Self-Adaptive Quantum Particle Swarm Optimization for Dynamic Environments

Gary Pamparà[1] and Andries P. Engelbrecht[2]

[1] Department of Computer Science, University of Pretoria, South Africa
gpampara@gmail.com
[2] Institute for Big Data and Data Science, University of Pretoria, South Africa
engel@cs.up.ac.za

Abstract. The quantum-inspired particle swarm optimization (QPSO) algorithm has been developed to find and track an optimum for dynamic optimization problems. Though QPSO has been shown to be effective, despite its simplicity, it does introduce an additional control parameter: the radius of the quantum cloud. The performance of QPSO is sensitive to the value assigned to this problem dependent parameter, which basically limits the area of the search space wherein new, better optima can be detected. This paper proposes a strategy to dynamically adapt the quantum radius, with changes in the environment. A comparison of the adaptive radius QPSO with the static radius QPSO showed that the adaptive approach achieves desirable results, without prior tuning of the quantum radius.

1 Introduction

Optimization algorithms have very different performance characteristics based on the underlying problem search space. Static environments have been widely studied and several algorithms have been shown to be applicable to a wide variety of static optimization problems. Notable examples include genetic algorithms [14], differential evolution [22] and particle swarm optimization (PSO) [16]. In the case where the underlying problem search space changes over time, the behavior characteristics of these algorithms are not always still applicable. A dynamic optimization problem describes such a problem search space, where the optimal value of the search space not only changes over time, but can drastically change in value and/or location. When applying an algorithm designed for a static environment to a dynamic environment, the inefficiencies of the algorithms become apparent with the algorithms unable to adapt to the changing search space. Common reasons for the inefficiency include a loss of diversity, outdated memory of previous best positions, and the inability of the algorithm to detect that the search space has actually changed [21].

Quantum particle swarm optimization (QPSO) [3] was proposed as an inherently dynamic variant of the PSO, capable of handling underlying environment changes by categorizing a set of particles within the algorithm as "quantum"

particles. These quantum particles follow a different position update strategy from normal particles: Quantum particles move probabilistically within a predefined quantum cloud around the global best particle position. The size of the quantum cloud determines the area within which a quantum particle is allowed to move. Unfortunately, the solution tracking ability of the QPSO is sensitive to the size of the radius. If the radius is too small, the area the quantum particles can explore for changes in the optimum, will be restricted; preventing the detection of an optima change outside of the cloud radius. Conversely, if the cloud radius is too large, a larger portion of the search space will be covered by the cloud, resulting in unnecessary exploration for small change severities, and resulting more in a random search.

This paper proposes a strategy to adapt the cloud radius value, starting with a large radius and reducing it to a small value. As soon as the environment changes, the radius is increased to a large value. This strategy helps to increase the exploration by the quantum particles whenever a change in the environment occurs, while moving towards exploitation the longer the environment remains unchanged.

The remainder of the paper is organized as follows: Section 2 discusses the necessary background information for the algorithm alteration proposed in Section 3. Following in Section 4, is a discussion on the experimental approach with results provided in Section 5. Lastly, concluding remarks are presented in Section 6.

2 Background

This section provides background information for the remainder of the paper. Section 2.1 discusses dynamic environments, with the moving peaks benchmark (MPB) discussed in Section 2.2. Sections 2.3 and 2.4 respectively discuss the PSO and QPSO algorithms, with previous radius management strategies discussed in Section 2.5.

2.1 Dynamic Environments

A problem space wherein optima move over time is regarded as a dynamic environment. The changes experienced by such environments can vary from subtle movements to extreme changes where optima seemingly move at random. Much work has been done to attempt to classify these problem space changes, based on the frequency and severity of the change observed, the type of movement the change undertook, and the trajectory that a change followed. Eberhart and Shi [10] and Hu and Eberhart [15] describe changes observed for an optimum as (a) Type I, where the measurable value of an optimum remains the same, but its position in the search space changes, (b) Type II, where the value of the optimum changes but the position in the search space remains constant, or (c) Type III, where both the position and the optimum value change. Angeline [1] categorizes the movement of an optimum as either linear, circular, or random.

Duhain and Engelbrecht [9] combine the previous classifications with spatial and temporal severity to create 27 unique environments. These environments are broadly classified into:

1. **Quasi-static** environments which have both low spatial and temporal changes.
2. **Progressive** environments which have small spatial, but frequent changes. The changes result in a search space where optima move gradually over time.
3. **Abrupt** environments which have infrequent and large spatial changes. The problem space almost seems to remain constant for a period of time before the next, large change is observed.
4. **Chaotic** environments experience large spatial adjustments that occur at a frequent interval.

