11

Dispersive Flies Optimisation: Modifications and Application

Mohammad Majid al-Rifaie

School of Computing and Mathematical Sciences, University of Greenwich
Old Royal Naval College, London, United Kingdom

Hooman Oroojeni M. J.

Department of Computing
Goldsmiths, University of London, London, United Kingdom

Mihalis Nicolaou

Computation-based Science and Technology Research Center
The Cyprus Institute, Nicosia, Cyprus

CONTENTS

11.1 Introduction

Information exchange and communication between individuals in swarm intelligence manifest themselves in a variety of forms, including: the use of different

update equations and strategies; deploying extra vectors/components in addition to the individuals' current positions; and dealing with tunable parameters. Ultimately the goal of the optimisers is to achieve a balance between global exploration of the search space and local exploitation of potentially suitable areas [1, 2] in order to guide the optimisation process.

One of the main motivations for studying dispersive flies optimisation (DFO) [3] is the algorithm's minimalist update equation which only uses the flies' position vectors for the purpose of updating the population. This is in contrast to several other population-based algorithms and their variants which besides using position vectors, use a subset of the following set of components (i.e. vectors): velocities and memories (personal best and global best) in particle swarm optimisation (PSO) [4], mutant and trial vectors in differential evolution (DE) [5], pheromone, heuristic vectors in Ant Colony Optimisation (ACO) [6], and so forth [7]. In addition to using only the position vectors at any given time, the only tunable parameter in DFO, other than the population size, is the *disturbance threshold*, Δ, which controls the component-wise restart in each dimension. This is again contrary to many well-known algorithms dealing with several (theoretically or empirically driven) *tunable parameters*, such as: learning factors, inertia weight in PSO, crossover or mutation rates, tournament and elite sizes, constricting factor in DE and/or Genetic Algorithms (GA) [8], heuristic strength, greediness, pheromone decay rate in ACO, impact of distance on attractiveness, scaling factor and speed of convergence in Firefly algorithm (FF) [9], and so on.

It is worthwhile noting that DFO is not the only minimalist algorithm and there have been several attempts to present 'simpler', more compact algorithms to better understand the dynamic of the population's behaviour as well as the significance of various communication strategies. Perhaps one of the most notable minimalist swarm algorithms is bare bones particle swarms (BB-PSO) [10] whose collapse has been studied in [11] along with the introduction of a dimensional jump or restart to the aforementioned algorithm, thus proposing the bare bones with jumps algorithm (BBJ). Another bare bones algorithm is barebones differential evolution (BBDE) [12] which is a hybrid of the barebones particle swarm optimiser and differential evolution, aiming to reduce the number of parameters; however, still with more components and parameters, and often at the expense of reduced performance level[1]. It is well understood that swarm intelligence techniques, including but not limited to PSO, are dependant on the tuning of their parameters (e.g. PSO relies on the choice of inertia weight [16, 17])[2]; as a result, adjusting a growing number of parameters becomes increasingly complex.

DFO has been applied to various problems, including but not limited to medical imaging [19], optimising machine learning algorithms [20, 21], train-

[1] Some of these shortcoming have been studied and at points overcome but at the expense of introducing additional parameters (see [13, 14, 15]).

[2] In a related work, Clerc [18] defines acceleration coefficient in order for PSO to be independent of computing the inertia weight.

ing and optimising deep neural network [22], computer vision and quantifying symmetrical complexities [23], building non-identical organic structures for gaming [24], identification of animation key points from 2D-medialness maps [25] and analysis of autopoiesis in computational creativity [26].

In this chapter, initially, DFO is briefly introduced in Section 11.2; then some of the modifications to the algorithms are discussed in Section 11.3; subsequently, the application of DFO in training neural networks for detecting a false alarm in ICU is presented in Section 11.4.

11.2 Dispersive flies optimisation

The swarming behaviour of the individuals in DFO consists of two tightly connected mechanisms, one is the formation of the swarms and the other is its breaking or weakening. The position vector of a fly in DFO is defined as:

$$\vec{x}_i^t = \left[x_{i0}^t, x_{i1}^t, ..., x_{i,D-1}^t\right], \qquad i \in \{0, 1, 2, ..., N\text{-}1\} \tag{11.1}$$

where i represents the i^{th} individual, t is the current time step, D is the problem dimensionality, and N is the population size. For continuous problems, $x_{id} \in \mathbb{R}$ (or a subset of $\mathbb{R}$).

In the first iteration, when $t = 0$, the d^{th} component of the i^{th} fly is initialised as:

$$x_{id}^0 = \mathrm{U}(x_{\min,d}, x_{\max,d}) \tag{11.2}$$

This, effectively generates a random value between the lower ($x_{\min,d}$) and upper ($x_{\max,d}$) bounds of the respective dimension, d.

On each iteration, dimensions of the position vectors are independently updated, taking into account:

- current fly's position
- current fly's best neighbouring individual (consider ring topology, where each fly has a left and a right neighbour)
- and the best fly in the swarm.

