A Review and Empirical Analysis of Particle Swarm Optimization Algorithms for Dynamic Multi-Modal Optimization

Simon Dennis
Computer Science Division
Stellenbosch University
Stellenbosch, South Africa
simondennis9@gmail.com

Andries Engelbrecht
Department of Industrial Engineering,
and Computer Science Division
Stellenbosch University
Stellenbosch, South Africa
engel@sun.ac.za

***Abstract*—A number of particle swarm optimization (PSO) variations have been developed to find multiple solutions to multi-modal optimization problems. These algorithms have been extensively evaluated in the literature. When dynamic optimization problems are considered, only a few PSO algorithms exist that have the ability to find and track multiple optima in dynamically changing search landscapes. These algorithms have not yet been rigorously evaluated on an extensive set of dynamic optimization problems. This paper presents a review of existing dynamic multi-modal PSO algorithms and conducts an empirical analysis of these algorithms on a set of dynamic optimization problems of varying dynamics. The best performing dynamic multi-modal PSO algorithms, with respect to different performance measures, are identified as an outcome of a formal statistical analysis.**

***Keywords*—Particle swarm optimization, multi-modal optimization, dynamic optimization problems**

I. Introduction

Multi-modal optimization [1], [2] deals with optimization tasks where the goal is to find all, or as many as possible, of the optima of a multi-modal problem, instead of finding just one of these optima. Many efficient evolutionary algorithms and swarm-based algorithms have been developed to find multiple solutions to static, single-objective optimization problems [1]–[3]. The task of finding multiple solutions is somewhat more difficult than finding a single sultion, specifically due to the requirement that candidate solutions should not all converge to just one point in the search space. When faced with a dynamic multi-modal optimization problem, the task of finding multiple solutions becomes even more difficult: in addition to locating the positions of optima, it is now also necessary to track optima over time, due to changes in the search landscapes. These changes may result in new optima appearing and existing optima disappearing. In addition, change severity and the frequency at which the search landscape changes further adds to the complexity of dynamic multi-modal optimization.

While research in the development of multi-modal optimization algorithms is in abundance, few studies can be found for dynamic multi-modal optimization [1], though a significant amount of work has been done to develop algorithms to find and track a single optimum in dynamic environments [4]–[6].

This paper focuses specifically on particle swarm optimization (PSO) [7], and the extent to which PSO variations have been developed for dynamic multi-modal optimization. The paper provides a review of PSO variations available in the literature for finding and tracking multiple solutions in dynamically changing search landscapes. Only six such PSO algorithms have been found. An empirical analysis of these algorithms are provided in this paper, with a comparison of these six algorithms on a large set of dynamic multi-modal optimization problems. The paper identifies the best PSO algorithms for dynamic multi-modal optimization with reference to offline error and the ratio of found optima. Based on a formal statistical analysis of the results, two of the six PSO algorithms are identified as the best performers.

The rest of the paper is organized as follows: Section II provides a review of the dynamic multi-modal PSO algorithms found in the literature. Section III reviews the moving peaks benchmark generator, which is used in the empirical analysis to produce instances of a large number of different multi-modal dynamic optimization problems. The empirical process is described in Section IV, and the results are presented and discussed in Section V.

II. Dynamic Multi-Modal Particle Swarm Optimization Algorithms

This section reviews dynamic multi-modal PSO algorithms identified in the literature.

A. Improved Speciation PSO

Li et. al. developed the speciation PSO for multi-modal optimization in static environments [8]. The improved speciation PSO (SPSO) adapts the original speciation PSO to find and track optima in dynamic environments [9].

SPSO groups the PSO population into sub-populations, known as species, on each iteration of the algorithm. These groups are formed using a process known as speciation: The list of all particles is sorted by each particle's fitness function value. The best particle is removed from the list and becomes a species seed – the particle from which a species is created. All particles within a certain radius, r_s, of this species seed are moved from the list of all particles into this species. This process of seed identification and species creation is then repeated until each particle is within a species. Species which

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

only contain one particle is combined to form a single species, called the general swarm. To encourage exploration of the search space in dynamic environments, a limit, p_{max}, is placed on the size of a species. Species containing more than p_{max} particles move their particles with the lowest fitness function value into the general swarm. The general swarm behaves as a standard PSO and focusses on exploring the search space.

