Perceived Simplicity and Complexity in Nature

Mohammad Majid al-Rifaie [1]

Abstract. The world of swarm intelligence and evolutionary computation has been witnessing an "explosion" in the number of new algorithms proposed, sometimes with noticeable contributions and at times very little to offer. These algorithms are often described as being nature-inspired, the validity of which should be judged on a one-to-one basis. In this work, an existing swarm intelligence algorithm, Dispersive Flies Optimisation or DFO, is presented as a case study along with some insights and structure comparison with some other swarm and evolutionary techniques, such as: particle swarm optimisation, genetic algorithm, evolutionary computation, ant colony optimisation, stochastic diffusion search and firefly algorithms. Despite recent attacks, this paper aims to shed light on the significance of nature-inspired observations and their contributions to the scientific communities of swarm intelligence and evolutionary computation, emphasising on the importance of offering sufficient analyses which justify the reason behind proposing new metaheuristics.

1 Background

There are many optimisation problems surrounding us in our everyday activities, ranging from negligible and trivial to vital and more complex. These problems have been often addressed using population based swarm intelligence and evolutionary computation techniques. Nature has been one of the main sources of inspiration for the creation of techniques applicable to complex search spaces and optimisation problems.

Arguably, throughout the history of mankind, nature has never stopped offering, at times, unsolicited inspirations to scientists, researchers and the curious. Observations, many of which made unintentionally, have been triggering the inquisitive minds for thousands of years. The drive for resolving problems and their often present state in the minds of scientists boost 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'.

In the worlds of swarm intelligence and evolutionary computation, 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, Particle Swarm Optimisation and Ant Colony Optimisation are only few such techniques belonging to the broader category of swarm intelligence and evolutionary computation; these processes investigate collective intelligence and aim at modelling intelligence by looking at individuals in a social context and monitoring their interactions.

[1] Goldsmiths, University of London, London SE14 6NW, United Kingdom. Email: m.majid@gold.ac.uk

A multi-agent approach to artificial intelligence was born in the 90s, when cooperation between agents became essential to having an *emerging* intelligence resulting from the interaction of a group of individuals [29]. It was time for sociologists, biologists and anthropologists to play their rules in helping AI with their models and social views on intelligence [43].

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 and evolutionary computation techniques are no exceptions in this regard; many of the most well known evolutionary and swarm intelligence algorithms exhibit clear links to nature. Natural examples of swarm intelligence that demonstrate 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 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 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 interact amongst themselves and the second one is the interaction of the individuals with the environment [14] (stigmergy). Interaction amongst 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, interactions help benefiting from discoveries and previous experience of all other members of the school or flock during search for food [41]. There are different forms of recruitment in social insects; it may take the form of local or global; one-to-one or one-to-many; and deploy stochastic or deterministic mechanisms. The nature of information exchange also varies in different environments and with different types of social insects and animals. Sometimes the information exchange is more complex where, for example, it might carry data about the direction, suitability of the target and the distance; or sometimes the information sharing is simply a stimulation forcing a

certain triggered action. What all these recruitment and information exchange strategies have in common is distributing useful information in their community.

In this paper, initially a simple swarm intelligence algorithm, Dispersive Flies Optimisation is presented, followed by six other swarm intelligence and evolutionary computation techniques. The aim is to emphasise the extent in which these algorithms rely on their inventors' observations in nature as well as a discussion on the simplicity and complexity of the algorithms' nature-inspired structure, based on the number of tunable parameters and other configurations.

2 Dispersive Flies Optimisation

Flies are insects of the order *Diptera*, which comprises a large order, containing an estimated $240,000$ species of mosquitoes, gnats, midges and others [40]. 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 [17] 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 [9]. Height change in mosquito swarm induced by a clarinet note [31] and the human voice [27] may have evolved as responses to the flight tone of female mosquitoes [33].

Swarms of flies are associated with visual markers ranging in size from cowpies and stones to church steeples [37]. The criteria used by insects to select markers may be quite subtle; it was noted in [26] that certain objects are used repeatedly by the mosquito *Aedes cataphylla* while similar objects nearby are neglected.

As explained in [19], 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. [18, 13]). 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.

