Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp. 529–538
DOI: 10.15439/2014F142
ACSIS, Vol. 2

Dispersive Flies Optimisation

Mohammad Majid al-Rifaie
Department of Computing
Goldsmiths University of London
London SE14 6NW, United Kingdom
Email: m.majid@gold.ac.uk

***Abstract*—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 – Dispersive Flies Optimisation or DFO – whose inspiration is beckoned from the swarming behaviour of flies over food sources in nature. The simplicity of the algorithm, which is the implementation of one such paradigm for continuous optimisation, facilitates the analysis of its behaviour. 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 three other well-known population based algorithms (Particle Swarm Optimisation, Differential Evolution algorithm and Genetic Algorithm). The convergence-independent diversity of DFO algorithm makes it a potentially suitable candidate for dynamically changing environment. In addition to diversity, the performance of the newly introduced algorithm is investigated using the three performance measures of accuracy, efficiency and reliability and its outperformance is demonstrated in the paper.**

I. Introduction

THROUGHOUT the history nature has been an inexplicable source of inspiration for scientists and researchers. Observations, many of which made unintentionally, have been triggering the inquisitive minds for hundreds of years. The task of resolving problems and its often present nature in the minds of scientists boosts the impact of these observations, which in cases led to discoveries. Among others, researchers in mathematics, physics and natural sciences have had their fair share of 'observations-leading-to-discoveries'.

Observing the magnificently choreographed movements of birds, behaviour of ants foraging, convergence of honey bees in search for food source and so forth has led several researchers to propose (inspired vs. identical) models used to solve various optimisation problems. Genetic Algorithm [1], Particle Swarm Optimisation [2] and Ant Colony Optimisation [3] are only few such techniques belonging to the broader category of swarm intelligence; it investigates collective intelligence and 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.

The work presented here aims at proposing a novel nature-inspired algorithm based on the behaviours of flies hovering over food sources. This model – Dispersive Flies Optimisation or DFO – is first formulated mathematically and then a set of experiments is conducted to examine its performance when presented with various problems.

II. Flies in Nature

Flies are insects of the order *Diptera*, which comprises a large order, containing an estimated $240,000$ species of mosquitoes, gnats, midges and others [4]. Flies exist in various types each exhibiting distinctive behaviour in different environments. What most flies have in common is their swarming behaviour which depends on several factors.

Swarming have been described in [5] where a difference of shape between low swarms over dung and high swarms over other markers have been logged. High swarms fluctuated in height; vertical movements of the swarms of *Anopheles franciscanus* (Culicidae) are said to be correlated with female presence at swarms [6]. Height change in mosquito swarm induced by a clarinet note [7] and the human voice [8] may have evolved as responses to the flight tone of female mosquitoes [9].

Swarms of flies are associated with visual markers ranging in size from cowpies and stones to church steeples [10]. The criteria used by insects to select markers may be quite subtle; it was noted in [11] that certain objects are used repeatedly by the mosquito *Aedes cataphylla* while similar objects nearby are neglected.

As explained in [12], various swarms of flies usually "flying in relation to a more or less conspicuous element of the landscape, a lakeshore, a road, a treetop, below the tip of a branch, in an opening in the forest canopy, above a cow, an outstanding leaf", and so on according to species (e.g. [13], [14]). Depending on the species, the size of the swarm may consist of a single individual or tens or thousands, related to a discrete swarm marker; or even countless millions in the zonal swarms of lake shores.

Several elements play a role in *disturbing* the swarms of flies; for instance, the presence of a threat causes the swarms to disperse, leaving their current marker; they return to the marker immediately after the threat is over. However, during this period if they discover another marker which matches their criteria closer, they adopt the new marker.

III. Dispersive Flies Optimisation

Dispersive Flies Optimisation (DFO) is an algorithm inspired by the swarming behaviour of flies hovering over food sources. As detailed in section II, the swarming behaviour of flies is determined by several factors and that the presence of threat could disturb their convergence on the marker (or the

978-83-60810-58-3/$25.00

optimum value). Therefore, having considered the formation of the swarms over the marker, the breaking or weakening of the swarms is also noted in the proposed algorithm.

In other words, the swarming behaviour of the flies, in Dispersive Flies Optimisation, consist of two tightly connected mechanisms, one is the formation of the swarms and the other is its breaking or weakening. 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 flies (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 flies are randomly initialised with a position for each flies in the search space.

On each iteration, the components of the position vectors are independently updated, taking into account the component's value, the corresponding value of the best neighbouring fly (consider ring topology) with the best fitness, and the value of the best fly in the whole swarm:

$$x_{id}^t = x_{nb,d}^{t-1} + \mathrm{U}(0,1) \times \left(x_{sb,d}^{t-1} - x_{id}^{t-1}\right) \tag{3}$$

where $x_{nb,d}^{t-1}$ is the value of the *n*eighbour's *b*est fly in the d^{th} dimension at time step $t-1$; $x_{sb,d}^{t-1}$ is the value of the *s*warm's *b*est fly in the d^{th} dimension at time step $t-1$; and $\mathrm{U}(0,1)$ is the uniform distribution between 0 and 1.

