electronics

Article

A New Artificial Duroc Pigs Optimization Method Used for the Optimization of Functions

Jacek M. Czerniak [1], Dawid Ewald [1], Marcin Paprzycki [2], Stefka Fidanova [3] and Maria Ganzha [4,*]

[1] Faculty of Computer Science, Kazimierz Wielki University, 85-064 Bydgoszcz, Poland; jczerniak@ukw.edu.pl (J.M.C.); dawid.ewald@ukw.edu.pl (D.E.)
[2] Systems Research Institute, Polish Academy of Sciences, 01-447 Warsaw, Poland; marcin.paprzycki@ibspan.waw.pl
[3] Institute of Information and Communication Technologies, Bulgarian Academy of Sciences, 1040 Sofia, Bulgaria; stefka@parallel.bas.bg
[4] Department of Mathematics and Information Science, Warsaw University of Technology, 00-662 Warsaw, Poland
* Correspondence: maria.ganzha@ibspan.waw.pl

Abstract: In this contribution, a novel optimization approach, derived from the behavioral patterns exhibited by Duroc pig herds, is proposed. In the developed metaheuristic, termed Artificial Duroc Pigs Optimization (ADPO), Ordered Fuzzy Numbers (OFN) have been applied to articulate and elucidate the behavioral dynamics of the pig herd. A series of experiments has been conducted, using eight standard benchmark functions, characterized by multiple extrema. To facilitate a comprehensive comparative analysis, experiments employing Particle Swarm Optimization (PSO), Bat Algorithm (BA), and Genetic Algorithm (GA), were executed on the same set of functions. It was found that, in the majority of cases, ADPO outperformed the alternative methods.

Keywords: fuzzy logic; OFN; ADPO; GA; PSO; BA

Citation: Czerniak, J.M.; Ewald, D.; Paprzycki, M.; Fidanova, S.; Ganzha, M. A New Artificial Duroc Pigs Optimization Method Used for the Optimization of Functions. *Electronics* **2024**, *13*, 1372. https://doi.org/10.3390/electronics13071372

Academic Editor: George A. Tsihrintzis

Received: 23 January 2024
Revised: 14 March 2024
Accepted: 29 March 2024
Published: 5 April 2024

1. Introduction to Pig Herd Optimization

Nowadays, it is almost impossible to count all the "animals" (or, more broadly understood, natural phenomena) that inspired optimization methods. Among them, some of the more "unusual ones" are as follows:

- Whale Optimization Algorithm (WOA) [1,2],
- Firefly Optimization Algorithm (FOA) [3],
- Orca Predation Algorithm (OPA) [4–6],
- Starling Murmuration Optimizer (SMO) [7–9],
- Hunting search meta-heuristic algorithm [10].

Moreover, nature-phenomena-inspired optimization algorithms have been applied to problems originating from a very wide spectrum of areas, for instance (the aim of this list is only to illustrate the breadth of the explored application areas; it is not exhaustive in any way):

- economic dispatch problem [11],
- workflow planning of construction sites [12],
- multimodal optimization with multiple optima [13],
- engineering optimization problems [14,15],
- medical science [16],
- automatic composition of semantic web services [17],
- pickup and delivery problem [18].

Moreover, recently, a novel trend can be observed. Here, a nature-inspired "mechanism" is combined with "non-standard elements". For instance, the resulting "hybrid methods" include (but are not limited to) the following:

- modification of the Artificial Bee Colony (ABC) algorithm by application of Ordered Fuzzy Numbers (OFN) – OFNBee [19,20], or
- Hybrid Fuzzy-Ant Colony optimization [21].