Dispersive Flies Optimisation (DFO) – first introduced in [1] – is an algorithm inspired by the swarming behaviour of flies hovering over food sources. The swarming behaviour of flies is determined by several factors including the presence of threat which disturbs their convergence on the marker (or the 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 DFO consists 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:

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

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

In the first generation, when $t = 0$, the i^{th} vector's d^{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 with the best fitness (consider ring topology), and the value of the best fly in the whole swarm:

$$x_{id}^t \quad = \quad x_{i_{nd}}^{t-1} + \mathrm{U}(0,1) \times (x_{sd}^{t-1} - x_{id}^{t-1}) \tag{3}$$

where $x_{i_{nd}}^{t-1}$ is the position value of $\vec{x}_i^{t-1}$'s best *n*eighbouring fly in the d^{th} dimension at time step $t-1$; x_{sd}^{t-1} is the value of the *s*warm's best 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 main 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 impacts of such disturbances is the displacement of flies 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 components of flies' position vectors are reset if a 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 disturbance to the otherwise permanent stagnation over a likely local minima. Algorithm 1 summarises the DFO algorithm[2].

Algorithm 1 Dispersive Flies Optimisation

while Function Evalutions $<$ Evaluations Allowed **do**
  for $i = 1 \rightarrow N$ **do**
    $\vec{x}_i$.fitness $\leftarrow f(\vec{x}_i)$
  end for
  $\vec{x}_s \ = \arg^*\min\left[f(\vec{x}_i)\right]$
  $\vec{x}_{i_n} = \arg^*\min\left[f(\vec{x}_{i_{\text{left}}}), f(\vec{x}_{i_{\text{right}}})\right]$*
  for $i = 1 \rightarrow N$ **do**
    for $d = 1 \rightarrow \mathrm{D}$ **do**
      $\tau_d \leftarrow x_{i_{nd}}^{t-1} + \mathrm{U}(0,1) \times (x_{sd}^{t-1} - x_{id}^{t-1})$
      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

* $\vec{x}_{i_{\text{left}}} = \vec{x}_{i-1}$ and $\vec{x}_{i_{\text{right}}} = \vec{x}_{i+1}$

In summary, DFO is a simple numerical optimiser over continuous search spaces. DFO is a population based stochastic algorithm,

[2] The source code can be downloaded from the following page: `http://doc.gold.ac.uk/mohammad/DFO/`

originally proposed to search for an optimum value in the feasible solution space. Despite the algorithm's simplicity, it is shown that DFO outperforms the standard versions of the well-known Particle Swarm Optimisation, Genetic Algorithm (GA) as well as Differential Evolution (DE) algorithms on an extended set of benchmarks over three performance measures of error, efficiency and reliability [1]. It is shown that DFO is more efficient in 84.62% and more reliable in 90% of the 28 standard optimisation benchmarks used; furthermore, when there exists a statistically significant difference, DFO converges to better solutions in 71.05% of problem set. Further analysis is also conducted to explore the diversity of the algorithm throughout the optimisation process, a measure that potentially provide more understanding on algorithm's ability to escape local minima. In addition to theoretical research on this algorithm, DFO has recently been applied to medical imaging [3]; furthermore, ongoing and current research are being conducted in the fields of image analysis, simulation and gaming [25], computational aesthetic measurements [2], (digital) arts [6, 10], protein folding, etc.

3 Population-based algorithms

In this section, six population-based nature-inspired algorithms are presented along with parameters and the associated configuration of each of the these algorithms. The presented algorithms are: particle swarm optimisation (PSO), differential evolution algorithm (DE), genetic algorithm (GA), ant colony optimisation (ACO), stochastic diffusion search (SDS) and firefly algorithm (FA).

3.1 Particle Swarm Optimisation

Particle swarm optimisation (PSO) is a population based optimisation technique developed in 1995 by Kennedy and Eberhart [24, 20]. 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 [21]. In particle swarms, although members of the swarm neither have knowledge about the global behaviour of the swarm nor a global information about the environment, the local interactions of the swarms result in complex collective behaviour, such as flocking, herding, schooling, exploration and foraging behaviour [32, 30, 8, 22].