The algorithm is characterised by two principle components: a dynamic rule for updating flies position (assisted by a social neighbouring network that informs this update), and communication of the results of the best found fly to other flies.

As stated earlier, the swarm is disturbed for various reasons; one of the positive impacts of such disturbances is the displacement of the disturbed flies 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 flies' 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 stagnation over a likely local minima.

Algorithm 1 summarises the DFO algorithm[1].

The next section briefly presents three population-based algorithms which will be used to compare the performance of DFO, and then the results of a series of experiments conducted on DFO over a set of benchmark functions are reported.

[1]The source code can be downloaded from the following page: http://doc.gold.ac.uk/~map01mm/DFO/

Algorithm 1 Dispersive Flies Optimisation

while FE $< 300,000$ **do**
  for $i = 1 \rightarrow NP$ **do**
    $\vec{x}_i$.fitness $\leftarrow f(\vec{x}_i)$
  end for
  $sb \leftarrow \{sb,\ \forall\ f(\vec{x}_{sb}) = \min\left(f(\vec{x}_1), f(\vec{x}_2), ..., f(\vec{x}_{NP})\right)\}$
  $nb \leftarrow \{nb,\ \forall\ f(\vec{x}_{nb}) = \min\left(f(\vec{x}_{\text{left}}), f(\vec{x}_{\text{right}})\right)\}$
  for $i = 1 \rightarrow NP$ **do**
    for $d = 1 \rightarrow \mathrm{D}$ **do**
      $\tau_d \leftarrow x_{nb,d}^{t-1} + \mathrm{U}(0,1) \times \left(x_{sb,d}^{t-1} - x_{id}^{t-1}\right)$
      if $(r < dt)$ **then**
        $\tau_d \leftarrow x_{min,d} + r\left(x_{max,d} - x_{min,d}\right)$
      end if
    end for
    $\vec{x}_i \leftarrow \vec{\tau}$
  end for
end while

IV. Population-Based Algorithms

The three algorithms introduced briefly in this section are variations of particle swarm optimisation (PSO), differential evolution algorithm (DE) and genetic algorithm (GA). One of the common features of these algorithms are the interactions between their population (i.e. information sharing), with the ultimate goal of finding the optima.

A. Particle Swarm Optimisation

Particle swarm optimisation (PSO) is population based optimization technique developed in 1995 by Kennedy and Eberhart [2]. 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 [15].

A swarm in PSO algorithm comprises of a number of particles and each particle represents a point in a multi-dimensional problem space. 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:

$$v_{id}^t = \chi\left(v_{id}^{t-1} + c_1 r_1\left(p_{id} - x_{id}^{t-1}\right) + c_2 r_2\left(g_{id} - x_{id}^{t-1}\right)\right) \tag{4}$$

$$x_{id}^t = v_{id}^t + x_{id}^{t-1} \tag{5}$$

where χ which is the constriction factor is set to 0.72984 which is reported to be be working well in general [16]; 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.

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 [17].

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:

$$\vec{x}_i^g = \left[x_{i,1}^g, x_{i,2}^g, ..., x_{i,D}^g\right], i = 1, 2, ..., NP \tag{6}$$

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:

$$x_{i,j}^0 = x_{min,d} + r\left(x_{max,d} - x_{min,d}\right) \tag{7}$$

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:

$$v_i^g = x_{best}^g + F\left(x_{r_1}^g - x_{r_2}^g\right) \tag{8}$$

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 and is set to 0.5.

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:

$$u_{i,j}^g = \begin{cases} v_{i,j}^g, & \text{if } r \leq CR \text{ or } j = r_d \\ x_{i,j}^g, & \text{otherwise} \end{cases} \tag{9}$$

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 and is set to 0.5, 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:

$$x_i^{g+1} = \begin{cases} u_i^g, & \text{if } f\left(u_i^g\right) \leq f\left(x_i^g\right) \\ x_i^g, & \text{otherwise} \end{cases} \tag{10}$$

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

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 [18], [19]. 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 [20] is utilised.

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

$$C(x, \alpha, \beta) = \frac{1}{\beta\pi\left(1 + \left(\frac{x-\alpha}{\beta}\right)^2\right)} \tag{11}$$

where $\alpha \leq 0$, $\beta > 0$, $-\infty < x < \infty$ (α and β are parameters that affect the mean and spread of the distribution). As specified in [19], 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):

$$\beta(t) = \frac{1}{1+t} \tag{12}$$

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

$$\text{offspring}_i = r \times \text{parent1}_i + (1-r) \times \text{parent2}_i \tag{13}$$

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 the experiments conducted in this paper, the probabilities 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.