Species update their positions by following the update equations of the quantum PSO [10]. The quantum PSO is a dynamic PSO algorithm inspired by quantum physics. Some particles in the species are considered quantum particles, while others are neutral particles. Neutral particles follow the standard PSO update equations. Quantum particles are positioned randomly in an area around the global best position. The best quantum particle will become the global best position if its position improves on the current global best position. This allows species to track the optima they have identified. When the neutral particles within a species have converged, quantum particles will search around the converged position for changes in the environment. If an improved position is found, the neutral particles will start moving towards the new global best position and will follow the optimum as it moves.

B. Memetic PSO

The memetic PSO (MPSO) was developed by Wang et. al. [11]. On each iteration, sub-swarms, known as species, are formed using a speciation mechanism. This speciation mechanism is similar to the speciation process used in SPSO: The list of all particles is sorted by each particle's fitness function value. The best particle is removed from the list and becomes a species seed. The control parameter, r_s, defines the maximum size of a species. At most r_s particles within a certain radius, r_0, of this species seed are moved from the list of all particles into this species. The order in which particles are selected depends on the particle's distance from the species seed, with closer particles being selected first. This speciation process is repeated until each particle is within a species. A species with a size equal to r_s is known as a full species. Species that are not full are combined into a single global species and roam the search space using normal PSO behaviour. The global species explores the search space. Non-global species exploit and track the optimum they are formed around.

Both the global species and non-global species follow the standard PSO update equations. To encourage exploitation within non-global species, the adaptive local search (LS) operator is used. Adaptive LS operates by attempting to improve the neighbourhood best position of each species. One of two search methods are used, depending on the distance between the neighbourhood best's position and its personal best. If this distance is smaller than the algorithm parameter, r_1, non-deterministic cognition-based local search (NCLS) is used. NCLS is a single-particle PSO where only the cognitive component of the velocity update is used. If this simplified PSO results in an improvement, the neighbourhood best position is changed to the improved value. The number of iterations is dependent on the algorithm parameter, n_{ls}. If the distance between the neighbourhood best position and its personal best position is larger than r_1, a random walk with direction exploitation (RWDE) is used. RWDE creates a new position by adding to the current position a vector with elements sampled from the uniform distribution in the range $[0, 1]$. If this new position improves the current neighbourhood best position, then the neighbourhood best position is updated. The number of iterations is also dependent on algorithm parameter n_{ls}.

On each iteration, the diversity of each non-global species is calculated. A species with a diversity less than the control parameter, r_2, is considered to be converged. The solution found by a converged species is added to a memory of solutions. The particles from which this species was formed are reinitialized elsewhere in the search space to search for optima elsewhere. On each iteration, the previous fitness value of the best particle in the entire population is checked against its current fitness value before this particle's position is changed. If these two fitness values are not equal, the algorithm assumes that an environment change has occurred. When an environment change is detected, all solutions in the memory are reintroduced into the search space as particles in the global swarm. This introduces knowledge of previous optimum positions into the population, encouraging particles in the global species to form species around these positions. While doing so, the global species will likely find the location of the new optimum if it is nearby. In this way, tracking of the optima is achieved.

C. Dynamic Vector-Based PSO

The dynamic vector-based PSO (DVBPSO), developed by Schoeman and Engelbrecht [12], is based on the vector-based PSO (VBPSO) [13], which was developed to find multiple solutions to static multi-modal optimization problems. It uses niching to find multiple optima in the search space. Niches in VBPSO are formed from a species seed, which is the best particle in the population not yet within a niche. For all other particles in the population which are not within a niche, two vectors are calculated, namely the personal best vector $\vec{v}_p$, i.e. the vector between the current position and personal best position, and the niche seed vector, $\vec{v_g}$, i.e. the vector between the current position and the species seed. If the dot product, $\vec{v}_p \cdot \vec{v}_g$ is greater than zero, i.e. the particle's personal best vector is in a similar direction to the niche seed vector, then it is included in the niche. This niching process is repeated until all particles are within a niche. New particles are initialized in niches with fewer than three particles to enable PSO to operate effectively within a niche. Niches operate independently using the standard PSO update equations.

