IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15821

A Comparative Analysis of Starling Swarm Algorithm for Optimisation in Dynamic Environment

Karatu Musa Tanimu, Wei Pang, and George M. Coghill

Department of Computing Science, King's Collage, University of Aberdeen, UK.

Abstract. Many real-world phenomena can be modelled as dynamic optimisation problems, where the environment of a problem is dynamic and therefore, conventional methods are not capable of dealing with such situations. In this paper, a novel starlings swarm algorithm is proposed for dealing with such dynamic environments. The proposed starlings algorithm consists of a swarm, which has two distinct behaviours to change in their environment that is mutually exclusive, namely: collective response in terms of communication within the seven nearest neighbours and murmuration in terms of patrolling the search space. This way a diverse population is created while the seven nearest neighbours progressively track the dynamic changes. We established that in problems with a moderate shift severity the best variants are those with topology similar to starlings and results confirmed a significant improvement in those cases that used starlings swarm algorithm, which eliminates the use of independent multi-swarm in the search space.

Keywords: Swarm Intelligence, starlings, Optimisation algorithm, bio-inspired, Environment dynamic.

I. INTRODUCTION

Many real-world optimisation problems are dynamic by nature. An example is the traffic condition in a city which changes continuously as new jobs are added to the schedule, and possible machines breaking down or wear out can also impact on the schedule. This kind of condition, a continuous situation, is time dependent; unlike optimisation in static environment, dynamic environments like this are tasked to keep track of the changes in the environment and update *the changes* with respect to time.

Researchers in dynamic optimisation attempted methods that have been successful in static optimisation [1, 11, 30, 32]. Particle Swarm Optimisation (PSO) [3,4,13] is noted as widely used in this context because of its effectiveness and ease of implementation. We infer from literature, when PSO is adapted for dynamic optimisation, it converges to an optimum but perform badly in a changing environment where the optima is dynamic. Consequently, their inability to adapt to changes in dynamic environment.

The two important issues that have been identified in literature as regards to adapting PSO for Dynamic Optimisations (DOPs) are, *outdated memory* and *diversity loss*. *Outdated memory* describes the situation when the global best solutions obtained so far by the algorithm are no longer true; the *diversity loss* occurs when the global optima shifted away from a converged swarm, in that case, if the swarm has a significant level of convergence, their ability to adapt to a change in the environment is severed. This behaviour is quite similar to being trapped in local optima in multi-modal optimisation problems. The first issue of adapting PSO for DOPs is perceived as relatively easy to solve (e.g. updating particle and swarm memories), the *diversity loss* problem has been found to be more challenging to deal with [7].

An effective multi-swarm PSO (mPSO) approach was proposed by Blackwell and Branke [5] to overcome the above issues. Basically, this approach includes several swarms moving independently in the search space with the aim of simultaneously exploring several optima at the same time. To solve the diversity loss issue, each swarm is equipped with two types of particles: classical (neutral) particles and charged or quantum particles. The latter ones are devoted to maintain the diversity during the run (e.g. in every algorithm iteration). Depending on the use of charged or quantum particles, one can obtain two slightly different algorithms: multi-swarm charged PSO (mCPSO) and multi-swarm quantum PSO (mQSO). The mPSO approach was tested on different multimodal problems showing a superior performance over state-of-art algorithms, such as: PSO with partial re-initialisation [19], hierarchical PSO [20], and self-organising scouts [9].

There have been successful attempt to solve the *diversity loss* problem [19, 20,9,5], we however believe that these approaches can be further improved. This motivated the development of a novel bio-inspired algorithm, "Starling Swarm Algorithm for Optimisation in Dynamic Environment". This approach is expected to have the ability of keeping track of the changes that occurred in the environment by following as closely as possible the dynamically changing environment and improve diversity.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026
DOI: 10.17148/IJARCCE.2026.15821

Starlings as prey have a way of tracking single or multiple predator attacks and responding to danger that may be visible to only a small fraction of individuals in the flock [2]; they can evade predators and thus, can detect and adapt to changing environment. We believe the collective response of starlings against peregrine falcon attack can inform a novel meta-heuristic algorithm. Starlings swarm apply their experience based on information transmitted among at least seven nearest neighbours about the predator to detect and track such perturbation [18]. The response to attacks is based on their detection ability, collective response, topology rule and murmuration.

The proposed starling algorithm consists of a swarm; the swarm has two distinct behaviours to changes in their environment which are, *collective response in terms of communication within their seven nearest neighbours* and *murmuration in terms of patrolling the search space*. A starling operation in these two behavioural modes is mutually exclusive (i.e. a starling can only operate in one of the modes at any given time).

II. BACKGROUND AND RELATED WORKS

In this research we focus on problems with dynamic objective functions, that is, the environment is changing over time. Denoting Ω as a feasible space, a Dynamic Optimisation Problem (DOP) can be defined as the set of fitness functions $f(x, t) : \Omega \rightarrow R(t \in N)$, where the goal is to find the set of global optimal solution x^* in the solution set X^t at every time t : $X^t = \{x^* \in \Omega | f(x*, t) \succeq f(x, t), \forall x \in \Omega\}$where is a comparison relation which implies *is better or equals*, thus, $\succeq \in \{\leqslant, \geqslant\}$.

Dynamism in an environment implies a recurrent or non-recurrent changes in the environment at a time interval, and these changes may occur discretely or continuously. Three kinds of dynamic problems are defined in [14] as follows: the changing location in problem search space of the optimal value(s), optimal value(s) changing while the location remain constant and both the location and the optimal value(s) varies. Furthermore, in a multi-dimensional search space, these variations can occur on one or more dimensions, either independently or simultaneously [10].

