Investigating Knowledge-Based Exploration-Exploitation Balance in a Minimalist Swarm Optimiser

Mohammad Majid al-Rifaie
School of Computing & Mathematical Sciences
University of Greenwich
London, United Kingdom
0000-0002-1798-9615

***Abstract*—One of the key challenges in evolutionary, swarm and population-based optimisers is the balance between exploration and exploitation. The reliance on both guided and stochastic search in these algorithms allows researchers to take different approaches to the topic. This work uses a minimalist, vector-stripped swarm optimiser to present a theoretical analysis on the behaviour of the particles. Being a population-based continuous optimiser, dispersive flies optimisation or DFO, bears several similarities with the well-known particle swarm optimisers, differential evolution algorithms and their bare-bones variants. The distinctive feature of this algorithm is its sheer reliance on particles positions to update the population. The minimalist nature of the algorithm reduces the challenges of understanding particles oscillation around constantly changing centres, particles' influence on one another, and their trajectories. This work presents a unified exploration-exploitation probability study which is derived from six scenarios in order to examine the population's dimensional behaviour in each iteration. This paves the way to propose and investigate adaptable, diversity promoting mechanisms. The proposed methods, which may be extendable to other optimisers, are then examined on a comprehensive set of benchmarks, and finally applied to high-dimensional tomographic reconstruction which is an important inverse problem in medical and industrial imaging.**

***Index Terms*—Exploration, exploitation, diversity, dispersive flies optimisation, DFO**

I. Introduction

Information exchange and communication between particles in swarm intelligence and evolutionary computation manifest themselves in a variety of forms, including the use of different update equations and strategies, deploying extra vectors in addition to the particles' current positions, and dealing with tunable parameters. Ultimately the goal of an optimiser is to achieve a balance between global exploration of the search space and local exploitation of potentially suitable areas in order to guide the optimisation process [1], [2].

The motivation for studying dispersive flies optimisation or DFO [3] is the algorithm's minimalist update equation which only uses particles' 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 the position vectors, use a subset of the following set of 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. 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 restart or disturbance threshold, Δ, which controls the component-wise restart in each dimension. This is contrary to many well-known algorithms dealing with several theoretically or empirically driven tunable parameters, such as: learning factors, inertia weight in PSO, crossover and mutation rates, tournament and elite sizes, and constricting factor in DE and/or Genetic Algorithms (GA) [7], heuristic strength, greediness and pheromone decay rate in ACO, impact of distance on attractiveness, scaling factor and speed of convergence in Firefly algorithm (FF) [8], and so on.

There have been several attempts to present compact algorithms to better understand the dynamic of population's behaviour as well as the relevance of various communication strategies but often with more components and parameters, and at the expense of performance. Perhaps one of the most notable minimalist swarm algorithm is bare-bones particle swarms (BB-PSO) [9] whose collapse has been studied along with the introduction of a component-wise jump or restart, thus proposing bare-bones with jumps algorithm (BBJ) [10]. Another algorithm is bare-bones differential evolution (BBDE) [11] which is a hybrid of the bare-bones particle swarm optimiser and differential evolution, aiming to reduce the number of parameters, albeit with more than only the position vector. It is well understood that swarm intelligence techniques are dependant on the tuning of their parameters, as a result, adjusting a growing number of parameters becomes increasingly complex.

This paper aims at providing a theoretical analysis of exploration and exploitation based on the existing knowledge of the swarm using the minimalist, vector-stripped algorithm; therefore, using this analysis to identify ways to measure exploration and exploitation probabilities, with the ultimate goal of controlling the behaviour of population by suggesting dimensional-based exploration-exploitation balance, without degrading the performance of the algorithm. Furthermore, the paper's proposed methods are applied to tomographic reconstruction, where images are reconstructed using tomography.

In this work, initially the swarm optimiser is presented in Section II, followed by the theoretical analysis in Section III which leads to proposing adaptable diversity mechanisms. Section IV presents the empirical experiments on a comprehensive set of benchmarks along with the results.

II. Background

DFO belongs to the broad family of population-based, swarm intelligence optimisers, which has been applied to various domains, including medical imaging [12], optimising machine learning algorithms [13], training deep neural networks [14], computer vision and quantifying symmetrical complexities [15], [16], beer organoleptic optimisation [17], solving diophantine equations [18] and analysis of autopoiesis in computational creativity [19].

In this algorithm, components of the position vectors are independently updated in each iteration, taking into account: current particle's position; current particle's best neighbouring individual (consider ring topology, where particles have left and right neighbours); and best particle in the swarm.

The update equation is

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

where,

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

As a diversity-promoting mechanism, individual components of the 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 disturbance or *restart threshold*, Δ. This ensures a restart to the otherwise permanent stagnation over likely local minima. In this method, which is summarised in Algorithm 1[1], each member of the population is assumed to have two neighbours and particles are not clamped to bounds, therefore, when out of bounds, are left unevaluated.

As a population-based continuous optimiser, DFO bears several similarities with other well-known swarm and evolutionary algorithms. Stemming from its bare-bones and vector-stripped nature, DFO allows for further analyses while demonstrating competitive performance despite being bare of "accessories". In terms of PSO, in many of the proposed variants, the algorithm 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. Furthermore, in addition

[1] Source code for the standard DFO: http://github.com/mohmaj/DFO

Algorithm 1 Dispersive Flies Optimisation

```
1: procedure DFO ($N, D, \vec{x}_{\min}, \vec{x}_{\max}, f$)*
2:     for $i = 0 \to N-1$ do                                  ▷ Initialisation
3:         for $d = 0 \to D-1$ do
4:             $x_{id}^0 \leftarrow \mathrm{U}(x_{\min,d}, x_{\max,d})$
5:         end for
6:     end for
7:     while ! termination criteria do                          ▷ Main DFO loop
8:         for $i = 0 \to N-1$ do
9:             $\vec{x}_i$.fitness $\leftarrow f(\vec{x}_i)$
10:        end for
11:        $\vec{x}_s$ = arg min $[f(\vec{x}_i)]$,   $i \in \{0,1,2,\ldots,N-1\}$
12:        for $i = 0 \to N-1$ and $i \neq s$ do
13:            $\vec{x}_{i_n}$ = arg min $[f(\vec{x}_{(i-1)\%N}), f(\vec{x}_{(i+1)\%N})]$
14:            for $d = 0 \to D-1$ do
15:                if $\mathrm{U}(0,1) < \Delta$ then
16:                    $x_{id}^{t+1} \leftarrow \mathrm{U}(x_{\min,d}, x_{\max,d})$
17:                else
18:                    $u \leftarrow U(0,1)$
19:                    $x_{id}^{t+1} \leftarrow x_{i_n d}^t + u(x_{sd}^t - x_{id}^t)$   ▷ Update eq.
20:                end if
21:            end for
22:        end for
23:    end while
24:    return $\vec{x}_s$
25: end procedure
```

