uses a PCX operator to generate the positions of quantum particles, namely the QPSO-PCX algorithm. For each quantum particle, the new position is determined by the lone offspring of a PCX operation using the global best position, the particle's personal best position, and a randomly generated position in the search space as the parents. More formally,

$$\mathbf{x}_i(t+1) = \tilde{\mathbf{x}}_i(t) \tag{6}$$

where $\tilde{\mathbf{x}}_i(t)$ is the result of Equation (5). Note that if the personal best and global best positions are the same, the PCX operator essentially degrades to performing a line search between the random position and the personal/global best position [9], [11]. However, given that the primary purpose of the PCX operator is diversity maintenance and not exploitation, such a situation does not inhibit performance in the same capacity as was noted by Padhye in static environments [11].

Two variants of QPSO-PCX are proposed in this work, differentiated by the selection of the mutated parent. In the QPSO-PCX$_r$ variant, the mutated parent is randomly selected from the parents, while the QPSO-PCX$_{\hat{y}}$ variant always mutates the parent corresponding to the global best position. The use of a random position in the search space facilitates unbounded and variably-sized diversity injection, thereby removing the limitations imposed by a fixed-sized radius. Since the two QPSO-PCX variants differ in how the random position influences the offspring, they have different exploration/exploitation trade-offs. The degree to which the mutated parent is exploited, and therefore the overall exploitation and exploration capabilities of QPSO-PCX, is directly controlled by the $\sigma_1$ and $\sigma_2$ parameters of the PCX operator shown in Equation (5). When $\sigma_1$ is small, the offspring is generated closer to the mutated parent. Conversely, larger values of $\sigma_2$ lead to offspring being biased towards the non-mutated parents. Thus, for the QPSO-PCX$_{\hat{y}}$, a small value of $\sigma_1$ leads to positions calculated near the global best, thereby increasing exploitation; larger values of $\sigma_2$ lead to a greater influence of the random position, enhancing exploration. For the QPSO-PCX$_r$ variant, the exploration/exploitation influence of the parameters is dependent upon the parent selected for mutation.

## IV. Experimental Procedure

To compare the performance of QPSO-PCX$_r$, QPSO-PCX$_{\hat{y}}$ and QPSO, benchmark environments exemplifying various degrees of dynamism were instantiated using the moving peaks benchmark (MPB) [16]. Single-peak MPB environments were created according to the parameters listed in Table I in 5, 10, 30, and 50 dimensions. The selection of these parameters corresponds to the four dynamic environment types proposed by Duhain and Engelbrecht [15]. Each algorithm configuration was executed 30 times in every environment for 1000 iterations using the parameters listed in Table II. To remove the influence of change detection mechanisms, particles re-evaluated their personal best positions at each iteration. Finally, to prevent invalid attractors, a particle's personal best position was only updated if the new position was within the feasible region of the search space.

Twenty-five configurations of the QPSO-PCX algorithms and fifteen configurations of the QPSO algorithm were examined as follows. For the QPSO-PCX algorithms, the parameters $\sigma_1$ and $\sigma_2$ were both evaluated for values of 0.1, 0.3, 0.5, 0.7, and 1.0. For QPSO, in addition to the typical uniform distribution, two additional probability distributions, namely the non-uniform, and triangular (mode = 0.5) distributions, were employed based on their demonstrated successes [8].

For each distribution, five values for $r_{cloud}$ were examined, namely 0.5, 1.0, 12.75, 25.0, and 50.0, based around the suggestion that $r_{cloud} \approx \frac{s}{2}$ leads to good performance [5], [7]. Values of $0.5$ and $1.0$ correspond to $\frac{s}{2}$ and $s$, respectively, for quasi-static and progressive environments, while $25.0$ and $50.0$ correspond to $\frac{s}{2}$ and $s$, respectively, for abrupt and chaotic environments. The radius of $12.75$ corresponds to the average of the aforementioned values of $\frac{s}{2}$ (i.e., $\frac{0.5+25}{2}$).

TABLE I: Environmental Control Parameters

| | Environment Type | | | |
|---|---|---|---|---|
| **Parameter** | Quasi-Static | Progressive | Abrupt | Chaotic |
| Peak Heights | $[30, 70]$ | $[30, 70]$ | $[30, 70]$ | $[30, 70]$ |
| Peak Widths | $[1, 12]$ | $[1, 12]$ | $[1, 12]$ | $[1, 12]$ |
| Height Severity | 1 | 1 | 10 | 10 |
| Width Severity | 0.05 | 0.05 | 0.05 | 0.05 |
| Change Severity ($s$) | 1 | 1 | 50 | 50 |
| $\lambda$ | 0 | 0 | 0 | 0 |
| Change Freq. (Iterations) | 200 | 1 | 200 | 5 |

TABLE II: Algorithmic Parameters

| **Parameter** | **Value** |
|---|---|
| Particles | 30 |
| Quantum Proportion | 50% (15 particles) |
| $\omega$ | 0.729844 |
| $c1, c2$ | 1.496180 |
| Topology | Global best (star) |
| Iteration Strategy | Synchronous |

### A. Performance Measures

When examining performance in dynamic environments, several aspects are important to measure, namely *accuracy* (the proximity to the optimum), *stability* (the reduction in solution quality when an environmental change occurs), *reactivity* (the ability of an optimizer to recover from an environmental change), and *exploitation capacity* (the ability to find quality solutions between environmental changes) [12]. The four performance measures employed are described below and were selected to address each of these aspects of performance.