Now if the changes in the environment only results in small deviations between two successive objective functions, then a good strategy will be to use the previously found best solutions as starting points of finding the new ones. For that reason the main objective of our algorithm is not only to find the current optima, but to also progressively track it as closely as possible and improve the diversity. The starlings swarm offers a unique approach that we conceive prospective in this respect, and has not been applied in an attempt to solve dynamic environment problems. Among the types of benchmarks for dynamic environment in literature, DF1 generator [29], Moving Peak Benchmark (MPB) [8] and the Generalised Dynamic Benchmark Generator [24] are arguably the popularly used benchmarks; in this paper we propose to use MPB and other benchmark functions.

The major advantage of MPB is the direct control it provides over the characteristics of environmental change. A typical MPB landscape consists of several peaks whose positions, heights, and widths can change over time. Conceptually, it can be represented as:

$$F(x,t) = \max_{i=1,\ldots,m} \left[\frac{H_i(t)}{1 + W_i(t)\|x - X_i(t)\|^2} \right], \qquad (1)$$

where $H_i(t)$, $W_i(t)$, and $X_i(t)$ represent the height, width, and location of peak *i* at time *t*.

The Moving Peaks Benchmark (MPB) is particularly suitable for evaluating optimisation algorithms in dynamic environments because it provides an intuitive and highly controllable representation of environmental change through variations in peak position, height, and width. Compared with the Generalised Dynamic Benchmark Generator, MPB offers greater interpretability when analysing an algorithm's ability to track moving optima, since environmental changes can be directly associated with observable modifications of individual peaks. Although DF1 also supports dynamically changing multimodal landscapes, MPB provides a well-established experimental framework in which the number of peaks, dimensionality, change frequency, and change severity can be systematically controlled. Consequently, MPB facilitates detailed assessment of change detection, population diversity, adaptation speed, and tracking error, making it particularly appropriate for studies concerned with continuous adaptation to non-stationary environments.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15821

A. Particle Swarm Optimisation

Particle Swarm Optimisation algorithms - PSO developed by Kennedy and Eberhart [13] is based on the collective group behaviour of organisms such as school of fish, insect swarming or birds flocking, whereby the group attempts to meet collective objectives through a two-way feedback among members within the group. PSO is used for problems where the function to be optimised is discontinuous, with too many non-linearly related parameters [15]; the algorithms operate in a sequence of few iterative steps defined on the behaviour of the swarm it emulates [10]. Each particle in the swarm senses potential solution in the search space and signal in proportional to the suitability of the candidate solution to the other particles in the swarm and the global best solution is adopted by the swarm, based on a fitness [17]. In each iteration, a particle's velocity and the particle's position is updated as follows:

$$\vec{V}_{k(t+1)} = \omega\vec{v}_{k(t)} + c_1.r_1(\vec{P}_{k(t)} - \vec{x}_{k(t)}) + c_2.r_2(\vec{P}_{g(k)} - \vec{x}_{k(t)}), \qquad (2)$$

$$\vec{X}_{i(t+1)} = \vec{x}_{i(t)} + \vec{V}_{k(t+1)}, \qquad (3)$$

Where ω is the *inertia weight* factor, c_1 expresses the confidence a particle has in itself and c_2 expresses the confidence a particle has in its neighbours, r_1 and r_2 are random real numbers between 0 and 1. The value of ω affects the overall behaviour of the swarm. Higher values of *inertia weight* makes the swarm better in exploration but slow in convergence. Lower values make the swarm converge faster but also increase the chances of a premature convergence.

For particle k after t iterations, $\vec{v}_{k(t)}$ is its current velocity. $\vec{x}_{i(t)}$ its current position. $\vec{P}_{k(t)}$ its personal best. $\vec{P}_{g(k)}$ the global best after t iterations. The standard Particle Swarm Optimisation algorithms and its variants have performed well for static environment. Yet, it is shown that PSO, like other evolutionary algorithms, must be modified to not only find the optimal solution in a short time but also to be capable of tracking the solution after a change in the environment occurs when applied to a dynamic environment [24]. To design a PSO algorithm for dynamic environments, we seek solution to the problems of *outdated memory* and *diversity loss* as mention earlier [6, 19, 20, 22, 25, 33].

B. Topological Rule

The introduction of topological rules into PSO for dynamic environment is a new approach. Montes et al [26] tests the implementation of three population topologies commonly used to modify standard PSO: the fully connected topology, in which every particle is a neighbour of any other particle in the swarm; the Von Neumann topology, in which each particle is a neighbour of four other particles; and the ring topology, in which each particle is a neighbour of two other particles. Francesco and Alessandro [16] proposed FSO, adopts a different, although similar approach based on recent naturalistic observations on the collective behaviour of the European Starlings (Sturnus Vulgaris) [2]. Francesco and Alessandro applied the collective response in a static environment and our approach applies the naturalistic behaviour in a dynamic environment. In real starlings each generic j^{th} starling communicate and follows the flight of a number, $N_{ctrlStar}$, no matter what their positions in the flock. A value $N_{ctrlStar}$ = 7 from our empirical study of starlings was found to be effective when dealing with predator attack. A new term due to the naturalistic observation as it is explained in the following in addition to the previous PSO Equations (1) and (2):

- N_{star} is the total number of starlings in the swarm; – Maximum value of the topological coefficient, $\Psi_{topology}$;
- Number of starlings in the swarm directly communicating with a single starling, $N_{ctrlStar}$ (topology rule);
- Each j^{th} starling is associated with group composed of randomly chosen $N_{ctrlStar}$ of other members of the swarm;
- A maximum value of the response velocity when a change is detected, V_{escMax}.