V. Experiments

This section presents a set of experiment investigating the performance of the newly introduced Dispersive Flies Optimisation (DFO) and discusses the results. Then, to understand whether disturbance plays an important role in the optimisation process, a *control* algorithm is presented DFO-c where no disturbance is inflicted upon the population of flies.

Recognising the lose of diversity as a common issue in all distribution based evolutionary optimisers (since 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 DFO 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 [16] and [21]. The first set, f_{1-14}, have been used by several authors [22], [16], [23] and it contains the three classes of functions recommended by Yao *et al.* [24]: unimodal and high dimensional, multimodal and high dimensional, and low dimensional functions with few local minima. In order not to initialise the flies on or near a region in the search space known to have the global optimum, *region scaling* technique is used [25], which makes sure the flies 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 [21] 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.

One hundred flies were used in the experiments and the termination criterion for the experiments is set to reaching $300,000$ function evaluations (FEs). There are 50 Monte Carlo simulations for each experiment and the results are averaged over these independent simulations. Apart from the disturbance threshold which is set to $dt = 0.001$, there are no adjustable parameters in DFO's update equation.

The aim of the experiments is to study and demonstrate the qualities of the newly introduced algorithm as a population based continuous optimiser. The behaviour of the DFO algorithm is compared against its control counterpart and some other population based algorithms (see Sections IV-A, IV-B and IV-C).

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:

$$\text{Error} = |f(\vec{x}_g) - f(\vec{x}_o)| \quad (14)$$

$$\text{Efficiency} = \frac{1}{n}\sum_{i=1}^{n} \text{FEs} \quad (15)$$

$$\text{Reliability} = \frac{n^{'}}{n} \times 100 \quad (16)$$

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 [26] to be a robust measure in the presence of outliers and is defined as:

$$\text{Diversity} = \frac{1}{NP}\sum_{i=1}^{NP}\sqrt{\sum_{j=1}^{D}\left(x_i^j - \bar{x}^j\right)^2} \quad (17)$$

$$\bar{x}^j = \frac{1}{NP}\sum_{i=1}^{NP} x_i^j \quad (18)$$

where NP is the number of flies 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 Dispersive Flies Optimisation

The error, efficiency and reliability results of DFO 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 ($f_{1-2,5-11}$ and $g_{1-2,7,9}$) are optimised with the specified accuracy. The figures in the table are expanded in the following categories:

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	$(-100, 100)^D$

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 two out of five 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, with 100% reliability for f_{10} and 32% for 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 50% of the benchmarks in this category (f_{5-9} and $g_{7,9}$), 71% of which achieve 100% success rate (all except f_7, g_7 with 28% and 10% success rates respectively). The optimiser exhibit a promising performance when dealing with the difficult Rosenbrock functions (f_3, g_6), reaching the error of 10^{-4} and 10^{-3} respectively. The algorithm performs exceptionally well in optimising the infamous Rastrigin functions, both common and shifted mode (i.e. f_5 and g_9), achieving 100% success rate; however it does show weakness in the more challenging g_{10} rotated version.

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

In order 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 DFO

Most swarm intelligence and evolutionary techniques commence with exploration and, over time (i.e. function evaluations or iterations), lean towards exploitation. Maintaining the right balance between exploration and exploitation phases has proved to be difficult. The absence of the aforementioned balance leads to a weaker diversity when encountering a local minimum and thus the common problem of pre-mature convergence to a local minimum surfaces.

Similar to other swarm intelligence and evolutionary algorithms, DFO commences with exploration and over time, through its mechanism (i.e. gradual decrease in the distance between the members of the population and as such,