The *error* of an optimizer, which measures its accuracy, is the absolute value of the difference between the global best fitness and the optimal fitness. The error measure is given by

$$e(t) = |f(\hat{\mathbf{y}}(\mathbf{t})) - f(\mathbf{o}(t))| \tag{7}$$

where $\mathbf{o}(t)$ is the optimal solution at time $t$.

The *collective mean error* (CME) is the mean error of the best solution, given by

$$\text{CME}(t) = \frac{\sum_{i=1}^{t} e(i)}{t}, \tag{8}$$

and incorporates the accuracy, stability, reactivity, and the exploitation capacity of an optimizer.

The *average best error before change* (ABEBC) measures the average error of the best solution at each iteration just before an environmental change occurred and strongly indicates the exploitation capacity. The ABEBC measure is given by

$$\text{ABEBC}(t) = \frac{\sum_{c=1}^{n_c(t)} e(c)}{n_c(t)} \tag{9}$$

where $c$ indicates an iteration just before a change occurs and $n_c$(t) denotes the number of environmental changes which have occurred up to time $t$.

Finally, the *average best error after change* (ABEAC) measures the average error of the best solution at each iteration just after an environmental change occurred and strongly indicates the stability. The calculation of the ABEAC measure differs from the ABEBC measure in Equation (9) only in that the error measure is averaged across iterations immediately following an environmental change.

### B. Statistical Analysis

Statistical analysis of results was performed using the normalized wins and losses approach [18]. For each performance measure, a Kruskal-Wallis test was performed using the average performance measure values at each iteration just before an environmental change occurred. If the Kruskal-Wallis test indicated that a difference existed among the strategies, pairwise Mann-Whitney U tests were performed to identify the individual differences. When the Mann-Whitney U test indicated that a difference existed among two strategies, the average performance measure values at each iteration just before a change occurred were used to assign wins and losses; the better performing algorithm was awarded a win, while the inferior algorithm was awarded a loss. To prevent skewed results, the wins and losses were normalized using the number of environmental changes. Finally, the strategies were ranked based on the difference between the number of wins and losses.

## V. Results and Discussion

The tabular results presented in this section depict the rank attained by the use of each parameter configuration, for both the QPSO-PCX and QPSO algorithms, with respect to each performance measure. The average rank across all performance measures is also provided for each algorithm configuration to give an indication of overall performance. Note that an entry with a higher rank indicates that a statistically significant performance improvement was observed relative to an entry with a lower rank. Therefore, the algorithm configuration which attained the highest rank (i.e., a rank of 1) demonstrated significantly superior performance over all other examined algorithm configurations.

Table III presents the experimental results aggregated across all examined environments. For the error measure, the QPSO-PCX$_r$ with $\sigma_1 = 0.1$ and $\sigma_2 = 0.7$ lead to the best performance overall. For the remainder of measures, namely the CME, ABEBC, and ABEAC, the QPSO-PCX$_{\hat{y}}$ $\sigma_1 = 0.3$ and $\sigma_2 = 0.1$ performed best.

Considering the average rank across all measures, shown in Table III, there is a clear preference for low values of $\sigma$ in the QPSO-PCX algorithms. In both variants of QPSO-PCX, the lowest average rank was observed when $\sigma_1 = 0.3$ and $\sigma_2 = 0.1$. Moreover, the four best configurations of both QPSO-PCX$_r$[1] and QPSO-PCX$_{\hat{y}}$ had either $\sigma_1 = 0.1$ or $\sigma_2 = 0.1$. Finally, it is noted that the lowest average rank for any QPSO configuration was 40.0 when the uniform

[1]Note that for QPSO-PCX$_r$, the fourth best configuration was a tie between $\sigma_1 = 0.5, \sigma_2 = 0.1$ and $\sigma_1 = 1.0, \sigma_2 = 0.7$

distribution with radius 12.75 was employed. Evidently, the performance of both QPSO-PCX variants was significantly better than QPSO across all environments.

Figure 1 shows the diversity profiles, measured as the average distance from the swarm centre [19], of the best ranked configuration for each algorithm. The diversity profiles demonstrate that the QPSO-PCX variants maintain substantially higher levels of diversity, and thus exploration, than QPSO. Another noteworthy observation was that the QPSO-PCX$_{\hat{y}}$ algorithm maintains a higher level of diversity than QPSO-PCX$_r$ with the same parametrization.

Figure 2 shows the mean error of the best ranked configuration for each algorithm. Notice that in the environments with low temporal severity, i.e., the quasi-static and abrupt environments, the QPSO-PCX variants responded slower to environmental changes. Despite the slower response of the QPSO-PCX algorithms, there was no significant difference in performance noted for any algorithm on the quasi-static and abrupt environments. However, when the progressive and chaotic, i.e., high temporal severity, environments were considered, the enhanced exploration proved beneficial and allowed the QPSO-PCX algorithms to more quickly adapt to the fast-paced environmental changes.