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 ($\vec{p}$). Whereas the best position so far found throughout the population, or the local neighbourhood, is called neighbourhood best (or global best, $\vec{g}$).

The standard PSO algorithm defines the position of each particle by adding a velocity to the current position. Here is the equation for updating the velocity of each particle:

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

$$x_{id}^{t} = v_{id}^{t} + x_{id}^{t-1} \quad (5)$$

where w is the inertia weight whose optimal value is problem dependent [34]; $\vec{v}_{id}^{t-1}$ 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 $\vec{x}_i$ in dimension d; and g_{id} is neighbourhood best at dimension d.
Therefore, PSO optimisation is based on particles' individual experience and their social interaction with the particle swarms. After updating particles' velocities, their new positions are determined.

3.2 Genetic Algorithm

In this work, a real-valued GA which has previously shown to work well on real-world problems [38, 39] in explained. The underlying process is similar to when GA is applied to combinatorial problems, where 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. One such recombination operator is arithmetic crossover and one such mutation operator is Cauchy mutation using an annealing scheme. At the end, in order to comb out the least fit individual, tournament selection [7] is utilised.

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

$$\text{offspring}_i = r \times \text{parent1}_i + (1-r) \times \text{parent2}_i \quad (6)$$

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 $U(0,1)$.

Therefore, the algorithm needs the probabilities of crossover and mutation of the individuals, p_c and p_m respectively, the tournament size of the tournament selection, and elitism with a certain elite size is deployed to maintain the best found solution in the population.

3.3 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 [35, 36].

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:

$$x_i^g = \left[x_{i,1}^g, x_{i,2}^g, ..., x_{i,D}^g\right], i = 1, 2, ..., N \quad (7)$$

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

$$x_{i,j}^0 = x_{min,j} + r\left(x_{max,j} - x_{min,j}\right) \quad (8)$$

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 j^{th} dimension, respectively. The evolutionary process (mutation, crossover and selection) starts after the initialisation of the population.

3.3.1 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, out of many variants, *DE/best/1* variation of mutation approaches is provided as a sample:

$$v_i^g = x_{best}^g + F\left(x_{r_1}^g - x_{r_2}^g\right) \tag{9}$$

where r_1 and r_2 are different from i and are distinct random integers drawn from the range $[1, N]$; 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.

3.3.2 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:

$$u_{i,j}^g = \begin{cases} v_{i,j}^g, & \text{if } \ r \leq p_c \ \text{ or } j = r_d \\ \\ x_{i,j}^g, & \text{otherwise} \end{cases} \tag{10}$$

where r is a uniformly distributed random number drawn from the unit interval $U\left(0, 1\right)$, 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 p_c, which is another control parameter, specifies the level of inheritance from v_i^g (mutant vector).

3.3.3 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:

$$x_i^{g+1} = \begin{cases} u_i^g, & \text{if } \ f\left(u_i^g\right) \leq f\left(x_i^g\right) \\ \\ x_i^g, & \text{otherwise} \end{cases} \tag{11}$$

where $f\left(x\right)$ is the fitness function.

3.4 Ant Colony Optimisation

Another well-known swarm intelligence algorithm is Ant Colony Optimisation (ACO) [16], which is a metaheuristic for solving combinatorial optimisation problems. The driving inspiration for this algorithm stems from the pheromone trail laying and following behaviour of real ants using pheromones as a mean for communication. This algorithm is one of the existing techniques in the literature which uses the indirect mode of communication, where no two ants exchange information directly, instead they use pheromone trails to guide the process and thus probabilistically construct solutions for the problem being solved. In other words, the artificial ants' search experience is shaped and evolved through the encounter with the pheromone trails.

Due to the popularity of the algorithm (and as it is often the case with swarm intelligence and evolutionary techniques) there have been several flavours of algorithms starting from the early 90s, including but not limited to: Ant System, Elitist AS, Ant-Q, ANTS, Population-based ACO, Beam-ACO, and so forth.