Therefore, the update equation is

$$x_{id}^{t+1} \quad = \quad x_{i_n d}^t + u(x_{sd}^t - x_{id}^t) \tag{11.3}$$

where,

- x_{id}^t: position of the i^{th} fly in d^{th} dimension at time step t
- $x_{i_n d}^t$: position of $\vec{x}_i^t$'s best *neighbouring* individual (in ring topology) in d^{th} dimension at time step t

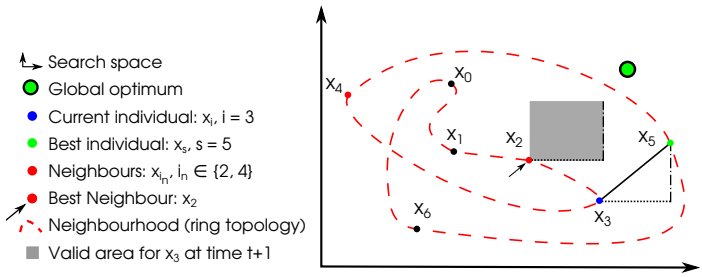

FIGURE 11.1
Sample update of x_i, where $i = 3$ in a 2D space.

- x_{sd}^t: position of the *swarm*'s best individual in the d^{th} dimension at time step t and $s \in \{0, 1, 2, ..., N\text{-}1\}$
- $u \sim \mathrm{U}(0, 1)$: generated afresh for each dimension update.

The update equation is illustrated in an example in Fig. 11.1 where $\vec{x}_3$ is to be updated. The algorithm is characterised by two main components: a dynamic rule for updating the population's position (assisted by a social neighbouring network that informs this update), and communication of the results of the best found individual to others.

As stated earlier, the position of members of the swarm in DFO can be restarted, with one impact of such restarts being the displacement of the individuals which may lead to discovering better positions. To consider this eventuality, an element of stochasticity is introduced to the update process. Based on this, individual dimensions of the population's position vectors are reset if a random number generated from a uniform distribution on the unit interval $\mathrm{U}(0, 1)$ is less than the *restart threshold*, Δ. This guarantees a restart to the otherwise permanent stagnation over likely local minima.

Algorithm 14 summarises the DFO algorithm. In this algorithm, each member of the population is assumed to have two neighbours (i.e. ring topology).

Algorithm 14 Dispersive flies optimisation.

1: **procedure** DFO $(N, D, \vec{x}_{\min}, \vec{x}_{\max}, f)$*
2: **for** $i = 0 \rightarrow N - 1$ **do** ▷ **Initialisation**: Go through each fly
3: **for** $d = 0 \rightarrow D - 1$ **do** ▷ Initialisation: Go through each dimension
4: $x_{id}^0 \leftarrow \mathrm{U}(x_{\min,d}, x_{\max,d})$ ▷ Initialise d^{th} dimension of fly i
5: **end for**
6: **end for**
7: **while** ! termination criteria **do** ▷ Main DFO loop
8: **for** $i = 0 \rightarrow N - 1$ **do** ▷ **Evaluation**: Go through each fly
9: $\vec{x}_i$.fitness $\leftarrow f(\vec{x}_i)$

```
10:         end for
11:         x⃗_s = arg min [f(x⃗_i)],   i ∈ {0, 1, 2, ..., N − 1}        ▷ Find best fly
12:         for i = 0 → N − 1 and i ≠ s do        ▷ Update each fly except the best
13:             x⃗_{i_n} = arg min [f(x⃗_{(i−1)%N}), f(x⃗_{(i+1)%N})]    ▷ Find best neighbour
14:             for d = 0 → D − 1 do                    ▷ Update each dimension
15:                 if U(0, 1) < Δ then                    ▷ Restart mechanism
16:                     x_id^{t+1} ← U(x_{min,d}, x_{max,d})        ▷ Restart within bounds
17:                 else
18:                     u ← U(0, 1)
19:                     x_id^{t+1} ← x_{i_n d}^t + u(x_sd^t − x_id^t)    ▷ Update the dimension value
20:                     if x_id^{t+1} < x_{min,d} or x_id^{t+1} > x_{max,d} then    ▷ Out of bounds
21:                         x_id^{t+1} ← U(x_{min,d}, x_{max,d})        ▷ Restart within bounds
22:                     end if
23:                 end if
24:             end for
25:         end for
26:     end while
27:     return x⃗_s
28: end procedure
```

* INPUT: swarm size, dimensions, lower/upper bounds, fitness function.

11.3 Modifications in DFO

In this section some of the immediate modifications to the algorithm are discussed. These modifications and variations were proposed after studying the behaviour of the population when applied to a number of problems.

