Swarm and Evolutionary Computation 74 (2022) 101125

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

journal homepage: www.elsevier.com/locate/swevo

A new moving peaks benchmark with attractors for dynamic evolutionary algorithms

Matthew Fox*, Shengxiang Yang, Fabio Caraffini*

Institute of Artificial Intelligence, School of Computer Science and Informatics, De Montfort University, Leicester LE1 9BH, United Kingdom

ARTICLE INFO

Keywords:
Benchmark
Attractor
Evolutionary dynamic optimization
Evolutionary algorithm
Performance metric

ABSTRACT

Prediction in evolutionary dynamic optimization (EDO), such as predicting the movement of optima, or when and how an environment will change, is a topic that is still under investigation and presents unsolved challenges. A few studies approach prediction based on re-initialising a population or requirement satisfaction problems such as Robust Optimization Over Time. The benchmark problems in these studies inherently use randomly changing parameters and therefore such randomness may make it difficult to compare these algorithms with other EDO approaches. In this paper, we introduce a new benchmark, called Moving Peaks Benchmark with Attractors, which incorporates an attractor heuristic that attracts peaks to a certain location in the environment into the moving peaks problem. The proposed benchmark is fully flexible where the dynamics of the attractors and the rate at which a peak is attracted to such attractors can be modified. By adjusting these characteristics, certain styles of movements can be achieved by a peak. We also introduce a new performance measure that focuses on the comparison of algorithms that use prediction. Seven EDO algorithms based on different working logics are chosen to give a wide representation of the state-of-the-art in this area. We argue that having predictable characteristics in the benchmark problem is more adequate for studying the performances and behaviours of those algorithms that embed prediction mechanisms. Experimental results obtained with the proposed benchmark show it's suitability for the EDO domain as all algorithms featuring prediction capabilities display higher accuracy than their competitors.

1. Introduction

Most real-world optimization problems deal with changing objective functions, constraints and problem environments over a period of time [1,11,20,47]. These time-varied optimization problems, known as Dynamic Optimization Problems (DOPs), require algorithmic solutions to continuously "chase" moving optima. Other challenges include detecting when the problem changes. DOPs are conceived from the belief that learning from previous evaluations, when assuming the problem has not changed significantly can increase the efficiency and reliability of the optimiser, thus allowing for a quicker and more informed convergence to a solution.

There are various methods of dealing with DOPs that include tracking moving optima (TMO) on each environmental change, which remains one of the most widely researched topic. Two strategies can be used to improve on TMO approaches: one is to use historical information [52] and the other is to maintain the population diversity [17]. However, TMO approaches do not take into account the cost of frequently changing solutions if the solution chosen is too costly or impossible to switch to.

There also exist DOPs that include requirement satisfaction problems as an objective to be optimised. For example, one approach that finds a solution that remains acceptable over several environmental changes is robust optimization over time (ROOT). Proposed by Yu et al. [50], it's goal is to find consecutive solutions over time that can have a varying degree of quality rather than finding the global optimum at each time interval. Approaches to solving ROOT problems include the prediction of the fitness landscape following future environmental changes to estimate how long a solution will remain acceptable. According to the ROOT metric, a solution is chosen according to either the average fitness of a solution given a predetermined length of environmental changes or the survival time, i.e., how long a solution's fitness remains above a set threshold [13]. A perfect predictor of the fitness landscape ensures that a perfect solution can be chosen, however, any errors within the predictor may lead to prediction deception.

The changes in DOPs could be predicted if factors exist that allow prediction based on the view that previous fitness landscapes correlate with future fitness landscapes with a small degree of changes. Such information could be exploited to allow an optimiser to increase diversity or reduce the convergence time to the best solution. Using prediction to

* Corresponding authors.
E-mail addresses: p13191539@my365.dmu.ac.uk (M. Fox), syang@dmu.ac.uk (S. Yang), fabio.caraffini@dmu.ac.uk (F. Caraffini).

https://doi.org/10.1016/j.swevo.2022.101125
Received 19 December 2021; Received in revised form 15 April 2022; Accepted 2 July 2022
Available online 16 July 2022

track moving optima is a common approach and includes influencing the diversity after an environmental change [36,45,51], or estimating the areas to re-initialize individuals after a change occurs [18]. Prediction methods are also integrated within problems where there are requirement satisfaction constraints on the fitness of a solution. It is used in two methods; in the ROOT framework where the goal is to find one solution that will be good enough across successive environmental changes. And also to solve time-linkage problems, where the impact of choosing a good solution now may impact the future environment, future landscape changes are predicted using historical information on individuals' fitnesses and approximated solutions as the training data.

Using benchmark problems with different characteristics, such as factors that change, cyclic/periodic/recurrent changes, or changes that can be detected, allows researchers to identify certain characteristics of algorithms. Appropriate benchmark problems should be designed to be flexible, simple, and allow for generalisation with real-world problems [31]. Current benchmarks are designed to exploit a certain type of fitness landscape with varying changes and include prediction only when these varying changes exhibit certain behaviours. Developing a benchmark problem that allows for predictable characteristics to compare the performance of algorithms to solve DOPs is greatly needed for furthering research in ROOT and EDO. There is also a small amount of literature on using prediction in the optimiser, a benchmark such as this may help future research into this topic by providing tools and measures to evaluate predictive characteristics.

In this paper, we propose a new benchmark problem that tests the validity of algorithms that use prediction methods as a heuristic to guide the search. The proposed moving peaks benchmark with attractors (MPBA) uses the modified moving peaks benchmark (MMPB) [5,21,49] as the base and extends the peaks centre movement by including an attractor function that attracts a peak towards a given location within the fitness landscape. By using a weighting factor to adjust the strength of the attraction and extensible functions to move the attractor location across the fitness landscape, we create a versatile benchmark problem that allows an optimiser to predict the movement of peaks. The attractor creates a fully configurable benchmark problem where the combination of movement types, change frequency and weighting of the attractor creates different movement effects on the optima, which allows for studies of EDO algorithms under various conditions. We test a selection of algorithms from the EDO literature against MMPB and MPBA to identify any differences and show that a better predictable benchmark will be more suitable to EDO algorithms that incorporate a prediction method, than those benchmarks that are fully random.

The main contributions of this paper are summarised as follows:

1. We introduce a new benchmark problem where the movement of the optima is predictable under varying conditions. This allows researchers to study the inclusion of prediction and other forecasting techniques within the optimiser using a benchmark that has a fully configurable degree of predictability. Currently, benchmark problems in DOPs either are random, or fully deterministic by definition. Having the flexibility to control how the optimiser responds to information that may be predictable will ensure that the optimiser is robust in most environments. The benchmark allows the researcher to choose the severity, the direction of the movement, and how random an optimum moves through the landscape.
2. A new performance metric is introduced to measure the performance of algorithms that incorporate prediction in their optimiser. This measure looks at the distance between the global optima and the closest individual after the environment has changed. Taking an average of this distance measure over some environmental changes shows how successful the individual initialization is in an optimiser that uses prediction.

This paper is organised as follows. In Section 2, related works and current benchmark problems within DOPs are discussed. Section 3 proposes our new benchmark problem and highlights some important features. The new performance measure is proposed in Section 4. In Section 5, we compare our proposed benchmark with MMPB using current methods to solve DOPs found in the literature. Finally, Section 6 concludes this paper.

2. Related work

2.1. Evolutionary dynamic optimization (EDO)

Many problems in evolutionary optimization can be modelled as DOPs where the variables of the problem change over time. Compared to stationary optimization, where the goal is to find the optimal solutions, the goal of EDO is to track the optima through the environment space across successive iterations. This stems from Branke [5] in which the author illustrated the issue of treating DOPs as a static problem for each environment, where the optimiser is restarted and converged to a new solution. Static optimization techniques were found to be impractical and inefficient for solving DOPs.

Given that previous fitness landscapes may be correlated with future landscapes, DOPs may be solved efficiently using various approaches. One method is to maintain diversity during environmental changes to keep a baseline of diversity to influence future searches in successive environments without losing information by converging to the solution quickly [3,42]. Another method is to understand how the environmental changes between successive landscapes, as areas in the environment that do not contain an individual, may change which will affect the solutions quality [9]. Finally, memory techniques allow the current population to use historical evaluations in guiding their search and preserving good solutions. Memory methods include an implicit memory scheme that stores more information than needed to be used in future iterations [6,8,48]. On the other hand, explicit memory schemes store and retrieve information based on a set of rules. A full survey on the approaches can be found in Nguyen et al. [32].

Prediction methods in EDO attempt to exploit patterns in changing environments that may be predictable. Such patterns may be found in historical data from previous fitness evaluations, or through approximation methods using interpolation. Currently, three main approaches to prediction in EDO exist; predicting the movement of the optima [18,36,45,51], and predicting when the next environmental change occurs and what environment will appear [39,40]. Some approaches directly influence the placement of individuals in new environments or included as part of operators within the algorithm.

In [36] the authors consider a Kalman based prediction to track moving optima under the assumption that real world application changes are not random and can be learnt. Their experiments show that prediction mechanisms are helpful in order to improve the tracking capabilities of an evolutionary algorithm, but they must not be overestimated because they can lead to an excessive convergence, which has a negative impact on the algorithm.

Hatzakis and Wallace [18] combines a forecasting technique with a evolutionary algorithm where the location of the optimal solution is estimated using a forecasting method, created using the sequence of prior optimum locations, in order to better place a new group of individuals around the estimated location. This approach aims to create a faster convergence to the new global optimum. The results indicate that the approach improves algorithm performance, especially in problems where the frequency of objective change is high.

Considering approaches where the next environmental change will occur, Simões and Costa [39] uses linear regression and Markov chains to estimate the generation when a change in the environment will occur and also to predict to which state (or states) the environment may change, respectively. They note that knowing á priori when a change will take place and which environment will appear next, we can introduce useful information in the population before change happens, avoiding the performances decrease usually observed with standard evolutionary algorithms.

Meier and Kramer [26] uses prediction as a third attractor in the movement function of a PSO algorithm to bias the movement of the particles towards the predicted optima regardless of the quality of the prediction and shows promising results. Further work proposes a new reinitialization mechanism placing individuals of an evolution strategy wider in dimensions where the prediction is likely to be inaccurate [27].

As DOPs are said to have a correlation between the old and new environment, transfer learning models of solutions or fitness can be used to correlate environment changes with previous environments. In [23] neural networks to implement a transfer model of environment changes, learnt from past solutions, which is then used to assist the optimization in the new environment. Jiang et al. [19] constructs a transfer model in objective space and then transferred the optimal objective values in past environments to new values in new environments. Neural networks was also used in Liu et al. [24], where a neural network based change prediction method to discover the change law of the optima in different subareas and predict new optima.

Enhancements in the multi-objective space include studies that predict characteristic points [14,34,35,46] who presents a multi-directional prediction strategy to enhance the performance of evolutionary algorithms in solving a dynamic multi-objective optimization problem. The population is clustered into a number of groups by a proposed classification strategy and used predict the moving location of the Pareto set.

2.2. Requirement satisfaction and prediction in EDO

A new perspective on solving DOPs is to find solutions that are robust over time instead of following the changing optima [50]. ROOT takes into account not only uncertainties in the parameter space but also the effect of these uncertainties over time in the time-space. Jin et al. [21] proposed a framework consisting of a population-based optimisation algorithm (POA), a database, a fitness approximator, and a fitness predictor. Studies in ROOT that use this framework use an autoregressive predictor to forecast future individuals' fitness. They found that given a good prediction a solution can remain good enough over some successive environmental changes. Future behaviours of the peaks in ROOT are predicted based on information found from each swarm and the data is used to pick the next robust solution when the current solution becomes unsatisfactory using a multi-swarm method [49]. The research also focused on the appropriate method of estimating future solutions.