* INPUT: N: swarm size, D: dimensions, $\vec{x}_{\min}$: lower bound, $\vec{x}_{\max}$: upper bound, f: fitness function.

to the position of particle i, $\vec{x}_i$, each PSO particle has an associated velocity, $\vec{v}_i$, and memory, $\vec{p}_i$, vectors. Other variants, including bare-bones PSOs were also introduced to simplify the algorithm, with the ultimate goal of offering insight into the underlying behaviour of the algorithm. In one such cases, one of the inventors of PSO, Kennedy, describes the process as "*strip[ping] away some traditional features*" with the hope of "*revealing the mysteries of the algorithm*" [9]. In this particular model, the velocity vectors are removed while the algorithm continues to benefit from the memory vectors; the work was carried out to shed light on the behaviour of the algorithm, yet at the cost of performance. There have been other contributions that have tried to further explore the simplified versions [10], [20], [21].

Following on from the above and to quote Kennedy [9] "*The particle swarm algorithm has just enough moving parts to make it hard to understand*", this work builds on of its key motivation to analyse a minimalistic algorithm to:

- reduce the challenges of understanding particles oscillating around the constantly changing centres (in each iteration, independently)
- understand particles' influence on one another (and their contribution to the swarm's next iteration)
- strip the parameters in the analysis to understand the trajectory of particles

To address these areas, the minimalist, vector-stripped features of the optimiser are used to analyse the algorithm's diversity and its exploration-exploitation behaviour.

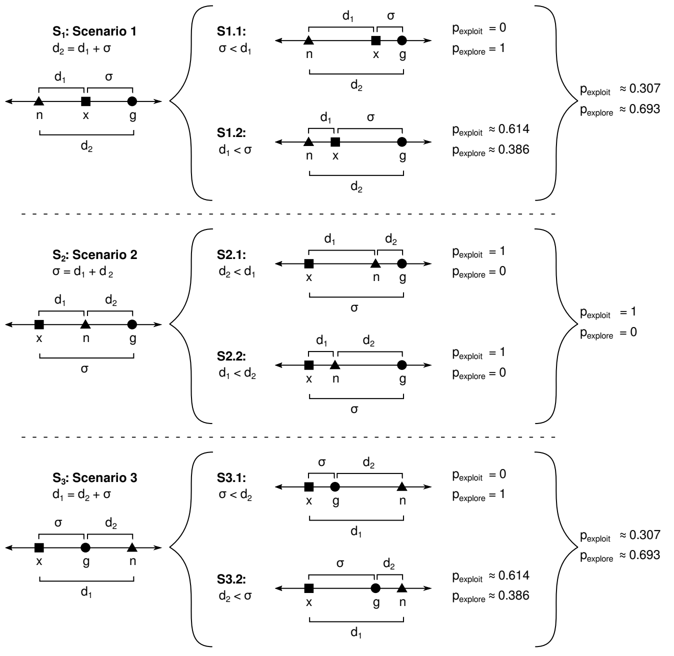


Fig. 1. Three scenarios for $x \leq g$, with $d_1 = |n - x|$, $d_2 = |g - n|$. The analysis also holds for the mirrored scenarios where $g \leq x$. These probabilities assume a start from the initial state. The analysis is extended to further reflect particles' dynamic in Section III-D.

III. Iteration-based exploitation analysis

As shown in the update equation, Eq. 1, for each particle, the search focus is $\vec{\mu} = \vec{x}_{i_n}$, and the spread, $\vec{\sigma} = \vec{x}_s - \vec{x}_i$, is the distance between the best particle in the swarm and the current particle. Therefore, the equation could be rewritten for each particle's dimension as

$$x = \mu + u\,\sigma \tag{2}$$

Considering one dimension of a problem and for the ease of readability in the remainder of this section, x refers to x_i^t; x' refers to x_i^{t+1}; g refers to x_s^t; and n refers to $x_{i_n}^t$.

When $x \leq g$, where g is the same for all x in the population for each iteration, the following three scenarios can be analysed:

- S_1: $n \leq x \leq g$ $S_{1.1}$: $\sigma \leq d_1$; $S_{1.2}$: $d_1 < \sigma$
- S_2: $x \leq n \leq g$ $S_{2.1}$: $d_2 \leq d_1$; $S_{2.2}$: $d_1 < d_2$
- S_3: $x \leq g \leq n$ $S_{3.1}$: $\sigma \leq d_2$; $S_{3.2}$: $d_2 < \sigma$

Fig. 1 illustrates these scenarios along with the 6 sub-scenarios. The mirrored counterparts can be envisaged for $g \leq x$. The analysis is first presented from the initial state, where the particles are initialised in the search space. Furthermore, *exploitation* refers to the approaching of x to g (i.e. $|g - x'| < |g - x|$). Analogously, *exploration* refers to the increasing distance between x and g (i.e. $|g - x'| > |g - x|$). Using these knowledge-based exploitation and exploration concepts, the analysis in this section focuses on each scenario and therefore, the overall impact on each iteration.

A. Scenario 1: $n \leq x \leq g$

In this scenario, the difference between $|n - x|$ and $|g - x|$, as well as the value of u in the update equation, determine whether x' approaches g. Given $d_1 = |n - x|$ and $d_2 = |g - n|$, in the first scenario, $d_2 = d_1 + \sigma$ (see Fig. 1-top).

