Penguins Huddling Optimisation

Mohammad Majid al-Rifaie

***Abstract*—In our everyday life, we deal with many optimisation problems, some of which trivial and some more complex. These problems have been frequently addressed using multi-agent, population-based approaches. One of the main sources of inspiration for techniques applicable to complex search space and optimisation problems is nature. This paper proposes a new metaheuristic – Penguin Huddling Optimisation or PHO – whose inspiration is beckoned from the huddling behaviour of emperor penguins in Antarctica. The simplicity of the algorithm, which is the implementation of one such paradigm for continuous optimisation, facilitates the analysis of its behaviour and the derivation of the optimal value for its single adjustable parameter in the update equation. A series of experimental trials confirms the promising performance of the optimiser over a set of benchmarks, as well as its competitiveness when compared against few other well-known population based algorithms.**

***Index Terms*—Emperor penguins huddling, multi-agent system, continuous optimisation, metaheuristics, nature inspired algorithms.**

I. Introduction

STUDIES of the behaviour of social insects and animals have suggested several new metaheuristics for use in collective intelligence and multi-agent systems. This has given rise to a concomitant increasing interest in distributed computation through the interaction of simple agents in nature-inspired optimisation techniques.

In swarm intelligence literature, search and optimisation are often used interchangeably. However, in computer science, search is defined in at least three broad (and overlapping) ways: data search, path search and solution search [1]. In the first definition, search refers to finding a (target) model in a search space, and the goal of the algorithm is to find a match, or the closest match to the target in the search space. This is defined as *data search* and is considered a classical meaning of search in computer science; one example of search in this category is Binary search (more such methods are described in [2]). In the second type, the goal is finding a path (*path search*) and the list of the steps leading to a certain solution is what the search algorithm tries to achieve. In this type of search, paths do not exist explicitly but are rather created during the course of the search (a simple example used in AI courses is the "8-puzzle" problem, where a set of tiles are numbered 1-8 and placed in a square leaving one space empty; classical algorithms used for solving problem of this nature are "depth-first search", "breadth-first search", "A*", etc. [3]). In the third definition, *solution search*, the goal is to find a solution in a large problem space of candidate solutions. Similarly to the path search, where paths do not exist explicitly, the search space consists of candidate solutions which are not stored explicitly but are rather created and evaluated during the search process. However, in contrast to the path search, the steps taken to find the solution are not the goal of the algorithm. Population-based swarm intelligence algorithms fit within this category; traditional examples are genetic algorithms, particle swarm optimisation, etc).

In optimisation, which is similar to the third definition of search, the model of the first definition is replaced with an objective or fitness function which is used to evaluate possible solutions. In both search and optimisation, the positions of the optima are not known in advance (even though the optima itself might be known a-priori). The task of the fitness function is to measure the proximity of the candidate solutions to the optima based on the criteria provided by each optimisation problem. The algorithm compares the output of the function to the output of the previously located candidate solutions and, in the case of a minimisation problem, the smaller the output the better the solution. Data search can be seen as a caste of optimisation if the objective function tests the equality of the candidate solution to the model.

Global or continuous optimisation is, however, concerned with locating the optimal, real-valued solution within the entire search space and one of the main difficulties that global optimisers face, is the existence of local optima within the problem space.

According to [4], global optimisation techniques are categorised into four groups:

- Incomplete: This technique uses clever intuitive heuristics for searching without presenting safeguards if the search gets stuck in a local minimum.
- Asymptotically complete: This technique reaches a global minimum with certainty or at least with probability one with the assumption of allowing to run indefinitely long, without providing means to know when a global minimum has been found.
- Complete: This technique reaches a global minimum with certainty, with the assumption of having exact computations and indefinitely long run time, and knows after a finite time that an approximate global minimum has been found (within prescribed tolerances).
- Rigorous: This technique reaches a global minimum with certainty and within given tolerances even in the presence of rounding errors, except in near-degenerate cases where the tolerances may be exceeded.

Along with the algorithm introduced in this paper, most of the population-based algorithms which do not guarantee an optimal global solution (while capable of escaping a local minimum in some cases) are defined as *incomplete* global optimisers. However, solely searching parts of the search space and using the knowledge obtained to update the potential

M. M. al-Rifaie is with the Department of Computing, Goldsmiths, University of London, London SE14 6NW, UK e-mail: m.majid@gold.ac.uk

solutions based on their heuristic rules allows population-based algorithms to be faster than other methods.

In this paper, a brief background on nature inspired and swarm intelligence techniques are presented (Section II) highlighting the role of nature in offering scientists valuable insights; then some information is offered on the huddling behaviour of emperor penguins on fast sea ice as observed by field researchers. Section III introduces Penguins Huddling Optimisation (PHO) as a novel approach for continuous optimisation problems. This section formulates the algorithms and offers some visuals illustrating the behaviour of the algorithm. In Section IV, a series of experiments serves to observe the performance of the newly introduced optimiser over the presented benchmarks, and a control algorithm is designed to validate the behaviour of PHO. The diversity of the algorithm is then monitored, followed by a parameter tuning process. At the end of the section, PHO is compared with a few population based, nature inspired algorithms. section VI concludes the paper by summarising the results and highlighting a few future research topics.

II. Nature and Swarm Intelligence

A multi-agent approach to AI was born in the 90s, when cooperation between agents became essential to have an *emerging* intelligence resulting from the interaction of a group of individuals [5]. It was time for sociologists, biologists and anthropologists to play their rules in helping AI with their models and social views on intelligence [6].

From the design of helicopters inspired by the behaviour of grasshoppers to the erection of sustainable buildings influenced by termites' construction skills, nature has been exhibiting an undeniable role in many of the greatest human inventions. Scientists, in search for methods to solve persistent problems, refer to nature and what it offers from inspiration to ideas and hints.

Swarm intelligence techniques are not exceptions in this regard; many of the most well known evolutionary and swarm intelligence algorithms (e.g. Genetic Algorithm [7], Particle Swarm Optimisation [8], Ant Colony Optimisation [9], etc.) demonstrate clear links to nature. Swarm Intelligence which investigates collective intelligence, aims at modelling intelligence by looking at individuals in a social context and monitoring their interactions with one another as well as their interactions with the environment. Natural examples of swarm intelligence that exhibit these forms of interaction include fish schooling, birds flocking, ant colonies in nesting and foraging, bacterial growth, animal herding, brood sorting by ants, etc. Therefore, swarm intelligence can be characterised as the communications between agents as well as the communication of agents with the environment while expecting an emergent phenomenon (intelligence).

Communication – social interaction or information exchange – observed in social insects and animals is important in swarm intelligence. In real social interactions, not just the syntactical information (i.e. contents) is exchanged between individuals but also semantic rules, tips and beliefs about how to process this information; in typical population based algorithms, however, only the syntactical exchange of information is considered, without necessarily changing the thinking process (e.g. rules and beliefs) of the participants.

In the study of the interaction of social insects, two important elements are the individuals and the environment, which lead to two integration schemes: the first one is the way in which individuals self-interact and the second one is the interaction of the individuals with the environment [10] (stigmergy). Self-interaction between individuals is carried out through recruitment. These recruitment strategies are used to attract other members of the society to gather around one or more desired areas, either for foraging purposes or for moving to a new nest site. In animals like fish or birds, self-interaction results in benefiting from discoveries and previous experience of all other members of the school of flock during search for food [11].

Next, some information is given about Emperor Penguins breeding on the fast sea ice of Antarctica, where the huddling behaviour helps survive the harsh temperature. Based on the information given, the proposed algorithm is formulated in Section III.

A. Emperor Penguins Huddling in Antarctica

The breeding of Emperor Penguins during the Antarctic winter night was among the most unexpected discoveries in the avian world [12]. The large size of the penguin chick at fledging and the time necessary for this development drives the Emperor Penguin to lay its single egg in late May and June and incubate it through the Antarctic night [13]. The rookeries are established on fast sea ice where no formal nest is built and the adult male incubates the egg in his brood pouch supported on his feet. The newly hatched chick is similarly protected until late December when fledging occurs. Depending on the latitude, from March through April the birds return again to the rookeries to form pairs and breed. Because of these reproductive commitments no other bird or mammal species is so obligated to remain in close proximity to the Antarctic continent and the fast sea ice that surrounds its shores.

Therefore, having to stay within this geographical region, huddling allows emperor penguins to conserve energy and survive their long winter fast while enduring the harsh climatic conditions [14].

Having known that the emperor penguin (*Aptenodytes forsteri*) is the sole bird that breeds during the Antarctic winter on fast-ice, far from the open sea where it exclusively feeds, the male emperor penguins have to fast for several months to complete their breeding cycle during the polar winter. Their ambulatory incubation and the absence of territoriality [15] permit them to form huddles throughout their fast.

Huddling, as a key factor enabling emperor penguins to preserve energy throughout their long winter fast, decreases energy expenditure; penguins deprived from huddling are only able to fast for about three months before reaching a critical body mass that would force them to abandon their egg to forage at sea [16].

Contrary to the classic view (e.g. as reflected in [17], [18], [19]), huddling episodes of emperor penguins are recently

shown to be discontinuous and of short, albeit variable, duration [20].