On the first iteration of DVBPSO, VBPSO is executed using randomly initialized particles to find the optima in the current environment. The fitness function value of these optima is calculated. On subsequent iterations, the fitness of each optimum is recalculated and is compared to the fitness from the first iteration. If these two fitness values are not equal, then the algorithm assumes an environment change has occurred. When an environment change occurs, VBPSO is executed with particles initialized at the previous optima and with three more particles per optimum initialized in the space around these optima. By using this re-initialization scheme, optima can be tracked because VBPSO is assisted with knowledge of where the previous optima were. The fitness of these new optima is calculated and is used to detect future changes in the

environment. When the next environment change is detected, VBPSO is executed under the same conditions.

D. Fractional GBest Multi-Swarm PSO

The fractional gbest multi-swarm PSO (FGBMSPSO) was developed by Pulkkinen et. al. [14]. FGBMSPSO uses a multi-swarm method to find multiple optima. Each sub-swarm of the multi-swarm method is created before execution of the algorithm begins. All particles in the population are distributed evenly among the sub-swarms. The number of sub-swarms is determined by the control parameter, n_{swarms}. Sub-swarms operate separately from each other. They follow the standard PSO rules with fractional gbest formation (FGF). FGF is a process that extends PSO by attempting to improve the gbest particle of a sub-swarm on each iteration. It does so by considering the positions which can be created by combining the dimensional components of different particles in the swarm. The best combination of dimensional components is known as the artificial gbest. If the artificial gbest has a better fitness function value than the current gbest, the current gbest is replaced with the artificial gbest.

The only interaction between each sub-swarm is a repelling mechanism, executed on each iteration, which prevents two sub-swarms from converging to the same optimum. The repelling mechanism operates by calculating the Euclidean distance between the gbest positions for each sub-swarm pair. If this distance is less than the minimum allowable distance between two sub-swarms, defined by control parameter r_{rep}, then the sub-swarm with the worse gbest position is randomly re-initialized in the search space. This repulsion mechanism encourages the algorithm to explore the search space for other optima, and not to waste computational effort exploiting an optimum that another sub-swarm is already exploiting.

On each iteration, the previous fitness function value of the best particle in the population is compared to its current fitness function value before the position of this particle is modified. If these two values are not equal, the algorithm assumes that an environment change has occurred. When an environment change occurs, each particle in the population obtains a new random velocity. This forces each sub-swarm out of the convergent state and allows it to track changes to the optimum which it was previously converged to.

E. Multi-Swarm Quantum PSO

The multi-swarm quantum PSO (MQSO) was developed by Blackwell and Branke [15]. MQSO uses a multi-swarm method to find multiple optima. Each sub-swarm of the multi-swarm method is created before execution of the algorithm begins. All particles in the population are distributed evenly among the sub-swarms. The number of sub-swarms is determined by the control parameter, n_{swarms}. Each sub-swarm acts separately and follows the rules of the quantum PSO [10]. This is the same quantum PSO used within the species of SPSO, described in section II-A. The quantum PSO enables MQSO to track changes in the optima as they occur.

To prevent multiple sub-swarms converging to the same optimum, an exclusion mechanism is used. The exclusion mechanism is the same as the repulsion mechanism of the FGBMSPSO, described in section II-D. It operates similarly by calculating the Euclidean distance between the gbest positions for each sub-swarm pair. If this distance is less than the minimum distance allowed between sub-swarms, defined by control parameter r_{excl}, then the sub-swarm with the worse gbest position is randomly re-initialized in the search space. This exclusion mechanism encourages sub-swarms to search for other optima, and not to waste unnecessary effort exploiting an optimum that another sub-swarm is already exploiting.

F. Adaptive Multi-Swarm PSO

The adaptive multi-swarm PSO (AMSPSO), developed by Li et. al. [16], uses a multi-swarm method where the number of swarms, and therefore the number of particles, changes dynamically over time. Particles are initially grouped into sub-swarms using a clustering process: The distance between each pair of particles in the population is calculated. Each sub-swarm is formed by grouping the particles which are closest to each other together into a sub-swarm. Each sub-swarm executes separately from the other sub-swarms to allow each sub-swarm to converge to a different optimum.

Sub-swarm behaviour is governed using standard PSO update equations and a global best improvement operator, called gBestLearn, which executes when a particle improves its personal best. When gBestLearn is executed on a particle, each dimension in the personal best has a small chance to be exchanged with the corresponding dimension value in the global best. If this new gbest position improves the fitness function value of the existing gbest, then the change is kept. Otherwise, the original gbest is kept. The chance to exchange dimensional values depends on the difference between the two values. Exchanges are more likely to occur when the difference is small, and less likely when the difference is large.