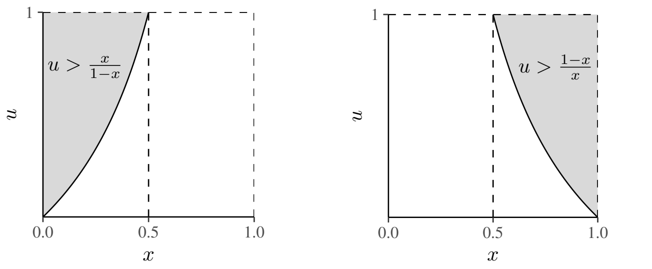


Fig. 2. Exploitation probability in S_1. Left: $n \leq x \leq g$; right: $g \leq x \leq n$

Depending on the proximity of x to either its best neighbour or the best particle in the swarm, two distinct cases need to be explored. Considering $\sigma \leq d_1$ ($S_{1.1}$ in Fig. 1), the exploitation probability, $p_{\text{exploit}} = 0$, and the exploration probability is $p_{\text{explore}} = 1$, where x moves away from g. On the other hand, in $S_{1.2}$, in Fig. 1, when $d_1 < \sigma$, p_{exploit} and p_{explore} depend on the proximity of the x to n as well as the randomly generated value of u. To analyse the probability of exploitation in this scenario, the space is scaled so that n is placed at the origin, g at 1, and x is uniformly distributed in $[0, 1]$. Therefore,

$$\begin{aligned}
n &= 0 \\
g &= 1 \\
x' &= 0 + u(1 - x) \\
P(\text{exploit}) &= P(|g - x'| < |g - x|) \\
&= P(1 - u(1 - x) < 1 - x) \\
&= P\left(u > \frac{x}{1 - x}\right) \\
&= 0.5 - \int_0^{0.5} \frac{x}{1 - x} dx \\
&= 0.5 - ([-x - \log(1 - x)]_0^{0.5}) \\
&\approx 0.307
\end{aligned}$$

This analysis holds for the mirrored case of scenario 1 (i.e. $g \leq x \leq n$). The plots in Fig. 2 illustrate the exploitation probabilities as shaded areas in S_1, and the mirrored S_1.

B. Scenario 2: $x \leq n \leq g$

Scaling the space, we have $x = 0$ and $g = 1$. Thus, there are two possible outcome cases for x':

1) $x \leq n \leq g \leq x'$
2) $x \leq n \leq x' \leq g$

Therefore, given $n \in [0, 1],\ u \sim \mathrm{U}(0, 1)$:

$$\begin{aligned}
x' &= n + u \\
P(\text{exploit}) &= P(|x' - 1| < 1) \\
&= P(|n + u - 1| < 1) = 1 \text{ Always holds}
\end{aligned}$$

$$\begin{aligned}
P(\text{exploit}) &= P(n + u - 1 < 1), && \text{for case (1)} \\
&= P(n + u < 2) = 1
\end{aligned}$$

$$\begin{aligned}
P(\text{exploit}) &= P(1 - (n + u) < 1), && \text{for case (2)} \\
&= P(-(n + u) < 0) = 1
\end{aligned}$$

The mirrored of this scenario (i.e. $g \leq n \leq x$) also holds with exploitation probability equal to 1.

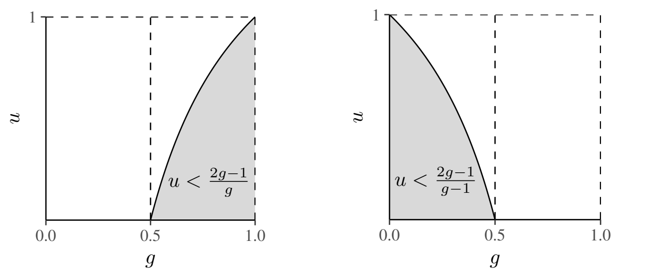


Fig. 3. Exploitation probability in S$_3$. Left: $x \le g \le n$; right: $n \le g \le x$

C. Scenario 3: $x \le g \le n$

Scaling the space, we have $x = 0$ and $n = 1$. Therefore,

$$\begin{aligned} x &= 0 \\ n &= 1 \\ P(\text{exploit}) &= P(ug + (1-g) < g) \\ &= P(u < \frac{2g-1}{g}) \\ &= \int_{0.5}^{1} (2 - \frac{1}{g}) dg \\ &= (2g - log(g))|_{0.5}^{1} \\ &\approx 0.307 \end{aligned}$$

The analysis for the mirrored cases in scenario 3 holds (i.e. $n \le g \le x$) and Fig. 3 illustrates the exploitation probabilities in S$_3$, and the mirrored S$_3$.

D. Unified exploitation analysis

The previous analysis provided the exploitation probabilities in a scaled environment where the acceptable range (bounds) of the search space in a problem domain did not play a role. This section analyses exploitation in relation to the feasible area of the search space.

Consider x to be uniform in $[-L, R]$ while g and n are fixed. Given this and as shown in Fig. 4, the areas highlighting exploitation can be plotted using A and B below:

$$\begin{aligned} g &= 0 \\ n &= 1 \\ A : u &= 1 - \frac{1}{|x|} \\ B : u &= \frac{1}{x} - 1 \end{aligned}$$

To proceed, the exploitation probabilities in the following four cases are presented individually:

1) $R, L \ge 1$
2) $R = L = 1$
3) $L \in [0, 1], R \in \left[\frac{1}{2}, 1\right]$
4) $L \in [0, 1], R \in \left[0, \frac{1}{2}\right]$

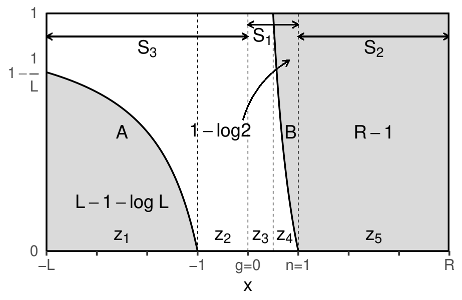


Fig. 4. Unified exploitation probability, p. The shaded areas in the graph represent exploitation, where particles in these areas at time t will be exploiting at time $t+1$.

$$\begin{aligned} R, L \ge 1: \; P(\text{exploit}) &= \frac{L + R - 1 - \log 2L}{L + R} \\ \text{put } L = R = x & \\ P(\text{exploit}) &= \frac{2x - 1 - \log 2x}{2x} \\ \lim_{x\to\infty} P(\text{exploit}) &= \lim_{x\to\infty} 1 - \frac{1}{2x} - \frac{\log 2x}{2x} \\ &= 1 \\ R = L = 1: \; P(\text{exploit}) &= \frac{1 - \log 2}{2} \\ L \in [0, 1], R \in \left[\frac{1}{2}, 1\right]: \; P(\text{exploit}) &= \frac{2R - 1 - \log 2R}{L + R} \\ L \in [0, 1], R \in \left[0, \frac{1}{2}\right]: \; P(\text{exploit}) &= 0. \end{aligned}$$