Since the results of combining OFNs with ABC optimization were quite promising [19,20], further exploration of this research direction (exploring hybrid optimization approaches) ensued. In this contribution, the behavior of a pig herd was selected as the nature-originating phenomenon. The idea originated from the fact that biologists and breeders, who studied the behavior of pigs, found that they are the "wisest" non-primate animals. As a matter of fact, in the scientific literature, it has even been claimed that their intelligence may exceed that of some primates. For instance, Stanley Curtis carried out an experiment related to pig memory. He placed a ball, a frisbee, and a sinker in front of several pigs, and was able to teach them to jump, sit next to, or retrieve any of these items [22–24]. Based on the performed experiments, he claimed that pigs are wiser than dogs. Moreover, it was also established that, in some video games, pigs were "better players" than chimpanzees [25]. Curtis' successors continued this work. Here, James Pettigrew [26] found that pigs not only have their own preferences for temperature [27], but also try, if they are able, on the basis of trial and error, to master ways of "managing" the heating in a room [28]. Next, Sarah Boysen argued [23] that pigs are able to concentrate with an intensity higher than in the case of chimpanzees. Finally, Michael Mendel observed [29] that guinea pigs can signal their strength and use this information to minimize aggressive behavior while establishing the order in the herd. He also stipulated that pigs show quite complicated social behaviors, such as competition, similar to those observed among some primate species [30]. In this context, anyone who has even briefly stayed near pigs could easily find that they are constantly communicating with each other.

Taking this into account, it was concluded that the abilities of pigs may be useful in solving optimization problems. The earliest attempts at exploring this research direction have been reported in [31–33]. However, to the best of our knowledge, nobody has tried to create a hybrid method, and to infuse Ordered Fuzzy Numbers (OFN)s into optimization algorithms inspired by pig behavior [34]. Since, as noted, combining OFNs and ABC optimization was quite successful, exploring the potential of a hybrid optimization method, based on (Duroc) pig herd behavior combined with the use of OFNs, is the main contribution of this work.

2. Introduction to Ordered Fuzzy Numbers and Their Application

Before describing the proposed approach to hybrid optimization, let us briefly introduce the main concepts related to the Ordered Fuzzy Numbers (OFN). This is undertaken, among others, because OFNs are not well-known and, therefore, without such an introduction, the proposed approach may not be easy to follow. However, it should be stressed that the interested readers should consult the references found in the next section for an in-depth discussion of the idea and its potential applications.

2.1. Concept of Ordered Fuzzy Numbers

The original approach to defining fuzzy numbers was proposed in 1993 by Witold Kośiński and his doctoral student P. Słysz [35–38]. Subsequent research with resulting publications by W. Kośiński and collaborators (P. Prokopowicz and D. Ślezak) introduced the concept of ordered fuzzy numbers and its potential applications [39–44]. Based on these publications and material summarized in [34], an *Ordered Fuzzy Number* (A) can be *conceptualized* as a quadruple $A = [f_A(0), f_A(1), g_A(1), g_A(0)]$, which forms a trapezoid, as depicted in Figure 1.

Based on this meta-level conceptualization, a precise definition of an Ordered Fuzzy Number can be formulated as follows:

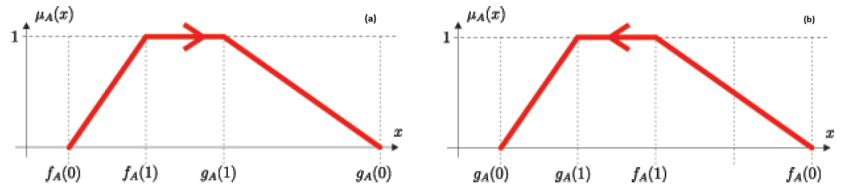

Figure 1. Visual representation of Ordered Fuzzy Numbers with (**a**) positive orientation and with (**b**) negative orientation.

Definition 1. *The Ordered Fuzzy Number A is defined as an ordered pair of functions:*

$$A = (x_{up}, x_{down})$$

where $x_{up}, x_{down} : [0,1] \longrightarrow R$ *are continuous functions (see, Figure 2). Functions* f_A, g_A *represent the parts* $up_A, down_A \subseteq R^2$*, respectively, such that it follows:*

$$up_A = \{(f_A(y), y) : y \in [0,1]\}$$
$$down_A = \{(g_A(y), y) : y \in [0,1]\}$$

Here, the orientation corresponds to the order of the graphs of f_A *and* g_A*.*

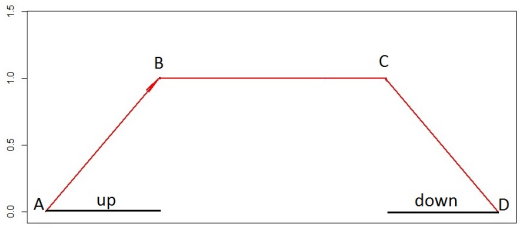

Figure 2. Graphical interpretation of the ordered fuzzy number definition.