For each j^{th} starling at each time step t, with t = 0,...,T_{max}, update in addition to the steps of the standard PSO update equation, the vector velocity component of each j^{th} starling is:

$$\vec{V}_{k(t+1)} = \omega\vec{v}_{k(t)} + c_1.r_1(\vec{P}_{k(t)} - \vec{x}_{k(t)}) + c_2.r_2(\vec{P}_{g(k)} - \vec{x}_{k(t)}) + \psi^j.S^j_{response(t)}, \qquad (4)$$

In Equation (3), the collective response rule is introduced according to [16], where topological coefficient defined by $\psi^j = \Psi_{topology,}$ and $\Psi_{topology}$ is a random value (0,1) for all values of j, and $S^j_{response(t)}$ is given as follows:

$$S^j_{response(t)} = \left(\frac{1}{N_{ctrlStar}}\sum_j^{N_{ctrlStar}} v_k^{q,j}(t)\right), \qquad (5)$$

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15821

The mean value of the k^{th} velocity components $v_k^{q,j}(t)$, of each j^{th} communicating starling by the q^{th} starlings, $k = 1,...,D$, where D is the starlings topology (Fig.1a). Fig. 1b form the basis for *Algorithm 1*.

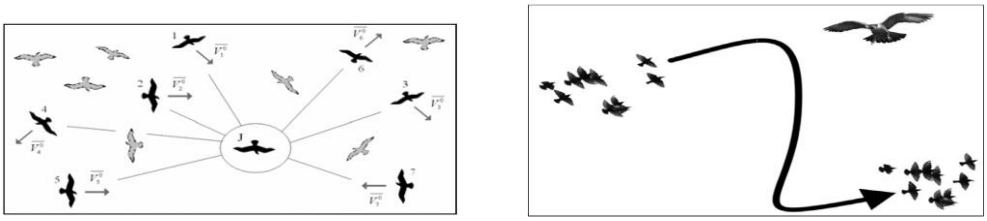


Fig.1: A group of seven interacting starlings with the j^{th} starling [16].

a. k^{th} velocity components of each j^{th} communicating starling by the q^{th} starlings
b. Starlings collective response to dynamic change

C. THE PROPOSED ALGORITHM

The proposed starling swarm algorithm maintains a swarm of starlings, the swarm has certain unique behaviours to change in their environment called collective response in terms of communication within their seven nearest neighbours and murmuration in terms patrolling the search space. These behaviours are mutually exclusive and have eliminated the use of multi-swarm. The proposed algorithm works thus: after initialising the swarm population, the particles begin searching the environment. At each iteration, velocity and position of a particle i in the swarm is updated using its best personal position ($pbest_i$) and the best position found by the swarm ($gbest_i$) using standard PSO update equations *(see Algorithm 2 below).*

However, if a change is detected by a particle i at time t_0, the particle will inform the swarm through $gbest_i$ and find its nearest neighbours (*see Algorithm 1 below*) based on topological rule. These neighbours will switch their mode from murmuration mode to response mode and collectively converge around the detected change. At this stage a communication typical of the Starlings swarm occur. While the swarm is aware of the situation, they continue to patrol the search space in the same manner the starlings keep constant vigilance with their murmuration. At time t_1 a new change is detected by another particle *j*, particle *j* finds its q^{th} nearest neighbours. The particles responding to the earlier change will switch to murmuration and join the rest of the particles to search other areas. While collective response speeds up convergence the murmuration improves adaptation to the changing optima.

Algorithm 1 : Collective response to change in Dynamic Environment

1: **for** *each response particle* t **do**
2: $p_i =$ *a random position of nearest neighbours with*
3: *topological radius* r_s
4: **if** *distance* $(p_i, optima_{new}) < r_s$ **then**
5: **for** $k = 1$ *to number of starlings* **do**
6: Find the nearest seven neighbours of k around $optima_{new}$ and keep their indexes (n) in the set N_k

$$x_k = \sum_{n \in N_k} x_n, \quad v_k = \sum_{n \in N_k} v_n$$

7: *end-for*
8: *Update the position and velocity of k neighbours*
9: *according to Equation* (3)
10: *Set the k neighbours of t to exploit* $optima_{new}$
11: *end If*

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15821

Algorithm 2 : Proposed Starlings Swarm Algorithm for Dynamic Environment

initialize the starlings population
Set the starlings to patrol the search space
for *each particle i in the starlings population* do
    Update position of particle i according to Equations$(1 \sim 2)$
    Update $pbest_i = p_i$
if *a change is detected in the environment* then
Apply Algorithm 1
end If
for *particles* $N - n$ do
    explore other areas for better solutions
if *a new change is detected* then
    Set $sbest_{star}$ *to the new change detected by the particles in the*
    starlings swarm
    $optima_{new} = sbest_{star}$
Apply Algorithm 1
for *response partilces* t do
    for *each particle i in the response particles* t do
     switch mode from response to exploring
continue until termination criteria is met

III. THE EXPERIMENTAL RESULTS

In this section we evaluate our improvements through experiments. We select the moving peaks benchmark (MPB) [8] as the first test problem (Scenario 2) as shown in Table 1 below and created a new scenario with a list of function objects.

Parameters	Scenario Settings
Number of peaks	5
Number of dimensions	10-200
Peaks heights	∈ [30,70]
Peaks widths	∈ [1,12]
Change frequency(Δe)	1000-5000
Change severity (sev)	0.0 - 3.0
Correlation coefficient	0.0
Peak function cone:	$f(\mathbf{x}) = \sqrt{\sum_{i=1}^{N} x_i^2}$

Table 1: Settings of Moving Peaks Benchmark's Scenarios

In this experiment we run similar experiments with [31] on our new algorithm, where a family of different problem instances (e.g. combining a certain number of peaks, topology, change frequency, shift severity etc.) are analysed. Each scenario of the problem will change 100 times every evaluations (Δe) - see appendix A and each run terminate when the algorithm consume 100 x Δe.