2.2 Moving Peaks Benchmark Function

The MPB [15] has been developed as a generator for dynamic environments, based on a set of input parameters. The generator can create problems spaces that adhere to a variety of classification goals, and is quite popular within literature [6, 5, 17, 19]. The generated problem search spaces are characterized by the number of peaks, within a given domain, with each peak maintaining a width, height, and location within the search space. The dynamic nature of the problem is achieved by varying the parameters of the MPB generator over time, resulting in a changed environment. The movement of the peaks is also determined by the state of the previous environment, where trajectory information of the peaks themselves is maintained.

2.3 Particle Swarm Optimization

The PSO algorithm, introduced by Kennedy and Eberhart [16], describes an optimization process which is modeled on the flocking behavior of birds. PSO is a population-based, stochastic algorithm which maintains a collection of entities, known as "particles". Each particle exhibits a simple behavior which determines its movement: (1) move towards the best position in the immediate neighborhood of particles, and (2) move towards a particle's own best, previously observed, position.

Particles move in an iterative manner throughout the problem search space, determining the magnitude of the movement through the application of a "velocity" vector (representing a step size and direction) and to the currently maintained position vector. The velocity vector implements the previously mentioned particle behavior, and is calculated as:

$$v_{ij}(t+1) = \omega v_{ij}(t) + c_1 r_{1j}(y_{ij}(t) - x_{ij}(t)) + c_2 r_{2j}(\hat{y}_{ij}(t) - x_{ij}(t)) \qquad (1)$$

where $v_{ij}(t)$ is the velocity of particle i in dimension j, with the current particle position given by $x_{ij}(t)$, at time step t; ω denotes the inertia coefficient, determining the extent to which a particle will continue to search in the same

direction, with c_1 and c_2 respectively specifying the influence of the social and cognitive components to the resultant velocity vector. The stochastic vectors, $\mathbf{r}_1$ and $\mathbf{r}_2$, are multiplied with the cognitive and social velocity components.

The movement of particle i through the search space, to the next position, is then determined by

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

The classical PSO algorithm displays several disadvantages when applied to dynamic environments, particularly related to the personal best position. There is no way to know, ahead of time, whether an environment change would (potentially) make the particle's best position obsolete or invalid [2]. Additionally, as the swarm converges on a solution, the dispersion of particles within the swarm (diversity) [25] decreases. This loss of diversity results in small step sizes, preventing particle movement to other areas of the search space.

2.4 Quantum Particle Swarm Optimization

The QPSO [3] was introduced to address the problems of the PSO within dynamic environments. The QPSO employs a percentage of particles that use a position update that is different to equation (2). These particles, referred to as *quantum particles*, move in a manner similar to electrons orbiting the nucleus of an atom. The movement of the quantum particles is done by sampling a probability distribution, centered at the global best position of the current swarm of particles. The movement is then

$$x_{ij}(t+1) \sim d(\hat{y}_{ij}(t), r_{cloud}) \tag{3}$$

where d is a probability distribution and r_{cloud} is a constant determining the size of the quantum "cloud" around the global best position. Any particle that is not updated using equation (3), is termed a neutral particle and follows the update equations of the classical PSO algorithm.

Although this approach does allow for a percentage of particles to behave in a manner that addresses concerns around diversity loss, an additional parameter, r_{cloud}, is introduced. Because r_{cloud} restricts the search area of quantum particles around the global best position, the parameter requires problem dependent tuning. After an environment change, a larger radius is desirable, as more exploration can occur within the quantum cloud, whereas a smaller radius would aid with solution refinement when the environment is unchanging.

2.5 Previously Suggested Radius Management Strategies

In order to prevent the problem dependent tuning of r_{cloud}, several strategies have been proposed. Blackwell *et al* [4] propose the use of different distributions to influence the movement of quantum particles within the quantum cloud. Although this approach does not hint at adjusting the radius of the quantum cloud,

the quantum particles may move to positions beyond the defined radius, based on the distribution being sampled. For example, sampling a standard Gaussian distribution would restrict 95% of the observed values to be within two standard deviations from the mean, and would on occasion allow for larger values.