In DOPs, the optimiser that chooses the solution at the present does not typically take into account the approximated solutions or the problem in the future. Therefore, a solution chosen now may directly impact future performance. These problems are called Dynamic Time Linkage Problems (DTLPs) [4]. To solve DTLPs, both the present and future behaviours of the problem must be taken into account. The DTLP approach [4] to solving time-linkage problems is a framework that learns to predict the future and optimises not only the current solution but also future predicted environments. It utilises a genetic algorithm and a predictor and was tested on 2 benchmark functions: a numerical optimization problem, and a dynamic pickup problem. The prediction approach predicts the trajectory of a solution for which the optimiser chooses the best solution that achieves the best overall result over a length of time.

Algorithms that also use prediction to introduce diversity within a population could also be said to fail under prediction-deception [4,31], where accurate results can only be found using a perfect prediction, and any errors in such prediction will deceive the optimiser towards false optima. For the case of ROOT and DTLPs, one issue is how far into the future should prediction be made and how much historical data can be used to form this prediction. For both cases, it is understood that the choice of lengths depends on the time-linkage timespan or the maximum survival times in ROOT. Another issue relates to the reliability of the predictive model when approximations are made further into the future. The longer the prediction, the less reliable the solution becomes. A choice that is made based on long timespans may be prediction-deceived as it leads the optimiser to the less preferred results.

2.3. Benchmark problems in EDO

Some studies in EDO research discuss the current benchmark features and have an extensive survey of problems giving their characteristics. Based on the surveys, the problems can be categorised into types of change, e.g., predictable changes, changes that are detected using few detectors, and cyclic/periodical/recurrent changes. Benchmarks that are flexible and simple allow for efficient analysis of EDO characteristics. The following benchmark problems are seen as the most popular ones.

Cobb and Grefenstette [10] proposed Switching Functions, where two landscapes are used in three ways: a linear translation of peaks in one landscape, global optimum randomly moves while one landscape is fixed, and switching between the two landscapes. In [5], Branke also proposed a landscape switching change in the Oscillating Peaks function, making the problem oscillate between some pre-fixed landscapes in given intervals. Furthermore, in Gaussian Peaks [16], a landscape with random peaks is subject to random movements where each peak changes in a predetermined gradual or abrupt way.

In the Moving Peaks Benchmark (MPB) problem [5], at each environmental change; the height, width and position of each peak randomly changes dependent on a severity factor. The MPB is a versatile and flexible benchmark and is used extensively in EDO and ROOT.

Similarly to MPB, Morrison and Jong [30] proposed DF1, with problem instances in which the width, height, and location of peaks change over time. Changes within this problem can be controlled by a logistic function to generate various step sizes. The benchmark is highly configurable where the number of dimensions, the number of peaks, and the dynamics of peaks are flexible [12,29,37].

In Dynamic Rotation [44], the landscape is combined with a visibility mask which allows a percentage of the search space to be masked with a predefined fitness value where the rest of the landscape has the original values. The landscape or the mask is rotated revealing and masking sections of the landscape.

The General Dynamic Benchmark Generator (GDBG) [22] is a benchmark generator that uses rotation as well as shifting to generate environmental changes. GDBG is a combination of existing ideas and functions to analyse a wide range of characteristics in DOPs. The set contains varying functions such as landscape rotation[38,42,44], and uses dynamic rules to control change steps [28].

Many problems in the literature are simply fitness landscapes of a particular design with a varying severity that can be predicted, leading to a partially predictable fitness landscape. Other generators or problems can have predictable features regarding the frequency and periodicity of the changes. However, prediction is not their main characteristic. The prediction of future environments is only reliable if a perfect prediction can be made [4]. Bad predictions may even lead to worse results than are obtained by optimising only the present. Hence, careful design and performance assessment of methods that predict the future are called for.

3. Proposed MPBA benchmark problem

3.1. Definitions

We focus our benchmark to help solve for prediction problems where it either predicts how the optimum move through the fitness landscape, or what environment will appear next. We do so by directly influencing the structure of the next environment. Algorithms that predict when the environment will change still could use this benchmark problem, however such dynamic is achieved only if the change frequency of the problem is deterministic.

We extend the MMPB [5,21,49] as it remains easily extensible and flexible to cover a wide variety of characteristics DOPs exhibit. In our

problem, a location separate to the optimum for which we call a 'attractor' is chosen in the landscape. By adjusting how the peak is moved towards this location in each environment change, we can produce different types and degrees of movements of the optimum. As the attractor remains fixed for a period of time, this movement is not fully random and therefore the EDO algorithm may be able to exploit this information with its prediction method.

In MMPB, at each environmental change, the height, width and position of each peak randomly change dependant on a severity factor. This test set is modified to allow each peak to have it's severity factor where it is possible to change some areas of the environment space more severely than others, making this an appropriate benchmark to test ROOT algorithms. The base equation of the MPB is described as follows:

$$F_t(\vec{x}) = \max_{i=1,2,\dots m} \left\{ H_t^i - W_t^i \cdot \left\| \vec{X} - \vec{C}_t^i \right\|_2 \right\} \tag{1}$$

where m is the number of peaks, X is a solution in the problem space, and H_t^i, W_t^i, C_t^i are the height, width and centre of the ith peak in the tth environment. The shift severity is a definable parameter regulating the severity (length) of the movement. This value controls the severity of change: the higher the value, the larger the changes and hence the more difficult the problem becomes. The MPB is modified by allowing each peak to move independently. The movement of the ith peak is mathematically expressed for the height (Eq. (2)), the width of the peak (Eq. (3)), the value of how far the peak is shifted (Eq. (4)) which is then used to move the centre of the peak (Eq. (5)) as follows:

$$H_{t+1}^i = H_t^i + h_s^i \cdot \mathcal{N}(0,1) \tag{2}$$

$$W_{t+1}^i = W_t^i + w_s^i \cdot \mathcal{N}(0,1) \tag{3}$$

$$\vec{V}_{t+1}^i = \mathrm{S}_s \cdot \frac{(1-\lambda)\cdot\vec{r} + \lambda \cdot \vec{V}_t^i}{\left\|(1-\lambda)\cdot\vec{r} + \lambda \cdot \vec{V}_t^i\right\|} \tag{4}$$

$$\vec{C}_{t+1}^i = \vec{C}_t^i + \vec{V}_{t+1}^i \tag{5}$$

where $N(0,1)$ is a random number from a Gaussian distribution with a zero mean and one variance, λ is the correlation coefficient, and $\vec{r}$ is a random vector drawn by a random number between $[-0.5, 0.5]$ for each dimension. h_s^i, w_s^i, and S_s denotes the height, width and shift severity.

Extending the MMPB, we introduce an attractor function, which includes an attractor weighting, and the Euclidean distance between the peaks centre and the attractor location, into the shift function as follows:

$$\vec{V}_{t+1}^i = \mathrm{S}_s \frac{(1-\lambda)\cdot\vec{r} + \lambda \cdot \vec{V}_t^i}{\left\|(1-\lambda)\cdot\vec{r} + \lambda \cdot \vec{V}_t^i\right\|} + w_a \cdot |\vec{A}_t^i - \vec{C}_t^i| \tag{6}$$

where S_s is the shift severity factor, w_a the weighting of the attractor. The Euclidean distance between the attractor, $\vec{A}_t^i$, and peak, $\vec{C}_t^i$, is used as the metric to influence the peaks movement. The purpose of the attractor is to guide the peaks through the landscape to create dynamics that could be predicted. Using the attractor weighting, as defined in Section 3.2, adjusts the severity of how a peak is attracted to the location, and the attractor movement in Section 3.3 adjusts how the peak can move through the environment.

3.2. Attractor weighting

An important attribute in the proposed benchmark problem is the attractor weighting which describes how quickly a peak is attracted to the attractor location. A higher weighting value ensures a quick convergence to the location whereas a low weighting produces a slow convergence, and the peak undergoes a stronger random walk. This adjustable parameter leads to the predictable movement we see in the MPBA problem. An ideal value should be chosen for the motion method and required predictability. However, given a higher weighting, the peak is attracted to the attractor location, and if the environment does not change when the peak reaches the location, the peak will continue with a small random walk around this attractor reducing the predictability of the problem.

Fig. 1 shows the effect of combining different weightings with attractor change using a density plot to show the average location of a peak over a number of runs given the same peak starting location and attractor dynamics. It shows that a balance between the change frequency and attractor weight is needed to ensure that the peak does not converge early to the attractor location. If we consider prediction approaches where the location of the peak is used as the training data, any randomness in the data, such as a peak moving around the attractor when converged, will cause errors in the prediction.

3.3. Attractor dynamics

We define the case where one attractor can influence one or more peaks and as such, we can increase or decrease the complexity and flexibility of the prediction problem. At present, this movement of individual optima has not been studied in DOPs that use prediction methods. However, future studies may benefit from the inclusion of this feature in the benchmark problem.

One attractor per peak allows the benchmark to be fully configurable. As shown in Fig. 2, it would be possible for each peak to move in it's style and direction. For example, in a group of optima, each individual moves in a circular pattern at different rates, and hence a predictive model could identify and capitalise on these patterns. Furthermore, by coupling one attractor location to multiple optima, over time a group of optimum converge to one point in the environment space, providing enough information to the predictive model that could forecast the movement of most optima.

Like individual peaks within the benchmark function, we also consider the motion and change frequency dynamics of a given attractor point. For a given problem, if the attractor location remains static for all environmental changes until the problem terminates, the peak will move to it's attractor location and remain in the immediate area, moving with a random motion. While this initial movement from the start to the attractor may be predictable, the random movement around the point may not be. We can define two dynamics that control an attractor location, the change frequency and the movement style.

Similar to the frequencies of environmental changes in the benchmark function, the change period defines how many fitness evaluations are made until the attractor location moves again. An attractor's location can change at the same change frequency of the benchmark function, or independently to the problem change frequency, such as a different constant rate or a varying rate taken from a random Gaussian distribution. We can define this constant change frequency in Algorithm 1.

Algorithm 1 Change attractors in a constant frequency.

Input: Current generation G, change frequency υ

```
if (G mod υ = 0) then
    Move attractors
end if
```

For problems where the goal is to track moving optima, a constant rate of change is ideal so that the need for an advanced method of detecting environmental changes is not required. However, algorithms that predict when an environmental change occurs and introduce diversity into the population may show an increase in quick convergence [39]. In this case, we define a random change frequency using either a uniform method (see Algorithm 2) or a Gaussian method (see Algorithm).