Choosing performance metric for evaluating algorithms in a dynamic environment is an open research topic [31]. However, some progress has been made in [11], [12], [28], [33]. In this research we use the offline error and offline performance metrics. These metrics are to be used to measure tracking capabilities which were originally suggested by [6] for static environment, but can easily be adapted to dynamic problems as follows:

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15821

Let $\mathbf{e}_t$ be the t^{th} evaluation, where t denotes a single function evaluation and T be the number of evaluations considered.

- **Current Error** e_t is the difference between current best individual (i.e best evaluation since last change) and the theoretical optimum.

$$e'_t = Opt_t - X_{best} \quad (5)$$

- **Offline Error** - is the average current error over all time step.

$$e^*_t = \frac{1}{T}\sum_{t=1}^{T} e'_t \quad (6)$$

- **Offline Performance** ($\mathbf{x}^*(T)$) - is calculated as the average of the best values found so far at each time step. It only considers individuals evaluated since the last change in the environment.

$$X^*(T) = \frac{1}{T}\sum_{t=1}^{T} e^*_t \quad (7)$$

With $e^*_t = max\{\mathbf{e_1}, \mathbf{e_2}, ..., \mathbf{e_t}\}$

The Starlings Particle Swarm Optimisation for Dynamic Environment - *sPSODE* approach has several parameters that need to be established *priori*. As a new approach based on [5], we will use in our algorithm a single swarm, containing 40 particles. While collective response with 8 nearest neighbours speed up convergence, 32 particles are diversified after a problem change. We performed 30 runs with different random seeds for the problem and algorithm. Other than the PSO parameters applied at swarm level, we have chosen the following values: inertia = 0.70 and $c_1 = c_2 = 1.47$ these values are said to achieve a reasonable performance as suggested by [23].

Taking into account the general parameter settings, we organise the experiments, thus:

- Effect of topology parameter in the diversity strategy.
- Algorithm: *sPSODE*
- Factors to study: *sev*,Δe
- Other problems with different landscapes.
- Algorithm: *mQSO, mQSOE, mPSOD, mPSODE, sPSODE*
- Factors to study: *peak function, sev*, Δe

The first two experiments are devoted to study the influence of parameters related with our strategies versus different shift severity (sev), which represents the distance that problem's optima will be shifted due to an environment change , and change frequencies (Δe) this represents the number of evaluations before a change occurs in the environment. The latter one adds an extra factor peak function, which defines the type of function used as peak (see Table 4). Taking the MPB as a template, we created new problem instances with different landscapes.

D. Effect of topology on the diversity strategy

As described in Sect. 2.2, there are 3 population topology commonly used to modify standard PSO: the fully connected topology, the von Neumann topology and the ring topology. The parameter $N_{ctrl\ star} = 7$ defines a new topology called collective response to changes - in starlings swarm. The main usefulness of this type of response is to solve the diversity loss in dynamic environment. So, hypothetically speaking topology should be of similar magnitude as the repulsion radius (r_{exp}) in [31].

In this regard, we selected the topologies in Sect. 2.2, $N_{ctrl\ star} \in \{2, 4, 7, 40\}$ to represent Ring, Von Neumann, Starlings and Fully connected topologies respectively. Compared to the size of the search space bounds = *[0,100]* and dimensionality = 5. These are relatively small values. We have also included a *rand* alternative, which randomly assigned to topology one of the severity or change frequency values in every algorithm iteration.

The off-line error achieved by the sPSODE algorithm with different configurations of topology (group size), varying the shift severity (*sev*) and the change frequency (Δe) is shown in Tables 2 and 3 below.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15821

Topology	Ring	Von Neumann	Starlings	Fully Connected
sev				
0.00	0.060±0.002	**0.032±0.015**	0.034±0.011	3.799±0.939
1.00	0.066±0.022	0.052±0.021	0.046±0.016	3.142±1.531
2.00	0.078±0.017	0.036±0.003	**0.027±0.021**	**3.800±1.838**
3.00	0.053±0.039	0.047±0.005	0.040±0.017	**3.888±1.709**
randSev	0.072±0.001	0.036±0.013	0.029±0.001	3.640±1.368

Table 2: Mean offline error (±SE) of sPSODE topologies under varying shift severity in MPB Scenario 2.

Topology	Ring	Von Neumann	Starlings	Fully Connected
Δe				
1000	0.221±0.018	0.149±0.085	**0.139±0.033**	4.186±2.136
2000	0.131±0.034	0.097±0.035	0.088±0.017	3.241±1.693
3000	0.082±0.006	0.073±0.044	0.052±0.008	3.790±2.087
4000	0.042±0.028	0.088±0.025	0.056±0.021	3.487±1.825
5000	0.052±0.020	0.020±0.001	0.047±0.001	3.662±1.992
randΔe	0.076±0.002	0.097±0.053	**0.026±0.003**	3.590±1.840

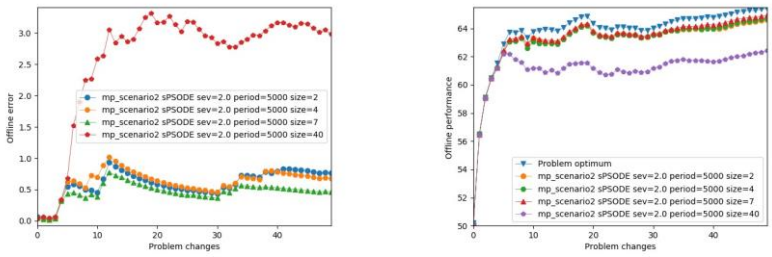
Table 3: Mean offline error (±SE) of sPSODE topologies under varying change frequencies in MPB Scenario 2.