TABLE II
DFO – DISPERSIVE FLIES OPTIMISATION

	Min.	Max.	Median	Mean	StdDev	Eff. (Rel.)
f_1	6.46E-47	1.97E-40	1.75E-43	1.07E-41	3.49E-41	46850 (100%)
f_2	2.24E-12	6.01E-10	6.46E-11	1.08E-10	1.26E-10	239850 (100%)
f_3	1.74E-04	1.45E+01	3.65E-01	2.17E+00	3.62E+00	∞ (0%)
f_4	3.89E-07	5.05E-03	2.87E-05	2.49E-04	7.81E-04	∞ (0%)
f_5	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	84850 (100%)
f_6	2.84E-14	6.39E-14	3.91E-14	3.88E-14	6.49E-15	121200 (100%)
f_7	0.00E+00	1.54E-01	1.85E-02	3.25E-02	3.74E-02	47450 (28%)
f_8	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	50950 (100%)
f_9	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	55550 (100%)
f_{10}	0.00E+00	2.22E-16	0.00E+00	4.00E-17	8.62E-17	1700 (100%)
f_{11}	0.00E+00	8.10E+01	8.10E+01	5.51E+01	3.82E+01	2100 (32%)
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	5.68E-14	2.27E-13	1.71E-13	1.49E-13	4.28E-14	45300 (100%)
g_2	4.55E-12	9.78E-10	3.88E-11	1.03E-10	1.57E-10	234100 (100%)
g_3	3.58E+05	3.22E+06	1.40E+06	1.38E+06	6.23E+05	∞ (0%)
g_4	1.40E+00	2.38E+02	2.18E+01	3.71E+01	4.74E+01	∞ (0%)
g_5	3.47E+03	1.82E+04	8.95E+03	9.26E+03	3.17E+03	∞ (0%)
g_6	1.66E-03	1.51E+02	3.06E+00	1.41E+01	3.05E+01	∞ (0%)
g_7	3.31E-11	2.64E-01	1.97E-02	2.93E-02	4.05E-02	236800 (10%)
g_8	2.00E+01	2.02E+01	2.01E+01	2.01E+01	3.11E-02	∞ (0%)
g_9	1.14E-13	2.27E-13	1.71E-13	1.52E-13	3.71E-14	89450 (100%)
g_{10}	1.29E+02	3.42E+02	2.34E+02	2.38E+02	5.62E+01	∞ (0%)
g_{11}	2.46E+01	4.02E+01	3.11E+01	3.12E+01	3.23E+00	∞ (0%)
g_{12}	9.73E+01	1.58E+04	2.34E+03	3.62E+03	3.51E+03	∞ (0%)
g_{13}	9.34E-01	2.01E+00	1.48E+00	1.48E+00	3.07E-01	∞ (0%)
g_{14}	1.23E+01	1.40E+01	1.35E+01	1.35E+01	3.69E-01	∞ (0%)

each agent's local and global best positions), 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 1 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 in exploration while converging to the optima whose fitness reach as low as 10^{-13} in g_1 and g_9.

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

The graphs in Fig. 2 illustrate the diversity of DFO-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 DFO-c is less than DFO, 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).

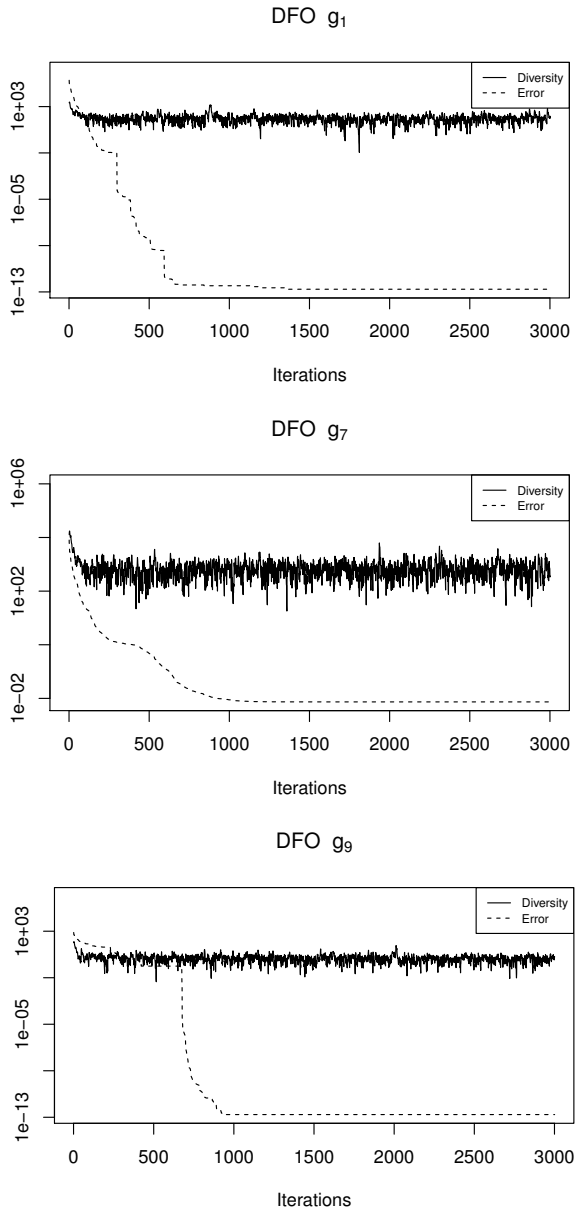

Fig. 1. DFO: diversity and error in $g_{1,7,9}$.

Note the gradual shrinkage of diversity in g_9 ($\approx 10^{-13}$) which is a clear indication of a premature convergence to a local minima with very poor chance of escape.