In the following sections, the performance is examined with respect to each dynamic environment type.

### A. Quasi-Static Environments

When quasi-static environments were considered, there was no significant difference noted for any algorithm on any of the performance measures. Although Figure 2a appears to indicate a difference in performance, especially for the QPSO-PCX$_{\hat{y}}$, it should be noted that the plots depict the mean error values while the (non-parametric) statistical tests examine median performance. Therefore, the higher mean error for QPSO-PCX$_{\hat{y}}$ can be attributed to anomalously poor runs.

### B. Progressive Environments

The results for progressive environments are provided in Table IV. For the error measure, the QPSO-PCX$_r$ algorithm with $\sigma_1 = 0.1$ and $\sigma_2 = 0.7$ demonstrated the best performance. The remainder of measures were most effectively minimized by QPSO-PCX$_r$ with $\sigma_1 = \sigma_2 = 0.1$. However, the best average rank was attained by neither of the aforementioned configurations, but rather by QPSO-PCX$_{\hat{y}}$ with $\sigma_1 = 0.1$ and $\sigma_2 = 0.3$.

In the case of progressive environments, there was a preference exhibited by both QPSO-PCX variants towards lower values of $\sigma$. For both variants, the best three configurations had either $\sigma_1 = 0.1$ or $\sigma_2 = 0.1$ while the worst configurations had $\sigma_1 = 1.0$. Considering the QPSO algorithm, the best performance (an average rank of 23.5) was depicted when the uniform distribution with radius 12.75 was employed. The performance of QPSO was evidently sub-par on progressive environments.

### C. Abrupt Environments

As with quasi-static environments, there was no significant difference noted among any of the algorithms for any performance measure on abrupt environments. Examining Figure 2c indicates that the best configuration of each algorithm performed similarly on abrupt environments. The mean error of the QPSO algorithm, in general, decreased quicker than the QPSO-PCX variants, indicating superior exploitation for the QPSO algorithm. However, this is to be expected given that the QPSO-PCX variants are meant to retain higher levels of diversity and are more focused on exploration than QPSO.

### D. Chaotic Environments

The results for chaotic environments are presented in Table V. With respect to the error measure, there was no significant difference noted among any of the algorithms. For the remainder of measures, QPSO-PCX$_r$ with $\sigma_1 = \sigma_2 = 0.3$ attained the best rank.

When considering the average ranks across all performance measures, the QPSO-PCX variants no longer depict an overwhelming tendency toward a value of 0.1 for $\sigma_1$ or $\sigma_2$. However, there was a definite preference for values of $\sigma \leq 0.5$ as values of $\sigma > 0.5$, in general, lead to the worst performance of QPSO-PCX configurations. Given that $\sigma$ values of 0.1 lead to quantum positions being more heavily biased towards the global best position (see Section III), the successes of slightly higher $\sigma$ values indicated that a greater degree of exploration was necessary to address the more severe changes exhibited by chaotic environments. For the QPSO algorithm, the best average rank of 35.3 was, again, demonstrated when using the uniform distribution with radius 12.75. As with the progressive environments, the performance of QPSO was significantly worse than that of the QPSO-PCX variants.

### E. Summary

In summary, both the QPSO-PCX$_r$ and QPSO-PCX$_{\hat{y}}$ variants showed drastic superiority over QPSO in progressive and chaotic environments. Given that both progressive and chaotic environments have high temporal severity, it is evidenced that the enhanced exploratory power of QPSO-PCX faired better than QPSO when the environmental changes were frequent. No significant difference in performance was noted for any performance measure on quasi-static and abrupt environments. In general, the performance of the QPSO-PCX$_{\hat{y}}$ variant was superior to that of QPSO-PCX$_r$. Furthermore, selecting the best parent for mutation (i.e., the QPSO-PCX$_{\hat{y}}$) lead to increased diversity relative to selecting a random parent for mutation.

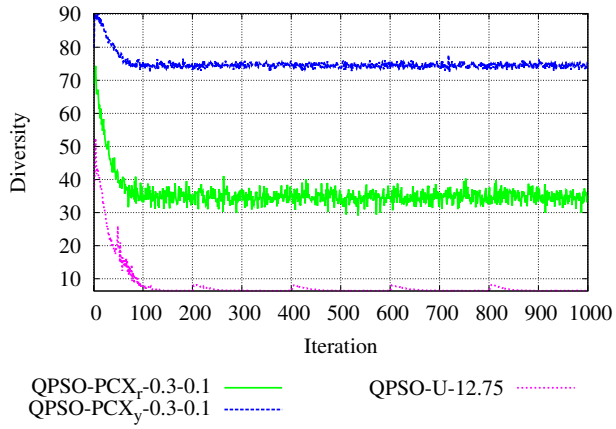
## VI. Conclusion

This paper presented the QPSO-PCX algorithm, a radius-free quantum particle swarm optimization (QPSO) algorithm which employed a parent centric crossover (PCX) operator to generate the positions of quantum particles. Two variants of QPSO-PCX were proposed, namely QPSO-PCX$_r$ and QPSO-PCX$_{\hat{y}}$ which differed in how the mutated parent was selected for the PCX operator. The main benefit, without regards to performance, of QPSO-PCX as opposed to QPSO is that the diversity maintenance is no longer limited by the pre-selected size of the quantum radius. Therefore, the QPSO-PCX is more well-suited to problems where the magnitude of environmental changes is not known *a priori*.