As detailed in [15], ACO's artificial ants are described to be stochastic solution construction procedures which are in charge of probabilistically building a solution by iteratively adding solution components to partial solutions. This process is facilitated by taking into account the following:

- heuristic information about the problem being addressed, if such information exists
- and the dynamically changing pheromone trails which reflect the ants' search experience

While the heuristic information / equation is problem dependant, here are some of the generic parameters of the algorithm when dealing with the classical Travelling Salesman Problem (TSP). Note that the following reflects on one instance of using an ACO algorithm on a certain problem:

- $T(r, s)$: the mount of pheromone on the edge connecting r to s.
- $H(r, s)$: heuristic value of the edge.
  In TSP, the shorter the distance between the nodes, the higher the heuristics. Therefore, for instance: $H(r, s) = \frac{1}{\text{distance}(r,s)}$
- $p_k(r, s)$: probability that ant, k, travels from node r to node s.
- β: heuristic strength
- α: greediness

$$p_k(r, s) = \frac{T(r, s)^\alpha \times H(r, s)^\beta}{\sum_{\text{Unvisited cities c}} T(r, c)^\alpha \times H(r, c)^\beta}$$

Using the global pheromone update, once a tour is complete, pheromones are updated along the edges using the following:

- $A_k(r, s)$: the mount of pheromone added by ant k on the edge connecting r to s.
- m: number of ants (population size)
- ρ: pheromone decay rate
- L_k: length of route complete by ant k

The amount of pheromone on the edge (r,s) is calculated using the following equation:

$$T(r, s) = \rho \times T(r, s) + \sum_{k=1}^{m} A_k(r, s)$$

where $A_k(r, s) = 1/L_k$

3.5 Stochastic Diffusion Search

This section introduces Stochastic Diffusion Search (SDS) [11], a swarm intelligence algorithm whose performance is based on simple interaction of agents. The SDS algorithm commences a search or optimisation by initialising its population and then iterating through two phases of test and diffusion.

In the test phase, SDS checks whether the agent hypothesis is successful or not by performing a hypothesis evaluation which returns a Boolean value. Later in the iteration, contingent on the precise recruitment strategy employed (in the diffusion phase), successful hypotheses diffuse across the population and in this way information on potentially good solutions spreads throughout the entire population of agents. In other words, each agent recruits another agent for interaction and potential communication of hypothesis.

In standard SDS, *passive recruitment mode* is employed. In this mode, if the agent is inactive, a second agent is randomly selected

for diffusion; if the second agent is active, its hypothesis is communicated (*diffused*) to the inactive one. Otherwise there is no flow of information between agents, instead a completely new hypothesis is generated for the first inactive agent at random. Therefore, recruitment is not the responsibility of the active agents.

The activity of agents are often determined by specifying a threshold for the fitness of the agents, based on which agents are labelled as active or inaction. Alternatively, in another approach, the activity of each agent is decided when its fitness is compared against a random agent (which is different from the selecting one); if the selecting agent has a better fitness (e.g. smaller value in minimisation problems) than the randomly selected agent, it will be flagged as active, otherwise inactive. Higher rate of inactivity boosts exploration, whereas a lower rate biases the performance towards exploitation. SDS algorithm has been explained using the 'mining game' metaphor in [4].

One of the main and inherent processes in SDS is partial function evaluation. This feature is helpful with many fitness functions which are decomposable to components that can be evaluated separately during the test phase of SDS. In partial function evaluation pFE is some function of the agent's hypothesis, where $pFE = f(h)$, and the evaluation of one or more of the components may provide partial information to guide the subsequent optimisation process.

3.6 Firefly algorithm

Firefly algorithm, first introduced in 2008 [44], is another new swarm intelligence algorithm which takes its inspiration from nature, and is more specifically based on the "idealised" flashing characteristics of the fireflies. The behaviour of the algorithm can be explained using the following assumptions derived from nature:

- fireflies are unisex and are attracted to each other
- attractiveness is proportional to brightness; therefore a less bright firefly would be attracted to and subsequently move towards the brighter one. If the brightness is the same, the fireflies move randomly. If the distance between fireflies increases, the brightness decreases which results in less attraction.
- the landscape of the objective function impacts the brightness of the fireflies

In FA, brightness is determined based on the fitness function, however attractiveness β is considered relative to the distance between two fireflies and the absorption of the light in the media. The light intensity I is determined using the following:

$$I = I_0 e^{-\gamma r} \tag{12}$$

where I_0 represents the light intensity at the starting point, γ is the light absorption coefficient, and r is the distance. And the attractiveness of a firefly is determined by the following:

$$\beta = \beta_0 e^{-\gamma r^2} \tag{13}$$

where β_0 is the attractiveness at $r = 0$.