Sub-swarms which converge to the same optimum are merged into the same sub-swarm. Therefore, the number of sub-swarms will decrease over time until there is one swarm at each optimum. If the number of sub-swarms does not decrease after a certain number of iterations, defined by control parameter δ, the number of particles is changed to optimize the search process. If more sub-swarms were created since the previous change in number of particles occurred, then the algorithm assumes that there are fewer sub-swarms than optima and increases the total number of particles. If less sub-swarms were created since the previous change in number of particles, the algorithm assumes there are more particles than necessary and the total number of particles is decreased. All new individuals, if any, are then grouped into sub-swarms using the same clustering process as before.

Sub-swarms which have a swarm diversity less than the control parameter η are considered converged. The behaviour of converged sub-swarms is similar to the MPSO described in section II-B. When a sub-swarm is converged, then its optimum is added to a list of converged positions. The sub-swarm is then re-initialized into the search space so that it may find new optima. Whenever the number of particles in the search space increases, particles from the list of converged positions are added first, and then randomly initialized particles fill the remaining required number of particles if necessary. This allows the positions of previous optima to be transferred to new sub-swarms being formed, and allows these optima to be tracked as their positions change.

III. Moving Peaks Benchmark Problem Generator

The moving peaks benchmark is a problem generator for dynamic optimization problems, developed by Branke [17]. This generator creates a search landscape consisting of multiple peaks of varying size. Each peak with position $\vec{x_i}$ periodically moves a random distance in a random direction, and randomly adjusts its width w_i and height h_i. The severity of these random changes is controlled with benchmark parameters x_s, w_s and h_s respectively. The parameters $\vec{x}_{max}$, $\vec{x}_{min}$, w_{max}, w_{min}, h_{max} and h_{min} define the upper and lower bounds of these changes respectively. When a dynamic multi-modal optimization algorithm is executed on a landscape produced by the moving peaks benchmark, the algorithm's objective is to locate the highest point of each peak, and to then track each peak's movement through the search space. Equations (1), (2) and (3) describe how peaks move in the moving peaks benchmark; $\vec{v}$ in equation (1) is a vector of random values sampled from the uniform distribution in the range $[0, 1]$ and scaled by x_s.

$$x_i(t+1) = \begin{cases} 2 \cdot x_{min} - x_i(t) - \vec{v} & x_i(t) + \vec{v} < x_{min} \\ 2 \cdot x_{max} - x_i(t) - \vec{v} & x_i(t) + \vec{v} > x_{max} \\ x_i(t) + \vec{v} & \text{otherwise} \end{cases} \quad (1)$$

$$h_i(t+1) = \min(h_{max}, \max(h_{min}, h_i(t) + h_s \cdot N(0,1))) \quad (2)$$

$$w_i(t+1) = \min(w_{max}, \max(w_{min}, w_i(t) + w_s \cdot N(0,1))) \quad (3)$$

In the above, $N(0, 10$ refers to the normal distribution with a zero mean and unit variance.

Du Plessis and Engelbrecht [18] extended the original moving peaks benchmark problem by including a mechanism to dynamically change the number of peaks n in the environment. When the peak positions are adjusted, n is either increased or decreased. This change is random, but its severity is controlled by the variability parameter, n_s. The maximum and minimum number of peaks are also defined by parameters n_{max} and n_{min} respectively. Equation 4 describes how the number of peaks changes in the moving peaks benchmark.

$$n(t+1) = \begin{cases} \max(n_{min}, n(t) - n \cdot n_s \cdot U(0,1)) & r < 0.5 \\ \max(n_{max}, n(t) + n \cdot n_s \cdot U(0,1)) & \text{otherwise} \end{cases} \quad (4)$$

In the above, $r \sim U(0, 1)$, and $U(0, 10)$ refers to the uniform distribution in the range $[0, 1]$.

IV. Empirical Procedure

This section describes the experimental process, empirical method, and performance measurements used used to compare the six dynamic multi-modal PSO algorithms.

A. Experiments

The six dynamic multi-modal optimization algorithms are evaluated and compared on 900 different parametrizations of the moving peaks benchmark generator. Each parametrization is a combination of the following parameters:

- Period, i.e. the number of iterations between environment changes: 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000. The total number of iterations is 10 times the period.
- Movement severity, x_s: 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100.
- Peak variability, n_s: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9. Note that each simulation starts with 10 initial peaks.