We can also define how the attractor location moves over a series of changes. The movement of the attractor influences the direction of travel the peak is attracted to, and in combination with varied weightings w_a, different movement styles of a peak can be achieved. The following shifts in this study are proposed; linear, randomised, and deterministic. These movement types encompass many changes that can be

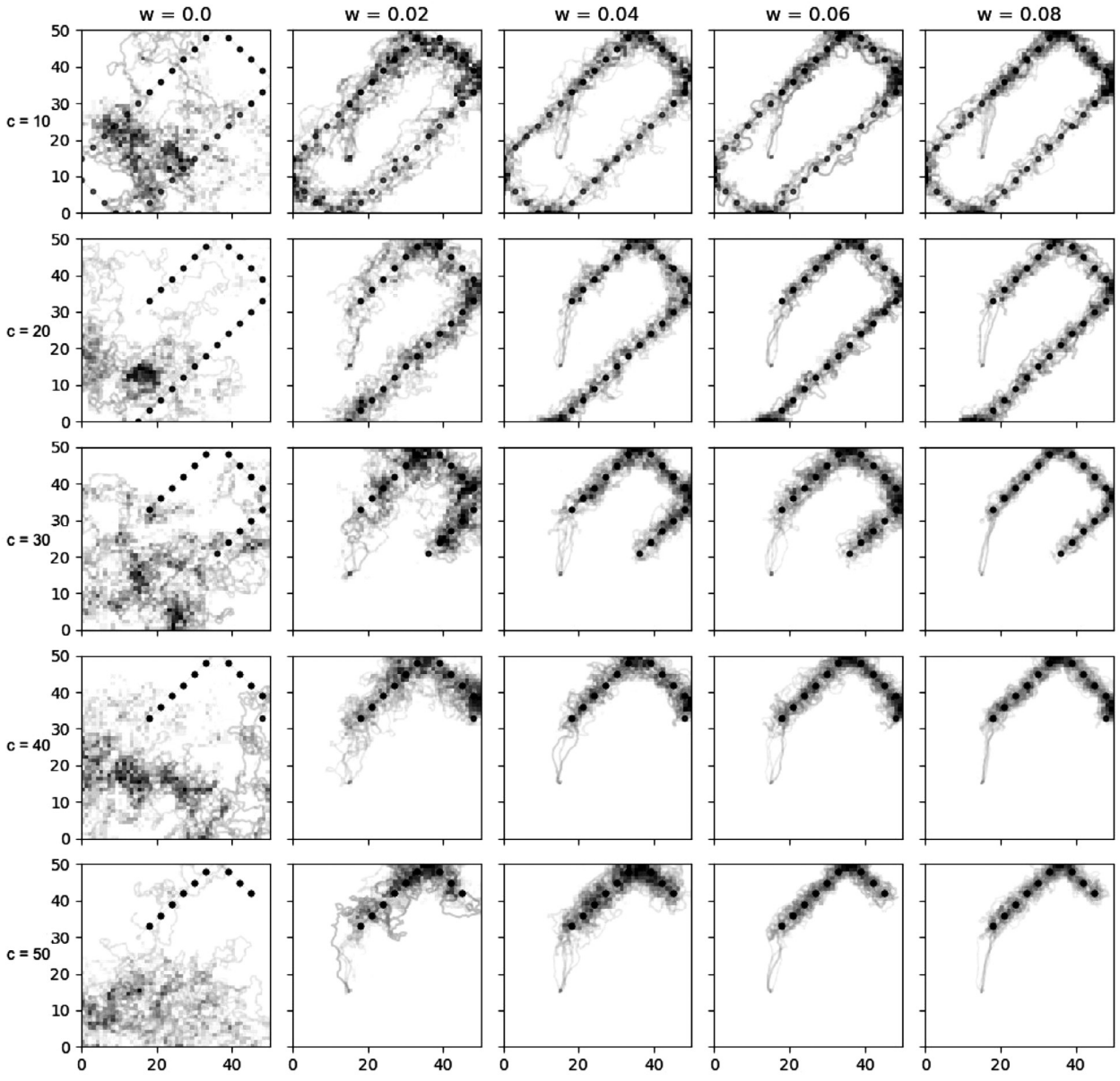

Fig. 1. The distribution of 100 samples of a peak's movement with varying attractor weight w and change frequency c. For each figure, the experiment was run 30 times, with the one peak initialised at a location of [15, 15], The attractor location is started at [15, 30], with a linear motion movement type, that follows the same direction in all examples. Each experiment has 500 environmental changes. The figure shows the traces of the peaks movement, where a higher density around a attractor location shows that most samples of the peaks were pulled towards the attractor. A peak is attracted to the attractor quickly given a higher attractor weight. A peak with a low attractor weight undergoes more of a random movement through the environment. The effect of the change frequency shows that with a high frequency the peaks stay around the attractor longer, whereas with a lower frequency the peak has a higher chance of a random walk while moving towards the attractors location. Balancing the change frequency, weighting and attractor location would ensure that the peak undergoes a smoother movement with varying degrees of randomness.

Algorithm 2 Change attractors in a uniform randomised frequency.

Input: Current generation G, change frequency υ, minimum change frequency m, maximum change frequency n

1: **if** ($G \mod \upsilon = 0$) **then**
2: $r = U(m, n)$
3: $\upsilon = r$
4: Move attractors
5: **end if**

Algorithm 3 Change attractors in a Gaussian randomised frequency.

Input: Current generation G, change frequency υ, distribution mean c, standard deviation of distribution s

1: **if** ($G \mod \upsilon = 0$) **then**
2: $r = N(c, s)$
3: $\upsilon = r$
4: Move attractors
5: **end if**

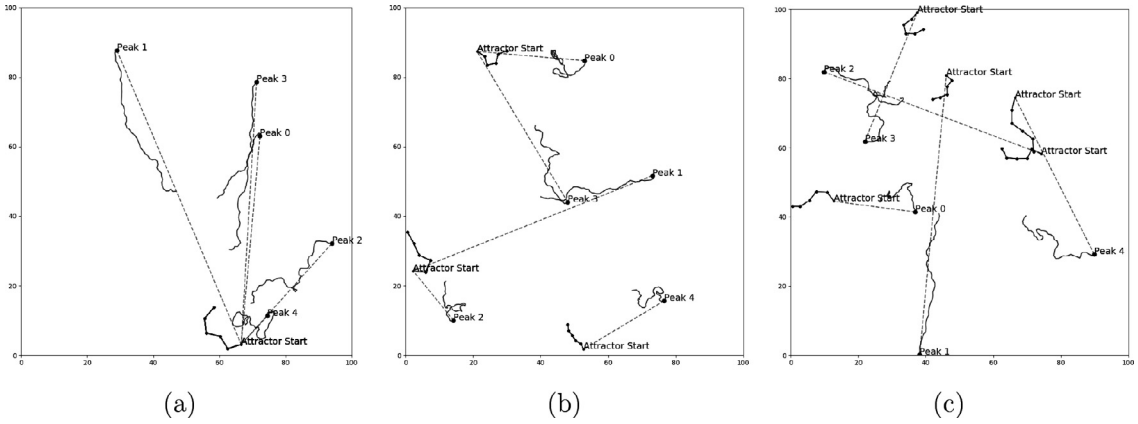

Fig. 2. A set of peaks movement given a varying number of attractors. Each peak is labelled and it's movement is represented by a thin-weighted line. The attractor's start and movement are shown as a thick-weighted line. The peaks attractiveness to an attractor location is represented using a dashed line. Each figure shows how a peak moves towards an attractor. The dynamics of the problem change when all peaks move towards an attractor versus when each peak has it's attractor. (a) 5 peaks attracted to one attractor. This figure shows that all peaks move towards the single attractor location at each environmental change. (b) 5 peaks attracted to 3 attractors. Each peak is initialised with attractiveness to a random attractor. (c) 5 peaks each with it's attractor. In this example, only one peak can belong to one attractor.

Algorithm 4 Linear motion of Attractors.

Input: Attractor location list Al, current movement value list m, environment bounds b

for all Attractor location, a in Al **do**
  if a not in bounds, b **then**
    $m = m \cdot -1$
  end if
  $a = a + m$
end for

Algorithm 5 Randomised motion of Attractors.

Input: Attractor location list Al, environment bounds b, distribution mean c, standard Deviation of distribution s

for all Attractor location, a in Al **do**
  $r = N(c, s)$
  $a = a + r$
  if a not in bounds, b **then**
    $a = b$
  end if
end for

Algorithm 6 Deterministic motion of Attractors.

Input: Attractor location list Al, deterministic location list Dl, current index i

for all Attractor location a in Al **do**
  if $i > length(Dl)$ **then**
    $l = Dl[1]$
  else
    $i = i + 1$
    $l = Dl[i]$
  end if
  $a = l$
end for

seen in other benchmark problems and as such are proposed as a general approach to how the attractor location can change. In most cases, the attractor location may change in a randomised fashion, which would partially replicate the MMPB problem but with slightly more predictable features.

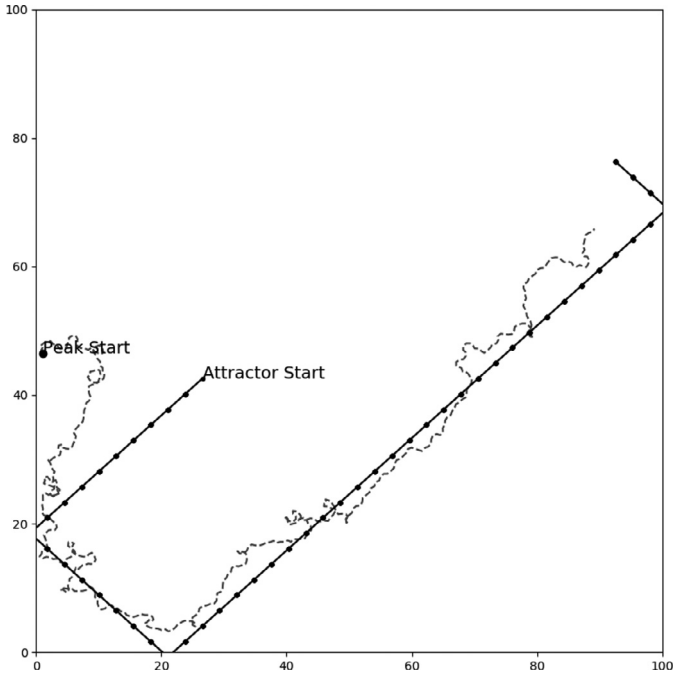

Fig. 3. One peak's movement represented as a dashed line, dependent on the linear movement style of the attractor. This example shows a peaks movement that is similar to the linear movement of the attractor.

Based on previous benchmarks where the peaks move in a linear fashion [39], an attractor can move given a direction and constant rate. In combination with a high attractor weighting, the same dynamic can be achieved. The linear motion is described in Algorithm and the effect of the attractor motion on the peak can be seen in Fig. 3. This type of movement may be easier to predict in comparison to other movement types since the direction and change frequency of the attractor remains constant.

While the above methods are suitable for dynamic problems that last for a long time period, understanding the peaks motion using a large memory is useful. From Fig. 4, a random attractor movement is constrained to the minimum and maximum bounds of the environment, such motion is not predictable and will not produce ideal training data

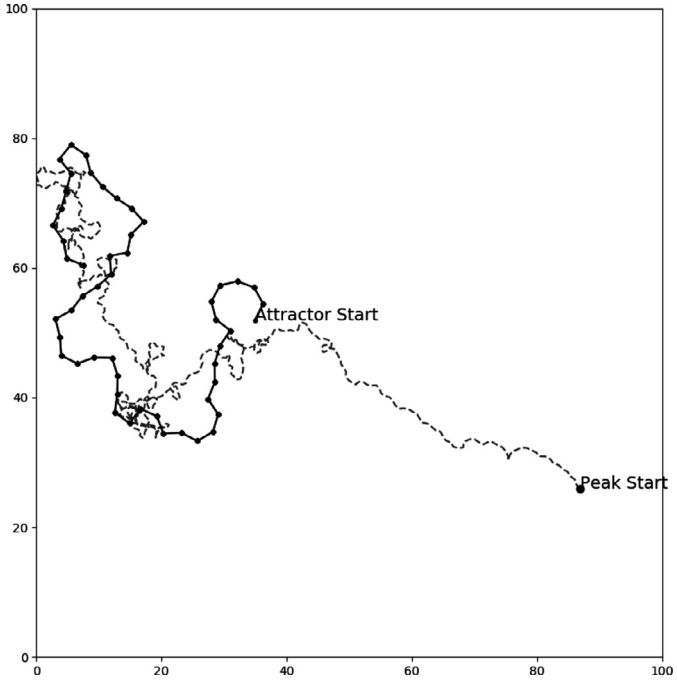

Fig. 4. One peak's movement represented as a dashed line, dependent on the random movement style of the attractor.