In order to compare the performance of DFO and its control counterpart, Table III presents the result of optimising the benchmarks using DFO-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 V-B for the definitions of the measures). The results show that in 89% of cases (where there is a significant difference between the two algorithms), DFO is performing significantly better than its control counterpart (DFO-c) which is stripped from the

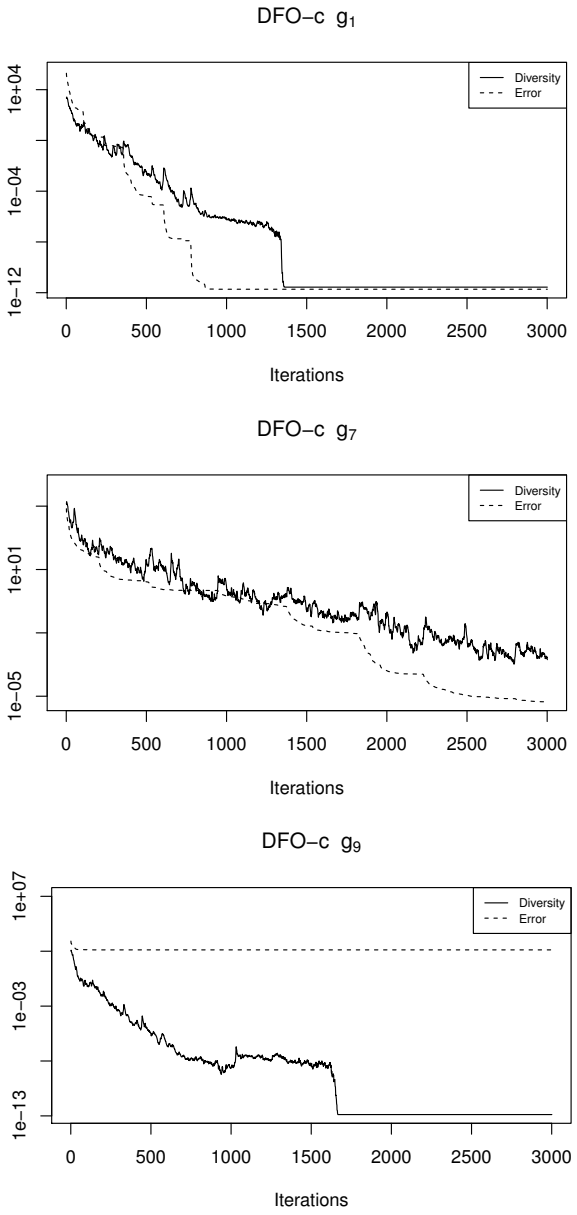

Fig. 2. DFO-c: diversity and error in $g_{1,7,9}$.

TABLE III
DFO-C – CONTROL DFO ALGORITHM

	Min.	Max.	Median	Mean	StdDev	Eff. (Rel.)
f_1	1.44E-56	3.09E-36	1.27E-45	9.65E-38	4.55E-37	65400 (100%)
f_2	7.29E-09	3.23E+01	1.28E-04	7.60E-01	4.60E+00	298200 (2%)
f_3	5.27E-05	1.61E+02	5.08E+00	1.67E+01	3.08E+01	∞ (0%)
f_4	4.48E-09	3.20E+03	1.55E+03	1.40E+03	8.66E+02	141500 (2%)
f_5	1.87E+02	4.17E+02	2.96E+02	2.94E+02	5.76E+01	∞ (0%)
f_6	1.97E+01	2.00E+01	1.98E+01	1.98E+01	5.24E-02	∞ (0%)
f_7	2.22E-16	6.00E+00	9.30E-02	3.51E-01	8.72E-01	64050 (8%)
f_8	1.03E-32	3.30E+02	2.14E+00	2.35E+01	5.84E+01	132950 (24%)
f_9	0.00E+00	1.57E+02	1.54E-01	5.35E+00	2.27E+01	176500 (30%)
f_{10}	0.00E+00	2.22E-16	0.00E+00	7.99E-17	1.08E-16	1700 (100%)
f_{11}	0.00E+00	8.10E+01	8.10E+01	5.99E+01	3.59E+01	2100 (26%)
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	5.68E-14	9.37E-05	1.14E-13	1.91E-06	1.33E-05	70600 (94%)
g_2	1.68E-09	2.23E+01	1.23E-04	4.63E-01	3.14E+00	257700 (2%)
g_3	2.18E+05	5.38E+06	1.67E+06	1.73E+06	9.39E+05	∞ (0%)
g_4	2.23E+02	1.74E+04	1.80E+03	2.91E+03	3.36E+03	∞ (0%)
g_5	5.79E+03	1.38E+04	8.50E+03	8.69E+03	2.00E+03	∞ (0%)
g_6	2.25E-04	9.53E+01	8.61E+00	1.68E+01	2.52E+01	∞ (0%)
g_7	3.01E-10	2.13E-01	3.02E-02	4.17E-02	4.41E-02	263900 (2%)
g_8	2.00E+01	2.02E+01	2.00E+01	2.01E+01	3.89E-02	∞ (0%)
g_9	8.36E+01	2.64E+02	1.62E+02	1.64E+02	4.61E+01	∞ (0%)
g_{10}	1.22E+02	4.93E+02	2.69E+02	2.71E+02	7.69E+01	∞ (0%)
g_{11}	1.98E+01	4.11E+01	3.10E+01	3.13E+01	3.97E+00	∞ (0%)
g_{12}	2.32E+02	1.38E+04	3.04E+03	4.78E+03	3.88E+03	∞ (0%)
g_{13}	4.79E+00	3.56E+01	1.47E+01	1.58E+01	6.47E+00	∞ (0%)
g_{14}	1.28E+01	1.45E+01	1.36E+01	1.37E+01	3.38E-01	∞ (0%)