For each problem instance, the search space boundary is defined as $[0, 100]^d$, where the number of dimenions, d, is 10.

Due to the random nature of PSO and the moving peaks benchmark, 48 independent trials are performed for each experiment. Each algorithm is executed with a total number of 50 particles to ensure enough particles are available for sub-populations to be formed with. The number of iterations executed by each algorithm is set to be 10 times the benchmark period to ensure that enough environment changes can occur for performance in the dynamic environment to be measured. Each algorithm is otherwise parametrized with the parameters used by the articles they were first proposed in [9], [11], [12], [14]–[16], and as summarized in the tables provided in the appendix. Control parameters were not tuned, due to the fact that environment changes are unknown, and it is not possible to tune control parameters for solving dynamic optimization problems [19], [20]. Control parameters that are optimal for a current environment is not necessarily optimal for the next environment, especially if there are significant changes in landscape characteristics.

B. Performance Measurements

Four performance measurements are recorded:

- **Offline error:** Offline error, originally developed by Branke for use in dynamic optimization algorithms [4], is the average distance over all environment changes between the optimum found by an algorithm and the actual optimum position. The distance is calculated at the moment before the environment changes. In a multi-modal environment, the distance is calculated for each sub-population as the distance to the closest known optimum from the solution represented by the global best position of that sub-population. The net offline error is calculated as the average of these distances over all sub-populations, averaged over all environment changes. The offline error is calculated to determine how good a solution discovered by an algorithm is relative to the most optimal solution possible.
- **Ratio of found peaks:** The ratio of the number of sub-populations which have found an optimum at or near a peak to the total number of peaks. This is calculated just prior to each environment change, and is averaged over all environment changes. This ratio of found peaks is calculated to determine how many of the optima in the search space are identified by the algorithm, indicating the algorithm's multi-modal abilities.

- **Offline diversity:** The diversity of each sub-population is measured just before each environment change. The diversity of each sub-population is calculated as the average distance of particles in that sub-population from the average over all of the particles in the sub-population. These sub-population diversities are averaged per environment. These averages are then averaged over all environment changes. The offline diversity is calculated to determine the level of convergence an algorithm achieves.
- **Infeasible particles**: Particles have shown to exhibit roaming behavior: within the very first iterations many particles leave the boundaries of the search space and explore infeasible space [21], [22]. This roaming behavior wastes unnecessary computational effort by exploring infeasible space. The average number of particles for which at least one dimension violates a boundary constraint is calculated as an average over all iterations. Note that nothing is done to prevent roaming behavior in order to evaluate the extent to which the different algorithms have particles that violate boundary constraints.

C. Empirical Method

The performance of each algorithm is evaluated over all experiments: The average and standard deviation over all trials is calculated for each experiment. Additionally, Friedman ranking [23] is calculated for each experiment, for each performance measurement over all trials of each algorithm. The ranking process is a pairwise comparison between each pair of algorithms where an algorithm's rank increases by one if it performs better than the other algorithm, or decreases by one if it performs worse than the other algorithm. The average Friedman rank over all experiments is calculated for each algorithm, for each performance measurement.

Since the objective of a dynamic multi-modal optimization algorithm is to accurately locate and track multiple optima, a low offline error is ranked higher than a high offline error. Similarly, a high ratio of found peaks is ranked higher than a low ratio of found peaks. Diversity is a measurement of how converged the sub-populations within the algorithm are, and therefore neither a high or low diversity is preferable due to both high and low convergence being advantageous, especially in a dynamic environment. Diversity is simply ranked with a low diversity being ranked higher than a high diversity.

V. Results and Discussion

This section analyses and compares the performance of the existing dynamic multi-modal optimization algorithms. Due to space limitations detailed tables with results are not provided, and only the observed trends are discussed. However, the section ends with a summary of the results over all the dynamic optimization problems and different problem dynamics in Table I.

A. General Trends

All of the algorithms have worse offline error performance for the highly variable environments compared to the slightly variable environments. All of the algorithms find fewer optima for highly variable environments than for slightly variable environments, except for MPSO which consistently finds a low 5% of the optima irrespective of the type of environment change. Diversity of the sub-populations within all of the algorithms is not affected by the severity of optima movement.