The experiments of Table 2 used Δe = 5,000, while in Table 3, *sev* = 2.0. The text in bold corresponds to the best and worst obtained values in Table 2 and best values in Table 3.

As expected, the best results for the first set of experiments correspond to configurations with starlings topology. It means that after each change, the starlings' topology is more adaptive and just need some diversity to find the shifted optima. For example, when the *sev* is 2.0 and *randSev*, the version with starlings topology is the best behaved and for most of the severities except for *sev* = 0.0, precisely because the starlings' topology is the optimal value closer to these severities. Regarding the experiments related with Δe, the first thing that stands out is that the configuration starlings topology do not seem to be affected when varying the change frequency for *rand*Δe. This result correlates with those obtained in the previous set of experiments, for which the best configuration for *randSev* was almost the same. Despite the results in Tables 2 and 3, it is also important to analyse the algorithm's behaviour over time. To that end, we plot (*figure 2*) the four topologies over the problem instance with *sev* = 2.0 and Δe = 5,000. In Fig. 1a and 1b, we plotted the evolution of the offline error and offline performance in terms of problem changes.

From this graph it is possible to notice that the main difference between configurations occurs after 30th problem change of the search (*Fig. 1a*). In particular, starlings topology is more accurate at that point. However, there are time windows for which the other topologies are temporarily as good as the starlings'. For example, before the 10th problem change (text it Fig. 1b), the performance of all the topologies fairly the same with that of starlings. These are indications that the exploration topology, as most of the parameters in stochastic algorithms, are optimal only in some parts of the optimisation process. For this reason, the *rand* configuration shows a good performance in general in both sets of experiments Table 2 and Table 3 above.

In summary, for the problem instances considered in this section, the best sPSODE configurations is that of starlings. An interesting aspect is that the *rand* option shows in general good results for the two factors studied, and particularly



(a) Offline error (b) Offline performance

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15821

Fig.2 (a) and (b) above: Evolution of offline error and offline performance for sPSODE in MPB scenario 2 (*Sev* = 3, Δe = 5000). Graphs shows the best (*topology* = 7) and worst (*topology* = 40) the starlings topology. To extend our study on the evolution of offline error and offline performance, above shows graph of uni-modal and multi-modal problems, plotted against number of problem changes.

E. Other problems with different landscapes

To extend research, we conduct further experiments on problems with different landscapes. The dynamics of these problems are essentially the same as the Scenario 2 of the MPB but with two distinct features: the boundaries of the search space and peak function. Table 4 below shows the seven selected functions for the peaks. These functions are used in the context of non-dynamic optimisation [27]. Cone, Sphere, Schwefel, and Quadric are Uni-modal, while Rastrigin, Griewank, Ackley and Weierstrass are multi-modal problems. We kept the settings of MPB's Scenario 2 , only for the number of peaks in multi-modal landscapes as in Table 1 above.

	Peak function	Formula	Range
Uni-modal	Cone	$f(\mathrm{x}) = \sqrt{\sum_{i=1}^{N} x_i^2}$	$[0, 100]^5$
	Sphere	$f(x) = \Sigma_{i=1}^{n} x_i^2$	$[0, 100]^5$
	Schwefel	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \Pi_{i=1}^{n} \lvert x_i \rvert$	$[0, 100]^5$
	Quadric	$f(x) = \sum_{i=1}^{n} (\sum_{j=1}^{n} x_j)^2$	$[0, 100]^5$
Multi-modal	Rastrigin	$f(x) = \Sigma_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-5.12, 5.12]^5$
	Griewank	$f(x) = \frac{1}{4000} \Sigma_{i=1}^{n} (x_i)^2 - \Pi_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$	$[-32, 32]^5$
	Ackley	$f(x) = -20exp(-0.2\sqrt{\frac{1}{n} \Sigma_{i=1}^{n} x_i^2}) - exp(\frac{1}{n} \Sigma_{i=1}^{n} \cos(2\pi x_i)) + 20 + \exp$	$[-32, 32]^5$
	Weierstrass	$f(x) = \Sigma_{i=1}^{n} (\Sigma_{k=0}^{k_{max}} [a^k \cos(2\pi b^k (x_i + 0.5))]) - n \Sigma_{k=0}^{k_{max}} [a^k \cos(\pi b^k)]$	$[-0.5, 0.5]^5$

Table 4: Mean offline error (±SD) across problems under varying change frequencies (Δe).

Since the configuration for the starlings diversity strategy is the best, we decided to employ sev = 2.0 configuration for *sev* and *5000* for Δe . For comparison purposes, we included the mQSO, mQSOE, mPSOD and mPSODE algorithms results in [31].

	Problem	sev	mQSO	mQSOE	mPSOD	mPSODE	sPSODE	Imp. rate
Uni-modal	Cone	0.0	0.34±0.05	**0.28±0.04**	0.49±0.06	0.43±0.06	0.72±0.36	17.65
		1.0	1.78±0.06	1.06±0.08	1.59±0.10	1.44±0.12	**0.77±0.36**	56.74
		2.0	2.36±0.10	1.51±0.10	2.39±0.12	2.25±0.11	**0.81±0.41**	65.68
		3.0	2.88±0.11	2.27±0.14	3.24±0.13	2.90±0.14	**0.75±0.34**	73.95
	Sphere	0.0	0.23±0.05	**0.21±0.05**	0.21±0.03	0.35±0.06	0.74±0.48	8.70
		1.0	0.92±0.06	**0.56±0.04**	0.90±0.04	0.99±0.07	0.83±0.60	39.13
		2.0	1.59±0.06	1.70±0.10	1.91±0.06	2.07±0.07	**0.76±0.39**	52.20
		3.0	2.60±0.08	3.09±0.12	3.12±0.09	3.93±0.15	**0.85±0.53**	67.31
	Schwefel	0.0	0.39±0.05	0.43±0.06	0.46±0.06	0.44±0.06	**0.09±0.19**	76.92
		1.0	3.00±0.10	1.80±0.11	2.72±0.10	2.50±0.12	**0.20±0.36**	93.33
		2.0	3.89±0.15	2.91±0.11	3.94±0.13	3.61±0.14	**0.28±0.70**	92.80
		3.0	4.77±0.1	4.13±0.17	5.11±0.18	4.69±0.17	**0.29±0.37**	93.92