Harrison *et al* [12] examine the effect of different distributions on the performance of the QPSO, concluding that the choice of distribution depends on the type of dynamic environment and that smaller values for r_{cloud} lead to better performances for QPSO. These conclusions also stated that the uniform distribution is, generally, a poor choice. Another study by Harrison *et al* [13] attempt to remove the parameter r_{cloud} and the probability distribution completely through the use of a parent centric crossover (PCX) [7] operator. The resulting algorithm completely removes the atom metaphor, replacing it with a crossover operator instead. As a result, the algorithm can not truly be classified as a QPSO variant. The algorithm does, however, address the problem of diversity loss because the PCX operator was originally designed to introduce diversity into the swarm of particles, but introduces two additional parameters: the deviations of two Gaussian distributions.

3 Self-Adaptive Quantum Particle Swarm Optimization

In order to allow the QPSO to manage a dynamically sized quantum cloud, adjusting automatically based on the current swarm and the current environment, some of the problems associated with the PSO in dynamic environments (outdated memory, diversity loss, etc) need to be addressed. The resulting algorithm is largely unchanged from the standard QPSO definition, with a few configuration changes applied:

- When a change in the environment occurs, the memory of particles (personal best positions) may have become invalid or obsolete. These invalid values need to be corrected, otherwise the particle may be attracted to an undesirable area of the search space. One such mechanism, is to reset the personal best position of the particle to the current position, and to re-evaluate the particle. Quantum particles do not depend on a personal best position and are re-initialized within the problem domain.
- The personal best position of a particle is updated, if and only if the current position is within the boundaries of the search space. This constraint on the update process prevents solutions that may be seemingly more optimal, but are located outside the defined problem search space to become personal best positions. Allowing infeasible personal best positions will result in particles being attracted to infeasible solutions, and / or fruitless search in infeasible space [11].
- The quantum cloud radius, r_{cloud}, is calculated by taking the maximum between the diversity of neutral and quantum particles. The diversity calculation for neutral particles only considers the particle personal best position, as these positions will be within the problem search space. Quantum particles are only considered if the current position of the particle is within the

problem bounds. Diversity is calculated as:

$$D(S(t)) = \frac{1}{n_s} \Sigma_{i=1}^{n_s} \sqrt{\Sigma_{j=1}^{n_x} (x_{ij}(t) - \bar{x}_j(t))^2} \quad (4)$$

where n_s is the number of the neutral, or quantum particles considered in the diversity calculation; $\bar{x}_j(t)$ is the average j-th dimension of the entire swarm, calculated as

$$\bar{x}_j(t) = \frac{\Sigma_{i=1}^{n_s} x_{ij}(t)}{n_s} \quad (5)$$

The resulting diversity value is then used as the cloud radius value. The cloud radius value is fed into a random distribution as the deviation, from which quantum particle positions are sampled. Scaling the calculated diversity by a constant to determine the cloud radius was not considered as it would introduce an additional constant, which would require problem dependent tuning.

- The neighborhood topology should facilitate information exchange, but at a slower rate than a fully connected topology. Slower information propagation through the swarm using topologies like the local best or Von Neumann, allows for larger parts of the search space to be covered, whilst still allowing for convergence. Fully connected topologies result in faster convergence which results in smaller diversity and cloud radius values.
- Particle neighborhoods should consist of both types of particles. Because quantum particles move within the quantum cloud at each iteration, it is advantageous to share information about the changed search space with the neutral particles in the local particle neighborhood.

4 Experimental Approach

The main objective of this paper is to demonstrate that the performance of the QPSO on a set of dynamic optimization problems is either better than, or unchanged, when the algorithm dynamically adapts the value of r_{cloud} compared to keeping the value at a predefined static value. This section describes the different considerations in order to prepare the experiments to evaluate the QPSO and the self-adaptive QPSO, with experiment design in Section 4.1, performance measures in Section 4.2, and Section 4.3 discussing the statistical process.