Using FA, each firefly i is paired with every other firefly, j, in the population and a decision is made as to whether to move i towards j and how far. The distance between fireflies are calculated using the Cartesian distance $r_{ij} = ||\vec{x}_i - \vec{x}_j||_2$, considering ℓ_2, and the following update equation:

$$\vec{x}_i = \vec{x}_i + \beta_0 e^{-\gamma r^m} (\vec{x}_j - \vec{x}_i) + \alpha \vec{\epsilon}_i \tag{14}$$

where the second term represents attraction (while m is usually set to 2, it could take other values with $m > 0$), and the third term is randomisation with the vector of random variable $\vec{\epsilon}_i$ being drawn from a Gaussian distribution.

4 Discussion and Opinion

Looking at the algorithms presented in the sections above, communication is seen as a key element shared by all population-based algorithms with varying flavours. Communication, at times one-to-one, one-to-many or many-to-many, mostly adapts to the problem domain and the related restrictions. The aim of the communication in the algorithms is the diffusion of 'good' information and therefore the investigation of promising areas. Given the unknown search space, and not to mention noisy environment, this task cannot be achieved without a degree of 'stochasticity' which allows for extra exploration of perhaps previously 'unnoticed' regions of the search space. Therefore, while communication is at the heart of population-based nature inspired algorithms, acknowledging the lack of "complete knowledge" about the search space, it becomes evident that 'guided communication' or a deterministic approach cannot be the only method.

Most of the deterministic algorithms are local searches which are efficient at converging to a local minima and given a fixed starting point, the solution returned is always the same. Despite the efficiency of such algorithms (and their good performance when dealing with unimodal problems), they are prone to being trapped in local minima, incapable of escaping towards global optima. Therefore, identifying the right balance between *guided* and *stochastic* communication is one of the primary challenges in introducing new algorithms or fine-tuning the existing ones.

Looking from a different perspective, a common aspect of population-based nature-inspired computation is adjusting the degree of "trust" at various stages in which the algorithms return solutions. This trust impacts the algorithms' follow-up resource allocation as to whether to be biased towards detailed exploitation of the currently found solution/s or instead boosting the exploration of the search space with the hope of identifying better solutions. Albeit different for various nature-inspired algorithms, the infamous trade-off between exploration and exploitation remains to be a key challenge.

Therefore, all the population-based algorithms are proposed with the goal of "striking" the right balance in defining the interwoven concepts of guided/stochastic communications and identifying the degree of trust in the solutions found at various stages of the optimisation process. With these in mind, researchers embark on a journey to find the "holy grail" which, with an optimistic view, would ultimately exhibit the ability to address a certain class of problems and not all (see No Free Lunch Theorem for a detailed discussion on this matter [42]). In this endeavour, researchers and scientists seek help from nature, which many view as a "stable" form of intelligence that has been evolving and surviving for millions of years.

An important question raised here is whether introducing a new algorithm which is not radically dissimilar to other algorithms contribute to the existing literature in swarm and evolutionary computation. There is a number of active researchers who are renouncing this trend as "threatening" and "away from scientific rigour". There are many reasons, and sometimes, valid arguments to disseminate this message in the community that merely "tweaking" some parameters and achieving an ever so slightly better performance on a certain set of problems could hardy be seen as an outstanding contribution. On the other hand, merely recommending to "put a lid" on the nature-inspired observations is an equally questionable argument.

In order to discuss this further, it is worth mentioning that one of the common features noticed in the development of algorithms, in general, and swarm intelligence and evolutionary computation algorithms, in particular, is the presence of many variants of the same techniques. The reason behind this persistent pattern could be associated with the flexibility of the algorithms in changing the update equations (i.e. altering the deterministic aspects of the algorithm), or injecting further stochasticity by introducing more complex disturbance mechanism or evolutionary operators. While the possibility of "extending" and "revising" an algorithm is welcome, it is equally important to be able to provide concrete analyses as to why the "new versions" are important contributions to the literature beyond the results on some "standard" benchmarks.