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15821

			6					
	Quadric	0.0	0.99±0.17	0.76±0.10	1.21±0.10	1.19±0.19	**0.65±0.44**	34.34
		1.0	1.47±0.11	1.63±0.17	1.54±0.10	1.85±0.16	**0.70±0.46**	52.38
		2.0	1.85±0.12	2.81±0.11	2.17±0.12	2.46±0.14	**0.73±0.32**	60.54
		3.0	2.68±0.15	4.32±0.25	3.28±0.10	4.95±0.23	**0.64±0.33**	76.12
Multi-modal	Rastrigin	0.0	4.86±0.33	4.68±0.35	3.65±0.30	4.08±0.25	**0.93±0.04**	80.86
		1.0	5.50±0.47	5.16±0.30	3.31±0.26	3.53±0.22	**2.27±0.26**	58.73
		2.0	5.45±0.33	5.64±0.34	3.69±0.16	4.19±0.27	**2.74±0.26**	49.72
		3.0	6.13±0.33	6.57±0.24	4.60±0.19	4.48±0.18	**2.94±0.30**	52.04
	Griewank	0.0	0.46±0.01	*0.40±0.01*	0.42±0.01	0.39±0.01	**0.03±0.01**	93.48
		1.0	0.81±0.01	0.81±0.01	0.78±0.01	0.74±0.01	**0.06±0.01**	92.59
		2.0	0.88±0.01	1.07±0.02	0.92±0.01	0.86±0.01	**0.06±0.01**	93.81
		3.0	0.98±0.01	1.33±0.02	1.05±0.01	1.01±0.02	**0.07±0.01**	92.86
	Ackley	0.0	1.35±0.6	2.25±0.23	0.70±0.10	0.67±0.10	**0.54±0.20**	60.00
		1.0	1.57±0.14	1.80±0.19	0.89±0.05	0.78±0.05	**0.55±0.14**	64.67
		2.0	2.27±0.20	3.14±0.10	1.77±0.07	1.60±0.09	**0.61±0.19**	73.13
		3.0	3.20±0.14	5.26±0.17	2.38±0.05	2.21±0.05	**0.64±0.21**	80.00
	Weierstrass	0.0	0.01±0.01	4E-3±2E-3	**1E-3±2E-4**	1E-3_4E-4	2E-3_8E-4	90.00
		1.0	1.28±0.01	0.73±0.01	1.12±0.01	0.87±0.01	**0.27**±0.07	80.00
		2.0	1.79±0.01	1.55±0.02	1.82±0.01	**1.34±0.01**	3.73±0.63	25.14
		3.0	2.17±0.01	1.88±0.02	2.30±0.01	1.53±0.01	4.56±0.57	25.49

Table 5: Offline error: mean ± standard deviation values for different problems varying shift severity (*sev*)

Tables 5 and 6 shows the results for the new problem instances and the implemented algorithms, varying the severity and the change frequency, respectively. The last column of these tables includes the improvement rate (Imp. rate) between our best algorithm and mQSO.

	Problem	**Δ*e***	**mQSO**	**mQSOE**	**mPSOD**	**mPSODE**	**sPSODE**	**Imp. rate**
Uni-modal	Cone	1000	4.27±0.14	3.58±0.14	4.57±0.17	4.27±0.21	**0.95±0.34**	77.75
		2000	3.05±0.12	2.21±0.12	2.89±0.11	2.70±0.12	**0.92±0.29**	69.83
		3000	2.40±0.08	1.53±0.07	2.34±0.12	2.19±0.16	**0.83±0.33**	65.42
		4000	1.95±0.07	1.13±0.06	1.88±0.08	1.81±0.14	**0.79±0.34**	59.49
	Sphere	1000	4.02±0.25	4.34±0.30	6.23±0.37	6.31±0.29	**0.64±0.38**	84.08