diversity inducing disturbance. Furthermore, in all multimodal functions (f_{3-9} and g_{6-12}), whenever there is a statistically significant difference between DFO and DFO-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, DFO is 79% more efficient than its control counterpart, and 92% more reliable.

E. Fine Tuning Disturbance Threshold

The role of disturbance in increasing the diversity of DFO population is discussed earlier (Section V-D). Also, the importance of disturbance is investigated on the optimisation capability of DFO 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. 3 illustrates the performance of DFO using these dt probabilities. Both set of benchmarks (i.e. f_{1-14} and g_{1-14}) 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 DFO with other Population-Based Optimisers

Having presented the performance of the DFO 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

TABLE IV
COMPARING DFO AND DFO-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 DFO is significantly outperforming its counterpart algorithm; and o–X shows that the algorithm compared to DFO is significantly better than DFO. 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.

	DFO – DFO-c		
	Error	**Efficiency**	**Reliability**
f_1	o – X	1 – 0	–
f_2	X – o	1 – 0	1 – 0
f_3	X – o	–	–
f_4	X – o	0 – 1	0 – 1
f_5	X – o	1 – 0	1 – 0
f_6	X – o	1 – 0	1 – 0
f_7	X – o	1 – 0	1 – 0
f_8	X – o	1 – 0	1 – 0
f_9	X – o	1 – 0	1 – 0
f_{10}	o – X	0 – 1	–
f_{11}	–	0 – 1	1 – 0
f_{12}	–	–	–
f_{13}	–	–	–
f_{14}	–	–	–
g_1	–	1 – 0	1 – 0
g_2	X – o	1 – 0	1 – 0
g_3	X – o	–	–
g_4	X – o	–	–
g_5	–	–	–
g_6	–	–	–
g_7	X – o	1 – 0	1 – 0
g_8	–	–	–
g_9	X – o	1 – 0	1 – 0
g_{10}	X – o	–	–
g_{11}	–	–	–
g_{12}	–	–	–
g_{13}	X – o	–	–
g_{14}	X – o	–	–
	16 – 2	11 – 3	11 – 1

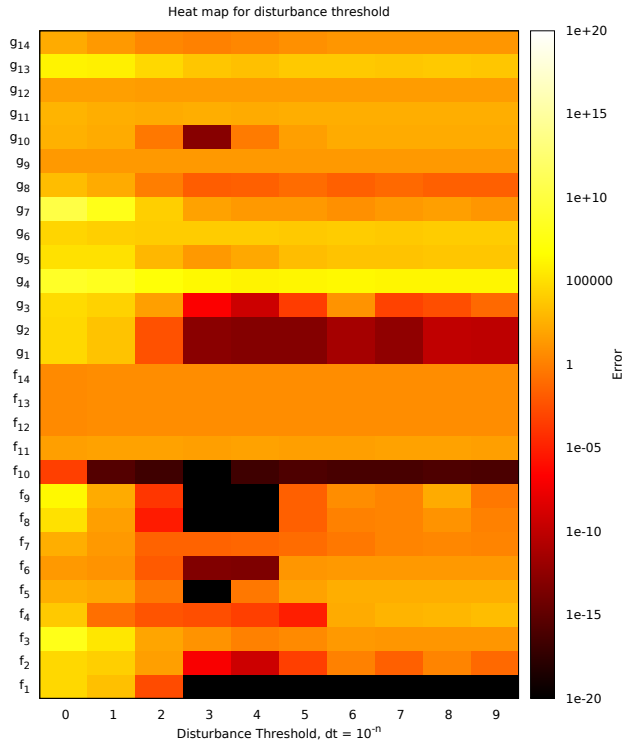

Fig. 3. Fine tuning disturbance threshold

TABLE V
DE (DIFFERENTIAL EVOLUTION), PSO (PARTICLE SWARM OPTIMISATION) AND GA (GENETIC ALGORITHM)