Having the above arguments in mind, it is equally vital to allow and respect scientists, researchers and students to observe the nature, animals and insects with the goal of finding ways to deal with the above challenges. It is important to encourage researchers to provide analysis as to why their proposed algorithms outperform or compete with others with the goal of identifying insights to the complex behaviour of the population in swarm and evolutionary computation techniques. The complexity of the algorithms could perhaps lead to better performance on some problems at the expense of even more complex task of understanding the behaviour of the algorithms. Often the number of components (e.g. various vectors) in the algorithms and the tunable parameters are good indicators of an algorithm's complexity. This inherent complexity results in both the hindrance of analysing the behaviour of the agents as well as the difficulty in finding the optimal parameter values.

Following on the above discussion, one of the principal reasons behind proposing Dispersive Flies Optimisation (DFO) has been introducing an algorithm that is simple but competitive enough compared to other more complex algorithms. Further theoretical analysis is ongoing to provide detailed explanation on the algorithm's optimisation behaviour. Other than the population size, DFO has one tunable parameter (disturbance threshold or dt). Additionally, other than the index of the best fly (s in $\vec{x}_s$), each fly i consists of merely its position vector ($\vec{x}_i$) along with the index of the best neighbouring fly (n in $\vec{x}_{i_n}$). To emphasis DFO's simplicity, a comparison is provided with some of the algorithms mentioned in this work. For instance, PSO, in many of the proposed variants commonly uses the following parameters:

- population size
- c_1, controlling the impact of cognitive component
- c_2, controlling the impact of social component
- χ or w, depending on the update equation

In addition to the position of the particles i, $\vec{x}_i$, each particle has an associated velocity $\vec{v}_i$ and memory, $\vec{p}_i$, vectors. Other variants of PSO, including barebone PSOs are also introduced by researchers, trying to simply the algorithm, with the goal of offering insight into the underlying behaviour of the algorithm. In one such cases, one of the inventor of the PSO algorithm, Kennedy, describes the process as "*strip[ing] away some traditional features*" with the hope of *revealing the mysteries of the algorithm* [23]. In this particular model the velocity vector is elegantly removed with the algorithm still benefiting from the memory vectors, a work that indeed shed light on the behaviour of the algorithm. Other contributions have tried to further explore the simplified version and add more to enhance its performance demonstrating the capability of the simplified version in contracts with the original model [12, 28, 5].

Other than PSO, the parameters and adjustable configurations of other aforementioned well-known algorithms are listed below.

- GA
  1. population size
  2. p_c: crossover rate
  3. p_m: mutation rate
  4. tournament size
  5. elite size
- DE
  - population size
  - p_c: crossover rate
  - equation used to calculate the mutation vector
  - F: constricting factor
- ACO
  - m: number of ants (population size)
  - β: heuristic strength
  - α: greediness
  - ρ: pheromone decay rate
- SDS [3]
  - population size
  - threshold for determining an agent's activity
  - the choice in using various recruitment modes (passive, active, dual)
  - the choice of increasing exploration by using context-sensitive or context-free mechanisms
  - the percentage of decomposable function using in partial function evaluation
- FA
  - population size
  - m: impact of distance on attractiveness
  - α which could be replaced with αS_k in cases where scales vary significantly in different dimensions, d. Therefore, given d dimensions ($k = 1, ..., d$), adding d extra parameters
  - γ determining the speed of convergence, in theory $\gamma \in [0, \infty)$ with $\gamma = 0$ maintaining a constant attractiveness of $\beta = \beta_0$

Similar to other swarm intelligence and evolutionary computation techniques, the above algorithms couple the stochasticity and determinism to both follow the "concrete instructions" of the algorithm as to where a good solution is likely to be, and a stochastic component, making sure that the often large search space is not left unexplored.

Therefore, while it is useful to propose powerful algorithms, with the potential drawback of analysis complexity, it is equally important to notice the elegance witnessed in nature, in social animals and insects. Key questions to ask here are: whether proposing simple algorithms with few tunable parameters, paves the way for more comprehensive analyses (thus a deeper understanding) and whether this would guide us closer to the sought after concept of *"communication of agents with simple rules leading to the complex emergence of intelligence"*.