11.3.1 Update equation

Diversity is the degree of convergence and divergence, which 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 to be a robust measure in the presence of outliers and is defined as [2]:

$$\text{DIVERSITY}^t = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\sum_{d=1}^{D}\left(x_{id}^t - \bar{x}_d^t\right)^2} \tag{11.4}$$

$$\bar{x}_d^t = \frac{1}{N}\sum_{i=1}^{N} x_{id}^t. \tag{11.5}$$

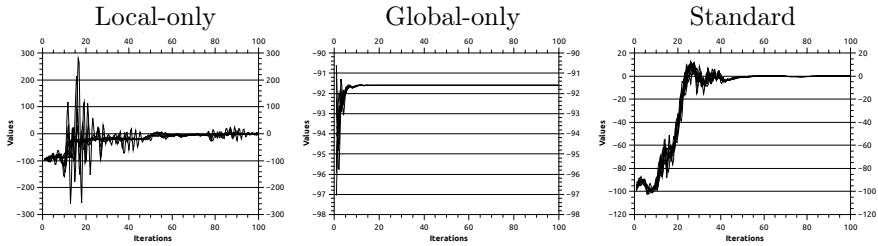

FIGURE 11.2
Diversity values in different update equations. Illustrating diversity values in one dimension when DFO, with $\Delta = 0$, is applied to the Sphere function: $f(\vec{x}) = \sum_{d=1}^{D} x_d^2$.

It is well understood that the disturbance threshold, Δ, impacts the diversity of the swarm directly. Assuming a predetermined value, diversity is proportionally applied to the swarm throughout the optimisation process. By merely changing the update equation, diversity could be altered. In addition to the standard update equation, local-only and global-only update equations can be seen below:

$$\text{Standard:} \quad x_{id}^{t+1} = x_{i_n d}^{t} + u(x_{sd}^{t} - x_{id}^{t}) \tag{11.6}$$

$$\text{Local-only:} \quad x_{id}^{t+1} = x_{i_n d}^{t} + u(x_{i_n d}^{t} - x_{id}^{t}) \tag{11.7}$$

$$\text{Global-only:} \quad x_{id}^{t+1} = x_{sd}^{t} + u(x_{sd}^{t} - x_{id}^{t}) \tag{11.8}$$

Diversity values of the swarm in each iteration of the above-mentioned update equations are illustrated in Fig. 11.2. For instance, in order to have an initially more "pronounced" exploration phase, the local-only update equation could be used. On the other hand, if an immediate convergence of the population is desirable, global-only structure can be utilised[3]. Other configurations have been tried by researchers depending on the problem, for instance, in [22], Eq. 11.9, below, is used, which removes current positions of each fly from the update equation, relying entirely on the best neighbour ($\vec{x}_{i_n}$) and the best member of the population ($\vec{x}_s$). In other words, this update equation, takes into account the position of the best neighbouring individual and the position of the best fly in the entire swarm to determine the updated position:

$$x_{id}^{t+1} = x_{i_n d}^{t} + u(x_{sd}^{t} - x_{i_n d}^{t}) \tag{11.9}$$

11.3.2 Disturbance threshold, Δ

Disturbance threshold, Δ, controls the diversity of the population and, in turn, influences the hill-climbing process. The standard DFO algorithm has

[3]Note: some of these features are shared amongst other swarm intelligence algorithms.

proposed a simple, single valued Δ for the entire dimensions. Therefore each dimension is exposed to the same probability of being restarted. However, depending on the nature of the problems and taking into account the dimensions' varying degree and need for exploration, a dimension-dependent Δ can be used. This would allow a tailored restart mechanism if such knowledge is available a-priori.

Furthermore, in order to adjust the level of exploration and exploitation, Δ can be tuned during the optimisation process based on the performance of the algorithm on a given problem. For instance, in [22], a mechanism is proposed for Δ to be re-adjusted depending on various performance factors and the swarm behaviour. This method is noted to have resulted in increased performance over the problem of detecting false alarms in Intensive Care Units (ICU).

The immediate impact of the disturbance threshold, as mentioned before, is the possibility of identifying better (and previously unexplored) solutions. This is achieved by randomly generating a value within the lower and upper bounds. Depending on the problems, the new position could be derived by other methods[4] where the updated value is not simply generated randomly between the bounds but rather tied to other associable values such as the current position of the fly, position of the best fly in the swarm, the best neighbouring fly or a combination of these. Following on this ethos, in [22], the position of the updated value is determined by the following:

$$x_{id}^{t+1} = \mathcal{N}(x_{i_n d}^{t}, \Delta^2) \tag{11.10}$$

therefore, restarting the value of the dimension based on a sample from a Gaussian with the mean set to $x_{i_n d}^{t}$ and variance to Δ^2.

11.4 Application: Detecting false alarms in ICU

Deep Neural Networks (DNN) and Deep Learning (DL) have been increasingly popular in solving various problems which have been otherwise difficult to address. However, one of main challenges is to design and train suitable neural networks.