Given the high significance of huddling in emperor penguins' survival, several studies in the 1960s attempted to explain how huddles form and break. It was suggested that huddling forms more frequently during unfavourable meteorological conditions, especially high wind speeds [17]. In other words, huddling occurrence increased with lower ambient temperatures. It is also suggested [18] that huddles break up because of a rise in ambient temperature, and/or birds struggling.

This work attempts to encapsulate and simulate few of the main conceptual behaviour of emperor penguins in Antarctica, namely forming and breaking of the huddles.

III. Penguins Huddling Optimisation

Penguins huddling optimisation (PHO) is an algorithm inspired by the huddling behaviour of emperor penguins breeding in Antarctica. As detailed in section II-A, penguins' goal is maintaining body temperature by staying within a close proximity of the favourable region which, in the proposed algorithm, is considered to be the position of the penguin with the best fitness. Therefore, the chief goal of the members of the population is to secure a place around the "most comfortable penguin", therefore forming a huddle whose centre changes dynamically as better positions are discovered over time. Having considered the formation of the huddle, the breaking or weakening of the huddle (as mentioned in section II-A) is also noted in the proposed algorithm.

The process of huddles' formation and breakage are the two mechanisms coupled in Penguin Huddling Optimisation. The algorithm and the mathematical formulation of the update equations are introduced below.

The position vectors of the population are defined as:

$$\vec{x}_i^t = \left[x_{i1}^t, x_{i2}^t, ..., x_{iD}^t\right], \qquad i = 1, 2, ..., NP \tag{1}$$

where t is the current time step, D is the dimension of the problem space and NP is the number of penguins (population size).

In the first generation, when $t = 0$, the i^{th} vector's j^{th} component is initialised as:

$$x_{id}^0 = x_{min,d} + r\left(x_{max,d} - x_{min,d}\right) \tag{2}$$

where r is a random number drawn from a uniform distribution on the unit interval $U(0,1)$; x_{min} and x_{max} are the lower and upper initialisation bounds of the d^{th} dimension, respectively. Therefore, a population of penguins are randomly initialised with a position for each penguin in the search space.

On each iteration, the components of the position vectors are independently updated, taking into account the component's value and the corresponding value of the penguin with the best fitness:

$$x_{id}^t = x_{gd}^{t-1} + \omega_{id}^t \tag{3}$$

$$\omega_{id}^t = \left|x_{gd}^{t-1} - x_{id}^{t-1}\right| \mathrm{N}(0,1) \tag{4}$$

where x_{gd}^{t-1} is the value of the best penguin's d^{th} component at time step $t-1$; ω_{id}^t is the huddling space, and $\mathrm{N}(0,1)$ is the Gaussian distribution between 0 and 1.

In other words, the update equation can be written as:

$$x = \mathrm{N}\left(\mu, \sigma^2\right) \tag{5}$$

where $\mathrm{N}\left(\mu, \sigma^2\right)$ is the normal distribution with mean μ which is the position of the best penguin, x_{gd}, and the standard deviation σ which is the distance between the current and best penguins, $|x_{gd} - x_{id}|$.

The algorithm is characterised by two principle components: a dynamic rule for updating penguin position (assisted by a social network deploying global neighbourhood that informs this update), and communication of the results of the updates to other penguins.

As stated earlier, the huddle is disturbed for various reasons; one of the positive impacts of such disturbances is the displacement of the disturbed penguin which may lead to discovering a better position. To consider this eventuality, an element of stochasticity is introduced to the update process. Based on this, individual components of penguins' position vectors are reset if the random number, r, generated from a uniform distribution on the unit interval $U(0,1)$ is less than the *disturbance threshold* or *dt*. This guarantees a proportionate disturbance to the otherwise permanent huddles.

Algorithm 1 summarises the PHO algorithm.

Algorithm 1 Penguin Huddling Optimisation

while FE $< 300,000$ **do**
  for $i = 1 \rightarrow NP$ **do**
    $\vec{x}_i$.fitness $\leftarrow f(\vec{x}_i)$
  end for
  $g \leftarrow \{g,\ \forall\ f(\vec{x}_g) = \min\left(f(\vec{x}_1), f(\vec{x}_2), ..., f(\vec{x}_{NP})\right)\}$
  for $i = 1 \rightarrow NP$ **do**
    for $d = 1 \rightarrow \mathrm{D}$ **do**
      $\tau_d \leftarrow x_{gd} + \omega_{id}$ (see Eq. 3)
      if $(r < dt)$ **then**
        $\tau_d \leftarrow x_{min,d} + r\left(x_{max,d} - x_{min,d}\right)$
      end if
    end for
  end for
  $\vec{x}_i \leftarrow \vec{\tau}$
end while

Prior to investigating the performance of the algorithm in the next section, few illustrations are presented with the aim of visually highlighting the behaviour of population when presented with a problem.

In order to illustrate the movements of each individual penguin within PHO the following scenario is considered where the fitness function is defined as the Euclidean distance from the penguin to the known optimal point in the search space. Fig. 1 visualises few iterations through which the population of 10 penguins are initialised in the corner of the search space (see the the grey discs in the first image), attempting to get to the optimal area (centre, which is highlighted with the green disc); the most successful penguin (i.e. closest to the optimum) in each iteration is shown in red. As the figure illustrates, in

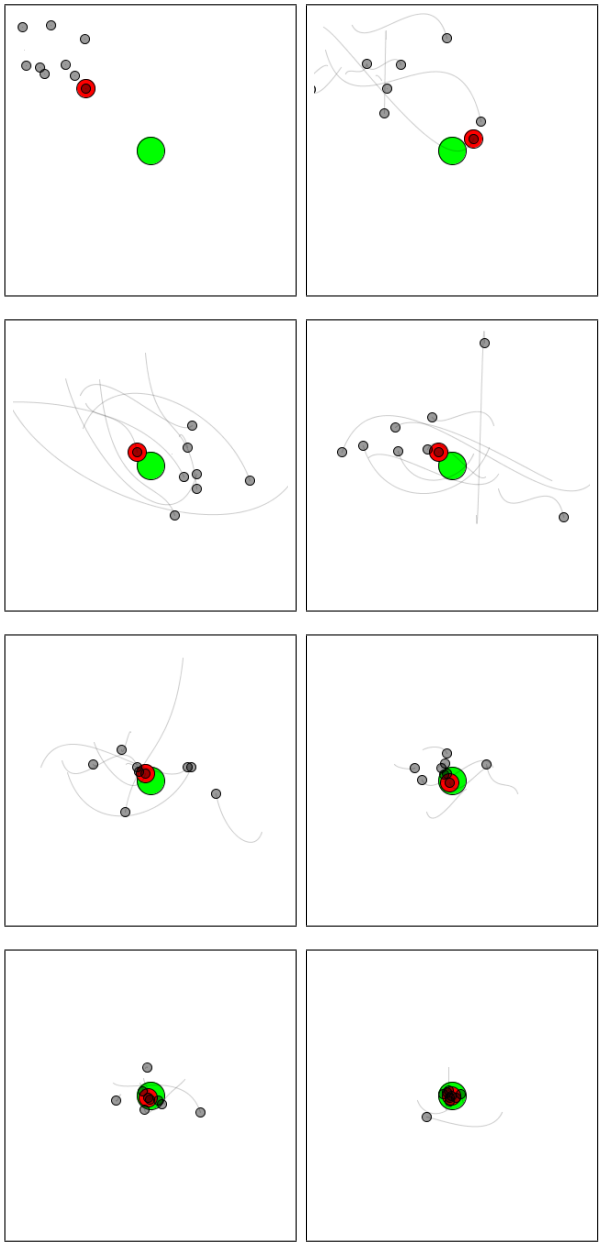

Fig. 1. PHO agents move towards the optimum position highlighted in green. The red disc represents the closest agent to the optimum at each iteration, and the grey discs are the population (see each row left to right).

each iteration penguins move towards the huddling space, and over time, decrease their distance from the optimum position and the huddle intensifies.

Another simulation (see Fig. 2), where the population is initialised in the top-left corner, highlights the areas of the search space covered by the penguins during 100 iterations where the penguins move through the search space in search of the optimal point. The same experiment is repeated in Fig. 3 but with two flipping optima (the probability that each optimum is on during each iteration is $p = 0.5$) showing the behaviour of the algorithm when presented with a problem of this nature.

The next section reports the results of a series of experiments conducted on PHO over a set of benchmark functions.

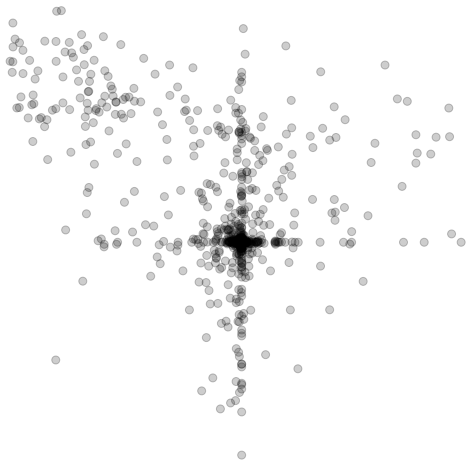

Fig. 2. Traces of PHO agents during 100 iterations of the optimisation process.