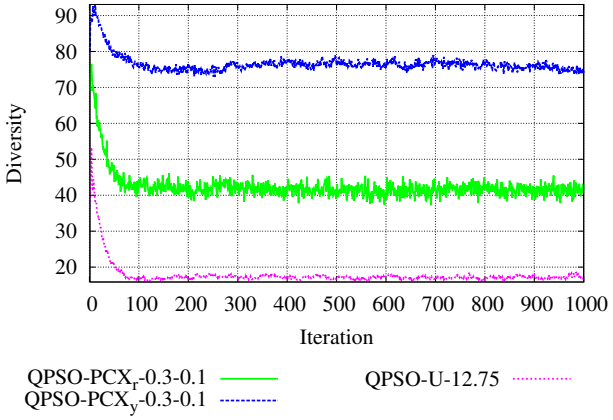
Various parameter configurations of the QPSO-PCX variants and QPSO algorithm were compared on a variety of

TABLE III: Rank Aggregated Across All Environments

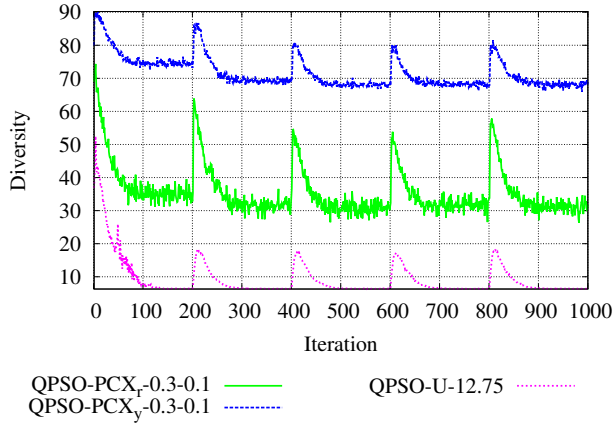
| Measure | $\sigma_1$ | QPSO-PCX$_r$ $\sigma_2$ 0.1 | 0.3 | 0.5 | 0.7 | 1.0 | QPSO-PCX$_{\hat{y}}$ $\sigma_2$ 0.1 | 0.3 | 0.5 | 0.7 | 1.0 | QPSO Radius | Distribution U | N | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Error | 0.1 | 10 | 59 | 61 | **1** | 13 | 17 | 2 | 19 | 27 | 11 | 0.5 | 50 | 54 | 46 |
| | 0.3 | 4 | 63 | 56 | 39 | 9 | 7 | 37 | 28 | 3 | 21 | 1.0 | 46 | 48 | 48 |
| | 0.5 | 40 | 25 | 16 | 28 | 64 | 18 | 57 | 12 | 24 | 22 | 12.75 | 28 | 28 | 28 |
| | 0.7 | 6 | 14 | 65 | 51 | 5 | 20 | 26 | 38 | 15 | 55 | 25.0 | 42 | 28 | 43 |
| | 1.0 | 62 | 8 | 60 | 28 | 28 | 52 | 58 | 22 | 28 | 53 | 50.0 | 45 | 41 | 44 |
| CME | 0.1 | 9 | 32 | 41 | 59 | 22 | 2 | 13 | 6 | 3 | 21 | 0.5 | 64 | 65 | 63 |
| | 0.3 | 5 | 27 | 36 | 16 | 18 | **1** | 19 | 11 | 26 | 10 | 1.0 | 61 | 62 | 60 |
| | 0.5 | 4 | 20 | 14 | 42 | 38 | 33 | 47 | 17 | 57 | 23 | 12.75 | 45 | 49 | 46 |
| | 0.7 | 15 | 34 | 39 | 29 | 35 | 28 | 8 | 24 | 48 | 55 | 25.0 | 53 | 50 | 54 |
| | 1.0 | 43 | 52 | 40 | 7 | 30 | 12 | 37 | 25 | 44 | 31 | 50.0 | 56 | 51 | 58 |
| ABEBC | 0.1 | 9 | 32 | 40 | 57 | 22 | 4 | 15 | 6 | 5 | 21 | 0.5 | 64 | 65 | 63 |
| | 0.3 | 3 | 24 | 36 | 16 | 18 | **1** | 20 | 11 | 27 | 10 | 1.0 | 61 | 62 | 60 |
| | 0.5 | 2 | 19 | 14 | 44 | 37 | 33 | 49 | 17 | 58 | 23 | 12.75 | 42 | 46 | 43 |
| | 0.7 | 13 | 34 | 39 | 29 | 35 | 28 | 8 | 25 | 50 | 55 | 25.0 | 52 | 48 | 53 |
| | 1.0 | 45 | 54 | 41 | 7 | 30 | 12 | 38 | 26 | 47 | 31 | 50.0 | 56 | 51 | 59 |
| ABEAC | 0.1 | 9 | 32 | 41 | 59 | 22 | 2 | 14 | 6 | 4 | 21 | 0.5 | 64 | 65 | 62 |
| | 0.3 | 5 | 26 | 38 | 16 | 18 | **1** | 20 | 11 | 27 | 10 | 1.0 | 61 | 63 | 60 |
| | 0.5 | 3 | 19 | 15 | 43 | 36 | 33 | 47 | 17 | 57 | 23 | 12.75 | 45 | 48 | 46 |
| | 0.7 | 13 | 34 | 39 | 29 | 35 | 28 | 8 | 24 | 49 | 55 | 25.0 | 52 | 50 | 54 |
| | 1.0 | 42 | 53 | 40 | 7 | 30 | 12 | 37 | 25 | 44 | 31 | 50.0 | 56 | 51 | 58 |
| Average | 0.1 | 9.3 | 38.8 | 45.8 | 44.0 | 19.8 | 6.3 | 11.0 | 9.3 | 9.8 | 18.5 | 0.5 | 60.5 | 62.3 | 58.5 |
| | 0.3 | 4.3 | 35.0 | 41.5 | 21.8 | 15.8 | **2.5** | 24.0 | 15.3 | 20.8 | 12.8 | 1.0 | 57.3 | 58.8 | 57.0 |
| | 0.5 | 12.3 | 20.8 | 14.8 | 39.3 | 43.8 | 29.3 | 50.0 | 15.8 | 49.0 | 22.8 | 12.75 | 40.0 | 42.8 | 40.8 |
| | 0.7 | 11.8 | 29.0 | 45.5 | 34.5 | 27.5 | 26.0 | 12.5 | 27.8 | 40.5 | 55.0 | 25.0 | 49.8 | 44.0 | 51.0 |
| | 1.0 | 48.0 | 41.8 | 45.3 | 12.3 | 29.5 | 22.0 | 42.5 | 24.5 | 40.8 | 36.5 | 50.0 | 53.3 | 48.5 | 54.8 |