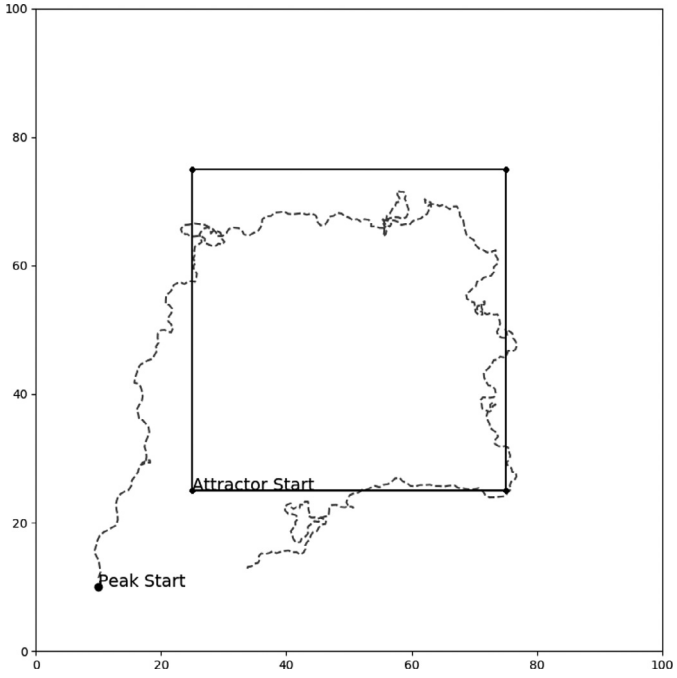

Fig. 5. One peak's movement represented as a dashed line, dependent on the deterministic, square movement style of the attractor. A cyclical motion of the peak can be created by varying the location, change frequency and weighting of the attractor.

for a prediction model. In most cases, a linear motion attractor location or deterministic approach is suitable.

In the randomised approach (see Algorithm), the attractor location undergoes a random walk with random step sizes across the fitness landscape. An optimiser's prediction method may be successful for the time period between two changes. However, when the attractor location changes, the historical data may not be useful.

We also define a deterministic approach, where the attractors locations are predetermined in the initialisation of the benchmark function, as shown in Algorithm. Using a deterministic approach, we can move a peak in a motion that creates an n-dimensional shape by tuning the weighting and attractor location parameters. Such a configuration achieves a highly flexible benchmark, allowing for various fitness landscape changes to be observed, as shown in Fig. 5. For example, by arranging a set of attractors in a circular pattern (described in Algorithm 7), the motion of the associated peak is observed to move in such a pattern. Two considerations must be made; firstly, the location of each attractor, and secondly, the order at which each attractor is set in the list, which affects such creation of the shape.

Algorithm 7 Generate points on a circle.

Input: Circle center c, circle radius r, number of points p, location list l

$s = 2 \cdot \pi / p$
for all $t = i, \cdot p$ **do**
  $a = t \cdot s$
  $x = c^x + r \cdot \sin(a)$
  $y = c^y + r \cdot \cos(a)$
  $l.insert(x, y)$
end for

In Fig. 6, we show five attractor settings and three types of attractor movement changes: Linear; Deterministic positioned as a square pattern; and another deterministic method positioned in a circular pattern. The figure highlights the effect of different peak movements over a set number of runs, represented as a density plot. In most cases, increasing the attractor weighting to a value of 0.015 achieves a balanced peak movement as it is both attracted to each attractor but does not converge to the point prematurely.

4. Performance metrics

In this section, four performance measures are presented. The performance measures are chosen based on how well an algorithm chooses the best solution, how quick the convergence to an optimum is, and the performance of the forecasting method. We choose the Offline Error (OE) [2], Best Error After Change ($BEAC$) [43], Absolute Recovery Rate (ARR), and our proposed measure, the Best Distance After Change ($BDAC$).

4.1. Offline error

The first metric considered is the offline error [2]. This is measured as the average over, at every evaluation, the error of the best solution found since the last change on the environment. A value equal, or close to zero is considered perfect. This is defined as follows:

$$OE = \frac{1}{G} * \sum_{i=1}^{i=G} \left(\frac{1}{N} * \sum_{j=1}^{j=N} BOG_{ij} \right) \tag{7}$$

where G is the total number of generations, N is the total number of runs and BOG_{ij} is the best fitness value found in the last change on the environment referring to the ith generation and jth run j of the algorithm at hand.

4.2. Absolute recovery rate

The absolute recovery rate [33] of an algorithm defines how quick it starts converging on the global optimum before the next change occurs, which is given as follows:

$$ARR = \frac{1}{K} \sum_{i=1}^{K} \frac{\sum_{j=1}^{p(i)} \left[f_{best}(i,j) - f_{best}(i,1) \right]}{p(i) \left[f^*(i) - f_{best}(i,1) \right]} \tag{8}$$

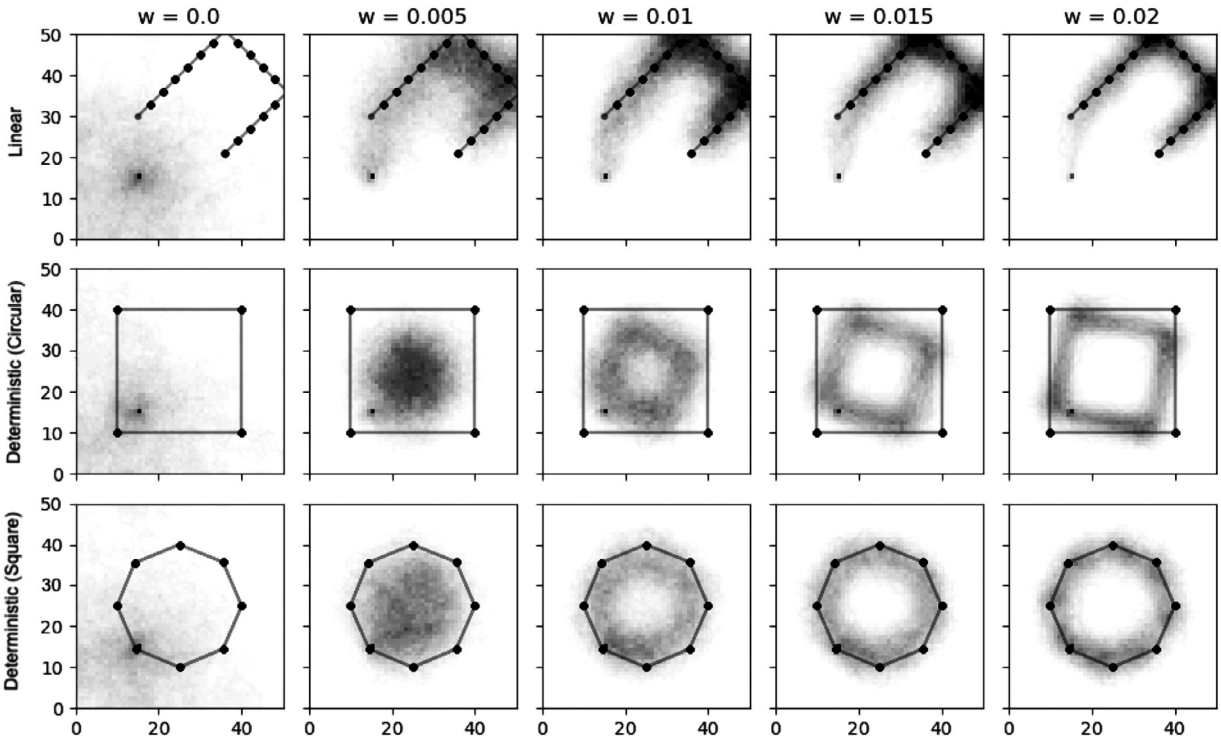

Fig. 6. Two-dimensional density distribution charts of MPBA peak movement over 100 environmental changes for three different values of w = 0,0.005,0.01 and attractor movement styles [Linear, Deterministic (Square), Deterministic (Circular)]. Peak starting location of (15, 15). Adjusting the weighting makes the peak undergo a more random movement, that over time shows the motion can be predicted.

where $f_{best}(i, j)$ is the fitness value of the best solution chosen by the algorithm until the jth generation of the change period i, K is the total number of changes, and $p(i)$ is the number of generations at each change period i and $f^*(i)$ is the global optimal value of the landscape at the ith change. This measure allows to test how well an algorithm predicts future environments to increase the rate of convergence performs. The ARR result would be 1 in the best case when the algorithm is able to recover and converge to the global optimum immediately after a change, and would be zero in case the algorithm is unable to recover from the change at all [41].

4.3. Best error after change

Using the best error after change, the performance of an algorithm in terms of how the population is distributed through the search space can be evaluated as follows:

$$\text{BEAC} = \frac{1}{K}\sum_{i=1}^{i=K}[f^*(i) - f_{best}(0, i)] \tag{9}$$

where K is the total number of changes and $f_{best}(0, i)$ is the best individual just after an environmental change in the ith change period and $f^*(i)$ is the global optimal value of the landscape at the ith change period.

The metric measures the difference of fitness between the global optima and the best individual in the population. A value equal to 0 will suggest the algorithm places an individual at the exact location of the global optima.

4.4. Best distance after change

While the previous measures evaluate the convergence rate and fitness rate, measuring the performance of algorithms that incorporate predictive measures is a challenge. To address this challenge, in this paper, we propose a measure that considers the distance between the global optima and the closest individual in the population after an environmental change has occurred. Other measures exist that measure the distance between optima and individuals. However, these metrics focus on the distance at the end of generation runs. Measuring the distance after a change occurs and when the individuals are repopulated is important to understand the effect of the predictive function within the optimiser, if the goal is to identify the best location to place the individuals. Similar to OE and BEAC, algorithms should aim for a value of 0, meaning that a individual in the new generation was placed at the exact location of the global optima. This Best Distance After Change (BDAC) measure is defined as follows:

$$\text{BDAC} = \frac{1}{K}\sum_{i=1}^{i=K} f_{closest}(0, i) \tag{10}$$

where K is the total number of changes and i is the closest individual to the global optimum after an environmental change. $f_{closest}(0, i)$ is the fitness value of the closest individual to the global optimum chosen by the algorithm in the first change period (0).

Considering that current EDO approaches aim to increase diversity or converge to a global optimum as quickly as possible, this measure is tolerant to both approaches. It also does not consider how long it takes to converge to the global optima therefore it is immune to convergence methods. In the case of diversity, typical approaches distribute the population around the previous optima, assuming that the new optimum is within this distribution of individuals in a population. The closest individual will be placed near the previous optima. Prediction methods will distribute individuals around the estimated location of the global optima and therefore it's centre could be assumed to be at the global optima given a perfect prediction.

With current performance measures, the result could be deceived if the optimiser converges to a local optimum where the fitness value is close to the global optimum. In these cases, the performance measures would show good results. At the same time, when a new population is initialised using an optimiser that does not predict the movement of the

Table 1
Parameter settings for MMPB and MPBA benchmark problems .