4.1 Experimental Design

Due to the complex nature of algorithms that operate in dynamic environments, it is beneficial to elaborate on the software approach used. All experiments used the software library *CIlib* [20] which allows for type-safe, repeatable experimentation, with perfectly reproducible results. The initial seed, from which 30 distinct seeds are then generated for the different independent algorithm runs, is listed in Table 2. Importantly, different pseudo-random number generators are used

Table 1: Environment parameters

Parameter	Static	Progressive	Abrupt	Chaotic
Peak height	[30, 70]	[30, 70]	[30, 70]	[30, 70]
Peak width	[1, 12]	[1, 12]	[1, 12]	[1, 12]
Height change severity	1	1	10	10
Width change severity	0.05	0.05	0.05	0.05
Change severity	1	1	10	10
Random movement % (λ)	0	0	0	0
Change freq. (iterations)	200	1	200	5

Table 2: QPSO algorithm parameters

Parameter	Value
Particles	40
Proportion quantum particles	50%
ω	0.729844
c_1, c_2	1.496180
Topology	l-best (size 3)
Iteration strategy	Synchronous
PRNG Seed	123456789L
Static radius values	$r_{cloud} \in [5, 10, 50]$
Quantum cloud distribution	Gaussian

for algorithms and dynamic environment updates. PSO parameter choices are based on the convergence properties described by Van den Bergh [24], with the l-best topology providing slower information propagation throughout the neutral particles. Sampling a Gaussian distribution centered at the global best position allows for quantum particle movement with a central tendency at the global best position, whilst still allowing unconstrained movement that may exceed the boundaries of the quantum cloud.

To compare the performance of the static QPSO with the self-adaptive QPSO, benchmark environments were defined that match the classifications of Duhain [8] and Duhain and Engelbrecht [9], using the MPB as the search space problem. Each problem space contained 10 peaks and used the parameters defined in Table 1. Each environment was also classified as a Type III environment [15]. Each algorithm configuration was executed 30 times for 1000 iterations. The domain of the problem search space was defined to be $[0, 100]$, with 5 dimensions. The static QPSO variants are identified by the size of the associated radius: QPSO-5, QPSO-10, and QPSO-50 for cloud radius of 5, 10 and 50 respectively. Other parameters, common to the QPSO algorithms, are listed in Table 2.

4.2 Dynamic Environment Performance Measures

Performance measures are required to quantify the performance of an algorithm within a dynamic environment. Performance measures of static environments do

not behave in a manner that allows for a valid performance quantification for dynamic environments. Duhain [8] recommend that better choices for performance measurement within a dynamic environment include the *accuracy* of the solutions over time, the *stability* (solution quality after an environment change), and algorithm *exploitative capacity*, which is the quality of the best solution between environment changes. Because the MPB defines a maximum peak value, the error produced by an algorithm, by comparing a solution to the search space optimum, is calculable. A set of "good" [8] measures are:

- The **collective mean error** (CME) [18] records the mean error of the best solution over the entire experiment. The measure is defined as:

$$\text{CME} = \frac{1}{n_t} \Sigma_{t=1}^{n_t} err_{t,m} \tag{6}$$

  where n_t is the number of iterations within an experiment, and $err_{t,m}$ is the difference between the optimum in environment m and the best solution, at time-step t. The CME is a good overall measure [18] that quantifies the overall performance of an algorithm within a dynamic environment.
- The **average best error before change** (ABEBC) is a measure that records the difference between the optimum value and the quality of the best particle, or error (provided that the target value is known). Knowledge of when an environment change occurs is a prerequisite for using the ABEBC measure, and the measure provides insight about the *exploitative capability* [9] of an algorithm on a given problem. Formally, the measure is defined by:

$$\text{ABEBC} = \frac{1}{n_c} \Sigma_{c=0}^{n_c} (err_{c,r-1}) \tag{7}$$

  where r is the number of iterations between two environmental changes and $err_{c,r-1}$ is the difference between the best fitness and the optimal fitness at iteration t after the last change c; n_c is the total number of environment changes.
- The **average best error after change** (ABEAC) [23] is a measure that determines the *stability* [5] of an algorithm within a dynamic environment. The measure is similar to the ABEBC, except that the error in fitness compared to the global optimum, determined directly after an environment change. As such, the measure favors algorithms that tend to prefer areas of the search space that do not change all that much. The measure is defined by

$$\text{ABEAC} = \frac{1}{n_c} \Sigma_{c=0}^{n_c} err_{c,0} \tag{8}$$

  where $err_{c,0}$ is the error at the iteration directly after an environment change.

4.3 Statistical Process

The performance of the static QPSOs and the self-adaptive QPSO was evaluated using the measures defined in Section 4.2. For each combination of error

measurement, a Mann-Whitney-U rank sum test with Holm correction was used to determine if a significant difference ($\alpha = 0.05$) existed between the algorithm performances. A value of 1 is allocated to an algorithm if the results are superior to the other, and the inferior algorithm is assigned a score of -1. These scores then determine the win/loss ratio of the algorithms.