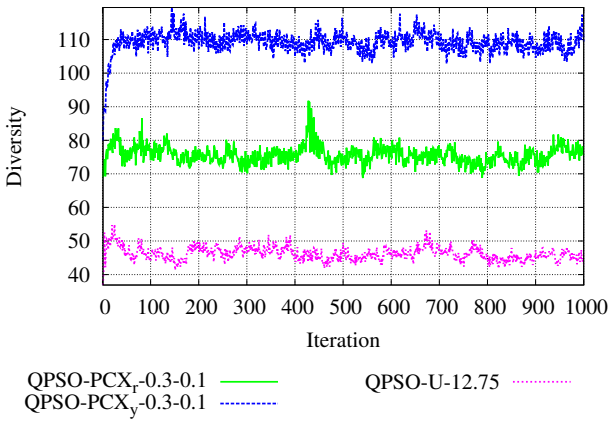

(a) Quasi-Static




(b) Progressive

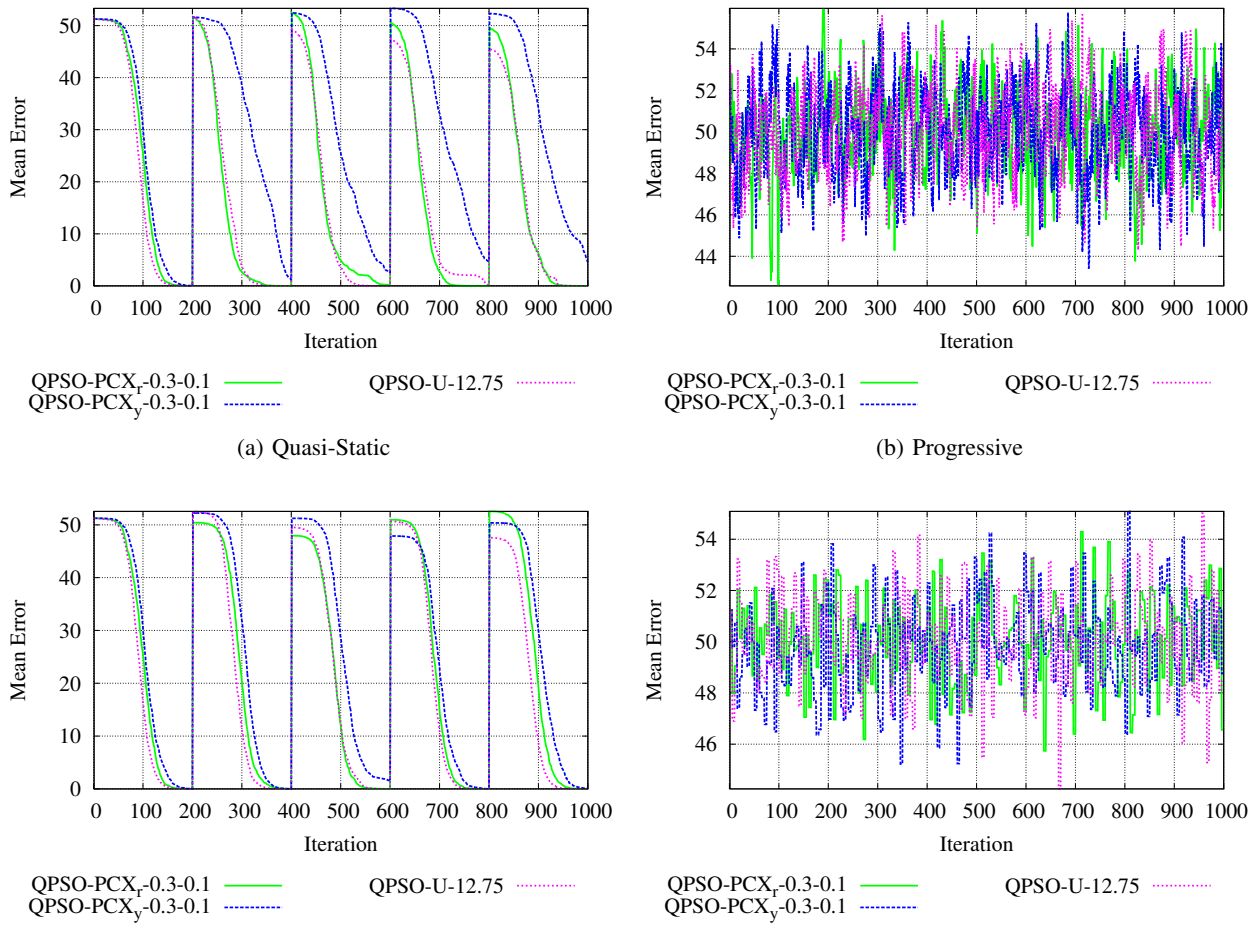



(c) Abrupt




(d) Chaotic

Fig. 1: Average swarm diversity for QPSO-PCX$_{\hat{y}}$ and QPSO-PCX$_r$ with $\sigma_1 = 0.3, \sigma_2 = 0.1$ and QPSO using the uniform distribution with radius 12.75 on 10D environments.




(a) Quasi-Static (b) Progressive

(c) Abrupt (d) Chaotic

Fig. 2: Mean error measure for QPSO-PCX$_{\hat{y}}$ and QPSO-PCX$_r$ with $\sigma_1 = 0.3, \sigma_2 = 0.1$ and QPSO using the uniform distribution with radius 12.75 on 10D environments.

TABLE IV: Ranking of Optimizers on Progressive Environments

| Measure | $\sigma_1$ | **QPSO-PCX$_r$** $\sigma_2$ 0.1 | 0.3 | 0.5 | 0.7 | 1.0 | **QPSO-PCX$_{\hat{y}}$** $\sigma_2$ 0.1 | 0.3 | 0.5 | 0.7 | 1.0 | **QPSO** Radius | Distribution U | N | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Error | 0.1 | 10 | 59 | 61 | **1** | 13 | 17 | 2 | 19 | 27 | 11 | 0.5 | 50 | 54 | 46 |
| | 0.3 | 4 | 63 | 56 | 39 | 9 | 7 | 37 | 28 | 3 | 21 | 1.0 | 46 | 48 | 48 |
| | 0.5 | 40 | 25 | 16 | 28 | 64 | 18 | 57 | 12 | 24 | 22 | 12.75 | 28 | 28 | 28 |
| | 0.7 | 6 | 14 | 65 | 51 | 5 | 20 | 26 | 38 | 15 | 55 | 25.0 | 42 | 28 | 43 |
| | 1.0 | 62 | 8 | 60 | 28 | 28 | 52 | 58 | 22 | 28 | 53 | 50.0 | 45 | 41 | 44 |
| CME | 0.1 | **1** | 32 | 37 | 50 | 6 | 10 | 2 | 4 | 24 | 20 | 0.5 | 53 | 54 | 51 |
| | 0.3 | 3 | 56 | 31 | 16 | 12 | 7 | 62 | 30 | 42 | 8 | 1.0 | 48 | 52 | 49 |
| | 0.5 | 15 | 40 | 25 | 58 | 18 | 17 | 47 | 13 | 57 | 21 | 12.75 | 22 | 26 | 23 |