B. Speciation Particle Swarm Optimization

SPSO has poor offline error performance on average. Offline error performance is particularly poor for slowly changing or large movement environments. On average, SPSO finds 23% of the optima. More optima are found for slowly changing environments – up to 42% of the optima. SPSO finds more optima than any other algorithm when the period between changes is 2000 or longer. The number of optima found by SPSO is not affected by the severity of optima movement within the environment. The diversity of the sub-populations is on average lower for slowly changing environments and higher for quickly changing environments. SPSO exhibits little roaming behavior. However, more particles search outside the search space for highly variable, large movement environments than for slightly variable, small movement environments.

C. Memetic Particle Swarm Optimization

MSPO has good offline error performance on average – the second best offline error compared to the other algorithms. Offline error performance is better for slowly changing environments than for quickly changing environments. MPSO has a better offline error than any other algorithm when the period between changes is 1000 or longer. Offline error performance is slightly worse for large movement environments compared to small movement environments. MPSO finds 5% of the optima on average, with a standard deviation of 5%. This implies that MPSO only ever locates one or two optima. This is the fewest number of optima found by any of the algorithms. The diversity of the sub-populations is similar for every environment. MPSO has almost no particles (only 0.1% on average) searching outside the search space. More MPSO particles search outside the search space in highly variable, large movement environments than in slightly variable, small movement environments.

D. Dynamic Vector-Based Particle Swarm Optimization

Over all of the problems, DVBPSO ranked third with respect to offline error performance. Change frequency does not affect DVBPSO's offline error. However, DVBPSO has a worse offline error for high movement environments compare to low movement environments. DVBPSO finds the largest number of optima on average, i.e. 40%. Fewer optima are found for high movement environments than for low movement environments. DVBPSO has similar sub-population diversity for all environments. Out of all the algorithms, DVBPSO exhibits the strongest roaming behavior. More DVBPSO particles search outside the search space in highly variable, large movement environments than in slightly variable, small movement environments.

E. Fractional GBest Multi-Swarm Particle Swarm Optimization

FGBMSPSO has the best offline error performance over all of the problems when compared to the other algorithms.

The severity of optima movement does not effect offline error performance. FGBMSPSO has better offline error performance for slowly changing environments than for quickly changing environments. FGBMSPSO finds 25% of the optima on average. The number of optima found by FGBMSPSO is not affected by the severity of optima movement. The diversity of the sub-populations is lower for slowly changing environments than for quickly changing environments. Diversity is larger for environments which are highly variable than for environments which are slightly variable. FGBMSPSO shows no roaming behavior.

F. Multi-Swarm Quantum Particle Swarm Optimization

Over all of the problems, MQSO has poor offline error performance, though better than AMSPSO and SPSO. Offline error is better for slowly changing environments than for quickly changing environments, and worse for high movement environments. MQSO finds the second lowest number of optima (only 18% of the optima). MQSO finds more optima for high movement environments than for low movement environments. MQSO sub-populations have a lower diversity for slowly changing environments than for quickly changing environments. MQSO exhbits stronger roaming behavior than FGBMSPSO, MPSO and SPSO, but less that the other algorithms. More MQSO particles search outside the search space in highly variable, large movement environments than in slightly variable, small movement environments. Additionally, more particles search outside the search space in slowly changing environments than in quickly changing environments.

G. Adaptive Multi-Swarm Particle Swarm Optimization

AMSPSO has poor offline error performance over all of the problems – the second worst. AMSPSO offline error performance is the worst for quickly and moderately changing environments when compared to the other algorithms. However, the offline error performance is better than SPSO for slowly changing environments. AMSPSO finds the second highest number of optima (28%) compared to the other algorithms. Only DVBPSO finds more optima than AMSPSO. More optima are found for slowly changing environments than for quickly changing environments. The same number of optima are found in low, medium and high movement environments. The diversity of the sub-populations AMSPSO is lower for quickly changing environments than for moderately changing environments. However, sub-population diversity for slowly changing environments is similar to sub-population diversity for moderately changing environments. Sub-populations diversity is larger for environments which are highly variable than for environments which are slightly variable. AMSPSO exhibts the second strongest roaming behavior. More particles search outside the search space for slowly changing environments than for quickly changing environments.

H. The Best Performing Algorithms

Table I presents the results for the performance measures over all DOP instances. The results are presented over all combinations of change frequencies, change severities, and variabilities in the number of peaks.