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15821

		2000	2.00±0.20	1.65±0.14	2.53±0.15	2.69±0.20	**0.73±0.33**	63.50
		3000	1.62±0.10	0.97±0.08	1.72±0.12	1.64±0.15	**0.73±0.33**	54.94
		4000	0.95±0.04	0.70±0.06	1.40±0.11	1.30±0.10	**0.67±0.40**	29.47
	Schwefel	1000	6.73±0.21	5.75±0.22	7.25±0.22	6.77±0.27	**0.13±0.13**	98.05
		2000	4.79±0.14	3.51±0.16	4.85±0.18	4.67±0.21	**0.06±0.08**	98.75
		3000	4.01±0.15	2.71±0.16	3.82±0.14	3.48±0.14	**0.10±0.12**	97.51
		4000	3.31±0.11	2.20±0.14	3.21±0.13	2.88±0.16	**0.16±0.20**	95.17
	Quadric	1000	3.26±0.12	3.51±0.18	4.40±0.30	4.03±0.24	**0.66±0.29**	79.75
		2000	2.03±0.11	2.10±0.13	2.28±0.13	2.41±0.18	**0.72±0.55**	64.00
		3000	1.70±0.11	1.74±0.20	2.01±0.11	1.92±0.13	**0.67±0.38**	60.59
		4000	1.46±0.12	1.50±0.14	1.62±0.11	1.63±0.13	**0.78±0.59**	46.58
Multi-modal	Rastrigin	1000	6.70±0.37	5.85±0.34	5.87±0.30	6.39±0.32	**4.70±0.32**	29.85
		2000	6.77±0.44	5.53±0.36	5.08±0.32	4.82±0.27	**3.52±0.33**	48.01
		3000	6.49±0.45	5.19±0.36	4.23±0.20	4.20±0.21	**2.29±0.24**	55.01
		4000	5.69±0.44	5.35±0.36	3.59±0.13	3.91±0.29	**2.48±0.21**	56.41
	Griewank	1000	2.50±0.05	*2.55±0.05*	2.62±0.04	2.65±0.06	**0.11±0.03**	95.60
		2000	1.37±0.02	1.46±0.03	1.51±0.02	1.41±0.05	**0.08±0.02**	94.16
		3000	1.15±0.02	1.18±0.02	1.12±0.02	1.04±0.01	**0.07±0.02**	93.91
		4000	0.78±0.01	0.89±0.01	0.90±0.01	0.87±0.01	**0.06±0.02**	92.31
	Ackley	1000	3.00±0.15	2.49±0.09	2.79±0.06	2.29±0.12	**0.79±0.16**	73.67
		2000	2.16±0.12	1.73±0.12	1.78±0.04	1.49±0.09	**0.70±0.21**	67.59
		3000	1.95±0.15	1.49±0.12	1.39±0.10	0.98±0.04	**0.59±0.19**	69.74
		4000	1.56±0.10	1.34±0.10	1.18±0.07	0.88±0.05	**0.49±0.16**	68.59
	Weierstrass	1000	2.64±0.01	2.20±0.01	2.55±0.01	2.38±0.01	**0.55±0.13**	79.17
		2000	2.03±0.01	1.49±0.01	1.91±0.01	1.67±0.01	**0.45±0.11**	77.83
		3000	1.12±0.01	1.55±0.02	1.58±0.01	**1.30±0.01**	**0.34±0.09**	80.00
		4000	0.89±0.00	1.88±0.02	1.35±0.01	1.05±0.01	**0.28±0.07**	80.69

Table 6: Offline error (mean ± SD) across problems with varying change frequency (Δe).

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026
DOI: 10.17148/IJARCCE.2026.15821

This improvement rate is computed for every problem instance *i* as follows:

$$Imp.rate_i = 100\left(1 - \frac{e_i, Best}{e_i, mQSO}\right) \quad (8)$$

where *ei,Best = min{*e_i*, mQSOE,* e_i*,mPSOD,* e_i*,mPSODE,* e_i*,sPSODE}* is the best (minimum) offline error obtained by the algorithms. If the improvement rate is equal to Zero then mQSO is better than the rest for that specific problem as in [31].

The results of the first set of experiments in Table 5 indicate that for less complex landscapes (those based on uni-modal functions: cone and sphere), algorithms based on quantum particles did better for severity 0.00 and 1.00. Our sPSODE variant is the best in 13 out of 16 problem instances. For those instances based on multi-modal functions, our method which incorporate starlings collective response diversity strategy after a problem change (sPSODE) is the most successful one, although when the Weierstrass function is used as a peak, this superiority over other variants (mQSO, mQSOE, mPSOD, mPSODE) is not significant except for severity 1.00. The best improvement rate values of our algorithm in multimodal functions is achieved in the Griewank function with values ranging from 92 to 94. For the case of uni-modal functions, best values correspond to the Schwefel function ranging from 76 to 93.

The differences of our algorithm with respect to mQSO, mQSOE, mPSOD and mPSODE are more pronounced in the second set of experiments with Δe. Table 6 above, shows that sPSODE is superior in all the problems. It is safe to say that for all problems starlings based algorithm is more appropriate. Here, starlings inspired algorithm reached better results in more problems than in the previous experiment (Table 5). See for example that the sPSODE algorithm clearly outperforms mQSO in both uni-modal and multi-modal problems, reaching improvement rates from 29 to 90. This is a further argument to confirm the superiority of the proposed strategy.

F. CONCLUSION AND FUTURE WORK

In this work we have proposed starlings collective response for improving the adaptation of a well-known approach for optimisation in dynamic environments: the multi-swarm PSO (mPSO). These strategies were oriented to manage the outdated memory and diversity loss. The strategy of diversification was developed based on the starlings murmuration and collective response to change in their environment. It was found through experiments that in problems with a moderate shift severity the best variants are those with topology similar to that of starlings communicating with their seven nearest neighbours through collective response. Moreover, on both unimodal and multi-modal problems this strategy seemed to be more effective than the constant generation of diversity shown by algorithms like mQSO, mPSODE and their variants. It has been observed that the use of this rule in the algorithms was more effective in both uni-modal and multi-modal functions.

In the future, we consider that self-adaptation is a promising research area to explore, where centralised or decentralised sharing information mechanisms could be used to promote those parameters configurations that are well suited at every stage of the search.