| | 0.7 | 14 | 34 | 61 | 9 | 41 | 39 | 27 | 60 | 43 | 63 | 25.0 | 35 | 29 | 36 |
| | 1.0 | 64 | 28 | 38 | 5 | 11 | 46 | 65 | 19 | 59 | 55 | 50.0 | 45 | 33 | 44 |
| ABEBC | 0.1 | **1** | 32 | 37 | 53 | 6 | 10 | 2 | 4 | 24 | 20 | 0.5 | 52 | 54 | 50 |
| | 0.3 | 3 | 55 | 31 | 16 | 12 | 7 | 62 | 30 | 43 | 8 | 1.0 | 48 | 51 | 49 |
| | 0.5 | 15 | 40 | 25 | 58 | 18 | 17 | 47 | 13 | 57 | 21 | 12.75 | 22 | 26 | 23 |
| | 0.7 | 14 | 34 | 61 | 9 | 41 | 39 | 27 | 60 | 42 | 63 | 25.0 | 35 | 29 | 36 |
| | 1.0 | 64 | 28 | 38 | 5 | 11 | 46 | 65 | 19 | 59 | 56 | 50.0 | 45 | 33 | 44 |
| ABEAC | 0.1 | **1** | 32 | 37 | 53 | 6 | 10 | 2 | 4 | 24 | 20 | 0.5 | 51 | 54 | 50 |
| | 0.3 | 3 | 55 | 31 | 16 | 13 | 7 | 62 | 30 | 43 | 8 | 1.0 | 48 | 52 | 49 |
| | 0.5 | 15 | 40 | 25 | 58 | 18 | 17 | 47 | 12 | 57 | 21 | 12.75 | 22 | 26 | 23 |
| | 0.7 | 14 | 34 | 61 | 9 | 41 | 39 | 27 | 60 | 42 | 63 | 25.0 | 35 | 29 | 36 |
| | 1.0 | 64 | 28 | 38 | 5 | 11 | 46 | 65 | 19 | 59 | 56 | 50.0 | 44 | 33 | 45 |
| Average | 0.1 | 3.3 | 38.8 | 43.0 | 39.3 | 7.8 | 11.8 | **2.0** | 7.8 | 24.8 | 17.8 | 0.5 | 51.5 | 54.0 | 49.3 |
| | 0.3 | 3.3 | 57.3 | 37.3 | 21.8 | 11.5 | 7.0 | 55.8 | 29.5 | 32.8 | 11.3 | 1.0 | 47.5 | 50.8 | 48.8 |
| | 0.5 | 21.3 | 36.3 | 22.8 | 50.5 | 29.5 | 17.3 | 49.5 | 12.5 | 48.8 | 21.3 | 12.75 | 23.5 | 26.5 | 24.3 |
| | 0.7 | 12.0 | 29.0 | 62.0 | 19.5 | 32.0 | 34.3 | 26.8 | 54.5 | 35.5 | 61.0 | 25.0 | 36.8 | 28.8 | 37.8 |
| | 1.0 | 63.5 | 23.0 | 43.5 | 10.8 | 15.3 | 47.5 | 63.3 | 19.8 | 51.3 | 55.0 | 50.0 | 44.8 | 35.0 | 44.3 |