The distinguishing feature of the *directed* Ordered Fuzzy Number lies in its orientation:

- When the directed OFN is aligned in the direction of increasing values along the X-axis, it is termed as having a positive orientation,
- Conversely, when the directed OFN is aligned in the opposite direction, it is referred to as having a negative orientation.
- Finally, a convex fuzzy number is a fuzzy set in the space of real numbers, which is normal, convex, has a bounded support, and a piecewise continuous membership function. It encompasses proper fuzzy numbers, fuzzy intervals, singleton sets, and fuzzy sets with an unbounded support.

2.2. Definition of Golden Ratio-Based Defuzzyfication Operator

Obviously, being ordered is an important element of OFNs. It can be said that it determines the "nature" of the number. Here, let us note that another important aspect of fuzzy numbers is the fuzzification/defuzzification process, i.e., the transformation from the real numbers to the fuzzy numbers and back. In this context, for the developed optimization algorithm, let us consider the use of a defuzzification method, which is based on the concept of the Golden Ratio, which in turn is derived from the Fibonacci series. Note that in the context of this work, fuzzification (mapping from real to fuzzy numbers) is not required, but it can be easily formulated if needed.

Overall, the inspiration for the defuzzification method originates from the observation of numerous occurrences of the Golden Ratio across various disciplines, such as biology, architecture, medicine, and art. Of particular interest is its presence in genetics, where the structure of the DNA molecule itself exhibits the Golden Ratio (with deoxyribonucleic acid molecules measuring 21 angstroms wide and 34 angstroms long for each full length of a single double helix cycle). This phenomenon, as well as an overall popularity of the Golden Ratio, points to an interesting role of the Fibonacci series, and the Golden Ratio

in particular, as the "universal design principles", utilized both by humans and in nature. Therefore, amidst the array of possible (and available) (de)fuzzification operators, it has been decided to employ one based on the Golden Ratio. Its definition is as follows:

Definition 2.

$$\underset{GR}{\varphi} = \begin{cases} min(supp(A)) + \frac{supp(A)}{\Phi} & if \quad ordered\ A\ is\ positve \\ max(supp(A)) - \frac{supp(A)}{\Phi} & if \quad ordered\ A\ is\ negative. \end{cases} \tag{1}$$

Here, GR is the defuzzification operator, while $supp(A)$ *is the support for the fuzzy set A in the universe X.*

Let us now define the Golden Ratio for the OFN. The mathematical Formula (1), as well as its graphic interpretation presented in Figure 3, applies to convex fuzzy numbers, as these numbers are being used in the proposed algorithm. Convex fuzzy numbers, as defined in fuzzy set theory, are sets characterized by both fuzziness and convexity. Fuzziness allows for the representation of uncertainty (or imprecision), while convexity ensures that the set remains continuous and possesses specific geometric properties. In the context of the proposed algorithm, convex fuzzy numbers serve as a flexible and robust means of representing uncertain or imprecise data, facilitating effective decision-making and problem-solving processes.

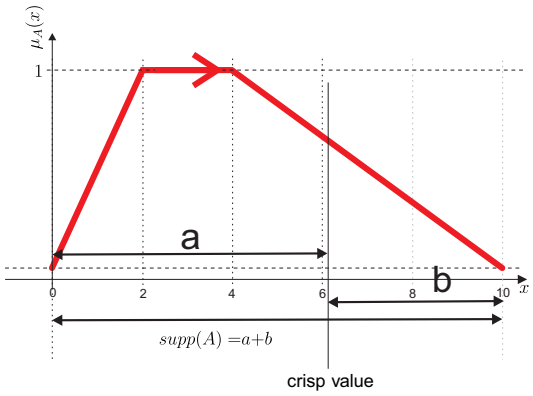

Figure 3. The OFN number A = [0, 2, 4, 10].