5 Results

This section contains the analysis of the obtained results for the four algorithms (QPSO-5, QPSO-10, QPSO-50, and self-adaptive QPSO). An analysis of the CME, ABEBC and ABEAC measures follow in Sections 5.1, 5.2 and 5.3 respectively. Section 5.4 analyzes the diversity and radius size of the self-adaptive QPSO.

5.1 Analysis of Collective Mean Error

Table 3 provides algorithm rankings with respect to the CME. For the CME measurement, the rankings indicate that the self-adaptive QPSO performed the best across the different environment types. QPSO-50 was the second best performing algorithm, followed by QPSO-10 and QPSO-5. As shown in Figure 1(a), a similar trend to the ranking data can be observed when comparing algorithm performances. For the abrupt and progressive environments all algorithms achieved similar performances, but the win/loss ratio favors the self-adaptive QPSO within these environments.

After an environment change, the self-adaptive QPSO has an increase in diversity, as illustrated in Figure 1(d). The increase in diversity results in a larger area for quantum particles to explore, and as the swarm starts to converge on an optimum, the radius value decreases. With a decreasing radius, quantum particles begin exploitation of the search space around the optimum. The error values in Figure 1(a) also show that the QPSO is sensitive to the frequency of environment change: the lower error values were achieved for the quasi-static environments where the frequency of change is low. Compared to the static QPSOs, it should be noted that the self-adaptive QPSO did not perform worse.

5.2 Analysis of Average Best Error Before Change

Figure 1(b) illustrates that all four algorithms manages to achieve values of less than 20 for the ABEBC within the quasi-static environments. For the other environments, the same trend of the CME measurement is evident, with none of the algorithms particularly providing a clearly better solution, and a similar spread of error values. Because the ABEBC demonstrates the exploratory capacity of an algorithm, it is clear that none of the algorithms were able to effectively locate a new solution before the environment changed. The self-adaptive QPSO achieved a comparable performance, compared to the static QPSOs.

5.3 Analysis of Average Best Error After Change

After an environment change, the QPSO-10 and self-adaptive QPSO managed to achieve median values that are lower than that of the other QPSO algorithms for the quasi-static environments. Unfortunately, for the other environment types, no one algorithm displayed a clear improvement, and all algorithms (including the self-adaptive QPSO) achieved equally poor results. These performances are illustrated in Figure 1(c).

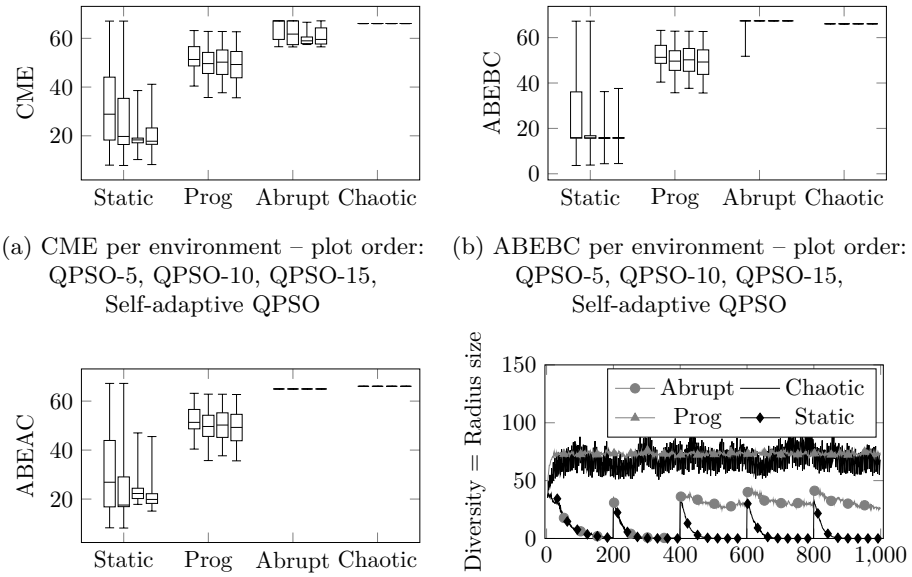
5.4 Analysis of the Dynamic Radius Size and Diversity

For the self-adaptive QPSO, the average radius size is illustrated in Figure 1(d) for each environment type over 1000 iterations. From the graph it is clear that the diversity (which is the cloud radius value), did change over the course of algorithm execution. For environments with high temporal severity (chaotic and progressive environments), the cloud radius size fluctuated at a large value which is roughly half of the problem domain. Due to the frequency of the environment changes, there is not enough time between the environment changes for particles to share enough information in order to attract the swarm to a specific region within the search space. Therefore, the re-initialization process maintains a large diversity.