	DE		**PSO**		**GA**	
	Error	**Eff. (Rel.)**	**Error**	**Eff. (Rel.)**	**Error**	**Eff. (Rel.)**
g_1	1.38E-13	21500 (100%)	5.23E-14	656236 (100%)	5.04E-05	∞ (0%)
g_2	1.72E-07	∞ (0%)	1.33E-01	∞ (0%)	1.21E+04	∞ (0%)
g_3	9.65E+06	∞ (0%)	1.52E+06	∞ (0%)	1.47E+07	∞ (0%)
g_4	4.92E-01	∞ (0%)	7.89E+03	∞ (0%)	5.13E+04	∞ (0%)
g_5	2.34E+03	∞ (0%)	5.04E+03	∞ (0%)	2.09E+04	∞ (0%)
g_6	2.30E+00	265800 (12%)	2.16E+01	∞ (0%)	7.23E+02	∞ (0%)
g_7	5.39E-01	∞ (0%)	1.04E-02	279653 (10%)	5.48E+03	∞ (0%)
g_8	2.09E+01	∞ (0%)	2.09E+01	∞ (0%)	2.04E+01	∞ (0%)
g_9	3.47E+01	∞ (0%)	9.59E+01	∞ (0%)	2.20E+01	∞ (0%)
g_{10}	1.47E+02	∞ (0%)	1.14E+02	∞ (0%)	1.39E+02	∞ (0%)
g_{11}	3.65E+01	∞ (0%)	3.00E+01	∞ (0%)	1.17E+01	∞ (0%)
g_{12}	5.85E+05	∞ (0%)	9.51E+03	∞ (0%)	8.14E+03	∞ (0%)
g_{13}	5.70E+00	∞ (0%)	5.35E+00	∞ (0%)	2.70E+00	∞ (0%)
g_{14}	1.34E+01	∞ (0%)	1.25E+01	∞ (0%)	1.39E+01	∞ (0%)

algorithms deployed for this comparison are Differential Evolution (DE), Particle Swarm Optimisation (PSO) and Genetic Algorithm (GA). These algorithms are briefly described earlier in Sections IV-A, IV-B and IV-C. Generic versions of each algorithm are used against the generic version of Dispersive Flies Optimisation. In this comparison, only the second and the more challenging set of benchmarks, g_{1-14} are used. Table V presents 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 VI shows the result of the statistical analysis comparing DFO with the other three optimisers. Based on this comparison, whenever there is a significant difference between the performance of DFO and the other algorithms, DFO significantly outperforms DE, PSO and GA in 66.67%, 58.33% and 85.71% of the cases, respectively. Table VII summaries the efficiency results of the three optimisers with that of DFO; note that only the efficiency of functions reaching the specified error is given. As shown in the table, DFO, in the majority of cases, outperforms the other algorithms. In other words, although, when compared with DE, DFO only outperforms marginally (60%), it outperforms both PSO and GA in all cases (100%). The reliability comparison of DFO with the other optimisers is given in Table VIII. DFO is shown to be the most reliable algorithm in this comparison.

TABLE VI
COMPARING ERROR IN DFO 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.

	DFO - DE	**DFO - PSO**	**DFO - GA**
g_1	–	o – X	X – o
g_2	X – o	X – o	X – o
g_3	X – o	–	X – o
g_4	o – X	X – o	X – o
g_5	o – X	o – X	X – o
g_6	o – X	X – o	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	–	o – X
g_{12}	X – o	X – o	X – o
g_{13}	X – o	X – o	X – o
g_{14}	–	o – X	X – o
$\sum$	8 – 4	7 – 5	12 – 2

TABLE VII
COMPARING EFFICIENCY IN DFO 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.

	DFO - DE	**DFO - PSO**	**DFO - 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 VIII
COMPARING RELIABILITY IN DFO 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.