Since OFNs are oriented, a formula for the case of the orientation "pointing in the other direction" is also needed. This case is illustrated in Figure 4.

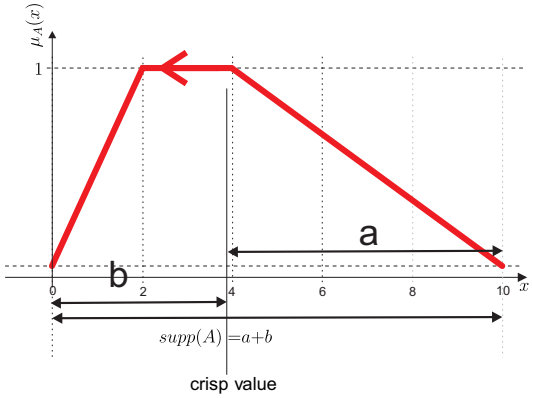

Figure 4. The OFN number B = [10, 4, 2, 0].

In this context, it is important to observe how the individual components of the two values of the line segments *a* and *b* are situated, relative to the orientation of the OFN. In the

scenario where the OFN is positive, a greater portion of the Golden Ratio (GR) originates from the base point $f(0)$. Conversely, in the case of a negative OFN, the base point shifts to $g(0)$. Now, let us proceed to define the additional technical aspects necessary for the GR defuzzification.

$$supp(A) = \begin{cases} \underset{A}{g}(0) - \underset{A}{f}(0) & if \quad ordered\ A\ is\ positve \\ \\ \underset{A}{f}(0) - \underset{A}{g}(0) & if \quad ordered\ A\ is\ negative \end{cases} \tag{2}$$

$$min(supp(A)) = \begin{cases} \underset{A}{f}(0) & if \quad ordered\ A\ is\ positve \\ \\ \underset{A}{g}(0) & if \quad ordered\ A\ is\ negative \end{cases} \tag{3}$$

$$max(supp(A)) = \begin{cases} \underset{A}{g}(0) & if \quad ordered\ A\ is\ positve \\ \\ \underset{A}{f}(0) & if \quad ordered\ A\ is\ negative \end{cases} \tag{4}$$

Note that use of the GR defuzzification operator in the context of (Duroc pigs inspired) optimization is one of the contributions of this work.

3. Proposed Artificial Duroc Pigs Optimization Method

The research on the Artificial Duroc Pigs Optimization (ADPO) algorithm dates back a long time, and the current article represents an extension of the earlier study, presenting the first version of the algorithm, and results obtained at that time [45]. Work on the APDO has been ongoing since its initial development, and its code has now been fully re-implemented in a completely new environment. This is another step in the evolution of the algorithm, which continues to progress in light of new discoveries and experiments. On the other hand, let us note that article [46] focuses on the application of fuzzy logic, particularly Ordered Fuzzy Numbers, in observing and detecting DDOS attacks, representing a completely different research area from that of the ADPO.

As mentioned, the primary focus of this study lies in amalgamating pig-inspired optimization with Ordered Fuzzy Number arithmetic, employing the Golden Ratio defuzzifier. Among the various breeds of pigs, Duroc pigs have been deliberately chosen. Originating around 1800 in New England, Duroc pigs represent an ancient strain of domestic pigs. Modern Duroc pigs typically exhibit medium-sized bodies, characterized by moderately long torsos and slightly flattened snouts [47,48]. In several regions, such as the UK, farms are established in such a way that Duroc pigs roam freely outdoors. During nighttime, they seek refuge in lightweight prefabricated shelters, spending the majority of their time roaming the fields. Feeding is typically carried out by workers who traverse through or around the herd, dispersing food randomly. Pigs communicate with loud "vocalizations" when food is discovered.

The observations related to the response of Duroc pig herds to a randomly scattered food supply became the basis of the proposed algorithm. Let us start from a meta-level description of the approach. Here, Figure 5 shows a schematic visualization of a simplified Duroc pig herd. The points symbolize the location of pigs on the field. Two of them (denoted as *C* and *E*) represent the case of roaring pigs, i.e., pigs that attempt to communicate with others (in their herd) that they found food.