Benchmark	Parameter	Admissible values
MMPB & MPBA	Number of Peaks 'm'	5, 10, 20
	Change Frequency ''	2500
	Shift Severity 'S_s'	$\mathcal{U}$[1] (0.5,3)
	Height Severity 'h_s'	$\mathcal{U}(1, 15)$
	Width Severity 'w_s'	$\mathcal{U}(0.1, 1.5)$
	Peaks Shape	Cone
	Number of Dimensions 'd'	2, 5, 10
	Correlations Coefficient 'λ'	0
	Peaks Location Range 'C_t^i'	[0, 50]
	Peak Height 'H_t^i'	[30, 70]
	Peak Width 'W_t^i,'	[1, 12]
	Initial Height Value	50
	Initial Width Value	6
	Number of environments	100
MPBA	Attractor Weighting - 'W_a'	[0, 0.005, 0.01]
	Change frequency '**c**'	[1, 2, 5, 10, 20]
	Attractor Movement Styles	Rnd, Linear, Square, Circle

[1] $\mathcal{U}(a, b,)$ indicates a uniformly distributed number in $[a, b]$.

optima, the randomness of the population would deceive any measure that takes into account the error, or distance from the global optimum as an individual may randomly be placed close to the global optimum. With an optimiser that uses prediction, it could be stated that most of the population would be initialised close to the optima over successive environment changes. Therefore, the value achieved for BDAC will be better than that for a non-predictive optimiser.

5. Experimental study

5.1. Compared algorithms

We compare seven dynamic optimisation algorithms taken from a range of literature in different areas of study. The performances of these EDO algorithms on the proposed MPBA and MMPB are assessed through the metrics outlined in Section 4.

The algorithms are chosen based on different characteristics and methods to solve DOPs. First, we choose four algorithms that deal with EDO: 1) a simple Genetic Algorithm (SGA) [15]; 2) Neural Network for Change Prediction (NNCP) [24]; 3) Dynamic PSO with partial restart and prediction (DynPSO) [25]; and 4) Self organising scouts, (SOS) [7]. We also use two other algorithms that incorporate a prediction method within their framework: 1) DQMOO/AR from Hatzakis and Wallace [18]; and 2) The ROOT framework proposed by Jin et al. [21]. Each algorithm uses prediction in a different method, DQMOO/AR uses autoregressive forecasting to influence the next generation of individuals to guide search to an optimum location by initialising individuals at the predicted optimum location. ROOT uses forecasting to estimate the fitness of an individual in future changes.

By comparison between MMPB and MPBA on DQMOO/AR and ROOT, and by other EDO approaches we can see whether the algorithms using our proposed benchmark function exhibited better performance results when the benchmark problem is predictable than using MMPB. Performances that improve when the predictability of the benchmark problem increases will confirm that using MPBA allows researchers to fully utilise the prediction components in their algorithms. However, if results were similar when the predictable factors are increased (for example, as we increase the attractor weighting) it will be shown that including these factors does not contribute to evaluating algorithms that use prediction.

As such, it is known that the goals for EDO algorithms and ROOT are different and it would be incorrect to compare the results with each other. We therefore include ROOT in the experiment to understand how well the ROOT algorithm performs with MPBA only with itself and we do not compare the results against other algorithms.

Each algorithm uses the respective parameters proposed in their studies. In the case of ROOT, we use a historical length p of 3, the prediction length, q, of 8 with the metric to optimise the survival length of a solution. Therefore we choose a survival threshold value of 40. We also define the parameters for the benchmark functions in Table 1.

5.2. Results

In order to compare the effectiveness of the MPBA problem, we test MMPB and our proposed benchmark, MPBA, using the proposed algorithms with three combinations of peaks (5, 10, 20), and dimensions (2, 5, 10). For all experiments, we use a total of 100 environmental changes, with 2500 fitness evaluations for each environment. Each experiment is repeated 30 times, and performance measures are averaged

Table 2
Results from EDO algorithms by found performance measures with a varied number of peaks and dimensions on the MPBA benchmark problem.

Metric	Algorithm	2 dimensional case with			5 dimensional case with			10 dimensional case with		
		5 peaks	10 peaks	20 peaks	5 peaks	10 peaks	20 peaks	5 peaks	10 peaks	20 peaks
OE	SGA	61.1528	63.5857	64.0598	51.5595	59.7447	53.4385	7.9524	33.7873	44.4827
	NNCP	2.2623	1.8543	1.3761	9.7939	7.7112	6.2822	19.9207	15.2768	10.1201
	DynPSO	2.1759	1.7217	1.2707	9.3749	8.2579	8.1403	16.8494	16.4264	13.2080
	SOS	41.5055	61.6210	63.2307	58.7672	35.9017	55.0555	34.0690	56.9809	58.5311
	DQMOO/AR	61.1878	63.3932	66.1575	48.2973	62.9116	51.6674	5.0271	7.9267	33.8455
	ROOT	54.4444	62.7267	62.7366	36.2214	51.9576	52.9165	19.6264	20.7326	23.5916
ARR	SGA	0.9350	0.9439	0.9442	0.6266	0.6093	0.6762	0.2545	0.2635	0.2260
	NNCP	0.8425	0.8326	0.9219	0.7702	0.7663	0.8052	0.5105	0.5580	0.6474
	DynPSO	0.8518	0.8448	0.9200	0.7683	0.7771	0.8000	0.5055	0.5563	0.6482
	SOS	0.8743	0.8893	0.9026	0.7650	0.7746	0.7897	0.5779	0.6779	0.6760
	DQMOO/AR	0.7645	0.7176	0.7237	0.6529	0.6434	0.6117	0.5193	0.5014	0.5052
	ROOT	0.7501	0.7306	0.7306	0.6490	0.6267	0.5800	0.5110	0.4989	0.4658
BEAC	SGA	15.6624	13.5314	12.2177	79.0046	76.7815	56.4205	159.5400	147.4190	108.1240
	NNCP	18.7501	17.1788	16.5312	59.8322	58.7864	63.5489	116.9712	104.7152	104.1027
	DynPSO	19.1174	16.9549	17.4090	55.9947	58.5579	64.5354	121.6022	110.6604	99.8927
	SOS	13.5697	10.8431	10.8024	15.2459	17.0265	15.8321	21.3898	18.8331	21.4261
	DQMOO/AR	17.0081	16.1959	11.4521	80.6930	56.8730	47.6302	142.4135	130.0634	103.1469
	ROOT	22.4282	18.4552	18.4553	90.1439	69.5346	53.8776	175.0942	125.9021	111.3809
BDAC	SGA	5.0001	4.8662	5.1024	29.1730	28.1593	28.9609	63.0752	67.3776	65.1060
	NNCP	5.9984	4.9119	4.4973	19.3544	19.0196	19.9262	35.2580	35.2565	37.5192
	DynPSO	6.1786	4.9335	4.5363	19.9683	19.4512	20.1058	35.3860	37.6248	37.1735
	SOS	27.4090	34.4512	37.8619	64.1490	77.0311	77.7639	99.9095	104.6710	115.2250
	DQMOO/AR	4.8158	5.0024	5.0300	24.4860	26.5906	26.6431	48.1034	55.5575	57.2986
	ROOT	5.13930	5.01092	5.01092	28.28805	27.71533	28.32473	60.81157	61.43584	61.43779

Table 3
Comparison of the effect of different attractor movement types in MPBA on the performance of compared algorithms.

Attractor movement type	Metric	SGA	NNCP	DynPSO	SOS	DQMOO/AR	ROOT
Random	OE	5.8434	5.5356	2.2562	1.9103	5.0318	0.5838
	ARR	0.9343	0.9016	0.9063	0.9089	0.7305	0.7716
	BEAC	6.0634	4.6401	5.5339	6.0753	5.1028	4.9529
	BDAC	23.9798	35.9756	23.1130	24.9508	18.8501	21.7059
Linear	OE	17.1063	4.9849	1.7716	1.6526	0.7249	0.6161
	ARR	0.9623	0.7613	0.8962	0.8968	0.9073	0.8042
	BEAC	4.7852	4.4017	4.1024	3.9774	3.5198	5.0649
	BDAC	31.1303	22.7834	15.9865	14.8060	10.5071	19.3378
Deterministic - Square	OE	4.5798	2.9668	1.6706	1.4774	2.3478	0.5456
	ARR	0.9886	0.8323	0.8981	0.9017	0.8569	0.8682
	BEAC	4.8182	5.4500	4.1951	4.2078	3.8472	4.9721
	BDAC	37.2906	31.7149	15.4954	15.2569	13.7828	23.2993
Deterministic - Circular	OE	12.8168	5.6655	1.5851	1.4442	1.3324	0.9418
	ARR	0.5135	0.7743	0.9043	0.8958	0.9009	0.8659
	BEAC	5.6212	5.1019	5.7672	4.3791	3.2799	4.7584
	BDAC	26.3361	34.8776	15.0304	15.0248	15.3557	22.2070

Table 4
OE results of different attractor change frequencies and weights of the MPBA problem.

Weight	Algorithm		Change frequency				
	Name	Indicator	1	2	5	10	20
0	SGA	Average	7.5838	5.5297	10.2335	1.0174	7.9171
		p-score	–	–	–	–	–
	SOS	Average	3.9660	16.8985	6.8822	5.6579	20.7659
		p-score	–	–	–	–	–
	NNCP	Average	3.9660	16.8985	6.8822	5.6579	20.7659
		p-score	–	–	–	–	–
	DynPSO	Average	2.3509	2.2981	2.2405	2.2992	2.2938
		p-score	–	–	–	–	–
	DQMOO/AR	Average	3.5522	3.0909	4.1500	2.8215	3.4053
		p-score	–	–	–	–	–
	ROOT	Average	3.8079	4.4125	3.0422	4.1196	3.6558
		p-score	–	–	–	–	–
0.005	SGA	Average	9.7034	4.4283	5.5910	5.7475	2.5220
		p-score	3.25E−06	1.11E−06	1.83E−06	7.67E−07	1.71E−06
	SOS	Average	7.9148	18.8174	16.7643	21.5184	17.1361
		p-score	2.77E−05	6.66E−07	1.03E−06	2.30E−06	1.77E−06
	NNCP	Average	1.5207	1.7128	1.8574	1.7454	1.9511
		p-score	0.0045	0.0025	0.0032	0.0012	0.0005
	DynPSO	Average	1.3982	1.7601	1.7268	1.8965	2.0112
		p-score	0.0003	0.0020	0.0005	0.0022	0.0008
	DQMOO/AR	Average	0.9738	1.9817	2.4833	2.9603	2.2150
		p-score	1.41E−05	0.0082	0.0003	0.7655	0.0026
	ROOT	Average	1.8223	1.5933	3.1031	1.6067	2.5210
		p-score	1.86E−06	1.98E−06	0.2802	2.03E−06	1.73E−06
0.01	SGA	Average	8.1866	3.9069	7.7702	13.3060	5.9055
		p-score	4.07E−06	1.50E−06	2.71E−06	2.27E−06	2.02E−08
	SOS	Average	11.1045	8.3810	9.5788	15.8454	14.4089
		p-score	2.13E−07	1.30E−06	9.30E−07	1.58E−06	1.29E−06
	NNCP	Average	2.0190	2.0105	1.8274	2.0422	2.1019
		p-score	0.0034	0.0045	0.0021	0.0008	0.0010
	DynPSO	Average	1.9798	1.9656	1.8823	1.9640	2.1578
		p-score	0.0041	0.0002	0.0022	0.0012	0.0020
	DQMOO/AR	Average	2.2640	2.3403	1.5999	2.7231	2.4222
		p-score	0.0017	0.1589	3.39E−05	0.6884	0.0207
	ROOT	Average	2.4273	2.6094	4.4938	5.4267	4.8426
		p-score	5.03E−07	9.69E−07	1.10E−06	2.59E−06	2.92E−06
0.015	SGA	Average	8.4445	8.7824	3.3662	5.7417	7.2666
		p-score	6.52E−07	1.42E−06	4.58E−07	1.87E−06	1.11E−06
	SOS	Average	4.2130	7.1745	9.9371	14.4100	7.2080
		p-score	0.0050	1.33E−06	2.95E−06	3.29E−07	3.90E−06
	NNCP	Average	2.3109	2.0629	2.0803	2.2814	2.2969
		p-score	0.0017	0.0067	0.0048	0.0052	0.0018
	DynPSO	Average	1.9898	2.1050	2.1278	2.1885	2.2178
		p-score	0.0028	0.0005	0.0005	0.0016	0.0060
	DQMOO/AR	Average	1.4630	1.7381	2.6173	2.9436	3.7768
		p-score	3.41E−05	0.0036	0.0008	0.5038	0.3820
	ROOT	Average	2.6201	2.9912	4.4177	5.0315	4.3611
		p-score	3.46E−06	3.18E−06	1.59E−06	1.32E−07	4.14E−07

Table 5
ARR results of different attractor change frequencies and weights of the MPBA problem.