In this section, a neuroevolution-based approach for training neural networks is used and applied to the problem of detecting false alarms in Intensive Care Units (ICU) based on physiological data. A common tool for training or

[4]Note that the disturbance threshold is applied on each single dimension of each fly in each iteration independently, and there are instances where generating a value between the lower and upper bounds results in unsuitable solutions (exploration), which are then re-absorbed by the swarm to previously known solutions. This, therefore, may result in the mechanism becoming less effective.

optimising a neural network is backpropagation (BP) with stochastic gradient-based learning. However, recent research [27] has shown that gradient-free, population-based algorithms can outperform the classical gradient based optimisation methods. This section demonstrates the use of DFO in optimising a deep neural network which is applied to the detection of false alarms in ICU. This application is reported in more detail in [22]. The overall goal is to reduce the number of suppressed true alarms by deploying and adapting DFO.

11.4.1 Problem description

The focus of this experiment is detecting abnormalities in the heart function, called arrhythmias that can be encountered in both healthy and unhealthy subjects. The ICU is equipped with monitoring devices that are capable of detecting dangerous arrhythmias, namely asystole, extreme bradycardia, extreme tachycardia, ventricular tachycardia and ventricular flutter/fibrillation.

Depending on the Association for the Advancement of Medical Instrumentation (AAMI) guidelines, appropriate measures should be taken within 10 seconds of the commencement of the event as these arrhythmias might cause death [28]. Triggering the alarm when an arrhythmia occurs may improve the chance of saving lives. Mis-configuring, defective wiring, staff manipulation, and patient manipulation or movement may increase the false alarm ratio to 86%. Clinically, 6% to 40% of the ICU alarms proved to have lower priority and do not require immediate measures [29]. False alarms stimulate the patient's mental discomfort [30] and the clinical staff's desensitisation thereby causing a slower response to the triggered alarms [31]. True alarms that have high priority and need an urgent response are only 2 - 9% of all ICU alarms [32]. Therefore, false alarm detection and elimination is an essential area of research.

The Physionet 2015 challenge [33] provides a training data set containing 750 recordings that are available to the public, as well as 500 hidden records for scoring. This dataset consists of life-threatening arrhythmia alarm records that have been collected from four hospitals in the United States and Europe. The recordings are sourced from the devices made by three major manufacturing companies of intensive care monitor devices. Each record is five minutes or five minutes and 30 seconds long at 250Hz and contains only one alarm. A team of expert annotators labeled them 'true' or 'false'. The commencement of the event is within the last ten seconds of the recordings.

In this challenge, participants can submit their code which would be evaluated according to two type of events: event 1 (real time) and event 2 (retrospective). All recordings have a sample rate of 250Hz and contain two ECG leads and one or more pulsative waveform (RESP, ABP or PPG). The ECGs may contain noise, and pulsatile channels may contain movement artefacts and sensor disconnections. In this experiment, the focus is on event 1, where data is trimmed to have a length of five minutes and a subset of 572 records

TABLE 11.1
Subset of Physionet dataset (572 out of 750 recordings) that contains the ECG leads II and V and PLETH signal. This subset is used to ensure that the NN models are trained on identical leads and pulsatile waveform.

Disease Name	True Alarm	False Alarm
Asystole	17	77
Bradycardia	35	37
Tachycardia	90	4
Ventricular Flutter/Fibrillation	6	40
Ventricular Tachycardia	54	212

are chosen. These contain the ECG leads II and V and PLETH signal. This decision is made to ensure that the NN models are trained on identical leads and pulsatile waveform. In this subset, there are 233 True alarms and 339 False alarms. In each n-fold, the dataset is divided into training, testing and validation using 70% (400), 20% (114), and 10% (58) respectively. Table 11.1 describes the dataset.

11.4.2 Using dispersive flies optimisation

In this experiment the flies' positions, which are the weights associated with the neural network (NN), are termed $\vec{w}$. In the first iteration, $t = 0$, the i^{th} vector's d^{th} component is initialised as $w_{id}^0 = \mathcal{N}(0,\ 1)$, where $\mathcal{N}$ denotes the Gaussian distribution. Therefore, the population is randomly initialised with a set of weights for each in the search space.

In each iteration of the original DFO equation, the components of the NN weights vectors are independently updated, taking into account the component's value, the corresponding value of the best neighbouring individual with the best Physionet score (consider ring topology), and the value of the best individual in the whole swarm. Therefore the updated equation is:

$$w_{id}^{t+1} \quad = \quad w_{i_n d}^t + u(w_{sd}^t - w_{id}^t) \tag{11.11}$$

where $w_{i_n d}^t$ is the weight (position) value of $\vec{w}_i^t$'s best *neighbouring* individual in the d^{th} dimension at time step t, w_{sd}^t is the value of the *swarm*'s best individual in the d^{th} dimension at time step t, and $u = U(0, 1)$ is a random number generated from the uniform distribution between 0 and 1. In this experiment, few configuration changes are applied to DFO for deep network optimisation. The adapted update equation takes into account the corresponding value of the best neighbouring individual and the value of the best in the whole swarm:

$$w_{id}^{t+1} \quad = \quad w_{i_n d}^t + u(w_{sd}^t - w_{i_n d}^t) \tag{11.12}$$

In DFO, swarms disturbance is regulated by *the disturbance threshold*, Δ, to control the behaviour of the population (exploration or exploitation) in the search space. One of the impacts of these disturbances is the displacement

of the individuals, which may lead to discovering a better Physionet score through finding better weights for the NN. Based on this, the individual components of the population's weights' vectors are reset if a random number, u is less than Δ. This approach guarantees a disturbance to the otherwise permanent stagnation over a possible local maximum. In the original DFO equation, the disturbance is done by updating the parameter with a random number in the acceptable range of lower and upper bounds. In this experiment, a parameter's disturbance is correlated with the current Δ and the best neighbour; that is, w_{id}^{t+1} is sampled from a Gaussian with the mean set to $w_{i_n d}$ and variance to Δ^2.