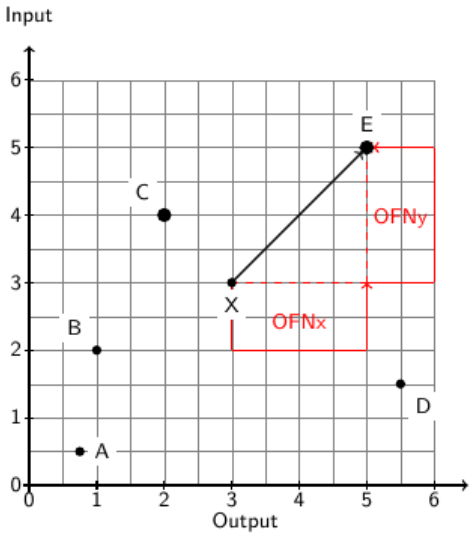

Figure 5. Graphical representation of the ADPO method.

In the ADPO algorithm, one more category of pigs, referred to as the scout pigs has been introduced. It represents pigs that roam around in search of food. Upon discovery, they communicate through loud roars. Upon hearing the roar, other pigs proceed to move in a straight line from their current position to the location of one of the roaring pig. Here, it is assumed that the field is flat and there are no obstacles to pig movement. In the scenario illustrated in Figure 5, a pig positioned at point X has the option to move towards point C or point E. Overall, in such cases, a specialized choice-preference algorithm can be proposed, or priorities can be determined based on factors such as the intensity of the roar, which may indicate the quantity of food available, or the proximity of the roaring pig. In the case of this contribution, the selection of the next destination, towards which a given pig will move, is dictated by the abundance of food only. Hence, the pig at point X will proceed towards point E. The direction of movement is determined using a simple equation representing a line passing through the two points.

However, it is also important to "capture" the distance from the current location to the food. Therefore, taking into account knowledge concerning optimization algorithms, it can be easily realized that taking into account locations that are too far could result in too early convergence (possibly to a local extremum). At the same time, focusing only on nearby locations would slow the search. It is exactly to establish the "right search space" where the OFN-based approach [49–51] is being applied, along with the Golden Ratio [52] defuzzification.

Of course, it is possible (at least in theory) to apply other defuzzification operators. This research direction is going to be explored in subsequent versions of the algorithm. However, as the basic characteristic of the GR operator is that the defuzzified numbers never go beyond the fuzzy number support, it was assumed that this approach should work well in the ADPO method.

The pseudocode of the optimization method for the Artificial Duroc Pigs, or the ADPO with Golden Ratio-based defuzzification, is presented next. Here, the following notation is applied:

- p_i—pig number i.
- P_i—pig number i from the subset of new individuals.
- s_{si}—pig scout number i, temporary collection of solutions.

ADPO Pseudocode:

Inputs:

Number of scouting pigs—$SP = 10$

Number of leaders [%]—$LP = 10\% * NP$
Number of pigs—$NP = 20$
A herd of scouting pigs—$\{s_1, \ldots, s_{SP}\}$
Herd of pigs—$\{p_1, \ldots, p_{NP}\}$

Outputs:
Solution in the form of a point $R(x, y)$

Functions:

- $SendToRandomPlaces()$—sending out NP pigs to random places,
- $SendToUniquePlaces()$—sending out SP pig scouts to other unexplored places,
- $SortGroup()$—arranging solutions from the finest to the poorest,
- $FindLeaderPigs()$—looking for the top performer(s), constituting $LP = 10\%$ of the target population,
- $ReturnOfPigsScouts()$—return of scout pigs,
- $NextPigsPositions()$—determining the subsequent position for each member of the herd, excluding the leaders.
- $RefillHerdPigs()$—completes the NP sample from the $NP + SP - LP$ set,
- $AvoidWeakPigs()$—deleting solutions that did not fit in the NP
- $IfNoEnd()$—a function that verifies the satisfaction of the termination condition; the algorithm will conclude upon reaching the specified accuracy or the maximum number of epochs.