Weight	Algorithm		Change frequency				
	Name	Indicator	1	2	5	10	20
0	SGA	Average	0.8986	0.8648	0.9127	0.9793	0.8988
		p-score	–	–	–	–	–
	SOS	Average	0.7919	0.8037	0.8229	0.7890	0.8114
		p-score	–	–	–	–	–
	NNCP	Average	0.7850	0.7871	0.8212	0.8102	0.8083
		p-score	–	–	–	–	–
	DynPSO	Average	0.7952	0.7985	0.8206	0.7857	0.8253
		p-score	–	–	–	–	–
	DQMOO/AR	Average	0.7825	0.7082	0.7431	0.7611	0.6990
		p-score	–	–	–	–	–
	ROOT	Average	0.7603	0.6996	0.7673	0.7608	0.7200
		p-score	–	–	–	–	–
0.005	SGA	Average	0.9184	0.8903	0.8963	0.9248	0.9373
		p-score	0.5999	0.2623	0.4653	0.0495	0.2059
	SOS	Average	0.7990	0.8068	0.8552	0.7892	0.7911
		p-score	0.9263	0.8451	0.1306	0.8612	0.3286
	NNCP	Average	0.7996	0.8123	0.8511	0.7851	0.7995
		p-score	0.0969	0.0473	0.1626	0.2144	0.0482
	DynPSO	Average	0.8094	0.7969	0.8487	0.7872	0.7833
		p-score	0.0597	0.0656	0.1529	0.2147	0.0691
	DQMOO/AR	Average	0.8052	0.8063	0.7928	0.7624	0.7002
		p-score	0.2134	0.0006	0.0207	0.2536	0.9099
	ROOT	Average	0.7947	0.7735	0.8015	0.7932	0.8011
		p-score	0.0316	0.0087	0.1470	0.2210	0.0013
0.01	SGA	Average	0.9090	0.9358	0.9287	0.9071	0.8959
		p-score	0.5577	0.0230	0.5440	0.0098	0.7499
	SOS	Average	0.8740	0.8274	0.7891	0.7331	0.8592
		p-score	0.0018	0.3493	0.2623	0.0978	0.1064
	NNCP	Average	0.8839	0.8275	0.7947	0.7271	0.8515
		p-score	0.0844	0.1568	0.1598	0.2302	0.0727
	DynPSO	Average	0.8723	0.8181	0.7846	0.7281	0.8548
		p-score	0.0727	0.1731	0.1479	0.2094	0.0862
	DQMOO/AR	Average	0.8094	0.7944	0.7660	0.7586	0.7690
		p-score	0.0822	0.0073	0.2802	0.9263	0.0117
	ROOT	Average	0.7615	0.7825	0.6919	0.7095	0.7107
		p-score	0.7036	0.0064	0.0015	0.0087	0.6435
0.015	SGA	Average	0.9599	0.8923	0.9382	0.9464	0.9487
		p-score	0.0472	0.4284	0.4284	0.3086	0.1528
	SOS	Average	0.8789	0.8436	0.8533	0.8253	0.8162
		p-score	0.0003	0.0786	0.1779	0.0897	0.8451
	NNCP	Average	0.8856	0.8549	0.8602	0.8435	0.8095
		p-score	0.0940	0.1289	0.1541	0.2204	0.0690
	DynPSO	Average	0.8733	0.8451	0.8632	0.8279	0.8271
		p-score	0.0892	0.1469	0.1605	0.2130	0.0667
	DQMOO/AR	Average	0.8564	0.8091	0.7376	0.8209	0.7392
		p-score	0.0014	0.0008	0.3820	0.0111	0.0598
	ROOT	Average	0.7775	0.7808	0.7413	0.6754	0.6964
		p-score	0.7189	0.0050	0.0937	0.0032	0.3185

over the runs. To build an adequate set of historical data for the forecasting methods, we also use 100 changes before the performance of the algorithm is measured. Furthermore, for MPBA we include the following combinations: five types of attractor movement dynamics, three types of change frequencies, and three attractor weightings. By testing these combinations, we can understand the effectiveness of MPBA as a benchmark problem for both DOPs that have fully random dynamics, and those that contain predictable elements that can be exploited.

5.2.1. *Effect of varying the number of peaks and dimensions on MPBA*

Table 2 (as well as Tables 1 to 5 in the attached supplementary material document available online with further numerical results) shows the effect of varying the number of dimensions and peaks for the compared algorithms. It can be observed that for most algorithms tested, the offline error increases as the number of dimensions increases. This is expected as the parameters of the algorithms, the population size and change frequency, for example, remain fixed over all experiments. As the number of dimensions increases, the fitness landscape becomes increasingly complex to search.

It can also be observed that as we increase the number of peaks, the fitness increases as the average error of the landscape increases. For OE, both ROOT and DQMOO/AR achieve worse results at higher dimensions than other algorithms. In particular, as the ROOT framework is optimising for an average fitness above a fitness threshold for a length of future environmental changes, this is expected. However, with DQMOO/AR, the number of dimensions further increases the number of states for the forecasting method to predict. OE results highlight that prediction is restricted to the forecasting of the global optima and as such is affected by how every optimum's height changes over time. DQMOO/AR's forecasting method is optimal with one optimum. When multiple optima exist, if the population converges to a local optimum, the forecast from the autoregression using this incorrect data will include large errors. Methods such as SOS, which incorporate swarm and memory methods, will achieve optimal results compared to DQMOO/AR and ROOT.

The BEAC metric reveals that methods that use memory schemes have the best performance to quickly converge to an optimum, as such methods re-initialise their population around the last known optima. In comparison, our proposed performance measure, BDAC, suggests the

Table 6
BEAC results of different attractor change frequencies and weights of the MPBA problem.

Weight	Algorithm		Change frequency				
	Name	Indicator	1	2	5	10	20
0	SGA	Average	20.2497	18.5514	16.1463	16.1771	20.8257
		p-score	–	–	–	–	–
	SOS	Average	15.5009	17.4034	14.0181	16.0127	18.5062
		p-score	–	–	–	–	–
	NNCP	Average	18.0927	19.1387	18.0632	18.0589	18.9421
		p-score	–	–	–	–	–
	DynPSO	Average	19.0014	18.9879	18.8598	18.7166	19.0558
		p-score	–	–	–	–	–
	DQMOO/AR	Average	18.4232	17.4373	18.7191	17.4702	19.7441
		p-score	–	–	–	–	–
	ROOT	Average	20.2023	20.9683	18.6403	25.5603	22.7006
		p-score	–	–	–	–	–
0.005	SGA	Average	19.1566	18.3044	16.6508	15.6566	18.4773
		p-score	2.02E−06	0.0002	2.29E−06	1.90E−06	2.05E−06
	SOS	Average	15.7952	15.7746	13.8126	17.8825	16.3644
		p-score	2.46E−05	3.38E−06	0.0010	9.17E−07	1.83E−06
	NNCP	Average	13.0047	14.0764	14.0042	16.5538	15.9515
		p-score	0.0011	0.0022	0.0038	0.0064	0.0021
	DynPSO	Average	15.9435	15.9730	16.7042	17.0150	18.0439
		p-score	0.0036	0.0010	0.0046	0.0017	0.0007
	DQMOO/AR	Average	19.2073	16.2948	16.5571	16.7927	17.3356
		p-score	8.08E−06	3.42E−06	1.18E−06	1.29E−06	4.78E−06
	ROOT	Average	22.9877	17.2174	21.1418	17.1855	17.1295
		p-score	1.81E−06	8.08E−07	3.00E−06	2.13E−06	7.41E−07
0.01	SGA	Average	17.9154	18.3590	17.8624	21.9122	18.7812
		p-score	2.41E−06	0.0030	2.12E−06	1.62E−06	1.63E−06
	SOS	Average	13.1451	17.8229	17.1413	20.6217	13.1488
		p-score	1.38E−06	1.28E−06	1.87E−06	2.20E−06	1.86E−06
	NNCP	Average	14.0704	13.9896	14.8578	15.4990	15.1196
		p-score	0.0004	0.0011	0.0008	0.0007	0.0056
	DynPSO	Average	16.9714	17.0090	18.0086	18.0827	18.4665
		p-score	0.0008	0.0009	0.0031	0.0014	0.0029
	DQMOO/AR	Average	18.8374	16.1519	17.6296	18.1110	20.6317
		p-score	0.0011	3.26E−06	2.53E−06	0.0028	7.75E−06
	ROOT	Average	18.9888	20.6851	20.1997	25.2803	23.9451
		p-score	8.54E−06	0.0001	2.61E−06	0.0003	2.08E−07
0.015	SGA	Average	13.9394	18.7521	16.8888	18.7312	19.6058
		p-score	1.12E−06	0.0041	1.38E−06	1.68E−06	1.15E−06
	SOS	Average	13.7732	13.6240	14.8224	16.3392	17.4719
		p-score	1.42E−06	1.53E−06	1.40E−06	2.57E−05	1.85E−06
	NNCP	Average	14.8870	16.1788	15.9810	16.8769	16.7043
		p-score	0.0001	2.98347E−05	0.0018	0.0028	0.0009
	DynPSO	Average	17.2969	17.5152	17.6415	18.0015	18.0799
		p-score	0.0010	0.0004	0.0039	0.0023	0.0003
	DQMOO/AR	Average	16.5861	18.6206	17.1618	20.2887	19.2660
		p-score	1.71E−06	5.28E−05	2.45E−06	1.55E−06	0.0032
	ROOT	Average	22.4351	20.9605	24.9969	24.2022	21.3686
		p-score	1.53E−06	0.9426	3.14E−06	1.41E−06	4.78E−07

opposite. The results show that DMQOO/AR, SGA and both DynPSO and NNCP achieve better results than SOS. This could be explained by the movement dynamics of the heights of optima. Considering that the height of optima lies within boundaries, one or more peaks may have similar heights. As the BEAC metric uses the difference between the global best individual that may have converged to a different peak and the global optimum, even if the individual is not on the correct optima, it may cause a good BEAC result. Our method, BDAC, uses the difference in distance from the individual to the optima. From the results, it is revealed that DMQOO/AR, DynPSO and NNCP initialises individuals near to the peak, whereas SOS and ROOT initialise individuals at a greater distance.