Algorithm 15 summarises the adapted DFO algorithm and Figure 11.5 demonstrates Δ's behaviour in 1000 iterations.

Algorithm 15 Adapted DFO for Training.

Input: population size N, model structure L, network weights $\vec{w}_i$, length of weights vector D, loss function $f(.)$.

$\Delta = 1$
while not converged **do**
  $\vec{w}_s = \arg\max [f(L(\vec{w}_i))], \quad i \in \{1, \ldots, N\}$
  for $i = 1 \rightarrow N$ and $i \neq s$ **do**
    $\vec{w}_{i_n} = \arg\max [f(L(\vec{w}_{i-1})), f(L(\vec{w}_{i+1}))]$
    for $d = 1 \rightarrow$ D **do**
      if $(U(0,1) < \Delta)$ **then**
        $w_{id}^{t+1} \leftarrow \mathcal{N}(w_{i_n d}^t, \Delta^2))$
      else
        $w_{id}^{t+1} \leftarrow w_{i_n d}^t + u(w_{sd}^t - w_{i_n d}^t)$
      end if
    end for
    Dynamically update Δ (see Section 11.4.3.2)
  end for
end while

Output: Best fly's weight vector, $\vec{w}_s$.

11.4.3 Experiment setup

This part details the experiment setup comprising the model configuration and DFO configuration.

11.4.3.1 Model configuration

The experiment conducted here utilises two neural network architectures: a stack of dense layers, and a stack of convolutional and dense layers. The model structures are described in Tables 11.2 and 11.3 in detail. In NN, forward propagation includes a set of matrix operations where parameters include weights that connect NN layers to each other, and bias. The focus is on finding optimal weights of the network that provide the highest classification accuracy

TABLE 11.2
Dense Model Structure.

Layer Name	No of Neurons	Weights Shape	Total Weights	Bias
Dense 1 (Input)	64	(1, 64)	64	64
Dense 2	64	(64, 64)	4096	64
Dense 3	32	(2112, 32)	67584	32
Dense 4 (Output)	2	(32, 2)	64	2

TABLE 11.3
Convolution-Dense Model Structure.

Layer Name	No of Neurons	Weights Shape	Total Weights	Bias
Convolution 1 (Input)	32	(2, 1, 32)	64	32
Dense 2 (Input)	64	(32, 64)	2048	64
Dense 3	64	(64, 64)	4096	64
Dense 4	32	(1024, 32)	32768	32
Dense 5 (Output)	2	(32, 2)	64	2

for this task. Depending on the type of neurons and input shape in a NN, the shape of the output weights of that neuron varies. This study uses two models: model 1 consists of four dense layers, and model 2 one convolution and four dense layers. For instance, in model 1 of our study, the first layer is dense with 64 neurons. In the first layer, input shape is (-1, 33, 1) and the output weight is (1, 64) where 1 is the third axis of the layer's input shape, and 64 is the number of neurons in this layer: $[(-1, 33, 1) \times (1, 64)] + (64) = (-1, 33, 64)$

Model 2 includes a convolution as the first layer. This model has a similar input shape (-1, 33,1). The first layer has 32 neurons, and the shape of connecting weights to the next layer is (-1, 2, 1, 32) where 2 is the convolution window size, 1 is the third axis of the input shape, and 32 is the number of neurons in the convolution layer.

11.4.3.2 DFO configuration

In the experiment reported here, DFO is used to find the optimal weights in both NN models. The number of parameters (i.e. weights and bias) in models 1 and 2 are 71970 and 39234 respectively (see Tables 11.2 and 11.3). Each member of the population (fly) has a set of parameters representing the weights (including biases) of the NN model. Once all the parameters of each fly are initialised and loaded onto the NN model, the fitness (score) of each fly is calculated using 5-fold cross-validation, by using the mean Physionet score. After each iteration, each fly's best neighbour, and the best fly in the swarm are identified. The best fly holds the highest Physionet score amongst the population.

Before updating each weight, a value u is sampled from a uniform distribution $U(0, 1)$. If u is less than Δ, the weight is updated with the fly's best neighbour as focus, therefore, $w_{id}^{t+1} \leftarrow \mathcal{N}(w_{i_n d}^{t}, \Delta^2)$, otherwise, the fly's weight is updated with the focus on the best fly in the swarm $w_{id}^{t+1} \leftarrow w_{i_n d}^{t} + u(w_{sd}^{t} - w_{i_n d}^{t})$.