TABLE V: Ranking of Optimizers on Chaotic Environments

| Measure | $\sigma_1$ | QPSO-PCX$_r$ $\sigma_2$ 0.1 | 0.3 | 0.5 | 0.7 | 1.0 | QPSO-PCX$_{\tilde{y}}$ $\sigma_2$ 0.1 | 0.3 | 0.5 | 0.7 | 1.0 | QPSO Radius | Distribution U | N | T |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Error | 0.1 | - | - | - | - | - | - | - | - | - | - | 0.5 | - | - | - |
| | 0.3 | - | - | - | - | - | - | - | - | - | - | 1.0 | - | - | - |
| | 0.5 | - | - | - | - | - | - | - | - | - | - | 12.75 | - | - | - |
| | 0.7 | - | - | - | - | - | - | - | - | - | - | 25.0 | - | - | - |
| | 1.0 | - | - | - | - | - | - | - | - | - | - | 50.0 | - | - | - |
| CME | 0.1 | 43 | 34 | 39 | 56 | 41 | 9 | 49 | 19 | 2 | 22 | 0.5 | 62 | 64 | 63 |
| | 0.3 | 21 | 11 | 38 | 25 | 32 | 4 | **1** | 10 | 14 | 27 | 1.0 | 61 | 60 | 59 |
| | 0.5 | 8 | 12 | 15 | 28 | 46 | 44 | 35 | 29 | 52 | 23 | 12.75 | 47 | 51 | 48 |
| | 0.7 | 24 | 33 | 18 | 42 | 30 | 16 | 3 | 6 | 40 | 36 | 25.0 | 53 | 50 | 55 |
| | 1.0 | 13 | 65 | 37 | 20 | 45 | 5 | 7 | 31 | 26 | 17 | 50.0 | 57 | 54 | 58 |
| ABEBC | 0.1 | 43 | 34 | 36 | 56 | 41 | 9 | 51 | 21 | 2 | 22 | 0.5 | 62 | 64 | 63 |
| | 0.3 | 19 | 11 | 35 | 25 | 31 | 5 | **1** | 12 | 15 | 27 | 1.0 | 60 | 61 | 59 |
| | 0.5 | 8 | 10 | 14 | 28 | 48 | 44 | 38 | 30 | 54 | 24 | 12.75 | 46 | 50 | 47 |
| | 0.7 | 23 | 33 | 18 | 42 | 29 | 16 | 4 | 6 | 40 | 39 | 25.0 | 53 | 49 | 55 |
| | 1.0 | 13 | 65 | 37 | 20 | 45 | 3 | 7 | 32 | 26 | 17 | 50.0 | 57 | 52 | 58 |
| ABEAC | 0.1 | 43 | 34 | 38 | 56 | 41 | 9 | 51 | 19 | 2 | 22 | 0.5 | 62 | 64 | 63 |
| | 0.3 | 21 | 12 | 39 | 25 | 31 | 4 | **1** | 11 | 14 | 27 | 1.0 | 60 | 61 | 59 |
| | 0.5 | 8 | 10 | 15 | 28 | 46 | 44 | 35 | 29 | 52 | 23 | 12.75 | 47 | 50 | 48 |
| | 0.7 | 24 | 33 | 18 | 42 | 30 | 16 | 5 | 6 | 40 | 36 | 25.0 | 54 | 49 | 55 |
| | 1.0 | 13 | 65 | 37 | 20 | 45 | 3 | 7 | 32 | 26 | 17 | 50.0 | 57 | 53 | 58 |
| Average | 0.1 | 32.5 | 25.8 | 28.5 | 42.3 | 31.0 | 7.0 | 38.0 | 15.0 | 1.8 | 16.8 | 0.5 | 46.8 | 48.3 | 47.5 |
| | 0.3 | 15.5 | 8.8 | 28.3 | 19.0 | 23.8 | 3.5 | **1.0** | 8.5 | 11.0 | 20.5 | 1.0 | 45.5 | 45.8 | 44.5 |
| | 0.5 | 6.3 | 8.3 | 11.3 | 21.3 | 35.3 | 33.3 | 27.3 | 22.3 | 39.8 | 17.8 | 12.75 | 35.3 | 38.0 | 36.0 |
| | 0.7 | 18.0 | 25.0 | 13.8 | 31.8 | 22.5 | 12.3 | 3.3 | 4.8 | 30.3 | 28.0 | 25.0 | 40.3 | 37.3 | 41.5 |
| | 1.0 | 10.0 | 49.0 | 28.0 | 15.3 | 34.0 | 3.0 | 5.5 | 24.0 | 19.8 | 13.0 | 50.0 | 43.0 | 40.0 | 43.8 |