The two best performing algorithms are FGBMSPSO and DVBPSO. FGBMSPSO has the best offline error performance, but only finds 25% of the optima. FGBMSPSO performs well for all environments and does not waste computational effort by searching outside the search space. DVBPSO finds the most optima – on average 40% of the optima. However, DVBPSO has average offline error performance compared to the other algorithms and has the storngest roaming behavior.

The DVBPSO also resulted in the smallest diversity measures. The small diversity measures of the DVBPSO is an indication that convergence may be premature, when considering the large difference between the offline error of the DVBPSO and the FGBMSPSO.

VI. Conclusions

This paper provided a review of particle swarm optimization (PSO) algorithms developed to find and track multiple optima in dynamically changing search landscapes. Six dynamic multi-modal PSO algorithms were discussed, and the performance of these algorithms were analyzed on a large number of dynamic optimization problems. These problems differed in three main aspects, namely change frequency, change severity, and variation in the number of peaks.

The results show that, over all of the dynamic optimization problems, the fractional global best multi-swarm PSO algorithm performed the best with reference to the offline error, with the memetic PSO performing second best. The dynamic vector-based PSO, on the other hand succeeded in locating and tracking most of the optima, albeit at a larger offline error. Future research should investigate the reasons for the premature convergence of the DVBPSO. Solutions to this problem may result in a dynamic multi-modal PSO algorithm that still finds the most optima, and delviers better offline error.

While the results presented in this paper do show that some dynamic multi-modal PSO algorithms were successful with respect to offline error, the best average offline errors is still large. More importantly, the best performance with respect to number of optima found is not good, with only 40% of the optima found on average. This finding provides scope for the development of more efficient dynamic multi-modal PSO algorithms.

References

[1] X. Li, M. Epitropakis, K. Deb, and A. Engelbrecht, "Seeking multiple solutions: An updated survey on niching methods and their applications," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 4, pp. 518–538, 2017.

[2] M. Preuss, *Multimodal optimization by means of evolutionary algorithms*, ser. Natural Computing Series. Springer, 2015.

[3] O. Shir, "Niching in evolutionary algorithms," in *Handbook of Natiral Computing*. Springer, 2012, pp. 1035–1069.

[4] J. Branke, *Evolutionary Optimization in Dynamic Environments*. Springer, 2002.

[5] T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 6, pp. 1–24, 2012.

[6] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: Algorithms and applications," *Swarm and Evolutionary Computation*, vol. 33, pp. 1–17, 2017.

[7] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the Sixth International Symposium on Micromachine and Human Science*, 1995, pp. 39–43.

TABLE I. PERFORMANCE OVER ALL PROBLEM INSTANCES

Performance Measure	Friedman Ranking					
	1st	**2nd**	**3rd**	**4th**	**5th**	**6th**
Offline error	FGBMSPSO	MPSO	DVBPSO	MQSO	AMSPSO	SPSO
	271.0320	345.6129	3956.5755	5056.3352	5223.2785	5956.8402
	(370.0163)	(414.4840)	(4035.1685)	(5146.6371)	(5284.5984)	(6224.5222)
Ratio of found optima	DVBPSO	AMSPSO	FGBMSPSO	SPSO	MQSO	MPSO
	0.3985	0.2779	0.2483	0.2344	0.1805	0.0501
	(0.4187)	(0.2862)	(0.2525)	(0.2471)	(0.1860)	(0.0501)
Offline diversity	DVBPSO	AMSPSO	FGBMSPSO	SPSO	MQSO	MPSO
	0.0232	14.0299	28.7500	44.0810	94.4542	453.2879
	(0.0566)	(14.1137)	(30.8633)	(46.6114)	(228.5872)	(454.1332)
Infeasible particles	FGBMSPSO	MPSO	SPSO	MQSO	AMSPSO	DVBPSO
	0.0000	0.0017	0.0348	0.1346	0.1577	0.1884
	(0.0000)	(0.0174)	(0.1146)	(0.1663)	(0.1712)	(0.1954)

[8] X. Li, J. Branke, and T. Blackwell, "Particle swarm with speciation and adaptation in a dynamic environment," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, 2006, pp. 51–58.

[9] D. Parrott and X. Li, "Locating and tracking multiple dynamic optima by a particle swarm model using speciation," *IEEE Transactions on Evolutionary Computation*, vol. 10, pp. 440–458, 2006.

[10] T. Blackwel and J. Branke, "Multi-swarm optimization in dynamic environments," in *Workshop on Applications of Evolutionary Computation*, 2004, pp. 489–500.