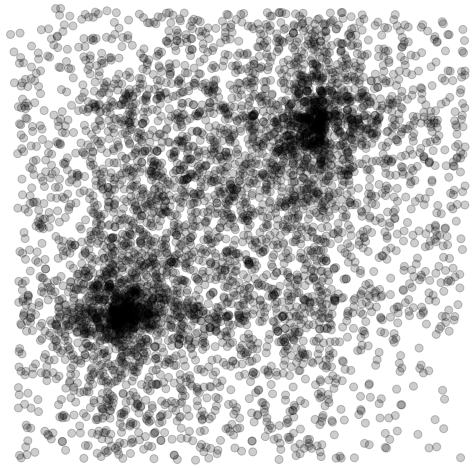

Fig. 3. Traces of PHO agents during 100 iterations dealing with two flipping optima.

IV. Experiments

The chief goal here is to present some experiments investigating the performance of Penguin Huddling Optimisation (PHO). The behaviour of PHO is compared against a control PHO (PHO-c) where no disturbance is induced on the population. Acknowledging the lose of diversity as a common problem with all distribution based evolutionary optimisers (since the dispersion reduces with convergence), the impact of disturbance on preserving the diversity of the population is also studied. Additionally, an optimal value for disturbance threshold, *dt*, is suggested. Afterwards the performance of PHO is compared against few other well-known population-based algorithms, namely Particle Swarm Optimisation (PSO), Differential Evolution (DE) and Genetic Algorithm (GA).

A. Experiment Setup

The benchmarks used in the experiments (see Table I) are divided in two sets, f_{1-14} and g_{1-14}; more details about these functions (e.g. global optima, mathematical formulas, etc.) are reported in [21] and [22]. The first set, f_{1-14}, have been used

by several authors [23], [21], [24] and it contains the three classes of functions recommended by Yao *et al.* [25]: unimodal and high dimensional, multimodal and high dimensional, and low dimensional functions with few local minima. In order not to initialise the penguins on or near a region in the search space known to have the global optimum, *region scaling* technique is used [26], which makes sure penguins are initialised at a corner of the search space where there are no optimal solutions.

The second test set, g_{1-14}, are the first fourteen functions of CEC 2005 test suite [22] and they present more challenging features of the common functions from the aforementioned test set (e.g. shifted by an arbitrary amount within the search space and/or rotated). This set has also been used for many researchers.

The experiments are conducted with the population size of 100 penguins. The termination criterion for the experiments is reaching $300,000$ function evaluations (FEs) which is one of the common termination strategies used. There are 50 Monte Carlo simulations for each trial and the results are averaged over these independent trials. Apart from the disturbance threshold which is set to $dt = 0.001$, there are no adjustable parameters in PHO's update equation.

The aim of the experiments is to demonstrate the qualities of the newly introduced algorithm as a population based continuous optimiser, and to compare its behaviour (in terms of the performance measures and statistical analysis presented in section IV-B) against a control algorithm and some other population based algorithms.

B. Performance measures and statistical analysis

In order to conduct the statistical analysis measuring the presence of any significant difference in the performance of the algorithms, Wilcoxon 1×1 non-parametric statistical test is deployed. The performance measures used in this paper are error, efficiency, reliability and diversity which are described below.

Error is defined by the quality of the best agent in terms of its closeness to the optimum position (if knowledge about the optimum position is known *a priori*, which is the case here). Another measure used is *efficiency* which is the number of function evaluations before reaching a specified error, and *reliability* is the percentage of trials where a specified error is reached. These performance measures are defined as below:

$$\text{ERROR} = |f(\vec{x}_g) - f(\vec{x}_o)| \quad (6)$$

$$\text{EFFICIENCY} = \frac{1}{n}\sum_{i=1}^{n} \text{FEs} \quad (7)$$

$$\text{RELIABILITY} = \frac{n'}{n} \times 100 \quad (8)$$

where $\vec{x}_g$ is the best position found and $\vec{x}_o$ is the position of the known optimum solution; n is the number of trials in the experiment and n' is the number of successful trials, FEs is the number of function evaluations before reaching the specified error, which in these experiments, set to 10^{-8}.

In this work, *diversity*, which is the degree of convergence and divergence, is defined as a measure to study the population's behaviour with regard to exploration and exploitation. There are various approaches to measure diversity. The average distance around the population centre is shown [27] to be a robust measure in the presence of outliers and is defined as:

$$\text{DIVERSITY} = \frac{1}{NP}\sum_{i=1}^{NP}\sqrt{\sum_{j=1}^{D}\left(x_i^j - \bar{x}^j\right)^2} \quad (9)$$

$$\bar{x}^j = \frac{1}{NP}\sum_{i=1}^{NP} x_i^j \quad (10)$$

where NP is the number of penguins in the population, D is the dimensionality of the problem, x_i^j is the value of dimension j of agent i, and $\bar{x}^j$ is the average value of dimension j over all agents.

C. Performance of Penguin Huddling Optimisation

The error, efficiency and reliability results of PHO performance over the benchmarks are reported in Table II. The first five columns detail the error-related figures and the last column highlights the median efficiency along with the reliability (shown between brackets) of the algorithm in finding the optima.

The algorithm exhibits a promising performance in optimising the presented problem set where half the benchmarks ($f_{1-2,4-11}$ and $g_{1-2,7,9}$) are optimised with the specified accuracy.

The figures in the table are expanded in the following categories:

1) Unimodal, high dimensional ($f_{1,2}, g_{1-5}$): The algorithm optimises 57% of the benchmarks in this category; while both functions in the first set are optimised ($f_{1,2}$), only 40% of the benchmarks in the second and more challenging set are optimised to the specified accuracy. All optimised benchmarks achieve 100% success.

2) Low dimensional and few local minima (f_{10-14}): In this category, 40% of the benchmarks are optimised, one with 100% (f_{11}). However, none of the Shekel functions (f_{12-14}) are optimised; Shekel is known to be a challenging function to optimise due to the presence of several broad sub-optimal minima; also the proximity of a small number of optima to the Shekel parameter $\vec{a}_i$ is another reason for the difficulty of optimising these set of functions.

3) Multimodal, high dimensional (f_{3-9}, g_{6-14}): The optimiser is able to optimise 57% of the benchmarks in this category ($f_{3-6,8,9}$ and $g_{7,9}$), 88% of which achieve 100% success rate (all except f_6 with 18% success rate). The optimiser exhibit a promising performance when dealing with the difficult Rosenbrock function, optimising the entire trials in the first set, f_3, and reaches the error of 10^{-3} in at least one run in the second set, g_6. The algorithm performs exceptionally well in optimising the infamous Rastrigin functions, both common and shifted mode (i.e. f_5 and g_9); however it does show weakness in the more challenging g_{10} rotated version.

TABLE I
BENCHMARK FUNCTIONS

Fn	Name	Class	Dimension	Feasible Bounds
f_1	Sphere/Parabola	Unimodal	30	$(-100, 100)^D$
f_2	Schwefel 1.2	Unimodal	30	$(-100, 100)^D$
f_3	Generalized Rosenbrock	Multimodal	30	$(-30, 30)^D$
f_4	Generalized Schwefel 2.6	Multimodal	30	$(-500, 500)^D$
f_5	Generalized Rastrigin	Multimodal	30	$(-5.12, 5.12)^D$
f_6	Ackley	Multimodal	30	$(-32, 32)^D$
f_7	Generalized Griewank	Multimodal	30	$(-600, 600)^D$
f_8	Penalized Function P8	Multimodal	30	$(-50, 50)^D$
f_9	Penalized Function P16	Multimodal	30	$(-50, 50)^D$
f_{10}	Six-hump Camel-back	Low Dimensioal	2	$(-5, 5)^D$
f_{11}	Goldstein-Price	Low Dimensioal	2	$(-2, 2)^D$
f_{12}	Shekel 5	Low Dimensioal	4	$(0, 10)^D$
f_{13}	Shekel 7	Low Dimensioal	4	$(0, 10)^D$
f_{14}	Shekel 10	Low Dimensioal	4	$(0, 10)^D$
g_1	Shifted Sphere	Unimodal	30	$(-100, 100)^D$
g_2	Shifted Schwefel 1.2	Unimodal	30	$(-100, 100)^D$
g_3	Shifted Rotated High Conditioned Elliptic	Unimodal	30	$(-100, 100)^D$
g_4	Shifted Schwefel 1.2 with Noise in Fitness	Unimodal	30	$(-100, 100)^D$
g_5	Schwefel 2.6 with Global Optimum on Bounds	Unimodal	30	$(-100, 100)^D$
g_6	Shifted Rosenbrock	Multimodal	30	$(-100, 100)^D$
g_7	Shifted Rotated Griewank without Bounds	Multimodal	30	$(-600, 600)^D$
g_8	Shifted Rotated Ackley with Global Optimum on Bounds	Multimodal	30	$(-32, 32)^D$
g_9	Shifted Rastrigin	Multimodal	30	$(-5, 5)^D$
g_{10}	Shifted Rotated Rastrigin	Multimodal	30	$(-5, 5)^D$
g_{11}	Shifted Rotated Weierstrass	Multimodal	30	$(-0.5, 0.5)^D$
g_{12}	Schwefel Problem 2.13	Multimodal	30	$(-\pi, \pi)^D$
g_{13}	Expanded Extended Griewank plus Rosenbrock	Expanded	30	$(-5, 5)^D$
g_{14}	Shifted Rotated Expanded Scaffer	Expanded	30	$(-100, 100)^D$

The success of the optimiser in optimising the notorious Rastrigin function in its common and shifted modes will be discussed in the context of PHO's dimension-to-dimension disturbance mechanism induced by the algorithm.