single-peak dynamic environment types. Four performance measures were employed to capture the accuracy, stability, reactivity, and exploitation capacity of the algorithms. Results indicated that the proposed QPSO-PCX variants drastically outperformed QPSO on progressive and chaotic environments while no algorithm proved superior on quasi-static and abrupt environments. In general, the QPSO-PCX$_{\tilde{y}}$ variant outperformed the QPSO-PCX$_r$ variant. It was also found that biasing the quantum position in the QPSO-PCX algorithms towards the global best solution lead to better performance, in general.

Given that this study aimed solely to demonstrate the superiority of QPSO-PCX over the standard QPSO, an immediate future study consists of comparing the performance of QPSO-PCX with other state-of-the-art dynamic optimization algorithms. Additional future work involves examining the performance of QPSO-PCX on a wider variety of dynamic environments, specifically multi-peak and real-world environments.

## REFERENCES


[1] J. Branke, *Evolutionary optimization in dynamic environments*. Springer Science & Business Media, 2012, vol. 3.

[2] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Joint Conference on Neural Networks*, vol. IV. IEEE, 1995, pp. 1942–1948.

[3] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*. Springer, 2010, pp. 760–766.

[4] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[5] T. Blackwell and J. Branke, "Multiswarms, exclusion, and anti-convergence in dynamic environments," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 459–472, 2006.

[6] ——, "Multi-swarm optimization in dynamic environments," in *EvoWorkshops*, vol. 3005. Springer, 2004, pp. 489–500.

[7] T. Blackwell, J. Branke, and X. Li, "Particle swarms for dynamic optimization problems," in *Swarm Intelligence*, ser. Natural Computing Series. Springer, 2008, pp. 193–217.

[8] K. R. Harrison, B. Ombuki-Berman, and A. P. Engelbrecht, "The effect of probability distributions on the performance of quantum particle swarm optimization for solving dynamic optimization problems," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*. IEEE, 2015, pp. 242–250.

[9] A. P. Engelbrecht, "Particle swarm optimization with crossover: a review and empirical analysis," *Artificial Intelligence Review*, vol. 45, no. 2, pp. 131–165, 2016.

[10] K. Deb and N. Padhye, "Development of efficient particle swarm optimizers by using concepts from evolutionary algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2010, pp. 55–62.

[11] N. Padhye, "Development of efficient particle swarm optimizers and bound handling methods," Master's thesis, Indian Institute of Technology, 2010.

[12] K. Weicker, "Performance measures for dynamic environments," in *Proceedings of the International Conference on Parallel Problem Solving from Nature*, ser. LNCS. Springer, 2002, vol. 2439, pp. 64–73.

[13] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 1. IEEE, 2001, pp. 94–100.

[14] X. Hu and R. Eberhart, "Tracking dynamic systems with pso: wheres the cheese," in *Proceedings of the Workshop on Particle Swarm Optimization*, 2001, pp. 80–83.

[15] J. G. O. L. Duhain and A. P. Engelbrecht, "Towards a more complete classification system for dynamically changing environments," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–8.

[16] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 3, 1999.

[17] K. Deb, D. Joshi, and A. Anand, "Real-coded evolutionary algorithms with parent-centric recombination," in *Proceedings of the Congress on Evolutionary Computation*, vol. 1. IEEE, 2002, pp. 61–66.

[18] M. Helbig and A. P. Engelbrecht, "Analysing the performance of dynamic multi-objective optimisation algorithms," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2013, pp. 1531–1539.

[19] O. Olorunda and A. P. Engelbrecht, "Measuring exploration/exploitation in particle swarms using swarm diversity," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2008, pp. 1128–1134.