For $\{x \in \mathbb{R} : -L \le x \le R\}$ and given the tendency of $L, R \ge 1$ in the scaled space (influenced by the proximity of g and n over time), the unified exploitation probability, $P(\text{exploit})$ or p, is summarised as:

$$\begin{aligned} p = P(\text{exploit}) &= P(|x' - g| < |x - g|) \\ &= \frac{L + R - 1 - \log 2L}{L + R} \end{aligned} \quad (3)$$

Based on this, an immediate line of research is to measure the iteration-based probabilities of exploitation to facilitate diversity adjustment. This, in addition to having an adaptable restart threshold, Δ_{dynamic} (as opposed to a pre-determined parameter value, Δ), allows for a dimensional diversity mechanism. Using this approach, the unified exploitation probability, p, is measured for each dimension and in each iteration. Using p, the component-wise restart is triggered when $r < \Delta_{\text{dynamic}}$ where $r = U(0, 1)$ and $\Delta_{\text{dynamic}} = 1/1500p$. The rationale is to take into account the previously reported empirical restart threshold of $\Delta = 0.001$ [3] where $\Delta_{\text{dynamic}} = \Delta$ when $p = 0.\bar{6}$, and higher when $p < 0.\bar{6}$ (see Fig. 5). The adapted algorithm, which benefits from the unified exploitation probability, is termed *unified* DFO or uDFO. Using this approach, enables the adaptive, dimension-dependant diversity to be present throughout the optimisation process, and reduced when the population is more inclined towards exploitation, be it local or global.

To demonstrate the evident effect of individual's restart on p over the iterations, Fig. 9 illustrates the behaviour of p during the optimisation process where the restart mechanism is triggered when $p > \{0.90, 0.95\}$.

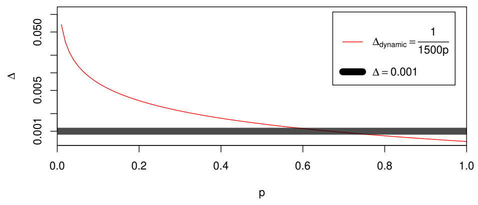


Fig. 5. Component-wise restart based on p, where the dynamic restart threshold is $\Delta_{\text{dynamic}} = 1/1500p$

In addition to the scenarios, S_{1-3}, which are based on the position of x in relation to n and g, Fig. 4 highlights the exploitation-related borderlines at $x \in \{-L, -1, 0, 0.5, 1, R\}$, and based on that, the search space is categorised into 5 *zones* (z_{1-5}). Using the zones provides a fitting way to investigate the behaviour of the individuals in the context of the unified exploitation probabilities as well as particle trajectories. In these zones, $z_{2,3}$ are explore-only, z_5 is exploit-only and $z_{1,4}$ influence both exploration and exploitation. In other words, zones impacting exploration are z_{1-4}, and zones impacting exploitation are $z_{1,4,5}$. Fig. 6 illustrates the visit-frequency of particle components in each zone over the iterations. Having these properties, investigating the state transitions from one zone at time t (x^t) to the next at time $t+1$ (x^{t+1}) provides each particle's dimensional trajectory, which is illustrated in Fig. 7 and summarised below:

- $x^t \in z_1 \rightarrow x^{t+1} \in z_5$
- $x^t \in z_2 \rightarrow x^{t+1} \in z_{5_{[1,2]}}$
- $x^t \in z_3 \rightarrow x^{t+1} \in z_4$
- $x^t \in z_4 \rightarrow x^{t+1} \in \{z_3, z_4\}$
- $x^t \in z_5 \rightarrow x^{t+1} \in \{z_1, z_2, z_3, z_4\}$

To show the trajectory density for each of these transitions, 1,000,000 component updates are initiated from each of the zones with $L, R = 4$. The associated density plots from different zone are shown in Fig. 8. The figure also presents the trajectory density of the independent updates across all zones, illustrating the densest area (see Fig. 8-bottom) which is in line with the search focus being n. State transition analysis allows for devising a strategy to control diversity through particle position's *zone-relocation*. Observing the density plot for z_5 in Fig. 8 or the state transition from z_5 in Fig. 7, it is evident that particles in z_5 at time t will be relocated to z_{1-4} at time $t+1$. Therefore, in another experiment, when the restart mechanism is triggered with $r < \Delta_{\text{dynamic}}$, components are relocated to z_5. Using this strategy, the components are effectively restarted to the exploit-only zone. As a result,

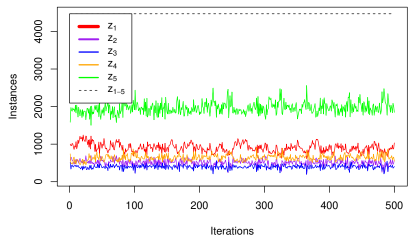
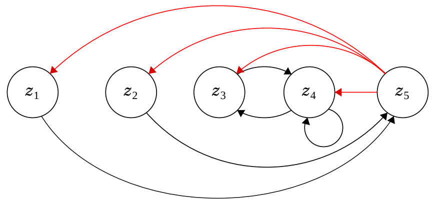


Fig. 6. Number of components in each zone over the iterations in a sample run optimising Rastrigin function, where z_5 is the most frequently visited zone, and z_3 is the least visited.



$$\begin{tikzcd}
z_1 \arrow[rrrr, bend right] & z_2 \arrow[rrr, bend right] & z_3 \arrow[r, bend left] & z_4 \arrow[l, bend left] \arrow[loop below] & z_5 \arrow[llll, bend right] \arrow[lll, bend right] \arrow[ll, bend right] \arrow[l]
\end{tikzcd}$$

Fig. 7. State transition of components between zones. Transitions from z_5 are highlighted.