To provide a better understanding of the behaviour of the algorithm, in the next section, the disturbance is discarded and the diversity of the algorithm is studied.

D. Diversity in PHO

Many swarm intelligence techniques start with exploration and, over time (i.e. function evaluations or iterations), lean towards exploitation. Preserving the balance between exploration and exploitation phases has proved to be a hard problem. The lack of such balance leads to weaker diversity when encountering a local minimum and thus a common problem exhibits itself in many optimisers, i.e. the pre-mature convergence of the population on a local minimum.

Similar to other swarm intelligence and evolutionary algorithms, PHO commences with exploration and over time, through its mechanism (i.e. gradual decrease in the distance between the members of the population and the best found position), moves towards exploitation. However, having implemented the disturbance threshold, a dose of diversity (i.e. dt) is introduced in the population throughout the optimisation process, aiming to enhance the diversity of the algorithm.

Figure 4 illustrates the convergence of the population towards the optima and their diversities in three random trials over three benchmarks (i.e. $g_{1,7,9}$ chosen from the second set) as examples from unimodal and multimodal functions.

The difference between the error and the diversity values demonstrates the algorithm's ability, whose fitness reaches as low as 10^{-13} in g_1 and g_9; thus emphasising the impact of disturbance in injecting diversity.

To investigate the impact of disturbance in increasing diversity, a control algorithm is proposed (PHO-c) where there is no disturbance ($dt = 0$) during the position update process.

The graphs in Fig. 5 illustrate the diversity of PHO-c populations in randomly chosen trials over three sample benchmarks (again $g_{1,7,9}$). The graphs illustrate that the diversity of the population in PHO-c is less than PHO, thus emphasising the impact of disturbance in injecting diversity, which in turn facilitates the escape from local minima (e.g. as demonstrated in case of the highly multimodal Rastrigin functions f_5, g_9).

TABLE II
PHO – PENGUIN HUDDLING OPTIMISATION

	Min.	Max.	Median	Mean	StdDev	Eff. (Rel.)
f_1	4.19E-47	7.85E-42	1.74E-44	5.06E-43	1.68E-42	47600 (100%)
f_2	7.24E-15	2.56E-12	2.08E-13	3.54E-13	4.94E-13	204350 (100%)
f_3	5.70E-05	4.24E+02	3.15E+00	4.23E+01	9.69E+01	9300 (100%)
f_4	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	73150 (100%)
f_5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	104250 (100%)
f_6	2.84E-14	6.39E-14	3.91E-14	4.24E-14	8.60E-15	48000 (18%)
f_7	0.00E+00	1.15E-01	2.46E-02	3.15E-02	2.99E-02	∞ (0%)
f_8	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	55200 (100%)
f_9	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	59650 (100%)
f_{10}	0.00E+00	2.22E-16	1.11E-16	1.11E-16	1.12E-16	1700 (100%)
f_{11}	0.00E+00	8.10E+01	8.10E+01	4.54E+01	4.06E+01	2200 (44%)
f_{12}	5.05E+00	5.05E+00	5.05E+00	5.05E+00	0.00E+00	∞ (0%)
f_{13}	5.27E+00	5.27E+00	5.27E+00	5.27E+00	0.00E+00	∞ (0%)
f_{14}	5.36E+00	5.36E+00	5.36E+00	5.36E+00	0.00E+00	∞ (0%)
g_1	1.14E-13	2.84E-13	1.71E-13	1.97E-13	4.90E-14	47200 (100%)
g_2	1.82E-12	7.84E-12	4.77E-12	4.69E-12	1.49E-12	194000 (100%)
g_3	2.02E+05	1.95E+06	6.50E+05	6.57E+05	3.06E+05	∞ (0%)
g_4	5.58E-04	5.43E+00	3.06E-02	2.00E-01	7.75E-01	∞ (0%)
g_5	6.51E+03	1.80E+04	1.20E+04	1.23E+04	2.73E+03	∞ (0%)
g_6	1.01E-03	1.30E+04	5.44E+00	7.41E+02	2.29E+03	∞ (0%)
g_7	1.53E-12	8.84E-02	1.85E-02	2.27E-02	1.90E-02	150600 (12%)
g_8	2.00E+01	2.01E+01	2.00E+01	2.00E+01	2.47E-02	∞ (0%)
g_9	5.68E-14	3.41E-13	1.71E-13	2.00E-13	5.77E-14	72400 (100%)
g_{10}	1.21E+02	4.29E+02	2.82E+02	2.76E+02	7.95E+01	∞ (0%)
g_{11}	2.28E+01	3.38E+01	2.74E+01	2.76E+01	2.76E+00	∞ (0%)
g_{12}	1.06E+00	2.03E+04	1.95E+03	3.34E+03	3.88E+03	∞ (0%)
g_{13}	5.42E-01	2.03E+00	1.07E+00	1.18E+00	3.53E-01	∞ (0%)
g_{14}	1.28E+01	1.40E+01	1.35E+01	1.34E+01	3.26E-01	∞ (0%)

It is noteworthy to emphasis that, over time, the diversity of the population in PHO-c shrinks, therefore if trapped in local minima, the likelihood of escaping the minima is lower. As shown in Fig. 5, g_9 is a clear example of premature convergence on a local minimum around which the population's diversity shrinks.

In order to compare the performance of PHO and its control counterpart, Table III presents the result of optimising the benchmarks using PHO-c. Additionally, a statistical analysis is conducted and the output is reported in Table IV where the performance is compared using the three aforementioned measures of error, efficiency and reliability (see Section IV-B for the definitions of the measures). The results show that in 88% of cases (where there is a significant difference between the two algorithms), PHO is performing significantly better than its control counterpart (PHO-c) which is stripped from the diversity inducing disturbance. Furthermore, in all multimodal functions (f_{3-9} and g_{6-12}), whenever there is a statistically significant difference between PHO and PHO-c, the former demonstrates significant outperformance over the later.

Following on the results from measuring error, Table IV also shows that in terms of efficiency and reliability measures, PHO is 73% more efficient that its control counterpart, and 92% more reliable.

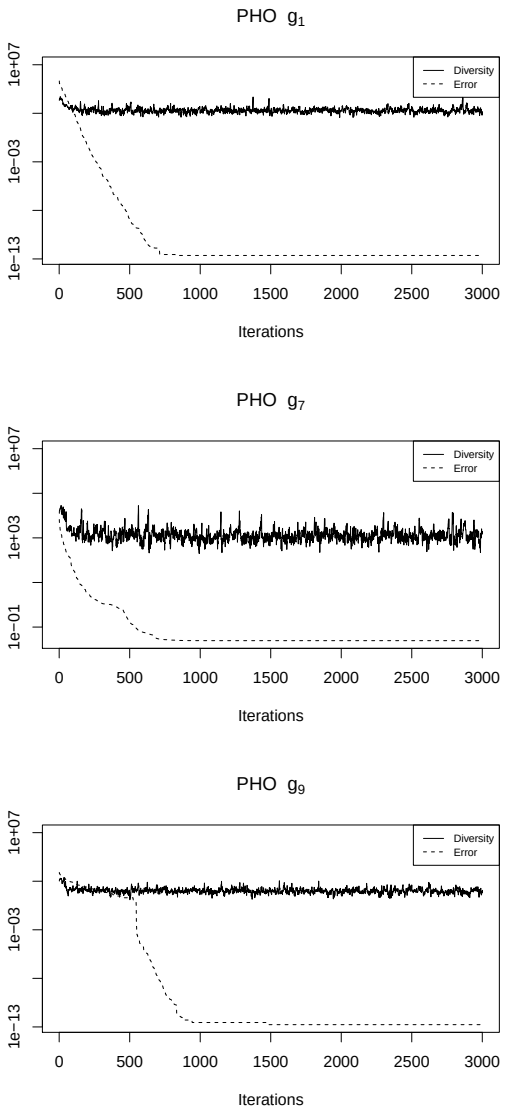

Fig. 4. PHO: diversity and error in $g_{1,7,9}$.

E. Fine Tuning Disturbance Threshold

The role of disturbance in increasing the diversity of PHO population is discussed earlier (Section IV-D). Also, the importance of disturbance is investigated on the optimisation capability of PHO by introducing a control algorithm which lacks the disturbance mechanism and the results demonstrate the positive impact of this mechanism.