An additional mechanism is also proposed to control the value of Δ. In the first phase or the parameter optimisation, Δ is set to 1 to bias the algorithm towards exploration; this process continues until there is no improvement in k iterations. Following this, Δ is set to zero, allowing the flies to converge to the best location. Once again, if no improvement is noticed in the Physionet score, Δ is increased by a random number between 0 and δ[5] ($\Delta \leftarrow \Delta + U(0, \delta)$). The algorithm is then set to run, if there is an improvement followed by a k iteration state of idleness, Δ is set to 0 again to exploit the recent finding. Alternatively if there is no improvement after the allowed idle time-frame, Δ is incremented further. In a situation where $\Delta > 1$, Δ is set to $U(0, 1)$, and the process continues as explained above until the termination points, which is $3,000$ iterations. Figures 11.3, 11.4 and 11.5 demonstrate the trend of improvement in accuracy, Physionet score and variations of Δ over the first 1000 iterations.

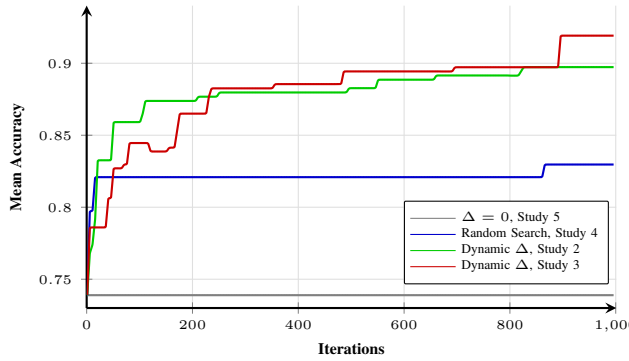

FIGURE 11.3
Model 1: 5-fold cross validation mean accuracy over 1000 iterations. Mean accuracy trend for random search, standard DFO, updated DFO, with dynamic and constant disturbance threshold (Δ).

11.4.4 Results

The result of this experiment, using DFO as a NN optimiser, is compared against the winning and 5$^{\text{th}}$ ranked entries in the Physionet challenge 2015 (Table 11.4), as well as the same NN architectures optimised with BP instead of the proposed gradient-free DFO training scheme.

As a benchmark, the accuracy and Physionet scores of [34] and [35] are used; they achieve (87.24%, 85.50%) and (87.78%, 80.09%) respectively. Various implementations of DFO, as well as random search are also investigated. For all experiments, 5-fold cross-validation is used. The DFO implementa-

[5]Note that throughout this in this paper, we use $k = 50$ and $\delta = 0.5$.

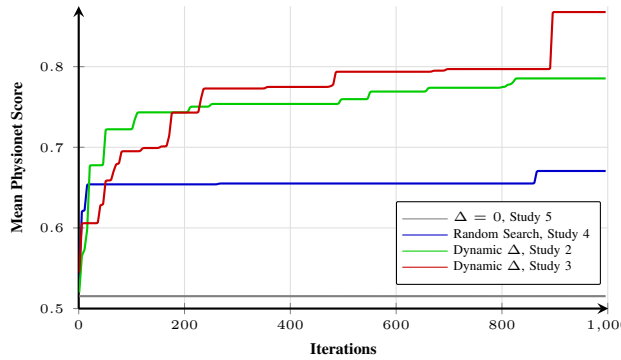

FIGURE 11.4
Model 1: 5-fold cross validation mean Physionet score over 1000 iterations. Mean Physionet score trend for random search, standard DFO, updated DFO, with dynamic and constant disturbance threshold (Δ).

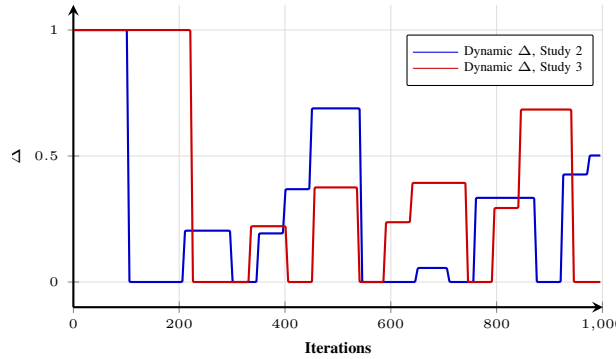

FIGURE 11.5
Model 1: 5-fold cross validation disturbance threshold (Δ) trend. Visualising Δ trend with considering dynamic and constant disturbance threshold (Δ) over 1000 iterations.

tion for Model 1 and 2 achieved the highest score among the other results (see Table 11.4). The resulting accuracy and Physionet scores are (91.91%, 86.77%) and (91.88%, 86.81%) respectively, therefore achieving better results than both back propagation-trained NN, as well as the challenge entries.