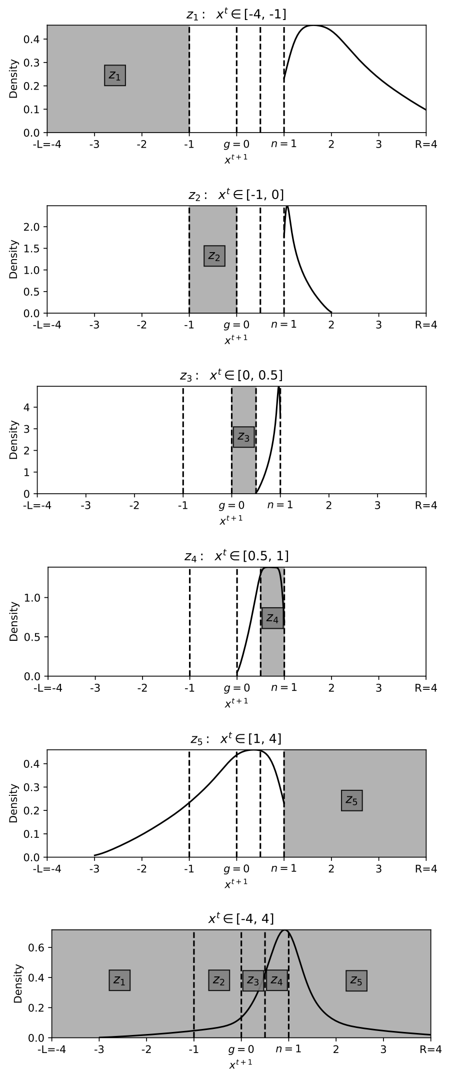
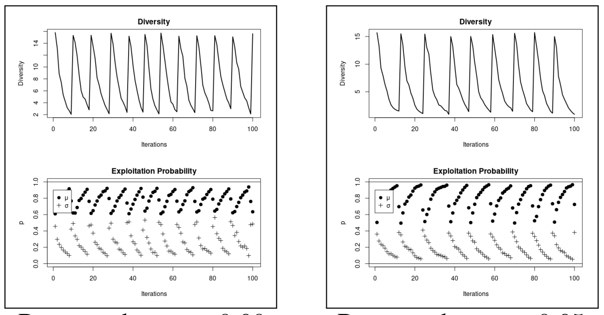


Fig. 8. Density plots for transition trajectory of 1 million independent components from each of the zones at time t (shaded) to $t+1$ in one-step updates. The dashed lines represent zones' boundaries. The x-axes represent the scaled positions in range $[-L, R]$ with $L, R = 4$, $g = 0$, $n = 1$, and the y-axes illustrate the trajectory density. For instance, the top graph shows the trajectory density of x^t values in z_1 which are originated from range $[-4, -1]$ at time t, and trajected to $[1, 4]$ at time $t+1$. The bottom graph presents the density plot across all zones, highlighting the focus as $\mu = n$. Note that the number of components initialised in each zone is equal.

while expecting lower diversity, the purpose of zone-relocation experiment is to examine the impact of 'targeted' restarts, with potential follow-up exploitation and visits to other zones. The adapted algorithm using the proposed zone-relocation strategy is termed uDFO$_{z5}$.



Restart when $p > 0.90$ Restart when $p > 0.95$

Fig. 9. Illustrating the link between exploitation probability, p, and diversity when restart commences at $p > \{0.90, 0.95\}$. In the bottom graphs, μ, and σ represent p's average and standard deviation in each iteration. As shown, increased diversity, which is the average distance around the population centre (see top graphs), decreases the p values (see bottom graphs), and vice versa.

IV. Empirical study and results

This section examines the results of the theoretical study of exploitation over a combined benchmark [22] which consists of functions presented in [23]–[25]. The combined benchmark CEC05 + CEC13 + ENG13 provides 84 unique problems whose details are provided in [22]. The benchmark includes functions with the following properties: U: unimodal, M: multimodal, S: separable, NS: non-separable, N: noisy, B: x^* on bounds (where x^* is the optimum), NV: x^* in narrow valley, UB: x^* outside initialisation volume, F: neutrality (has flat areas), HC: high conditioning, SD: sensitivity (f has one or more sensitive directions), A: asymmetric, D: deceptive (x^* is far from next local optimum) and C: composition.

In this section, uDFO and uDFO$_{z5}$ are compared against the standard DFO (with $\Delta = 0.001$) and DFO$_{\Delta=0}$ (i.e. without the restart mechanism) where the population size is $N_{DFO} = 150$. Furthermore, standard PSO algorithm in two neighbourhood structures, global PSO (GPSO) and local PSO (LPSO) are also used with the parameters derived from [25], where the population size, $N_{PSO} = 30$, $\omega = 0.729844$, $c = 1.49618$ and the initial $v = 0$. Each algorithm is run 50 times on each test function and the termination criterion is set to reaching 150,000 function evaluations. The problems' dimensionality is constant in all trials and is set to $D = 30$.

The metrics used to evaluate the results are *error*: best function value and proximity to known optimal values; the population's terminal *diversity*: mean distance between individuals and centroid (in PSO, the memory or personal best vectors are used, as opposed to DFOs where particles positions are used); and *last improvements*: significant activity late in the run, which indicates a potential move leading to hill climbing or escaping local minima. Measuring the significance of improvement depends on the nature of the test function (details for measuring this metric are provided in [22]).

In total, $25,200$ trials (6 algorithms $\times$ 84 test functions $\times$ 50 runs) are analysed by grouping them in terms of functions and function properties. To analyse the performance of the algorithms over the test functions, Wilcoxon [26] non-parametric tests of significance ($p < 0.05$) is used.

Additionally, the algorithms are applied to tomographic reconstruction (reconstruction of images by tomography), which is an important inverse problem in medical and industrial imaging [27]. In this problem, downsampled standard test images, the Shepp-Logan image phantoms [28], are reconstructed by using two projections. The images have the following dimensions: 25D (5×5), 100D (10×10), 255D (15×15), 400D (20×20) and 625D (25×25).

A. Results

Table I(a) summarises the performance of the algorithms on 84 test functions where 'win' and 'loss' of uDFO against other algorithms are considered when there is a recorded statistically significant outperformance in terms of the error values. The results illustrate uDFO's outperformance in 68%, 78%, 73% and 61% of the cases with statistically significant difference, when compared against DFO, DFO$_{\Delta=0}$ and GPSO and LPSO respectively. Understanding the potential reduced rate of the restart mechanism at the tail end of p, uDFO presents a higher

TABLE I
Summary of the results for uDFO.