The aim of this section is to recommend a value for the disturbance threshold, dt. The range of disturbance probabilities used in this experiment is between 1 to 10^{-9} and the values were chosen according to:

$$dt_n = 10^{-n}, \quad 0 \leq n \leq 9$$

Fig. 6 illustrates the performance of PHO using these dt probabilities. Both set of benchmarks (i.e. f_{1-14} and g_{1-14})

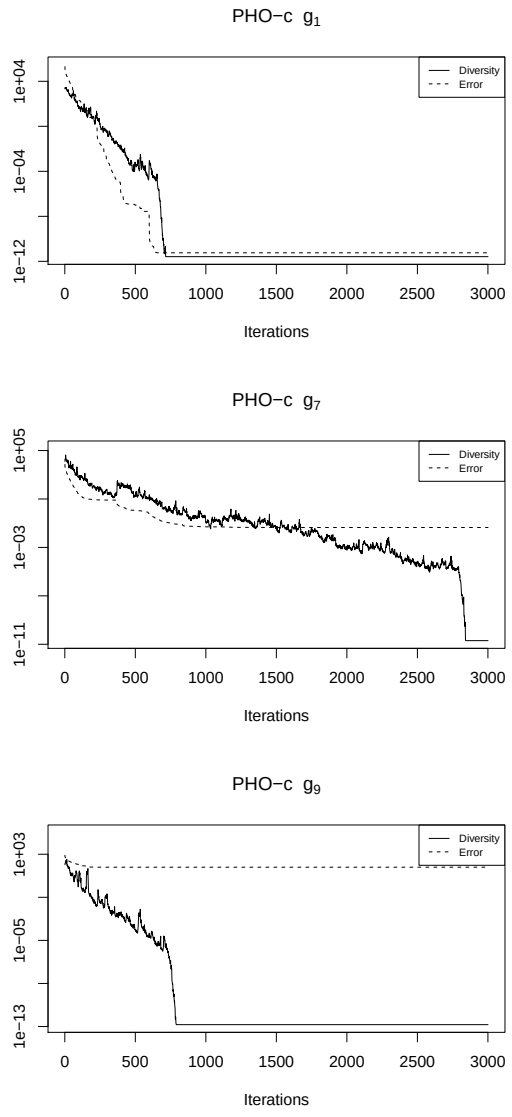

Fig. 5. PHO-c: diversity and error in $g_{1,7,9}$.

have been used to find a suitable value for the disturbance threshold. As the heat map highlights, the optimal range is $10^{-2} < dt < 10^{-4}$ and the overall recommended value of $dt = 10^{-3}$ is suggested as a good compromise.

F. Comparing PHO with other Population-Based Optimisers

Having presented the performance of the PHO algorithm (taking into account the three performance measures of error, efficiency and reliability, as well as the diversity of its population and the impact of disturbance on its behaviour), this section focuses on contrasting the introduced algorithm with few well-known optimisation algorithms. The three population algorithms deployed for this comparison are Differential Evolution (DE), Particle Swarm Optimisation (PSO) and Genetic Algorithm (GA). These algorithms are briefly described in Appendices A, B and C. Generic versions of each algorithm are used against the generic version of Penguin Huddling Optimisation. respectively.

TABLE III
PHO-C – CONTROL PHO ALGORITHM

	Min.	**Max.**	**Median**	**Mean**	**StdDev**	**Eff. (Rel.)**
f_1	7.07E-74	3.97E-43	5.09E-68	7.94E-45	5.62E-44	55350 (100%)
f_2	1.58E-13	2.10E+02	2.51E-10	4.25E+00	2.96E+01	246050 (76%)
f_3	1.12E-04	7.53E+02	3.99E+00	4.67E+01	1.42E+02	∞ (0%)
f_4	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	8750 (100%)
f_5	7.16E+01	2.77E+02	1.75E+02	1.78E+02	4.02E+01	∞ (0%)
f_6	1.95E+01	1.99E+01	1.98E+01	1.98E+01	9.34E-02	∞ (0%)
f_7	8.88E-16	1.01E+01	7.09E-02	6.53E-01	1.70E+00	56050 (12%)
f_8	1.12E-28	1.67E+00	9.99E-20	1.50E-01	3.89E-01	84700 (66%)
f_9	8.76E-27	1.86E+04	1.08E-15	3.73E+02	2.63E+03	96800 (58%)
f_{10}	0.00E+00	2.22E-16	2.22E-16	1.64E-16	9.84E-17	1600 (100%)
f_{11}	0.00E+00	8.10E+01	8.10E+01	5.35E+01	3.88E+01	1900 (34%)
f_{12}	5.05E+00	5.05E+00	5.05E+00	5.05E+00	0.00E+00	∞ (0%)
f_{13}	5.27E+00	5.27E+00	5.27E+00	5.27E+00	0.00E+00	∞ (0%)
f_{14}	5.36E+00	5.36E+00	5.36E+00	5.36E+00	0.00E+00	∞ (0%)
g_1	2.84E-13	6.91E-03	3.13E-12	1.40E-04	9.77E-04	53050 (92%)
g_2	3.69E-12	8.07E+03	6.88E-07	1.71E+02	1.14E+03	232850 (28%)
g_3	2.68E+05	3.56E+06	1.15E+06	1.35E+06	7.28E+05	∞ (0%)
g_4	1.61E-02	3.25E+01	1.05E+00	3.26E+00	5.62E+00	∞ (0%)
g_5	5.99E+03	1.86E+04	1.11E+04	1.08E+04	2.65E+03	∞ (0%)
g_6	7.42E-04	1.99E+04	5.42E+00	7.49E+02	3.11E+03	∞ (0%)
g_7	3.47E-12	1.52E-01	2.95E-02	3.41E-02	2.78E-02	207300 (4%)
g_8	2.00E+01	2.01E+01	2.00E+01	2.00E+01	2.74E-02	∞ (0%)
g_9	4.18E+01	1.72E+02	9.80E+01	1.01E+02	2.93E+01	∞ (0%)
g_{10}	1.13E+02	4.81E+02	2.99E+02	3.02E+02	7.10E+01	∞ (0%)
g_{11}	2.31E+01	3.75E+01	3.06E+01	3.06E+01	3.67E+00	∞ (0%)
g_{12}	1.76E+00	1.78E+04	9.83E+02	2.91E+03	3.67E+03	∞ (0%)
g_{13}	4.68E+00	1.93E+01	1.02E+01	1.09E+01	3.70E+00	∞ (0%)
g_{14}	1.27E+01	1.39E+01	1.34E+01	1.34E+01	2.62E-01	∞ (0%)

In this comparison, only the second and the more challenging set of benchmarks, g_{1-14} are used. Tables V, VI, VII present the optimising results of the aforementioned algorithms, and as shown, the algorithms have optimised some of the benchmark to the specified accuracy, 10^{-8}.

Table VIII shows the result of the statistical analysis comparing PHO with the other three optimisers. Based on this comparison, whenever there is a significant difference between the performance of PHO and the other algorithms, PHO significantly outperforms DE, PSO and GA in 69%, 62% and 86% of the cases, respectively.

Table IX summaries the efficiency results of the three optimiser with that of PHO; note that only the efficiency of functions reaching the specified error is given. As shown in the table, PHO, in the majority of cases, outperforms the other algorithms. In other words, although, when compared with DE, PHO only outperforms marginally (60%), it outperforms both PSO and GA in all cases (100%).

The reliability comparison of PHO with the other optimisers is given in Table X. PHO is shown to be the most reliable algorithm in this comparison. While PHO outperforms DE in 75% of cases, it show 100% outperformance when compared with PSO and GA.

TABLE IV
COMPARING PHO AND PHO-C PERFORMANCE

Based on Wilcoxon 1×1 Non-Parametric Statistical Test, if the *error* difference between each pair of algorithms is significant at the 5% level, the pairs are marked. X–o shows PHO is significantly outperforming its counterpart algorithm; and o–X shows that the algorithm compared to PHO is significantly better than PHO. In terms of the *efficiency* and *reliability* measures, 1 – 0 (or 0 – 1) indicates that the left (or right) algorithm is more efficient/reliable. The figures, n – m, in the last row present a count of the number of X's or 1's in the respective columns.

	PHO – PHO-c		
	Error	**Efficiency**	**Reliability**
f_1	o – X	1 – 0	–
f_2	X – o	1 – 0	1 – 0
f_3	–	1 – 0	1 – 0
f_4	–	0 – 1	–
f_5	X – o	1 – 0	1 – 0
f_6	X – o	1 – 0	1 – 0
f_7	X – o	0 – 1	0 – 1
f_8	X – o	1 – 0	1 – 0
f_9	X – o	1 – 0	1 – 0
f_{10}	X – o	0 – 1	–
f_{11}	–	0 – 1	1 – 0
f_{12}	–	–	–
f_{13}	–	–	–
f_{14}	–	–	–
g_1	X – o	1 – 0	1 – 0
g_2	X – o	1 – 0	1 – 0
g_3	X – o	–	–
g_4	X – o	–	–
g_5	o – X	–	–
g_6	–	–	–
g_7	–	1 – 0	1 – 0
g_8	–		–
g_9	X – o	1 – 0	1 – 0
g_{10}	–	–	–
g_{11}	X – o	–	–
g_{12}	–	–	–
g_{13}	X – o	–	–
g_{14}	–	–	–
	14 – 2	11 – 4	11 – 1

In order to compare the diversity of the PHO algorithm with the other three optimisers, three benchmarks were chosen from unimodal and multimodal categories ($g_{1,7,9}$). The result of this comparison is illustrated in Fig. 7. It is shown that DE has the least diversity in both uni- and multimodal functions. On the other hand, the diversity of the population in PSO decreases as the population converges towards an optimum (see g_1); however, when convergence does not occur (e.g. in $g_{7,9}$), PSO maintain its high diversity throughout the optimisation process. GA shows a similar pattern to that of PSO in multimodal functions, which is the gradual diversity decrease over time; however it maintains a higher diversity for the unimodal function than PSO (perhaps attributable to the difference in the fitness of the best positions found in both algorithms). In