The quasi-static environment plot shows that the radius reduced to a small value and increased as the environment changed (every 200 iterations), albeit a small change. The size of the cloud radius for the abruptly changing environment reduced similarly to the quasi-static environment, but at 400 iterations, increased to a value under half of the problem domain size and remained there for the remainder of the algorithm execution. It is not clear why this behavior is observed. As expected, the size of the cloud radius remained large for the progressive and chaotic environments where the frequency of environment change is high.

6 Conclusion

This paper investigated if the QPSO could be able to dynamically adapt and maintain the value of the quantum cloud radius, without requiring that the value be defined ahead of time, and without tuning the value for a given optimization problem. A new strategy was suggested, whereby the cloud radius value is based on the diversity of the particle swarm. By allowing the cloud radius value to dynamically adapt during the execution of the algorithm, it was shown that the self-adaptive strategy ranked well against three static quantum cloud radius QPSO algorithms. Even though the results indicated that the self-adaptive strategy did not provide significantly improved results when compared to the static radius QPSO algorithms, the results did indicate that the self-adaptive cloud radius does, generally, perform well and should be preferred. In future work, the influence of different distributions on the performance of the self-adaptive QPSO, and refinements to the adaptive cloud radius strategy, will be explored.



(a) CME per environment – plot order: QPSO-5, QPSO-10, QPSO-15, Self-adaptive QPSO

(b) ABEBC per environment – plot order: QPSO-5, QPSO-10, QPSO-15, Self-adaptive QPSO

(c) ABEAC per environment – plot order: QPSO-5, QPSO-10, QPSO-15, Self-adaptive QPSO

(d) Average diversity/radius size over algorithm iterations

Fig. 1: Measurements over environment types

Table 3: Algorithm performance ranking

Problem	Measure	QPSO-5	QPSO-10	QPSO-50	Self-adaptive QPSO
Quasi-static	CME (Win/Loss)	(0/-3)	(1/-2)	(2/-1)	(3/0)
	ABEBC (Win/Loss)	(0/-3)	(1/-2)	(3/0)	(2/-1)
	ABEAC (Win/Loss)	(0/-3)	(3/0)	(1/-2)	(2/-1)
	Win+Loss	-9	-1	3	5
	Rank	1	2	3	4
Progressive	CME (Win/Loss)	(0/-2)	(2/-1)	(0/-1)	(2/0)
	ABEBC (Win/Loss)	(0/-2)	(2/-1)	(0/-1)	(2/0)
	ABEAC (Win/Loss)	(0/-2)	(2/-1)	(0/-1)	(2/0)
	Win+Loss	-6	3	-3	6
	Rank	1	3	2	4
Abrupt	CME (Win/Loss)	(0/-3)	(1/-2)	(2/0)	(2/0)
	ABEBC (Win/Loss)	(0/-3)	(1/-2)	(2/-1)	(3/0)
	ABEAC (Win/Loss)	(0/-3)	(1/-2)	(2/-1)	(3/0)
	Win+Loss	-9	-3	4	8
	Rank	1	2	3	4
Chaotic	CME (Win/Loss)	(0/-3)	(1/-2)	(2/-1)	(3/0)
	ABEBC (Win/Loss)	(0/-3)	(1/-2)	(2/-1)	(3/0)
	ABEAC (Win/Loss)	(0/-3)	(1/-2)	(2/-1)	(3/0)
	Win+Loss	-9	-3	3	9
	Rank	1	2	3	4
Win/Loss total		-33	1	7	28