(a) Error

Algorithms	Win	Loss	Tie	Win Rate (significant cases)
uDFO (vs DFO)	**21**	10	53	68%
uDFO (vs DFO$_{\Delta=0}$)	**40**	11	33	78%
uDFO (vs GPSO)	**47**	17	20	73%
uDFO (vs LPSO)	**45**	29	10	61%

(b) Last improvements

Algorithms	Win	Loss	Tie	Win Rate (significant cases)
uDFO (vs DFO)	9	**29**	46	24%
uDFO (vs DFO$_{\Delta=0}$)	**54**	2	28	96%
uDFO (vs GPSO)	**37**	14	33	73%
uDFO (vs LPSO)	18	**32**	34	36%

(c) Diversity

Algorithms	Win	Loss	Tie	Win Rate (significant cases)
uDFO (vs DFO)	0	**84**	0	0%
uDFO (vs DFO$_{\Delta=0}$)	**84**	0	0	100%
uDFO (vs GPSO)	**60**	21	3	74%
uDFO (vs LPSO)	22	**59**	3	27%

The numbers indicate uDFO's wins and losses when compared against other algorithms. uDFO exhibits outperformance in the majority of significant cases.

TABLE II
Summary of the results for uDFO$_{z5}$.

(a) Error

Algorithms	Win	Loss	Tie	Win Rate (significant cases)
uDFO$_{z5}$ (vs DFO)	**25**	14	45	64%
uDFO$_{z5}$ (vs DFO$_{\Delta=0}$)	**39**	11	34	78%
uDFO$_{z5}$ (vs GPSO)	**43**	17	24	72%
uDFO$_{z5}$ (vs LPSO)	**45**	29	10	61%
uDFO$_{z5}$ (vs uDFO)	4	**13**	67	24%

(b) Last improvements

Algorithms	Win	Loss	Tie	Win Rate (significant cases)
uDFO$_{z5}$ (vs DFO)	9	**29**	46	24%
uDFO$_{z5}$ (vs DFO$_{\Delta=0}$)	**52**	2	30	96%
uDFO$_{z5}$ (vs GPSO)	**36**	11	37	77%
uDFO$_{z5}$ (vs LPSO)	22	**31**	31	42%
uDFO$_{z5}$ (vs uDFO)	**8**	6	70	57%

(c) Diversity

Algorithms	Win	Loss	Tie	Win Rate (significant cases)
uDFO$_{z5}$ (vs DFO)	0	**84**	0	0%
uDFO$_{z5}$ (vs DFO$_{\Delta=0}$)	**84**	0	0	100%
uDFO$_{z5}$ (vs GPSO)	**57**	21	6	73%
uDFO$_{z5}$ (vs LPSO)	22	**60**	2	27%
uDFO$_{z5}$ (vs uDFO)	0	**45**	39	0%

The numbers indicate uDFO$_{z5}$'s wins and losses when compared against other algorithms. uDFO$_{z5}$ exhibits outperformance in the majority of significant cases when compared against DFO, DFO$_{\Delta=0}$, GPSO and LPSO.

TABLE III
PERFORMANCE COMPARISON BY FUNCTION PROPERTIES.

(a) uDFO

	f Property	Total	uDFO	DFO	uDFO	DFO$_{\Delta=0}$	uDFO	GPSO	uDFO	LPSO
U	Unimodal	22	**14**	0	8	8	**14**	6	**17**	3
M	Multimodal	62	7	**10**	**32**	3	**33**	11	**28**	26
S	Separable	18	**8**	1	**13**	5	**10**	5	**11**	3
NS	Non-separable	66	**13**	9	**27**	6	**37**	12	**34**	26
N	Noisy	3	0	0	**3**	0	0	**1**	1	**2**
B	x^* on bounds	4	**2**	0	1	1	**3**	0	2	2
NV	x^* in narrow val	3	0	**1**	0	0	**2**	0	**2**	1
UB	x^* out init vol	2	0	**1**	**2**	0	1	1	1	1
F	Neutrality	8	0	**2**	**6**	0	**2**	1	1	**7**
HC	High condition	2	0	**1**	0	0	1	1	1	1
SD	Sensitivity	2	**2**	0	**2**	0	**2**	0	**2**	0
A	Asymmetric	20	**4**	1	**7**	0	**9**	4	7	7
D	Deceptive	2	0	**1**	**1**	0	**1**	0	**1**	0
C	Composition	19	2	**3**	**9**	0	**5**	4	2	**13**
	$\sum$	233	**52**	30	**111**	23	**120**	46	**110**	92
	%		**63%**	37%	**83%**	17%	**72%**	28%	**54%**	46%

(b) uDFO$_{z5}$

	f Property	Total	uDFO$_{z5}$	DFO	uDFO$_{z5}$	DFO$_{\Delta=0}$	uDFO$_{z5}$	GPSO	uDFO$_{z5}$	LPSO
U	Unimodal	22	**16**	1	7	**8**	**13**	6	**18**	3
M	Multimodal	62	9	**13**	**32**	3	**30**	11	**27**	26
S	Separable	18	**9**	5	**12**	5	**9**	5	**12**	4
NS	Non-separable	66	**16**	9	**27**	6	**34**	12	**33**	25
N	Noisy	3	0	**1**	**2**	0	0	**1**	1	**2**
B	x^* on bounds	4	**2**	0	1	**2**	**2**	1	2	2
NV	x^* in narrow val	3	**1**	0	**1**	0	**2**	0	**2**	1
UB	x^* out init vol	2	1	1	**1**	0	1	1	1	1
F	Neutrality	8	0	1	**6**	0	1	1	1	**7**
HC	High condition	2	0	0	0	0	1	1	1	1
SD	Sensitivity	2	**2**	0	**2**	0	**2**	0	**2**	0
A	Asymmetric	20	3	**4**	**7**	0	**7**	3	7	7
D	Deceptive	2	0	**1**	**1**	0	**1**	0	**1**	0
C	Composition	19	2	2	**9**	1	3	**4**	2	**13**
	$\sum$	233	**61**	38	**108**	25	**106**	46	**110**	92
	%		**62%**	38%	**81%**	19%	**70%**	30%	**54%**	46%

Bold type indicates significantly lower function error by the algorithm for greater number of function instances with a given property.

number of last improvement cases and higher termination diversity against DFO$_{\Delta=0}$ and GPSO, as shown in Tables I(b) and I(c), however, the contrary can be observed with DFO and LPSO. The rationale is the consistent value of the restart threshold in standard DFO throughout the optimisation (given $\Delta = 0.001$) and the well understood higher diversity of local neighbourhood population in LPSO [29].