TABLE V
DE – DIFFERENTIAL EVOLUTION

	Min.	**Max.**	**Median**	**Mean**	**StdDev**	**Eff. (Rel.)**
g_1	5.68E-14	2.84E-13	1.14E-13	1.38E-13	4.46E-14	21500 (100%)
g_2	1.11E-08	8.36E-07	8.40E-08	1.72E-07	2.22E-07	∞ (0%)
g_3	1.88E+06	2.43E+07	9.40E+06	9.65E+06	5.37E+06	∞ (0%)
g_4	3.06E-03	7.49E+00	8.28E-02	4.92E-01	1.40E+00	∞ (0%)
g_5	6.94E+02	4.69E+03	2.32E+03	2.34E+03	8.01E+02	∞ (0%)
g_6	2.27E-13	9.02E+00	2.58E+00	2.30E+00	2.21E+00	265800 (12%)
g_7	9.38E-03	1.71E+00	3.71E-01	5.39E-01	4.43E-01	∞ (0%)
g_8	2.08E+01	2.11E+01	2.09E+01	2.09E+01	5.86E-02	∞ (0%)
g_9	9.95E+00	6.17E+01	3.43E+01	3.47E+01	1.13E+01	∞ (0%)
g_{10}	3.08E+01	2.34E+02	1.64E+02	1.47E+02	5.41E+01	∞ (0%)
g_{11}	6.97E+00	4.14E+01	3.92E+01	3.65E+01	7.82E+00	∞ (0%)
g_{12}	4.15E+05	7.10E+05	5.97E+05	5.85E+05	7.87E+04	∞ (0%)
g_{13}	1.54E+00	1.13E+01	5.67E+00	5.70E+00	3.12E+00	∞ (0%)
g_{14}	1.29E+01	1.39E+01	1.35E+01	1.34E+01	2.32E-01	∞ (0%)

TABLE VI
PSO – PARTICLE SWARM OPTIMISATION

	Min.	**Max.**	**Median**	**Mean**	**StdDev**	**Eff. (Rel.)**
g_1	0.00E+00	5.68E-14	5.68E-14	5.23E-14	1.56E-14	656236 (100%)
g_2	1.18E-02	6.61E-01	9.16E-02	1.33E-01	1.15E-01	∞ (0%)
g_3	5.51E+05	2.55E+06	1.55E+06	1.52E+06	4.44E+05	∞ (0%)
g_4	2.09E+03	1.41E+04	7.49E+03	7.89E+03	3.04E+03	∞ (0%)
g_5	2.78E+03	7.52E+03	4.90E+03	5.04E+03	9.12E+02	∞ (0%)
g_6	1.05E-02	1.55E+02	1.17E+01	2.16E+01	3.22E+01	∞ (0%)
g_7	7.20E-11	3.69E-02	9.86E-03	1.04E-02	9.06E-03	279653 (10%)
g_8	2.07E+01	2.10E+01	2.09E+01	2.09E+01	6.78E-02	∞ (0%)
g_9	5.57E+01	1.44E+02	9.36E+01	9.59E+01	2.21E+01	∞ (0%)
g_{10}	4.78E+01	1.73E+02	1.16E+02	1.14E+02	2.77E+01	∞ (0%)
g_{11}	2.65E+01	3.41E+01	2.99E+01	3.00E+01	2.05E+00	∞ (0%)
g_{12}	9.55E+02	3.78E+04	7.38E+03	9.51E+03	6.98E+03	∞ (0%)
g_{13}	2.84E+00	9.75E+00	5.35E+00	5.35E+00	1.25E+00	∞ (0%)
g_{14}	1.17E+01	1.31E+01	1.25E+01	1.25E+01	2.64E-01	∞ (0%)

TABLE VII
GA – GENETIC ALGORITHM

	Min.	**Max.**	**Median**	**Mean**	**StdDev**	**Eff. (Rel.)**
g_1	2.42E-05	7.97E-05	5.22E-05	5.04E-05	1.51E-05	∞ (0%)
g_2	7.47E+03	1.76E+04	1.17E+04	1.21E+04	2.66E+03	∞ (0%)
g_3	3.53E+06	3.10E+07	1.32E+07	1.47E+07	5.73E+06	∞ (0%)
g_4	3.75E+04	8.35E+04	4.97E+04	5.13E+04	9.60E+03	∞ (0%)
g_5	1.62E+04	2.54E+04	2.08E+04	2.09E+04	2.39E+03	∞ (0%)
g_6	7.49E+01	3.09E+03	7.90E+02	7.23E+02	5.87E+02	∞ (0%)
g_7	4.13E+03	7.80E+03	5.41E+03	5.48E+03	7.45E+02	∞ (0%)
g_8	2.01E+01	2.06E+01	2.04E+01	2.04E+01	8.74E-02	∞ (0%)
g_9	1.10E+01	4.39E+01	2.06E+01	2.20E+01	7.53E+00	∞ (0%)
g_{10}	6.07E+01	2.30E+02	1.37E+02	1.39E+02	3.49E+01	∞ (0%)
g_{11}	4.44E+00	1.69E+01	1.18E+01	1.17E+01	2.76E+00	∞ (0%)
g_{12}	5.80E+02	3.08E+04	5.74E+03	8.14E+03	7.04E+03	∞ (0%)
g_{13}	1.14E+00	4.17E+00	2.68E+00	2.70E+00	5.54E-01	∞ (0%)
g_{14}	1.31E+01	1.43E+01	1.39E+01	1.39E+01	2.72E-01	∞ (0%)

TABLE VIII
COMPARING ERROR IN PHO WITH DE, PSO AND GA

Based on Wilcoxon 1×1 Non-Parametric Statistical Test, if the difference between each pair of algorithms is significant at the 5% level, the pairs are marked. X–o shows that the left algorithm is significantly better than the right one; and o–X shows that the right one is significantly better than the left. n – m in the row labeled Σ is a count of the number of X's in the columns above.

	PHO - DE	**PHO - PSO**	**PHO - GA**
g_1	o – X	o – X	X – o
g_2	X – o	X – o	X – o
g_3	X – o	X – o	X – o
g_4	X – o	X – o	X – o
g_5	o – X	o – X	X – o
g_6	o – X	–	X – o
g_7	X – o	o – X	X – o
g_8	X – o	X – o	X – o
g_9	X – o	X – o	X – o
g_{10}	o – X	o – X	o – X
g_{11}	X – o	X – o	o – X
g_{12}	X – o	X – o	X – o
g_{13}	X – o	X – o	X – o
g_{14}	–	o – X	X – o
$\sum$	9 – 4	8 – 5	12 – 2

TABLE IX
COMPARING EFFICIENCY IN PHO WITH DE, PSO AND GA

In this table, 1 – 0 (0 – 1) indicates that the left (right) algorithm is more efficient. The figures, n – m, in the last row present a count of the number of 1's in the respective columns. Note that non-applicable functions have been removed from the table.

	PHO - DE	**PHO - PSO**	**PHO - GA**
g_1	0 – 1	1 – 0	1 – 0
g_2	1 – 0	1 – 0	1 – 0
g_6	0 – 1	–	–
g_7	1 – 0	1 – 0	1 – 0
g_9	1 – 0	1 – 0	1 – 0
$\sum$	3 – 2	4 – 0	4 – 0

TABLE X
COMPARING RELIABILITY IN PHO WITH DE, PSO AND GA

In this table, 1 – 0 (0 – 1) indicates that the left (right) algorithm is more reliable. The figures, n – m, in the last row present a count of the number of 1's in the respective columns. Note that non-applicable functions have been removed from the table.

	PHO - DE	**PHO - PSO**	**PHO - GA**
g_1	–	–	1 – 0
g_2	1 – 0	1 – 0	1 – 0
g_6	0 – 1	–	–
g_7	1 – 0	1 – 0	1 – 0
g_9	1 – 0	1 – 0	1 – 0
$\sum$	3 – 1	3 – 0	4 – 0

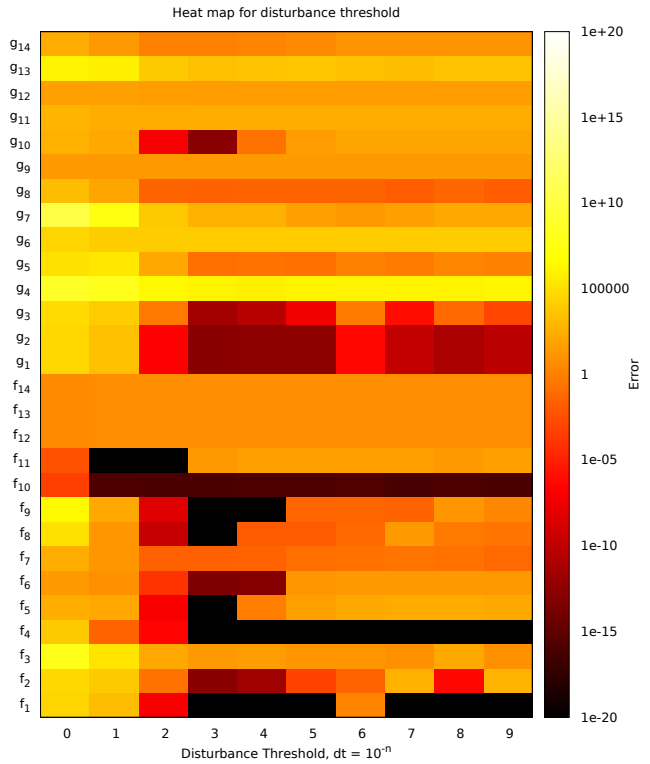

Fig. 6. Fine tuning disturbance threshold

terms of PHO, diversity is less convergence-dependent and more stable across all modalities.