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

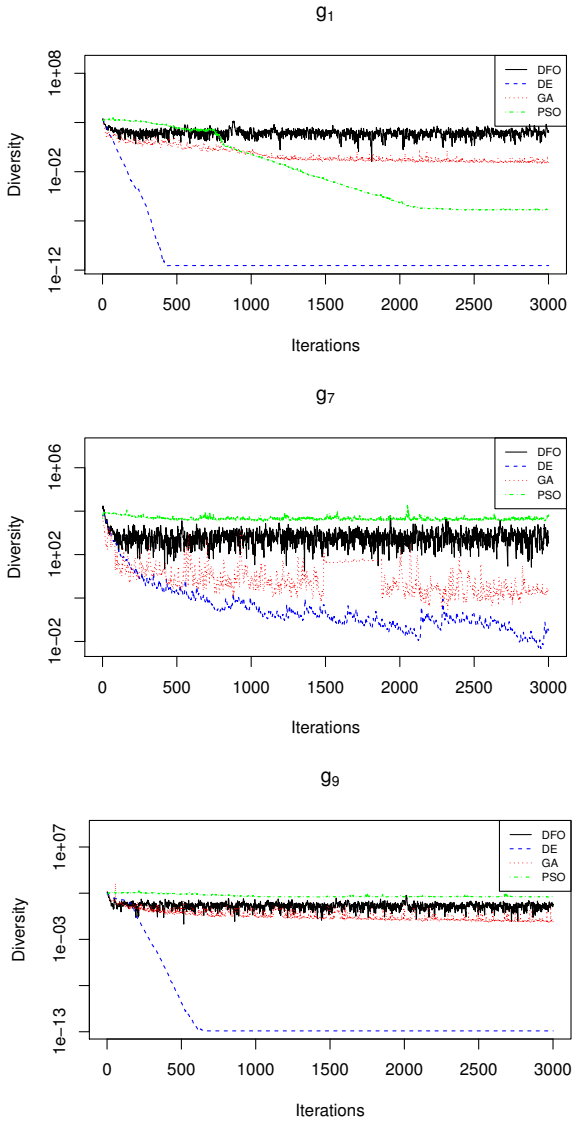

Fig. 4. Diversity of the population in DFO, DE, PSO and GA over three random trials in $g_{1,7 \text{ and } 9}$.

While DFO outperforms DE in 75% of cases, it show 100% outperformance when compared with PSO and GA. In order to compare the diversity of the DFO 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. 4. 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

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

VI. Conclusion

Dispersive Flies Optimisation (DFO), a simple numerical optimiser over continuous search spaces, is a population based stochastic algorithm, proposed to search for an optimum value in the feasible solution space; despite its simplicity, the algorithm's competitiveness 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 induced 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 injecting a component-wise disturbance (or restart) in the flies, aiming to preserve a balance between exploration and exploitation.

In addition to diversity, DFO's performance has been investigated using 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, DFO is more efficient and reliable in 84.62% and 90% of the cases, respectively; furthermore, when there exists a statistically significant difference, DFO converges to better solutions in 71.05% 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, dt. Additionally, optimising multi-objective real world problems is yet to be researched; this would be a continuation of an earlier set of works on the deployment of population-based algorithms for detecting metastasis in bone scans and calcifications in mammographs [27]. At last, but not least, given the demonstrated stable and convergence-independent diversity of Dispersive Flies Optimisation (in the context of the presented benchmarks), another exciting future research is to investigate the performance of DFO in the context of dynamic optimisation problems.

References

[1] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1989.

[2] 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.

[3] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *Computational Intelligence Magazine, IEEE*, vol. 1, no. 4, pp. 28–39, 2006.

[4] B. M. Wiegmann and D. K. Yeates, *Tree of Life: Diptera*. The Tree of Life Web Project, 1996.

[5] J. Downes, "Observations on the swarming flight and mating of culicoides (diptera: Ceratopogonidae) 1," *Transactions of the Royal Entomological Society of London*, vol. 106, no. 5, pp. 213–236, 1955.

[6] J. N. Belkin, N. Ehmann, and G. Heid, "Preliminary field observations on the behavior of the adults of anopheles franciscanus mccracken in southern california," *Mosq News*, vol. 11, pp. 23–31, 1951.

[7] H. T. Nielsen, "Swarming and some other habits of mansonia perturbans and psorophora ferox (diptera: Culicidae)," *Behaviour*, pp. 67–89, 1964.

[8] F. Knab, "The swarming of culex pipiens," *Psyche: A Journal of Entomology*, vol. 13, no. 5, pp. 123–133, 1906.

[9] L. M. Roth, "A study of mosquito behavior. an experimental laboratory study of the sexual behavior of aedes aegypti (linnaeus)," *American Midland Naturalist*, vol. 40, no. 2, pp. 265–352, 1948.

[10] R. T. Sullivan, "Insect swarming and mating," *The Florida Entomologist*, vol. 64, no. 1, pp. 44–65, 1981.

[11] W. Klassen and B. Hocking, "The influence of a deep river valley system on the dispersal of aedes mosquitos," *Bulletin of Entomological Research*, vol. 55, no. 02, pp. 289–304, 1964.

[12] J. Downes, "The swarming and mating flight of diptera," *Annual review of entomology*, vol. 14, no. 1, pp. 271–298, 1969.

[13] J. Downes, "Assembly and mating in the biting nematocera," *Intern. Congr. Entomol. Proc. 10th, Montreal*, pp. 425–34, 1958.

[14] R. L. Blickle, "Observations on the hovering and mating of tabanus bishopp," *Stone. Ann. Entomol. Soc. 52*, pp. 183–90, 1958.

[15] F. Heppner and U. Grenander, "A stochastic nonlinear model for coordinated bird flocks." *American Association for the Advancement of Science, Washington, DC(USA).*, 1990.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] T. Back, D. B. Fogel, and Z. Michalewicz, *Handbook of evolutionary computation*. IOP Publishing Ltd., 1997.

[21] 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.

[22] 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

[23] 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.

[24] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *Evolutionary Computation, IEEE Transactions on*, vol. 3, no. 2, pp. 82–102, 1999.

[25] 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.

[26] 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.

[27] 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