REFERENCES

[1]. P.J, Angeline, "Tracking extrema in dynamic environments". *In International Conference on Evolutionary Programming*, Vol. 13, p. 335-345, 1997. Springer, Berlin, Heidelberg.
[2] M. Ballerini, N. Cabibbo, et al. "Interaction ruling animal collective behaviour depends on topological rather than metric distance: Evidence from a field study". *Proceedings of the national academy of sciences*, Vol. 4, p.1232–1237, 2008.
[3] A. Banks, J. Vincent, and C. Anyakoha. "A review of particle swarm optimisation", *Natural Computing*, p. 467–484, 2007.
[4] A. Banks, J. Vincent, and C. Anyakoha. A review of particle swarm optimisation. Part II: hybridisation, combinatorial, multi-criteria and constrained optimisation, and indicative applications", *Natural Computing* p.109-124, 2008.
[5] T. Blackwell and J. Branke. "Multi-swarms, exclusion, and anti-convergence in dynamic environments". *IEEE transactions on evolutionary computation*, Vol. 4, p. 459-472, 2006.
[6] T. M Blackwell. "Swarms in dynamic environments. In Genetic and Evolutionary Computation Conference." *Springer*, p. 1-12, 2003.
[7] T. M Blackwell and P. J Bentley. "Dynamic search with charged swarms. In Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation." *Morgan Kaufmann Publishers Inc.*, p. 489-500, 2003.
[8] J. Branke. "Memory enhanced evolutionary algorithms for changing optimisation problems". Vol. 3, p. 1875-1882, 1999.

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 8, August 2026

DOI: 10.17148/IJARCCE.2026.15821

[9] J. Branke and H. Schmeck. "Designing evolutionary algorithms for dynamic optimisation problems. In Advances in evolutionary computing." *Springer*, p. 239-262, 2003.
[10] M. Ghamisi. "Fractional order Darwinian particle swarm optimisation: applications and evaluation of an evolutionary algorithm." *Springer*, p. 1-12, 2015.
[11] D. Dasgupta and D. R. McGregor. "Non-stationary Function Optimisation using the Structured Genetic Algorithm." Vol. 2, p. 145-154, 1992.
[12] E. Alan De-Jong. "Analysis of the behaviour of a class of genetic adaptive systems." 1975.
[13] R. Eberhart and J. Kennedy. 1995. "A new optimiser using particle swarm theory. In Micro Machine and Human Science," *Proceedings of the Sixth International Symposium on. IEEE*, p. 39-43, 1995.
[14] R. C. Eberhart and Y. Shi. "Tracking and optimising dynamic systems with particle swarms." p. 94-100, 2001.
[15] D. Floreano and C. Mattiussi. "Bio-inspired artificial intelligence: theories, methods, and technologies." *MIT press.*
[16] F. R. Fulginei and A. Salvini. "The flock of starlings optimisation: influence of topological rules on the collective behaviour of swarm intelligence. In Computational Methods for the Innovative Design of Electrical Devices." *Springer*, p. 129-145, 2010.
[17] A. H. Gandomi, et al. "Cuckoo search algorithm: a meta heuristic approach to solve structural optimisation problems." *Engineering with computers* vol. 29, 17-35, 2013.
[18] B. G. Hogan, et al. "The confusion effect when attacking simulated three-dimensional starling flocks." *Royal Society Open Science*, vol. 4, p. 160-564, 2017.
[19] X. Hu and R. C. Eberhart. "Adaptive particle swarm optimisation: detection and response to dynamic systems. In Evolutionary Computation," *Proceedings of the 2002 Congress on, Vol. 2. IEEE*, vol. 2, p. 1666-1670, 2002.
[20] S. Janson and M. Middendorf. "A hierarchical particle swarm optimiser for dynamic optimisation problems," p. 513-524, 2004.
[21] S. Janson and M. Middendorf. 2006. "A hierarchical particle swarm optimiser for noisy and dynamic environments." *Genetic Programming and Evolvable Machines* Vol. 7, p. 329-354, 2006.
[22] J. Kennedy and R. Mendes. "Population structure and particle swarm performance. In Evolutionary Computation, 2002. CEC'02." *Proceedings of the 2002 Congress on, Vol. 2. IEEE*, Vol. 2, p. 1671-1676, 2002.
[23] l. M and J. Kennedy. "The particle swarm - explosion, stability and convergence in a multidimensional complex space." *IEEE Transactions on Evolutionary Computation*, Vol. 6, p. 58-73, 2002.
[24] C. Li and S. Yang. "A generalised approach to construct benchmark problems for dynamic optimisation." *In Asia-Pacific Conference on Simulated Evolution and Learning. Springer*, p. 391-400, 2008.
[25] R. Lung and D. Dumitrescu. "A collaborative model for tracking optima in dynamic environments." *In Evolutionary Computation, CEC 2007. IEEE Congress on. IEEE*, p. 564-567, 2007.
[26] M. M Roldan, et al. 2009. "Frankenstein's PSO: A composite particle swarm optimisation algorithm." *IEEE Transactions on Evolutionary Computation.* Vol. 13, p. 1120-1132, 2009.
[27] Y. Xin, et al. "Evolutionary programming made faster." *IEEE Transactions on Evolutionary computation*, p. 82-102, 1999.
[28] Ronald W Morrison. "Performance measurement in dynamic environments. In GECCO workshop on evolutionary algorithms for dynamic optimisation problems." *Citeseer*. 2003.
[29] R. W Morrison, et al. "A test problem generator for non-stationary environments." p. 2047-2053, 1999.
[30] I. Moser and R. Chiong. "Dynamic function optimisation with hybridised extremal dynamics." *Memetic Computing*, Vol. 2, p. 137-148, 2010.
[31] P. Novoa-Herna´ndes, Carlos Crus Corona, and David A Pelta. "Efficient multi-swarm PSO algorithms for dynamic environments." *Memetic Computing* vol. 3, p. 163, 2011.
[32] D. Pelta, C. Crus, and J.L Verdegay. "Simple control rules in a cooperative system for dynamic optimisation problems." *International Journal of General Systems* vol.38, p. 701-717, 2009.
[33] R. Thomsen. "Multimodal optimisation using crowding-based differential evolution. In Evolutionary Computation" *IEEE,* Vol. 2. p. 1382-1389, 2004.