V. PHO AND OTHER POPULATION BASED ALGORITHMS

Penguin Huddling Optimisation fits in the category of population based optimisers. As stated earlier, among the well-known algorithms in this category are genetic algorithms, particle swarm optimisation, differential evolution algorithms, etc. Some of the similarities between these methods are listed below:

- initialisation of the population
- using fitness function as a way to evaluate the quality of each member of the population
- deploying evolutionary operations (e.g. mutation, crossover and selection) in each generation
- producing the offspring population from the parent population (or calculating the updated positions)

The main difference between most population-based algorithms is the update strategy they adopt through their update equation(s). Penguin Huddling Optimisation, in addition the simplicity of its implementation, and its different update formula, emphasises on preserving diversity through the disturbance threshold, dt, it deploys in the algorithm. The combination of these differences gives rise to the differing performance of the introduced algorithm.

VI. CONCLUSION

Penguin Huddling Optimisation (PHO) is a multi-agent population based stochastic optimiser which is proposed to search for an optimum value in a continuous solution space;

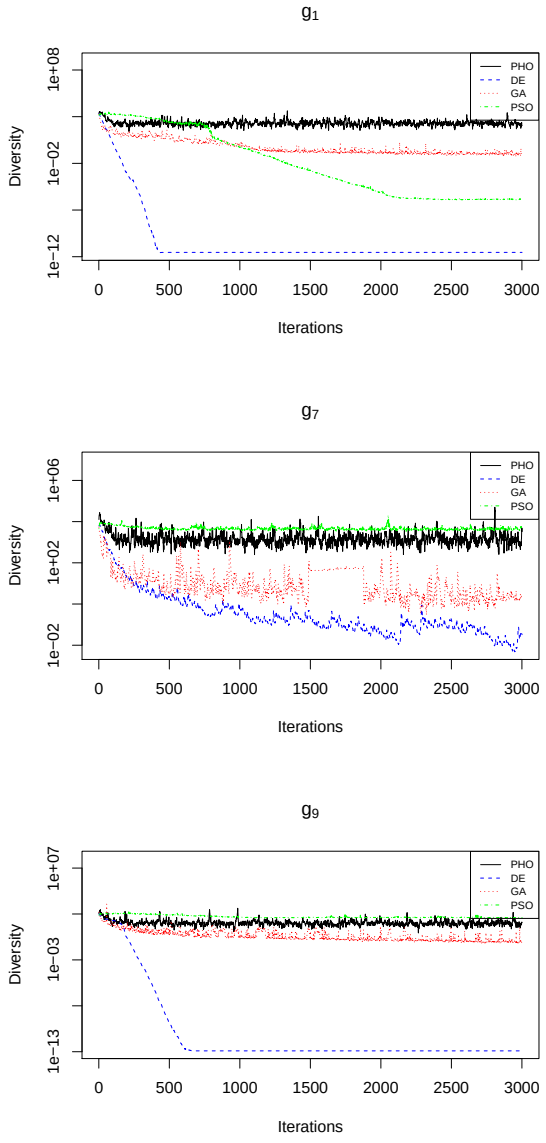

Fig. 7. Diversity of the population in PHO, DE, PSO and GA over three random trials in $g_{1,7 \text{ and } 9}$.

despite its simplicity, the algorithm's promising performance over an exemplar set of benchmark functions is demonstrated.

As part of the study and in an experiment, a control algorithm is proposed to investigate the behaviour of the optimiser. In this experiment, the algorithm's internal disturbance mechanism shows the ability to maintain a stable and convergence-independent diversity throughout the optimisation process. Additionally, a suitable value is recommended for the *disturbance threshold* which is the only parameter in the update equations to be optimised. This parameter controls the level of diversity by inducing a component-wise disturbance (or restart) in the huddling process, aiming to preserve a balance between exploration and exploitation.

In addition to diversity, PHO's performance has been investigated and compared against three other performance measures (i.e. error, efficiency and reliability). Using these measures, it is established that the newly introduced algorithm, outperforms few generic population based algorithms (i.e. differential evolution, particle swarm optimisation and genetic algorithm) in all of the aforementioned measures over the presented benchmarks. In other words, PHO is more efficient and reliable in 84.62% and 90.91% of the cases, respectively; furthermore, when there exists a statistically significant difference, PHO converges to better solutions in 72.5% of problem set.

A. Future Research

Much further research remains to be conducted on this simple new concept and paradigm.

Among the possible future research are investigating the algorithm for an adaptive disturbance threshold and an adaptive adjustment of the huddling space, ω. Optimising the swarm size and analysing its impact on the convergence behaviour of the algorithm is another topics to be explored. Additionally, optimising multi-objective real world problems is yet to researched; this is a continuation of earlier works on the deployment of population-based algorithms for detecting metastasis in bone scans and calcifications in mammographs [28], [29]. Given a distributed, autonomous agent system, the problem of obtaining a global knowledge is challenging and thus deploying a the local neighbourhood network and analysing the performance of the algorithm would be a topic for future research. At last, but not least, given the demonstrated stable and convergence-independent diversity of Penguin Huddling Optimisation (in the context of the presented benchmarks), research and experiments on the capability of the introduced algorithm to dynamic optimisation problems is yet to be conducted.

APPENDIX A
PARTICLE SWARM OPTIMISATION

Particle swarm optimisation (PSO) is population based optimization technique developed in 1995 by Kennedy and Eberhart [8]. It came about as a result of an attempt to graphically simulate the choreography of fish schooling or birds flying (e.g. pigeons, starlings, and shorebirds) in coordinated flocks that show strong synchronisation in turning, initiation of flights and landing, despite the fact that experimental researches to find leaders in such flocks failed [30].

A swarm in PSO algorithm comprises of a number of particles and each particle represents a point in a multi-dimensional problem space. Particles in the swarm explore the problem space searching for the optimal position, which is defined by a fitness function. The position of each particle, $\vec{x}$, is thus dependent on the particle's own experience and those of its neighbours. Each particle has a memory, containing the best position found so far during the course of the optimisation, which is called personal best or $\vec{p}$. Whereas the best position so far found throughout the population, or the local neighbourhood, is called neighbourhood best.

A standard particle swarm version, Clerc-Kennedy PSO (PSO-CK) or constriction PSO defines the position of each particle by adding a velocity to the current position. Here is

the equation for updating the velocity and position of each particle:

$$v_{id}^{t} = \chi\left(v_{id}^{t-1} + c_1 r_1\left(p_{id} - x_{id}^{t-1}\right) + c_2 r_2\left(g_{id} - x_{id}^{t-1}\right)\right) \quad (11)$$

$$x_{id}^{t} = v_{id}^{t} + x_{id}^{t-1} \quad (12)$$

where χ which is the constriction factor is set to 0.72984 which is reported to be be working well in general [21]; v_{id}^{t-1} is the velocity of particle i in dimension d at time step $t-1$; $c_{1,2}$ are the learning factors (also referred to as acceleration constants) for personal best and neighbourhood best respectively (they are constant); $r_{1,2}$ are random numbers adding stochasticity to the algorithm and they are drawn from a uniform distribution on the unit interval $U(0,1)$; p_{id} is the personal best position of particle x_i in dimension d; and g_{id} is neighbourhood best. In the experiments reported in this work, local neighbourhood is used.

PSO algorithm is based on particles' individual experience and their social interaction with the particle swarms.

APPENDIX B
DIFFERENTIAL EVOLUTION ALGORITHM

Differential evolution (DE), an evolutionary algorithms (EAs), is a simple global numerical optimiser over continuous search spaces which was first introduced by Storn and Price [31].

DE is a population based stochastic algorithm, proposed to search for an optimum value in the feasible solution space. The parameter vectors of the population are defined as follows:

$$\vec{x}_i^{g} = \left[x_{i,1}^{g}, x_{i,2}^{g}, ..., x_{i,D}^{g}\right], i = 1, 2, ..., NP \quad (13)$$

where g is the current generation, D is the dimension of the problem space and NP is the population size. In the first generation, (when $g = 0$), the i^{th} vector's j^{th} component could be initialised as:

$$x_{i,j}^{0} = x_{min,d} + r\left(x_{max,d} - x_{min,d}\right) \quad (14)$$

where r is a random number drawn from a uniform distribution on the unit interval $U(0,1)$, and x_{min}, x_{max} are the lower and upper bounds of the d^{th} dimension, respectively. The evolutionary process (mutation, crossover and selection) starts after the initialisation of the population.

1) Mutation: At each generation g, the mutation operation is applied to each member of the population x_i^g (target vector) resulting in the corresponding vector v_i^g (mutant vector). In this work, ***DE/best/1*** variation of mutation approaches is used:

$$v_i^{g} = x_{best}^{g} + F\left(x_{r_1}^{g} - x_{r_2}^{g}\right) \quad (15)$$

where r_1 and r_2 are different from i and are distinct random integers drawn from the range $[1, NP]$; In generation g, the vector with the best fitness value is x_{best}^g; and F is a positive control parameter for constricting the difference vectors.