5.2.2. Effect of varying attractor movement styles

In Table 3 (and Table 5 in the supplementary material document), the performance of the algorithms under different attractor movements is shown. For all runs, we only include one peak with one attractor, and two dimensions to highlight the performance of the algorithms under an environment that is easy to search. Taking into account the algorithms' performance in OE and ARR, the algorithms ROOT, DynPSO and NNCP is successful in most cases. The results suggest that DQMOO/AR shows the largest improvement for the BEAC and BDAC metrics, where a deterministic-circular movement achieves better results than a linear and deterministic-square movement. A square deterministic shape, however, is slightly worse than a circular shape. This result may be explained by considering how the autoregressive model is built in ROOT. Given a square or circular deterministic movement, each dimension component of the individuals may move according to a sinusoidal pattern. These patterns can be forecasted by the AR model to some degree of accuracy, thus allowing the algorithm for returning satisfactory results. In the case of a linear movement, unless the forecast model is multi-dimensional, the prediction of the movement of the peaks will not be perfect. For our comparison algorithms that use prediction; DQMOO/AR, NNCP and DynPSO shows better results using an attractor movement type that is not random. For ROOT, this shows varied results. However, this is due to the underlying optimisation goal being different from tracking moving optima.

Table 7
BDAC results of different attractor change frequencies and weights of the MPBA problem.

Weight	Algorithm		Change frequency				
	Name	Indicator	1	2	5	10	20
0	SGA	Average	5.2565	5.6400	4.9568	5.1339	5.2909
		p-score	–	–	–	–	–
	SOS	Average	39.4506	42.7475	39.4731	42.3682	44.7401
		p-score	–	–	–	–	–
	NNCP	Average	6.0763	5.9952	6.0901	6.0316	6.0049
		p-score	–	–	–	–	–
	DynPSO	Average	5.9876	5.9909	6.0535	6.0239	6.0204
		p-score	–	–	–	–	–
	DQMOO/AR	Average	5.0089	5.4199	5.1225	5.1696	5.4041
		p-score	–	–	–	–	–
	ROOT	Average	5.1928	4.9296	5.1247	5.4533	5.0543
		p-score	–	–	–	–	–
0.005	SGA	Average	4.9491	5.2298	5.1164	4.9519	5.1514
		p-score	0.0001	3.59E−06	0.0230	0.0017	0.0047
	SOS	Average	35.8729	43.6156	36.3299	35.3178	36.4606
		p-score	1.45E−06	7.30E−07	1.30E−06	8.11E−07	1.75E−06
	NNCP	Average	4.3963	4.2484	4.3982	4.3759	4.4154
		p-score	0.0023	0.0021	0.0039	5.9677E−05	0.0001
	DynPSO	Average	4.1069	4.2253	4.3156	4.3552	4.5205
		p-score	0.0043	0.0040	0.0039	0.0058	0.0039
	DQMOO/AR	Average	4.5251	5.0730	4.8409	5.0158	4.8949
		p-score	3.08E−06	5.51E−06	8.12E−05	0.0087	1.07E−06
	ROOT	Average	4.8932	5.0086	4.9353	4.9937	4.9118
		p-score	2.07E−05	0.1204	0.0018	1.94E−06	0.0270
0.01	SGA	Average	4.9042	4.8841	4.8953	4.7950	5.2034
		p-score	4.45E−05	2.41E−06	0.2059	2.79E−06	0.0786
	SOS	Average	22.6834	36.6053	35.5441	36.3808	30.4558
		p-score	4.56E−06	1.60E−06	3.02E−06	1.23E−06	2.69E−06
	NNCP	Average	4.3698	4.3570	4.7174	4.4839	4.6123
		p-score	0.0030	0.0052	0.0045	0.0030	0.0070
	DynPSO	Average	4.2459	4.4574	4.4817	4.5281	4.6404
		p-score	0.0044	0.0049	0.0039	0.0039	0.0030
	DQMOO/AR	Average	4.7348	5.0066	5.0544	5.1296	4.9002
		p-score	5.29E−05	3.04E−06	0.1156	0.4779	2.89E−06
	ROOT	Average	4.9458	4.9796	5.3327	5.3678	4.9045
		p-score	2.47E−05	0.5038	0.0002	0.0387	0.0093
0.015	SGA	Average	4.8068	5.1177	4.8182	4.9024	4.9770
		p-score	5.79E−06	2.70E−06	0.0207	0.0003	0.0002
	SOS	Average	30.6760	30.1227	23.0714	26.6972	30.4328
		p-score	2.37E−06	1.85E−06	6.22E−06	1.34E−06	2.20E−06
	NNCP	Average	4.5846	4.5953	4.8291	5.1429	5.1992
		p-score	0.0030	0.0039	0.0056	0.0040	0.0030
	DynPSO	Average	4.4850	4.4916	4.5418	4.7232	4.7973
		p-score	0.0032	0.0040	0.0042	0.0036	0.0029
	DQMOO/AR	Average	4.6900	4.9544	5.0277	5.1889	5.2534
		p-score	9.35E−06	4.21E−06	0.0333	0.4405	0.0196
	ROOT	Average	4.9566	5.1138	4.9744	4.9998	5.2591
		p-score	3.36E−05	0.0012	0.0117	1.49E−06	0.0002

5.2.3. Effect of varying attractor weightings and change frequencies

Tables 4 –7 (as well as the extended results shown in Tables 6 to 21 in the supplementary material file) consider the effect of changing the attractor weights and change frequency with a linear movement on the chosen algorithms using our metrics. Two dimensions and one peak were used. For the case of an attractor weight value of 0, the MPBA benchmark function acts the same as the MMPB benchmark problem as each peak is not attracted to an attractor and undergoes the same random movement.

From Table 4, NNCP and DynPSO is the best in most situations, and as the weight is increased, while SOS achieve worse results. When the weight is greater than 0, DMQOO/AR and ROOT produce similar results. As the change frequency increases, a low weighting produces similar results. However, with a high weight and a high change frequency, OE decreases. The reason for this is if an optimum converges to an attractor before the attractor moves to another location, the optimum will randomly move around the attractor location. Additionally, it can be observed from Table 5 that the ARR for NNCP and DynPSO in all cases is better than that of other algorithms. However, there is little difference between the relationship with change frequency and weight.

Comparing BEAC in Table 6 and BDAC in Table 2 shows a clear difference between the two performance measures. First, given BEAC, NNCP is the best algorithm considering all weights and change frequencies whereas ROOT is worse in most cases. Overall, prediction methods achieve better results in comparison to algorithms that do not incorporate prediction. This is the opposite case seen with BDAC in Table 7. As suggested previously, these results may be based on the difference in heights of the global optima to other optima directly impacting the validity of the BEAC results, whereas BDAC shows that the distance from the global optima to the closest individual is not sensitive to other optima. The value of BDAC improves for DMQOO/AR as the weight increases in low change frequencies.

From the Table 8, the results for DynPSO, NNCP and a small range of results for DQMOO/AR suggest that the benchmark does utilise the prediction components of the algorithms better than that of MMPB. This

Table 8
Pairwise Mann-Whitney U tests (significance level of 0.05) on the different change frequencies and weights versus a weight of 0, which has the same behaviour as the MMPB. ▲ denotes a significant result, where - does not. For each algorithm, the column contains the metrics OE, ARR, BDAC, and BEAC (from left to right).

Weight	Algorithm name	Change frequency				
		1	2	5	10	20
0.005	SGA	- - - -	- - - -	- - - -	- - - -	- - - -
	SOS	- - - -	- - - -	- - - -	- - - -	- - - -
	NNCP	▲-▲▲	▲-▲▲	▲-▲▲	▲-▲▲	▲-▲▲
	DynPSO	▲▲▲▲	▲▲▲▲	▲▲▲▲	▲-▲▲	▲-▲-
	DQMOO/AR	- - -▲	▲▲-▲	-▲-▲	-▲- -	- - - -
	ROOT	- - - -	▲- - -	- - - -	▲- -▲	- - -▲
0.01	SGA	- - - -	- - - -	- - - -	- - - -	- - - -
	SOS	- - - -	- - - -	- - - -	- - - -	- - - -
	NNCP	- -▲▲	▲▲▲▲	▲-▲▲	▲-▲▲	▲-▲▲
	DynPSO	-▲▲-	▲▲▲▲	▲-▲▲	▲-▲-	▲-▲-
	DQMOO/AR	- - - -	-▲▲▲	▲- -▲	- - - -	-▲▲▲
	ROOT	▲- - -	▲▲- -	-▲- -	-▲- -	- - - -
0.015	SGA	- - - -	- - - -	- - - -	- - - -	- - - -
	SOS	- - - -	- - - -	- - - -	- - - -	- - - -
	NNCP	-▲▲▲	▲▲▲▲	▲▲▲▲	▲-▲▲	▲▲▲▲
	DynPSO	-▲▲▲	▲▲▲▲	▲▲▲▲	▲▲▲-	▲-▲-
	DQMOO/AR	-▲▲-	▲▲▲-	▲▲▲-	- - - -	- - - -
	ROOT	▲- - -	▲- - -	- - - -	- - - -	- - - -

table shows the pairwise Mann-Whitney U-Test where the combination of the weight and change frequency is paired with the same change frequency but a weighting of 0. When MPBA has a weighting of 0, the movement of the peaks exhibit the same behaviour of the normal MMPB benchmark. For algorithms that does not have a predictive component, such as SGA, there is no significance in the results meaning that the algorithm does not produce better results with MPBA. In comparison, ROOT shows a small amount of significance in some problems.

5.3. Discussion

It is difficult to compare the success of the algorithms as most of the tested optimisation methods do not incorporate the same prediction within their algorithmic structure. While it is appropriate to compare such algorithms in an EDO setting by measuring error, convergence speed and accuracy to a global optimum, in the case of measuring prediction typical EDO approaches still achieve good results given a performance measure that could predict accuracy. Hence, in this work, we propose a metric that measures the distance between the global optima and closest individual after an environment has changed, suggesting that an algorithm that uses prediction will re-initialise individuals as close to the predicted location of the new global optima.

In this study, DQMOO/AR, DynPSO and NNCP is equipped to re-initialise individuals using prediction. Other re-initialisation methods, such as covering or distributing individuals around previously known optima and randomly distributing the population in the whole search space may randomly achieve good results if they are re-initialised close to the global optima, therefore results may be deceiving. Moving the peak in large steps may produce worse results for non-predictive algorithms, where algorithms that use prediction may demonstrate ideal results. Using higher attractor weights makes MPBA suitable for these problems.

Finding a good balance for the change frequency and attractor weight ensures that the peak is continually moving. As previously discussed, if the weight is too high the peak may converge to the attractor location too soon and move around this point. Conversely, a low weight would lead to another undesired case where the peak undergoes too much of a random walk. In both cases, any benefit of predictability is lost. However, the MPBA benchmark is flexible as it allows for both cases, allowing researchers to study predictability under different situations in EDO problems.

6. Conclusions and future work

This paper presents a new benchmark problem, denoted MPBA, which is useful to evaluate evolutionary dynamic optimisers that incorporate forecasting methods to predict how the optima move across the fitness landscape in successive environmental changes. Such methods aim to influence the convergence rate of a population, or in the case of requirement satisfaction problems such as Robust Optimisation over Time, aim to predict the optima to forecast the future fitness values of individuals.