[11] H. Wang, S.Yang, W. Ip, and D. Wang, "A memetic particle swarm optimisation algorithm for dynamic multi-modal optimisation problems," *International Journal of System Science*, vol. 43, no. 7, pp. 1268–1283, 2012.

[12] I. Schoeman and A. Engelbrecht, "Niching for dynamic environments using particle swarm optimization," in *Proceedings of the International Conference on Simulated Evolution and Learning*, 2006, pp. 134–141.

[13] ——, "A novel particle swarm niching technique based on extensive vector operations," *Natural Computing*, vol. 9, no. 3, pp. 683–701, 2010.

[14] J. Pulkkinen, S. Kiranyaz, and M. Gabbouj, "Dynamic multi-swarm particle swarm optimization with fractional global best formulation," in *AI and Machine Consciousness, Proceedings of the 13th Finnish Artificial Intelligence Conference STeP*, 2008, pp. 52–59.

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

[16] C. Li, S. Yang, and M. Yang, "An adaptive multi-swarm optimizer for dynamic optimization problems," *Evolutionary Computation*, vol. 22, no. 4, pp. 559–594, 2014.

[17] J. Branke, "Memory enhanced evolutionary algorithms for changing optimization problems," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 3, 2009, pp. 1875–1882.

[18] M. du Plessis and A. Engelbrecht, "Self-adaptive differential evolution for dynamic environments with fluctuating numbers of optima," in *Metaheuristics for Dynamic Optimization*, ser. Studies in Computational Intelligence. Springer, 2013, vol. 433, pp. 117–145.

[19] B. Leonard and A. Engelbrecht, "On the optimality of particle swarm parameters in dynamic environments," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2013.

[20] K. Harrison, A. Engelbrecht, and B. Ombuki-Berman, "Optimal parameter regions and the time-dependence of control parameter values for the particle swarm optimization algorithm," *Swarm and Evolutionary Computation*, vol. 41, pp. 20–35, 2018.

[21] A. Engelbrecht, "Roaming behavior of unconstrained particles," in *Proceediings of the BRICS-CCI*, 2013.

[22] S. Helwig and R. Wanka, "Theoretical analysis of initial particle swarm behavior," in *Proceedings of the Tenth International Conference on Parallel Problem Solving from Nature*, 2008, pp. 889–898.

[23] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 675–701, 1937.

APPENDIX

This appendix summarizes the control parameter values for the algorithms used in this study.

TABLE II. COMMON PARAMETERS FOR EVERY ALGORITHM, UNLESS STATED DIFFERENTLY

Algorithm Parameter	Value
Number of Particles	50
Number of Iterations	10 × benchmark period
w	0.7298
c_1	1.4962
c_2	1.4962

TABLE III. PARAMETERS FOR IMPROVED SPECIATION PSO

Algorithm Parameter	Value
c_1	2.05
c_2	2.05
r_s	0.5
r	0.5
$rCore$	0
p_{max}	10

TABLE IV. PARAMETERS FOR MEMETIC PSO

Algorithm Parameter	Value
r_0	10
r_s	2
p_{ls0}	1
p_{lsmin}	0.1
p_{lsmax}	1
n_{ls0}	5
n_{lsmin}	1
n_{lsmax}	5
β	0.2
γ	0.5
r_1	0.01
r_2	0.001
λ	0.5
σ	0.5

TABLE V. PARAMETERS FOR DYNAMIC VECTOR-BASED PSO

Algorithm Parameter	Value
g	10
k	10
c	500

TABLE VI. PARAMETERS FOR FRACTIONAL GBEST MULTI-SWARM PSO

Algorithm Parameter	Value
n_{swarms}	6
r_{rep}	10

TABLE VII. PARAMETERS FOR MULTI-SWARM QUANTUM PSO

Algorithm Parameter	Value
c_1	2.05
c_2	2.05
n_{swarms}	6
n_{neutral}	10
r_{cloud}	10
r_{excl}	10

TABLE VIII. PARAMETERS FOR ADAPTIVE MULTI-SWARM PSO

Algorithm Parameter	Value
w	0.6
c_1	1.7
c_2	1.7
α	3
β	0.5
δ	10
η	0.0001
Change in particles step size	10
Maximum no. of particles	300
Minimum no. of particles	70
Maximum sub-swarm size	7
Initial no. of particles	100