The behaviour of DFO, while having the constant Δ value of 0 and random search, is also investigated. Their accuracy and Physionet scores are (73.89%, 51.53%) and (84.16, 68.03) respectively. The models optimised via neuroevolution with DFO outperforming both (i) the networks trained by BP, as well as (ii) the winning entry of the Physionet challenge.

TABLE 11.4
Accuracy and Physionet score over 5-fold cross validation for first [34] and fifth rank [35] in Physionet challenge 2015, NN optimised with BP and adapted DFO algorithm with constant and dynamic disturbance threshold (Δ).

Author	**Method**	**Mean Accuracy**	**Physionet 2015 Score**
By [35]	SVM, BCTs, DACs	87.24% (+/- 2)	85.50% (+/- 3)
By [34]	Fuzzy Logic	87.78% (+/- 4)	80.09% (+/- 8)
Our Study 1	Dense Network, BP	87.70% (+/- 3)	75.35% (+/- 7)
Our Study 2	Dense NN, Standard DFO, Dynamic Δ, using Eq. 11.11	90.02% (+/- 3)	79.23% (+/- 5)
Our Study 3	**Dense NN, Adapted DFO, Dynamic Δ, using Eq. 11.12**	**91.91% (+/- 4)**	**86.77% (+/- 4)**
Our Study 4	Dense NN, Random Search	84.16% (+/- 3)	68.03% (+/- 5)
Our Study 5	Dense NN, Standard DFO, $\Delta = 0$ (i.e. no disturbance)	73.89% (+/- 2)	51.53% (+/- 4)
Our Study 6	Convolution-Dense NN, BP	88.21% (+/- 3)	76.34% (+/- 6)
Our Study 7	Convolution-Dense NN, Standard DFO, Dynamic Δ, using Eq. 11.11	89.15% (+/- 4)	77.21% (+/- 5)
Our Study 8	**Convolution-Dense NN, Adapted DFO, Dynamic Δ using Eq. 11.12**	**91.88% (+/- 2)**	**86.81% (+/- 4)**
Our Study 9	Convolution-Dense NN, Random Search	82.39% (+/- 5)	66.40% (+/- 7)
Our Study 10	Convolution-Dense NN, Standard DFO, $\Delta = 0$ (i.e. no disturbance)	73.90% (+/- 3)	52.53% (+/- 2)

11.5 Conclusions

This paper presents the standard DFO along with some of the existing and potential modifications to the algorithm: mainly the update equation, as well as its disturbance mechanism. These modifications are proposed in order to achieve a balance between exploration and exploitation at various stages of the optimisation process. The paper also presents an application of DFO in optimising a neural network which is applied real-world data from a Physionet challenge with promising results, outperforming both the classical backpropagation method as well as the winning entries of the corresponding Physionet challenge.

References

1. I.C. Trelea. "The particle swarm optimization algorithm: convergence analysis and parameter selection". Information Processing Letters, vol. 85(6), pp. 317-325, 2003.
2. O. Olorunda, A.P. Engelbrecht. "Measuring exploration/exploitation in particle swarms using swarm diversity" in *Proc. of IEEE Congress on Evolutionary Computation, CEC 2008*, pp. 1128-1134, 2008.

3. M.M. al-Rifaie. "Dispersive Flies Optimisation" in *Proc. of the 2014 Federated Conference on Computer Science and Information Systems*, vol. 2, pp. 529-538, 2014.
4. J. Kennedy. "The particle swarm: social adaptation of knowledge" in *Proc. of IEEE International Conference on Evolutionary Computation*, pp. 303-308, 1997.
5. R. Storn, K. Price. "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces". Journal of Global Optimization, vol. 11(4), pp. 341-359, 1997.
6. M. Dorigo, G.D. Caro, L.M. Gambardella. "Ant algorithms for discrete optimization". Artificial Life, vol. 5(2), pp. 137-172, 1999.
7. M.M. al-Rifaie. "Perceived Simplicity and Complexity in Nature" in AISB 2017: Computational Architectures for Animal Cognition, pp. 299-305, 2017.
8. T. Back, D.B. Fogel, Z. Michalewicz. *Handbook of Evolutionary Computation*. IOP Publishing Ltd. 1997.
9. X.-S. Yang. "Firefly algorithms for multimodal optimization" in *Proc. of International Symposium on Stochastic Algorithms*, pp. 169-178, 2009.
10. J. Kennedy. "Bare Bones Particle Swarms" in *Proc. of Swarm Intelligence Symposium, 2003 (SIS'03)*, pp. 80-87, 2003.
11. T. Blackwell. "A study of collapse in Bare Bones Particle Swarm Optimisation". IEEE Transactions on Evolutionary Computing, vol. 16(3), pp. 354-372, 2012.
12. M.G.H. Omran, A.P. Engelbrecht, A. Salman. "Bare bones differential evolution". European Journal of Operational Research, vol. 196(1), pp. 128-139, 2009.
13. M.M. al-Rifaie, T. Blackwell. "Bare Bones Particle Swarms with jumps". Lecture Notes in Computer Science, vol. 7461, pp. 49-60, 2012.
14. R.A. Krohling. "Gaussian particle swarm with jumps" in *Proc. of The 2005 IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1226-1231, 2005.
15. M.M. al-Rifaie, T. Blackwell. "Cognitive Bare Bones Particle Swarm Optimisation with jumps". International Journal of Swarm Intelligence Research (IJSIR), vol. 7(1), pp. 1-31, 2016.
16. F. van den Bergh. "An Analysis of Particle Swarm Optimizers". PhD Thesis, University of Pretoria, South Africa, 2002.
17. F. van den Bergh, A.P. Engelbrecht. "A study of particle swarm optimization particle trajectories". Information Sciences, vol. 176(8), pp. 937-971, 2006.