Table II presents performance comparison of uDFO$_{z5}$ with other algorithms. As expected, in terms of error, the winning rates of uDFO$_{z5}$ and uDFO are similar when compared against other algorithms, although the latter offers better overall performance. The last rows in Tables II(a,b,c) compare uDFO$_{z5}$ and uDFO, demonstrating the largest number of ties (see figures underlined) as indicators of similarities, which are likely to be influenced by the coverage similarity of holistic and zone-based restarts. However, as expected and explained earlier, uDFO exhibits higher diversity than uDFO$_{z5}$.

In order to analyse the error-related strengths and weaknesses of uDFO and uDFO$_{z5}$, each of the algorithm pairs are broken down in Table III based on fourteen function properties; the total number of function properties (shared by the test functions) is 233. The results demonstrate an overall outperformance of uDFO and uDFO$_{z5}$, where the most visible contribution of the unified exploitation approaches can be seen for functions with the following properties {U,S,NS,SD}, while being competitive in {M,NV,A}, and less effective for {N,C}. Among the suitable function properties is non-separable or NS, where variables interact, making it challenging to decompose the problem into sub-problems; this property is amongst the more demanding in the benchmark and in real-world fitness functions. Further analysis is required to better understand the function properties in the context of the algorithms performance.

TABLE IV
TOMOGRAPHIC RECONSTRUCTION: PERFORMANCE COMPARISON.

(a) uDFO

Algorithms	D=25	D=100	D=225	D=400	D=625
uDFO vs DFO	–	uDFO	uDFO	DFO	uDFO
uDFO vs GPSO	–	uDFO	uDFO	uDFO	uDFO
uDFO vs LPSO	uDFO	uDFO	uDFO*	uDFO*	uDFO*

(b) uDFO$_{z5}$

Algorithms	D=25	D=100	D=225	D=400	D=625
uDFO$_{z5}$ vs DFO	-	uDFO$_{z5}$	uDFO$_{z5}$	uDFO$_{z5}$	uDFO$_{z5}$
uDFO$_{z5}$ vs GPSO	-	uDFO$_{z5}$	uDFO$_{z5}$	uDFO$_{z5}$	uDFO$_{z5}$
uDFO$_{z5}$ vs LPSO	uDFO$_{z5}$	uDFO$_{z5}$	uDFO$_{z5}$*	uDFO$_{z5}$*	uDFO$_{z5}$*
uDFO$_{z5}$ vs uDFO	-	uDFO$_{z5}$	uDFO$_{z5}$	uDFO$_{z5}$	uDFO$_{z5}$

*: LPSO does not compute solutions for $D = \{255, 400, 625\}$. This is due to a large number of particles components' off-shooting out of bounds.

TABLE V
TOMOGRAPHIC RECONSTRUCTION: ERROR VALUES

	Algorithm	Min	Max	Median	Mean	StdDev
D=25	**uDFO**	0.00E+00	0.00E+00	0.00E+00	**0.00E+00**	0.00E+00
	uDFO$_{z5}$	0.00E+00	0.00E+00	0.00E+00	**0.00E+00**	0.00E+00
	DFO	0.00E+00	0.00E+00	0.00E+00	**0.00E+00**	0.00E+00
	GPSO	0.00E+00	0.00E+00	0.00E+00	**0.00E+00**	0.00E+00
	LPSO	0.00E+00	5.24E-32	3.08E-33	7.36E-33	1.14E-32
D=100	uDFO	1.6806E-14	1.6884E-14	1.6810E-14	1.6818E-14	1.6127E-17
	uDFO$_{z5}$	1.6806E-14	1.6808E-14	1.6806E-14	**1.6806E-14**	4.2304E-19
	DFO	1.98E-14	1.55E-11	5.65E-14	4.53E-13	2.18E-12
	GPSO	9.00E+01	2.48E+02	2.05E+02	1.94E+02	3.99E+01
	LPSO	1.17E+02	2.23E+02	1.67E+02	1.67E+02	2.53E+01
D=225	uDFO	6.02E-10	1.98E-08	4.49E-09	5.77E-09	4.34E-09
	uDFO$_{z5}$	2.09E-11	1.38E-09	1.94E-10	**2.53E-10**	2.40E-10
	DFO	1.45E-07	1.49E-06	4.15E-07	4.75E-07	2.48E-07
	GPSO	5.54E+02	7.08E+02	6.39E+02	6.42E+02	3.41E+01
	LPSO	NA	NA	NA	NA	NA
D=400	uDFO	1.71E-05	4.93E-05	2.77E-05	2.92E-05	7.14E-06
	uDFO$_{z5}$	1.60E-07	2.47E-06	6.14E-07	**7.21E-07**	4.41E-07
	DFO	1.32E-05	5.18E-05	2.59E-05	2.64E-05	7.72E-06
	GPSO	1.60E+03	1.86E+03	1.77E+03	1.76E+03	6.46E+01
	LPSO	NA	NA	NA	NA	NA
D=625	uDFO	1.89E-03	4.03E-03	2.75E-03	2.73E-03	4.65E-04
	uDFO$_{z5}$	1.33E-05	6.11E-05	2.97E-05	**3.13E-05**	1.03E-05
	DFO	1.01E-02	2.33E-02	1.73E-02	1.72E-02	2.67E-03
	GPSO	3.89E+03	4.41E+03	4.15E+03	4.15E+03	1.09E+02
	LPSO	NA	NA	NA	NA	NA

Finally, the proposed approaches are trialled on tomographic construction, taking into account problems with larger dimensionality (Tables IV and V). Each algorithm is run 50 times for each problem, therefore a total of $1,250$ trials are conducted (5 algorithms $\times$ 5 problems $\times$ 50 runs). Barring the lowest dimensional problem (25D), the results illustrate the overall competitiveness of uDFO in 92% (11 out of 12), and uDFO$_{z5}$ in 100% (12 out 12) of the algorithm-problem pairs in high-dimensional problems (see Tables IV-a and IV-b respectively).

In summary, while the performance of uDFO and uDFO$_{z5}$ are similar on the lower dimensional problem, uDFO$_{z5}$ demonstrates better performance in all higher-dimensional problems (i.e. 100D, 255D, 400D, 625D) with wider performance gaps as the dimensionality grows (see Table V). Further experiments are needed to verify the extendibility of performance in other high-dimensional problems.