[3] It is important to note that the parameters required for the "standard" SDS are the population size and the threshold for determining the activity of the agents.

5 Conclusion

In this paper, several swarm intelligence and evolutionary algorithms are presented and the process through which they lead the optimisation is explained. Dispersive Flies Optimisation (DFO) is presented as a case study representing a simple algorithm which other than the population size, only has one tunable parameter providing the means for the flies to further their exploration of the search space.

Discussing the pros and cons of introducing new algorithms, this work highlights the need for giving importance to *diligent* nature-inspired observations which could lead to the enrichment of swarm intelligence and evolutionary computation communities with the provisos that such observations allow in-depth analyses leading to better understanding of the complex formation of solutions and the "emergence of intelligence" by following simple rules.

REFERENCES

[1] Mohammad Majid al-Rifaie, 'Dispersive flies optimisation', in *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems*, ed., M. Paprzycki M. Ganzha, L. Maciaszek, volume 2 of *Annals of Computer Science and Information Systems*, pp. pages 529–538. IEEE, (2014).

[2] Mohammad Majid al-Rifaie, 'On symmetry, aesthetics and quantifying symmetrical complexity', in *Evolutionary and Biologically Inspired Music, Sound, Art and Design*, Lecture Notes in Computer Science, Springer Berlin Heidelberg, (2017). Accepted and in press.

[3] Mohammad Majid al-Rifaie and Ahmed Aber, 'Dispersive flies optimisation and medical imaging', in *Recent Advances in Computational Optimization*, 183–203, Springer, (2016).

[4] Mohammad Majid al-Rifaie and Mark Bishop, 'The mining game: a brief introduction to the stochastic diffusion search metaheuristic', *The Society for the Study of Artificial Intelligence and the Simulation of Behaviour Quarterly (AISBQ)*, **130**, (2010).

[5] Mohammad Majid al-Rifaie and Tim Blackwell, 'Cognitive bare bones particle swarm optimisation with jumps', *International Journal of Swarm Intelligence Research (IJSIR)*, **7**(1), 1–31, (2016).

[6] Mohammad Majid al Rifaie, Frédéric Fol Leymarie, William Latham, and Mark Bishop, 'Swarmic autopoiesis and computational creativity', *Connection Science*, 1–19, (2017).

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

[8] O. Burchan Bayazit, Jyh-Ming Lien, and Nancy M. Amato, 'Roadmap-based flocking for complex environments', in *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, p. 104, Washington, DC, USA, (2002). IEEE Computer Society.

[9] John N Belkin, NORMAN Ehmann, and Graham Heid, 'Preliminary field observations on the behavior of the adults of anopheles franciscanus mccracken in southern california', *Mosq News*, **11**, 23–31, (1951).

[10] J. Mark Bishop and Mohammad M. al Rifaie, 'Autopoiesis, creativity and dance', *Connection Science*, **29**(1), 21–35, (2017).

[11] J.M. Bishop, 'Stochastic searching networks', in *Proc. 1st IEE Conf. on Artificial Neural Networks*, pp. 329–331, London, UK, (1989).

[12] T. Blackwell, 'A study of collapse in bare bones particle swarm optimisation', *IEEE Transactions on Evolutionary Computing*, **16**(3), 354–372, (2012).

[13] R. L. Blickle, 'Observations on the hovering and mating of tabanus bishopp', *Stone. Ann. Entomol. Soc. 52*, 183–90, (1958).

[14] E. Bonabeau, M. Dorigo, and G. Theraulaz, 'Inspiration for optimization from social insect behaviour', *Nature*, **406**, 3942, (2000).

[15] Marco Dorigo, Mauro Birattari, and Thomas Stutzle, 'Ant colony optimization', *IEEE computational intelligence magazine*, **1**(4), 28–39, (2006).

[16] Marco Dorigo and Gianni Di Caro, 'Ant colony optimization: a new meta-heuristic', in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 2, pp. 1470–1477. IEEE, (1999).

[17] JA Downes, 'Observations on the swarming flight and mating of culicoides (diptera: Ceratopogonidae) 1', *Transactions of the Royal Entomological Society of London*, **106**(5), 213–236, (1955).