We also introduce a new performance measure that measures the distance between the closest individual in the population and the global optima at the first iteration after an environmental change. This measure is based on the idea that an optimiser with a predictor either places new individuals around the estimated location of the global optima to achieve a quick convergence or spreads the population around the optima to increase diversity. An optimiser that does not use prediction may not place an individual or focus on the re-initialisation of individuals in the new optimum location, thus a worse result of this performance measure is achieved.

MPBA exploits the movement of the optima to create a benchmark problem that can be predicted. The problem uses an attractor location for each optimum, or a set of optima and given an attractor weighting, over each environmental change the optima will be attracted to the location as determined rates. MPBA is fully configurable, as the attractor location can move by following deterministic paths or randomly. By adjusting the attraction weighting, change frequency and movement type of the attractor, we can create predictable optima movements while retaining some degree of random walk allowing EDO algorithms that incorporate prediction methods to be effectively evaluated.

When using our benchmark problem to test a range of algorithms from EDO studies we find that while EDO algorithms that use prediction produce better results than those algorithms that do not use prediction, there is an improvement in prediction-based algorithms as the benchmark problem becomes more predictable.

We envisage several extensions for this work, such as investigating movement patterns by varying the change frequency, attractor weighting and movement styles. Although EDO algorithms have previously demonstrated good performance on this proposed benchmark problem, further work into incorporating more advanced methods of prediction or time-series forecasting in EDO is needed.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Matthew Fox: Conceptualization, Methodology, Software, Data curation, Formal analysis, Validation, Visualization, Investigation, Writing – original draft, Writing – review & editing. **Shengxiang Yang:** Conceptualization, Methodology, Validation, Writing – review & editing. **Fabio Caraffini:** Methodology, Validation, Visualization, Supervision, Writing – original draft, Writing – review & editing.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.swevo.2022.101125.

References

[1] V.S. Aragón, S.C. Esquivel, An evolutionary algorithm to track changes of optimum value locations in dynamic environments, J. Comput. Sci. Technol. 4 (03) (2004) p.127–133.

[2] D. Ayvaz, H. Topcuoglu, F. Gürgen, A comparative study of evolutionary optimization techniques in dynamic environments, in: GECCO '06, Association for Computing Machinery, New York, NY, USA, 2006, p. 13971398, doi:10.1145/1143997.1144213.

[3] T.M. Blackwell, Swarms in dynamic environments, in: Genetic and Evolutionary Computation Conference, Springer, 2003, pp. 1–12.

[4] P.A.N. Bosman, Learning, anticipation and time-deception in evolutionary online dynamic optimization, in: Proceedings of the 7th Annual Workshop on Genetic and Evolutionary Computation, 2005, pp. 39–47.

[5] J. Branke, Memory enhanced evolutionary algorithms for changing optimization problems, in: Proceedings of the 1999 Congress on Evolutionary Computation, vol. 3, 1999, pp. 1875–1882.

[6] J. Branke, Evolutionary approaches to dynamic optimization problems - updated survey, 2001.

[7] J. Branke, Evolutionary Optimization in Dynamic Environments, vol. 3, Springer Science & Business Media, 2012.

[8] J. Branke, H. Schmeck, Designing evolutionary algorithms for dynamic optimization problems, in: Advances in Evolutionary Computing, Springer, 2003, pp. 239–262.

[9] H.G. Cobb, An Investigation Into the Use of Hypermutation as An Adaptive Operator in Genetic Algorithms Having Continuous, Time-Dependent Nonstationary Environments, Technical Report, Naval Research Lab Washington DC, 1990.

[10] H.G. Cobb, J.J. Grefenstette, Genetic Algorithms for Tracking Changing Environments, Technical Report, Naval Research Lab Washington DC, 1993.

[11] C. Cruz, J.R. González, D.A. Pelta, Optimization in dynamic environments: a survey on problems, methods and measures, Soft Comput. 15 (7) (2011) 1427–1448.

[12] S.C. Esquivel, C.A.C. Coello, Particle swarm optimization in non-stationary environments, in: Ibero-American Conference on Artificial Intelligence, Springer, 2004, pp. 757–766.

[13] H. Fu, B. Sendhoff, K. Tang, X. Yao, Finding robust solutions to dynamic optimization problems, in: Proceedings of the 16th European Conference on Applications of Evolutionary Computation, EvoApplications'13, Springer-Verlag, Berlin, Heidelberg, 2013, pp. 616–625, doi:10.1007/978-3-642-37192-9_62.

[14] X. Fu, J. Sun, A new learning based dynamic multi-objective optimisation evolutionary algorithm, in: 2017 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2017, pp. 341–348.

[15] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Pearson Education (US), (1989).

[16] J.J. Grefenstette, Evolvability in dynamic fitness landscapes: a genetic algorithm approach, in: Proceedings of the 1999 Congress on Evolutionary Computation, vol. 3, 1999, pp. 2031–2038.

[17] J.J. Grefenstette, et al., Genetic algorithms for changing environments, in: PPSN, vol. 2, 1992, pp. 137–144.

[18] I. Hatzakis, D. Wallace, Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, 2006, pp. 1201–1208.

[19] M. Jiang, Z. Huang, L. Qiu, W. Huang, G.G. Yen, Transfer learning-based dynamic multiobjective optimization algorithms, IEEE Trans. Evol. Comput. 22 (4) (2017) 501–514.

[20] Y. Jin, J. Branke, et al., Evolutionary optimization in uncertain environments—Asurvey, IEEE Trans. Evol. Comput. 9 (3) (2005) 303–317.

[21] Y. Jin, K. Tang, X. Yu, B. Sendhoff, X. Yao, A framework for finding robust optimal solutions over time, Memet. Comput. 5 (1) (2013) 3–18, doi:10.1007/s12293-012-0090-2.

[22] C. Li, S. Yang, T.T. Nguyen, E.L. Yu, X. Yao, Y. Jin, H.G. Beyer, P.N. Suganthan, Benchmark generator for CEC 2009 competition on dynamic optimization, 2008.

[23] X.-F. Liu, Z.-H. Zhan, T.-L. Gu, S. Kwong, Z. Lu, H.B.-L. Duh, J. Zhang, Neural network-based information transfer for dynamic optimization, IEEE Trans. Neural Netw. Learn. Syst. 31 (5) (2019) 1557–1570.

[24] X.-F. Liu, Z.-H. Zhan, J. Zhang, Neural network for change direction prediction in dynamic optimization 6 (2018) 72649–72662.

[25] A. Meier, Prediction-Based Nature-Inspired Dynamic Optimization, Universität Oldenburg, 2020 Ph.D. thesis.

[26] A. Meier, O. Kramer, Recurrent neural network-predictions for PSO in dynamic optimization, in: Proceedings of the Genetic and Evolutionary Computation Conference, 2018, pp. 29–36.

[27] A. Meier, O. Kramer, Predictive uncertainty estimation with temporal convolutional networks for dynamic evolutionary optimization, in: International Conference on Artificial Neural Networks, Springer, 2019, pp. 409–421.

[28] R.W. Morrison, Performance measurement in dynamic environments, GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, 2003.

[29] R.W. Morrison, Designing Evolutionary Algorithms for Dynamic Environments, Springer Science & Business Media, 2013.

[30] R.W. Morrison, K.A. De Jong, A test problem generator for non-stationary environments, in: Proceedings of the 1999 Congress on Evolutionary Computation, vol. 3, 1999, pp. 2047–2053.

[31] T.T. Nguyen, Continuous Dynamic Optimisation Using Evolutionary Algorithms, University of Birmingham, 2011 Ph.D. thesis.

[32] T.T. Nguyen, S. Yang, J. Branke, Evolutionary dynamic optimization: a survey of the state of the art, Swarm Evol. Comput. 6 (2012) 1–24.

[33] T.T. Nguyen, X. Yao, Continuous dynamic constrained optimization—The challenges, IEEE Trans. Evol. Comput. 16 (6) (2012) 769–786.

[34] M. Rong, D. Gong, Y. Zhang, Y. Jin, W. Pedrycz, Multidirectional prediction approach for dynamic multiobjective optimization problems, IEEE Trans. Cybern. 49 (9) (2018) 3362–3374.

[35] M. Rong, D.-w. Gong, Y. Zhang, A multi-direction prediction approach for dynamic multi-objective optimization, in: International Conference on Intelligent Computing, Springer, 2016, pp. 629–636.

[36] C. Rossi, M. Abderrahim, J.C. Díaz, Tracking moving optima using Kalman-based predictions, Evol. Comput. 16 (1) (2008) 1–30.

[37] S. Saleem, R. Reynolds, Cultural algorithms in dynamic environments, in: Proceedings of the 2000 Congress on Evolutionary Computation, vol. 2, 2000, pp. 1513–1520.

[38] R. Salomon, Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms, BioSystems 39 (3) (1996) 263–278.

[39] A. Simões, E. Costa, Evolutionary algorithms for dynamic environments: Prediction using linear regression and Markov chains, in: International Conference on Parallel Problem Solving from Nature, Springer, 2008, pp. 306–315.

[40] A. Simões, E. Costa, Improving prediction in evolutionary algorithms for dynamic environments, in: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, 2009, pp. 875–882.

[41] N.T. Thanh, Y. Xin, Evolutionary optimization on continuous dynamic constrained problems-an analysis, in: Evolutionary Computation for Dynamic Optimization Problems, Springer, 2013, pp. 193–217.

[42] R. Tinós, S. Yang, A self-organizing random immigrants genetic algorithm for dynamic optimization problems, Genet. Program. Evolvable Mach. 8 (3) (2007) 255–286.

[43] K. Trojanowski, Z. Michalewicz, Searching for optima in non-stationary environments, in: Proceedings of the 1999 Congress on Evolutionary Computation, vol. 3, 1999, pp. 1843–1850.

[44] K. Weicker, N. Weicker, Dynamic rotation and partial visibility, in: Proceedings of the 2000 Congress on Evolutionary Computation, vol. 2, 2000, pp. 1125–1131.

[45] Y.G. Woldesenbet, G.G. Yen, Dynamic evolutionary algorithm with variable relocation, IEEE Trans. Evol. Comput. 13 (3) (2009) 500–513.

[46] Y. Wu, Y. Jin, X. Liu, A directed search strategy for evolutionary dynamic multiobjective optimization, Soft Comput. 19 (11) (2015) 3221–3235.

[47] S. Yang, Evolutionary computation for dynamic optimization problems, in: Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, Association for Computing Machinery, New York, NY, USA, 2015, p. 629649, doi:10.1145/2739482.2756589.

[48] S. Yang, X. Yao, Population-based incremental learning with associative memory for dynamic environments, IEEE Trans. Evol. Comput. 12 (5) (2008) 542–561.

[49] D. Yazdani, T.T. Nguyen, J. Branke, J. Wang, A new multi-swarm particle swarm optimization for robust optimization over time, in: Applications of Evolutionary Computation. Cham, 2017, pp. 99–109, doi:10.1007/978-3-319-55792-2_7.

[50] X. Yu, Y. Jin, K. Tang, X. Yao, Robust optimization over time a new perspective on dynamic optimization problems, in: Congress on Evolutionary Computation, 2010, pp. 1–6, doi:10.1109/CEC.2010.5586024.

[51] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, E. Tsang, Prediction-based population re-initialization for evolutionary dynamic multi-objective optimization, in: International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2007, pp. 832–846.

[52] H. Zhou, X. Yuan, H. Qu, W. Cui, B. Chen, Visual clustering in parallel coordinates, Comput. Graph. Forum 27 (3) (2008) 1047–1054.