2) Crossover: Crossover operation, improves population diversity through exchanging some components of v_i^g (mutant vector) with x_i^g (target vector) to generate u_i^g (trial vector). This process is led as follows:

$$u_{i,j}^{g} = \begin{cases} v_{i,j}^{g}, & \text{if } r \le CR \text{ or } j = r_d \\ x_{i,j}^{g}, & \text{otherwise} \end{cases} \quad (16)$$

where r is a uniformly distributed random number drawn from the unit interval $U(0,1)$, r_d is randomly generated integer from the range $[1, D]$; this value guarantees that at least one component of the trial vector is different from the target vector. The value of CR, which is another control parameter, specifies the level of inheritance from v_i^g (mutant vector).

3) Selection: The selection operation decides whether x_i^g (target vector) or u_i^g (trial vector) would be able to to pass to the next generation $(g+1)$.In case of a minimisation problem, the vector with a smaller fitness value is admitted to the next generation:

$$x_i^{g+1} = \begin{cases} u_i^{g}, & \text{if } f\left(u_i^{g}\right) \le f\left(x_i^{g}\right) \\ x_i^{g}, & \text{otherwise} \end{cases} \quad (17)$$

where $f(x)$ is the fitness function.

APPENDIX C
GENETIC ALGORITHM

In this work, we use a real-valued Genetic Algorithm (GA) which has previously shown to work well on real-world problems [32], [33]. The GA works in the following way: the individuals are first randomly initialised and their fitness is evaluated through an objective function. Afterwards, in a iterative process, each individual has a probability of being exposed to recombination or mutation (or both). These probabilities are p_c and p_m respectively. The recombination operator used is arithmetic crossover and the mutation operator used is Cauchy mutation using an annealing scheme. At the end, in order to comb out the least fit individual, tournament selection [34] is utilised.

The reason behind using Cauchy mutation operator vs. the well-known Gaussian mutation operator is the thick (or heavy) trails of the Cauchy distribution that allows it to generate considerable changes, more frequently, compared to the Gaussian distribution. The Cauchy distribution is defined by:

$$C(x, \alpha, \beta) = \frac{1}{\beta\pi\left(1 + \left(\frac{x-\alpha}{\beta}\right)^2\right)} \quad (18)$$

where $\alpha \le 0$, $\beta > 0$, $-\infty < x < \infty$ (α and β are parameters that affect the mean and spread of the distribution). As specified in [33], all of the solution parameters are subject to mutation and the variance is scaled with $0.1\times$ the range of the specific parameter in question.

In order to decrease the value of β as a function of the elapsed number of generations t, an annealing scheme was applied (α was set to 0):

$$\beta(t) = \frac{1}{1+t} \tag{19}$$

As for the arithmetic crossover, the offspring is generated as a weighted mean of each gene of the two parents:

$$\text{offspring}_i = r \times \text{parent1}_i + (1-r) \times \text{parent2}_i \tag{20}$$

where offspring_i is the i'th gene of the offspring, and parent1_i and parent2_i refer to the i'th gene of the two parents, respectively. The weight r is drawn from a uniform distribution on the unit interval $U(0,1)$.

In this experiment, the probability of crossover and mutation of the individuals is set to $p_c = 0.7$ and $p_m = 0.9$ respectively. The tournament size of the tournament selection is set to two, and elitism with an elite size of one is deployed to maintain the best found solution in the population.

ACKNOWLEDGMENT

The author would like to thank the reviewers for their valuable comments which helped improve this work.

REFERENCES

[1] M. Mitchell, *An introduction to genetic algorithms, 1996*. MIT press, 1996.
[2] D. E. Knuth, *The art of computer programming. Vol. 3, Sorting and Searching*. Addison-Wesley Reading, MA, 1973.
[3] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial intelligence: a modern approach*. Prentice hall Englewood Cliffs, 1995, vol. 2.
[4] A. Neumaier, "Complete search in continuous global optimization and constraint satisfaction," *Acta Numerica*, vol. 13, no. 1, pp. 271–369, 2004.
[5] M. Lungarella, F. Iida, J. Bongard, and R. Pfeifer, *50 years of artificial intelligence: essays dedicated to the 50th anniversary of artificial intelligence*. Springer, 2007.
[6] M. Wooldridge, "An introduction to multiagent systems. 2002," *West Sussex, England: John Wiley and Sons Ltd*, vol. 348, 2002.
[7] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989.
[8] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. IV. Piscataway, NJ: IEEE Service Center, 1995, pp. 1942–1948.
[9] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *Computational Intelligence Magazine, IEEE*, vol. 1, no. 4, pp. 28–39, 2006.
[10] E. Bonabeau, M. Dorigo, and G. Theraulaz, "Inspiration for optimization from social insect behaviour," *Nature*, vol. 406, p. 3942, 2000.
[11] E. Wilson, *Sociobiology: The new synthesis*. Belknap Press, 1975.
[12] E. A. Wilson, "On the whales, seals, and birds of ross sea and south victoria land," 1905.
[13] B. Stonehouse and T. W. Glenister, *The Emperor Penguin, Aptenodytes Forsteri Gray*. Published for the Colonial Office by HMSO, 1953, vol. 10.
[14] C. Gilbert, G. Robertson, Y. Le Maho, and A. Ancel, "How do weather conditions affect the huddling behaviour of emperor penguins?" *Polar Biology*, vol. 31, no. 2, pp. 163–169, 2008.
[15] P. Jouventin, *Comportement et structure sociale chez le manchot empereur*. Expéditions Polaires Françaises, 1971.
[16] A. Ancel, H. Visser, Y. Handrich, D. Masman, and Y. L. Maho, "Energy saving in huddling penguins," *Nature*, vol. 385, pp. 304–305, 1997.
[17] J. Prévost, *Ecologie du manchot empereur*, 1961.
[18] J.-L. Mougin and E. P. Françaises, *Observations écologiques à la colonie de manchots empereurs de Pointe Géologie (Terre Adélie) en 1964*, 1966.
[19] R. Kirkwood and G. Robertson, "The occurrence and purpose of huddling by emperor penguins during foraging trips," *Emu*, vol. 99, no. 1, pp. 40–45, 1999.
[20] C. Gilbert, G. Robertson, Y. Le Maho, Y. Naito, and A. Ancel, "Huddling behavior in emperor penguins: dynamics of huddling," *Physiology & behavior*, vol. 88, no. 4, pp. 479–488, 2006.
[21] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proc of the Swarm Intelligence Symposium*. Honolulu, Hawaii, USA: IEEE, 2007, pp. 120–127.
[22] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Nanyang Technological University, Singapore and Kanpur Genetic Algorithms Laboratory, IIT Kanpur, Tech. Rep., 2005.
[23] J. Peña, "Theoretical and empirical study of particle swarms with additive stochasticity and different recombination operators," in *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, ser. GECCO '08. New York, NY, USA: ACM, 2008, pp. 95–102. [Online]. Available: http://doi.acm.org/10.1145/1389095.1389109
[24] C.-Y. Lee and X. Yao, "Evolutionary programming using mutations based on the lévy probability distribution," *Evolutionary Computation, IEEE Transactions on*, vol. 8, no. 1, pp. 1–13, 2004.
[25] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *Evolutionary Computation, IEEE Transactions on*, vol. 3, no. 2, pp. 82–102, 1999.
[26] D. Gehlhaar and D. Fogel, "Tuning evolutionary programming for conformationally flexible molecular docking," in *Evolutionary Programming V: Proc. of the Fifth Annual Conference on Evolutionary Programming*, 1996, pp. 419–429.
[27] O. Olorunda and A. P. Engelbrecht, "Measuring exploration/exploitation in particle swarms using swarm diversity," in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*. IEEE, 2008, pp. 1128–1134.
[28] M. M. al-Rifaie and A. Aber, "Identifying metastasis in bone scans with stochastic diffusion search," in *Information Technology in Medicine and Education (ITME)*. IEEE, 2012. [Online]. Available: http://dx.doi.org/10.1109/ITiME.2012.6291355
[29] M. M. al-Rifaie, A. Aber, and A. M. Oudah, "Utilising stochastic diffusion search to identify metastasis in bone scans and microcalcifications on mammographs," in *Bioinformatics and Biomedicine (BIBM 2012), Multiscale Biomedical Imaging Analysis (MBIA2012)*. IEEE, 2012, pp. 280–287. [Online]. Available: http://dx.doi.org/10.1109/BIBMW.2012.6470317
[30] F. Heppner and U. Grenander, "A stochastic nonlinear model for coordinated bird flocks." *American Association for the Advancement of Science, Washington, DC(USA).*, 1990.
[31] R. Storn and K. Price, "Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces," 1995, tR-95-012, [online]. Available: http://www.icsi.berkeley.edu/ storn/litera.html.
[32] R. Thomsen, "Flexible ligand docking using evolutionary algorithms: investigating the effects of variation operators and local search hybrids," *Biosystems*, vol. 72, no. 1-2, pp. 57–73, 2003.
[33] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 2, 2004, pp. 1980–1987.
[34] T. Back, D. B. Fogel, and Z. Michalewicz, *Handbook of evolutionary computation*. IOP Publishing Ltd., 1997.