[18] JA Downes, 'Assembly and mating in the biting nematocera', *Intern. Congr. Entomol. Proc. 10th, Montreal*, 425–34, (1958).

[19] JA Downes, 'The swarming and mating flight of diptera', *Annual review of entomology*, **14**(1), 271–298, (1969).

[20] R.C. Eberhart and J. Kennedy, 'A new optimizer using particle swarm theory', in *Proceedings of the sixth international symposium on micro machine and human science*, volume 43. New York, NY, USA: IEEE, (1995).

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

[22] Charles H. Janson, 'Experimental evidence for spatial memory in foraging wild capuchin monkeys,cebus apella', *Animal Behaviour*, **55**, 1229–1243, (1998).

[23] J. Kennedy, 'Bare bones particle swarms', in *Proceedings of Swarm Intelligence Symposium, 2003 (SIS'03)*, pp. 80–87. IEEE, (2003).

[24] J. Kennedy and R. C. Eberhart, 'Particle swarm optimization', in *Proceedings of the IEEE International Conference on Neural Networks*, volume IV, pp. 1942–1948, Piscataway, NJ, (1995). IEEE Service Center.

[25] Michael King and Mohammad Majid al-Rifaie, 'Building simple non-identical organic structures with dispersive flies optimisation and a* path-finding', in *AISB 2017: Games and AI*, University of Bath, Bath, U.K., (2017). Accepted and in press.

[26] W Klassen and B Hocking, 'The influence of a deep river valley system on the dispersal of aedes mosquitos', *Bulletin of Entomological Research*, **55**(02), 289–304, (1964).

[27] Frederick Knab, 'The swarming of culex pipiens', *Psyche: A Journal of Entomology*, **13**(5), 123–133, (1906).

[28] Renato A Krohling and Eduardo Mendel, in *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pp. 3285–3291. IEEE, (2009).

[29] Max Lungarella, Fumiya Iida, Josh Bongard, and Rolf Pfeifer, *50 years of artificial intelligence: essays dedicated to the 50th anniversary of artificial intelligence*, Springer, 2007.

[30] M.J. Mataric, *Interaction and Intelligent Behavior*, Ph.D. dissertation, Department of Electrical, Electronics and Computer Engineering, MIT, USA, 1994.

[31] Hedvig Tetens Nielsen, 'Swarming and some other habits of mansonia perturbans and psorophora ferox (diptera: Culicidae)', *Behaviour*, 67–89, (1964).

[32] Craig W. Reynolds, 'Flocks, herds, and schools: A distributed behavioral model', *Computer Graphics*, **21**(4), 25–34, (1987).

[33] Louis M Roth, 'A study of mosquito behavior. an experimental laboratory study of the sexual behavior of aedes aegypti (linnaeus)', *American Midland Naturalist*, **40**(2), 265–352, (1948).

[34] Y. Shi and R. C Eberhart, 'Parameter selection in particle swarm optimization', *Lecture notes in computer science*, 591–600, (1998).

[35] Rainer Storn and Kenneth 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.

[36] Rainer Storn and Kenneth Price, 'Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces', *J. Global Optim.*, **11**, 341–359, (1997).

[37] Robert T Sullivan, 'Insect swarming and mating', *The Florida Entomologist*, **64**(1), 44–65, (1981).

[38] R. Thomsen, 'Flexible ligand docking using evolutionary algorithms: investigating the effects of variation operators and local search hybrids', *Biosystems*, **72**(1-2), 57–73, (2003).

[39] 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*, volume 2, pp. 1980–1987, (2004).

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

[41] E.O. Wilson, *Sociobiology: The new synthesis*, Belknap Press, 1975.

[42] David H. Wolpert and William G. Macready, 'No free lunch theorems for optimization', *IEEE Transactions on Evolutionary Computation*, **1**(1), 67–82, (April 1997).

[43] M. Wooldridge, 'An introduction to multiagent systems. 2002', *West Sussex, England: John Wiley and Sons Ltd*, **348**, (2002).

[44] Xin-She Yang, 'Firefly algorithms for multimodal optimization', in *International symposium on stochastic algorithms*, pp. 169–178. Springer, (2009).