Among the limitations of the approach is the need for *a-priori* knowledge of the bounds to feasible solutions. Whilst setting indicative bounds in many real-world problems is practically possible, further investigation is needed in this area. Additionally, although the main computational expense is associated with function evaluation, the impact of calculating exploitation probability, p, on the computational cost is a topic for an ongoing research. Furthermore, having tested the approaches on a comprehensive set of test functions as well as identifying a number of suitable function properties, one of the next steps is applying the methods to other complex real-world problems with known function properties.

V. Conclusion

This work provides a theoretical, iteration-based analysis of particles' movements to measure the knowledge-based, dimensional exploitation probabilities. In addition to better understanding the particles' behaviour, the work focuses on providing a strategy to control the population's interaction in the search space. This is attempted through a unified exploitation probability, p, through (1) uDFO algorithm which uses a holistic restart, and (2) uDFO$_{z5}$ which is trialled for the purpose of examining zone-relocation restart mechanism. Both methods allow adaptable dimensional control of the particles. The proposed approaches are then examined over 84 test functions with a combined 233 function properties, where uDFO performs better in 68%, 78%, 73% and 61% of cases with statistically significant difference when compared against DFO, DFO$_{\Delta=0}$ and GPSO and LPSO respectively; and uDFO$_{z5}$ in 64%, 78%, 72% and 61% of the significant cases. The performance is then investigated on the high-dimensional tomographic reconstruction problems where uDFO and uDFO$_{z5}$ exhibited better performance in 92% and 100% of the high-dimensional algorithm-problem pairs respectively. Potential future work includes extending the exploitation analyses to other swarm optimisers and investigating unbounded problems. Studying the effect of the presented approaches on dynamically changing environments and exploring the combinations of function properties, which benefit from the analysis, are also topics for future research.

References

[1] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information processing letters*, vol. 85, no. 6, pp. 317–325, 2003.

[2] O. Olorunda and A. P. Engelbrecht, "Measuring exploration/exploitation in particle swarms using swarm diversity," in *Evolutionary Computation, 2008. CEC 2008.* IEEE, 2008, pp. 1128–1134.

[3] M. M. al-Rifaie, "Dispersive flies optimisation," in *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems*, M. P. M. Ganzha, L. Maciaszek, Ed., vol. 2. IEEE, 2014, pp. pages 529–538.

[4] J. Kennedy, "The particle swarm: social adaptation of knowledge," *Evolutionary Computation, 1997., IEEE International Conference on*, pp. 303–308, 1997.

[5] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[6] M. Dorigo and G. Di Caro, "Ant colony optimization: a new meta-heuristic," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 2. IEEE, 1999, pp. 1470–1477.

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

[8] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *International symposium on stochastic algorithms*. Springer, 2009, pp. 169–178.

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

[10] T. Blackwell, "A study of collapse in bare bones particle swarm optimisation," *IEEE Transactions on Evolutionary Computing*, vol. 16, no. 3, pp. 354–372, 2012.

[11] M. G. Omran, A. P. Engelbrecht, and A. Salman, "Bare bones differential evolution," *European Journal of Operational Research*, vol. 196, no. 1, pp. 128–139, 2009.

[12] M. M. al-Rifaie and A. Aber, "Dispersive flies optimisation and medical imaging," in *Recent Advances in Computational Optimization*. Springer, 2016, pp. 183–203.

[13] H. Alhakbani, "Handling class imbalance using swarm intelligence techniques, hybrid data and algorithmic level solutions," Ph.D. dissertation, Goldsmiths, University of London, London, United Kingdom, 2018.

[14] H. Oroojeni, M. M. al-Rifaie, and M. A. Nicolaou, "Deep neuroevolution: Training deep neural networks for false alarm detection in intensive care units," in *European Association for Signal Processing (EUSIPCO) 2018*. IEEE, 2018, pp. 1157–1161.

[15] M. M. al-Rifaie, A. Ursyn, R. Zimmer, and M. A. J. Javid, "On symmetry, aesthetics and quantifying symmetrical complexity," in *International Conference on Evolutionary and Biologically Inspired Music and Art*. Springer, 2017, pp. 17–32.

[16] P. Aparajeya, F. F. Leymarie, and M. M. al-Rifaie, "Swarm-based identification of animation key points from 2d-medialness maps," in *Computational Intelligence in Music, Sound, Art and Design*. Cham: Springer International Publishing, 2019, pp. 69–83.

[17] M. M. al-Rifaie and M. Cavazza, "Beer organoleptic optimisation," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 255–256.

[18] B. Lazov and T. Vetsov, "Sum of three cubes via optimisation," *arXiv preprint arXiv:2005.09710*, 2020.

[19] M. M. al-Rifaie, F. F. Leymarie, W. Latham, and M. Bishop, "Swarmic autopoiesis and computational creativity," *Connection Science*, pp. 1–19, 2017.

[20] R. A. Krohling and E. Mendel, "Bare bones particle swarm optimization with gaussian or cauchy jumps," in *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*. IEEE, 2009, pp. 3285–3291.

[21] M. M. al-Rifaie and T. Blackwell, "Cognitive bare bones particle swarm optimisation with jumps," *International Journal of Swarm Intelligence Research (IJSIR)*, vol. 7, no. 1, pp. 1–31, 2016.

[22] T. Blackwell and J. Kennedy, "Impact of communication topology in particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 4, pp. 689–702, 2019.

[23] 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," 2005.

[24] J. Liang, B. Qu, P. Suganthan, and A. G. Hernández-Díaz, "Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization," *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report*, no. 34, pp. 281–295, 2013.

[25] A. P. Engelbrecht, "Particle swarm optimization: Global best or local best?" in *2013 BRICS congress on computational intelligence and 11th Brazilian congress on computational intelligence*. IEEE, 2013, pp. 124–135.

[26] F. Wilcoxon, S. Katti, and R. A. Wilcox, "Critical values and probability levels for the wilcoxon rank sum test and the wilcoxon signed rank test," *Selected tables in mathematical statistics*, vol. 1, pp. 171–259, 1970.

[27] P. P. Bruyant, "Analytic and iterative reconstruction algorithms in spect," *Journal of Nuclear Medicine*, vol. 43, no. 10, pp. 1343–1358, 2002.

[28] L. A. Shepp and B. F. Logan, "The fourier reconstruction of a head section," *IEEE Transactions on nuclear science*, vol. 21, no. 3, pp. 21–43, 1974.

[29] S. Cheng and Y. Shi, "Diversity control in particle swarm optimization," in *2011 IEEE Symposium on Swarm Intelligence*. IEEE, 2011, pp. 1–9.