18. M. Clerc, J. Kennedy. "The particle swarm-explosion, stability, and convergence in amultidimensional complex space". IEEE Transactions on Evolutionary Computation, vol. 6(1), pp. 58-73, 2002.
19. M.M. al-Rifaie, A. Aber. "Dispersive Flies Optimisation and Medical Imaging" in Recent Advances in Computational Optimization, Studies in Computational Intelligence book series (SCI, volume 610), pp. 183-203, Springer, 2016.
20. H. Alhakbani. "Handling Class Imbalance Using Swarm Intelligence Techniques, Hybrid Data and Algorithmic Level Solutions". PhD Thesis, Goldsmiths, University of London, London, United Kingdom, 2018.
21. H.A. Alhakbani, M.M. al-Rifaie. "Optimising SVM to classify imbalanced data using dispersive flies" in *Proc. of the 2017 Federated Conference on Computer Science and Information Systems, FedCSIS 2017*, pp. 399-402, 2017.
22. H. Oroojeni, M.M. al-Rifaie, M.A. Nicolaou. "Deep neuroevolution: training deep neural networks for false alarm detection in intensive care units" in *Proc. of European Association for Signal Processing (EUSIPCO) 2018*, pp. 1157-1161, 2018.
23. M.M. al-Rifaie, A. Ursyn, R. Zimmer, M.A.J. Javid. "On symmetry, aesthetics and quantifying symmetrical complexity" in *Proc. of International Conference on Evolutionary and Biologically Inspired Music and Art*, pp. 17-32, 2017.
24. M. King, M.M. al-Rifaie. "Building simple non-identical organic structures with dispersive flies optimisation and A* path-finding" in *Proc. of AISB 2017: Games and AI*, pp. 336-340, 2017.
25. P. Aparajeya, F.F. Leymarie, M.M. al-Rifaie. "Swarm-Based Identification of Animation Key Points from 2D-medialness Maps" in Computational Intelligence in Music, Sound, Art and Design, pp. 69-83, Springer, 2019.
26. M.M. al-Rifaie F.F. Leymarie, W. Latham, M. Bishop. "Swarmic autopoiesis and computational creativity". Connection Science, pp. 1-19, 2017.
27. F.P. Such, V. Madhavan, E. Conti, J. Lehman, K.O. Stanley, J. Clune. "Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning". arXiv preprint arXiv:1712.06567, 2017.
28. Association for the Advancement of Medical Instrumentation and others. "Cardiac monitors, heart rate meters, and alarms". American National Standard (ANSI/AAMI EC13: 2002) Arlington, VA, pp. 1-87, 2002.

29. S.T. Lawless. "Crying wolf: false alarms in a pediatric intensive care unit". Critical Care Medicine, vol. 22(6), pp. 981-985, 1994.

30. S. Parthasarathy, M.J. Tobin. "Sleep in the intensive care unit". Intensive Care Medicine, vol 30(2), pp. 197-206, 2004.

31. M.-C. Chambrin. "Alarms in the intensive care unit: how can the number of false alarms be reduced?" Critical Care, vol. 5(4), pp. 184, 2001.

32. C.L. Tsien, J.C. Fackler. "Poor prognosis for existing monitors in the intensive care unit". Critical Care Medicine, vol. 25(4), pp. 614-619, 1997.

33. G.D. Clifford, I. Silva, B. Moody, Q. Li, D. Kella, A. Shahin, T. Kooistra, D. Perry, R.G. Mark. "The PhysioNet/Computing in Cardiology Challenge 2015: reducing false arrhythmia alarms in the ICU" in *Proc. of Computing in Cardiology Conference (CinC)*, pp. 273-276, 2015.

34. F. Plesinger, P. Klimes, J. Halamek, P. Jurak. "False alarms in intensive care unit monitors: detection of life-threatening arrhythmias using elementary algebra, descriptive statistics and fuzzy logic" in *Proc. of Computing in Cardiology Conference (CinC)*, pp. 281-284, 2015.

35. C.H. Antink, S. Leonhardt. "Reducing false arrhythmia alarms using robust interval estimation and machine learning" in *Proc. of Computing in Cardiology Conference (CinC)*, pp. 285-288, 2015.