References

1. Angeline, P.J.: Tracking extrema in dynamic environments. In: International Conference on Evolutionary Programming. pp. 335–345. Springer (1997)
2. Blackwell, T.: Particle swarm optimization in dynamic environments. In: Evolutionary computation in dynamic and uncertain environments, pp. 29–49. Springer (2007)
3. Blackwell, T., Branke, J.: Multi-swarm optimization in dynamic environments. In: Workshops on Applications of Evolutionary Computation. pp. 489–500. Springer (2004)
4. Blackwell, T., Branke, J., Li, X.: Particle Swarms for Dynamic Optimization Problems, pp. 193–217. Springer Berlin Heidelberg, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-74089-6_6
5. Branke, J.: Memory enhanced evolutionary algorithms for changing optimization problems. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406). vol. 3, p. 1882 Vol. 3 (1999). https://doi.org/10.1109/CEC.1999.785502
6. Branke, J.: The moving peaks benchmark. URL: http://www.aifb.uni-karlsruhe.de/~jbr/MovPeaks/movpeaks (1999)
7. Deb, K., Joshi, D., Anand, A.: Real-coded evolutionary algorithms with parent-centric recombination. In: Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on. vol. 1, pp. 61–66 (May 2002). https://doi.org/10.1109/CEC.2002.1006210
8. Duhain, J.G.: Particle swarm optimisation in dynamically changing environments-an empirical study. Master's thesis, University of Pretoria (2011)
9. Duhain, J.G., Engelbrecht, A.P.: Towards a more complete classification system for dynamically changing environments. In: Evolutionary Computation (CEC), 2012 IEEE Congress on. pp. 1–8. IEEE (2012)
10. Eberhart, R.C., Shi, Y.: Tracking and optimizing dynamic systems with particle swarms. In: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546). vol. 1, pp. 94–100 vol. 1 (2001). https://doi.org/10.1109/CEC.2001.934376
11. Engelbrecht, A.: Roaming behavior of unconstrained particles. In: Proceedings - 1st BRICS Countries Congress on Computational Intelligence, BRICS-CCI 2013. pp. 104–111 (09 2013)
12. Harrison, K., Ombuki-Berman, B.M., Engelbrecht, A.P.: The effect of probability distributions on the performance of quantum particle swarm optimization for solving dynamic optimization problems. In: 2015 IEEE Symposium Series on Computational Intelligence. pp. 242–250 (Dec 2015). https://doi.org/10.1109/SSCI.2015.44
13. Harrison, K.R., Ombuki-Berman, B.M., Engelbrecht, A.P.: A radius-free quantum particle swarm optimization technique for dynamic optimization problems. In: 2016 IEEE Congress on Evolutionary Computation (CEC). pp. 578–585 (July 2016). https://doi.org/10.1109/CEC.2016.7743845
14. Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press (1975)
15. Hu, X., Eberhart, R.: Tracking dynamic systems with pso: where's the cheese. In: Proceedings of the workshop on particle swarm optimization. pp. 80–83 (2001)
16. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proceedings of the IEEE International Joint Conference on Neural Networks. vol. IV, pp. 1942–1948. IEEE (1995)

17. Li, C., Yang, S., Nguyen, T., Yu, E., Yao, X., Jin, Y., Beyer, H., Suganthan, P.: Benchmark generator for cec 2009 competition on dynamic optimization. Tech. rep., University of Leicester, UK (2008)
18. Morrison, R.W.: Performance measurement in dynamic environments. In: GECCO workshop on evolutionary algorithms for dynamic optimization problems. pp. 5–8. Citeseer (2003)
19. Moser, I., Chiong, R.: Dynamic Function Optimization: The Moving Peaks Benchmark, pp. 35–59. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-30665-5_3
20. Pamparà, G., Nepomuceno, F., Leonard, B.: Cilib v2.0.1 (Oct 2014). https://doi.org/10.5281/zenodo.12371, https://doi.org/10.5281/zenodo.12371
21. van der Stockt, S., Engelbrecht, A.P.: Analysis of hyper-heuristic performance in different dynamic environments. In: Computational Intelligence in Dynamic and Uncertain Environments (CIDUE), 2014 IEEE Symposium on. pp. 1–8. IEEE (2014)
22. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization **11**(4), 341–359 (Dec 1997). https://doi.org/10.1023/A:1008202821328, https://doi.org/10.1023/A:1008202821328
23. Trojanowski, K., Michalewicz, Z.: Searching for optima in non-stationary environments. In: Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on. vol. 3, pp. 1843–1850. IEEE (1999)
24. Van Den Bergh, F.: An Analysis of Particle Swarm Optimizers. Ph.D. thesis, Pretoria, South Africa, South Africa (2002), aAI0804353
25. Van Den Bergh, F., Engelbrecht, A.P.: A study of particle swarm optimization particle trajectories. Information sciences **176**(8), 937–971 (2006)