BEGIN
step 1: $\forall_{p_i} \in < p_1, \ldots, p_{NP} >$ **SendToRandomPlaces(**p_i**)**
step 2: $\forall_{s_i} \in < s_1, \ldots, s_{SP} >$ **SendToUniquePlaces(**s_i**)**
step 3: $\forall_{P_i} \in < p_1, \ldots, p_{NP} \cup s_1, \ldots, s_{SP} >$ **SortGroup(**P_i**)**
step 4: **FindLeaderPigs()** $\{pl_1, pl_2, \ldots, pl_{LP}\} \in P$
step 5: **RefillHerdPigs()**
step 6: **AvoidWeakPigs()**
step 7: **NextPigsPositions()**
$\forall_{p_i} \in < p_1, \ldots, p_{NP} > P_i \longrightarrow P_i'$
niech $P(x_1, y_1)$ i $R(x_2, y_2)$
$P(x_1, y_1) \longrightarrow P'(x_1', y_1')$

$OFN_x[x_1, x_1, x_2, x_2]$ i $OFN_y[y_1, y_1, y_2, y_2]$

$$x_1' = GR(OFN_x) = \begin{cases} min(supp(OFN_x)) + \frac{|supp(OFN_x)|}{\Phi} & , \\ for\ (OFN_x)\ positive & \\ & \\ max(supp(OFN_x)) - \frac{|supp(OFN_x)|}{\Phi} & , \\ for\ (OFN_x)\ negative & \end{cases}$$

$$y_1' = GR(OFN_y) = \begin{cases} min(supp(OFN_y)) + \frac{|supp(OFN_x)|}{\Phi} & , \\ for\ (OFN_y)\ positive & \\ & \\ max(supp(OFN_y)) - \frac{|supp(OFN_x)|}{\Phi} & , \\ for\ (OFN_y)\ negative & \end{cases}$$

step 8: **IfNoEnd()**$\longrightarrow$**GoTo(Step2)**
END

A single "round" of the ADPO algorithm progresses in seven steps. At the beginning, pigs are "sent" to $NP = 20$ random places. This number of locations (20) resulted in the

best overall performance in both the experiments reported in what follows and in other completed ones. However, it is important to note that NP should be seen as one of the hyperparameters of the proposed approach. In other words, for different optimization problems this number may vary and cannot be treated as a "universal constant". For instance, increasing complexity of the problem, combined with the need to find a solution within a larger space may (naturally) affect the size of the "optimal NP". Nevertheless, exploring this aspect is out of the scope of this work.

In the subsequent phase, to mitigate potential undesired outcomes associated with the use of a pseudorandom generator, additional $SP = 10$ scouts are dispatched to distinct locations, where no pigs from the initial step are present. Again, the value of SP is another hyperparameter of this method. Next, all $NP + SP$ found local solutions (amounts of found food) are sorted in descending order. In the fourth step, the best 10% of solutions are collected (becoming "leaders"). Then, the selected "leader pigs" are saved, overwriting the original list of the NP individuals. The memory used to store the intermediate information, which is no longer necessary, is released in the sixth step. In the seventh step, pig movement takes place. Its details have been described above in the context of Figure 5.

In the proposed algorithm, OFNs are utilized to determine the maximum feasible movement of individual pigs. This involves applying the Golden Ratio defuzzifying operator to the coordinates, ordinate, and abscissa, thereby determining the length of the actual motion. Following the assessment of a stopping condition, to ascertain whether further evolution is required, the algorithm may progress to the next "round". However, if a solution is identified as satisfactory, it is returned, and the algorithm concludes.

4. Experimental Setup

Since the current implementation of the ADPO method is new, its efficiency has to be (re)evaluated for a set of well-tested benchmark functions in the 2D space where the accuracy can be easily assessed. Since the reported results are promising, application of ADPO to more complex scenarios is planned in the near future.

To compare the performance of ADPO to the "canonical nature-inspired algorithms", the following methods were selected (names of specific packages used in their implementations are provided in parenthesis):

- PSO—Particle Swarm Optimization (package—*psoptim* [53,54]),
- BAT—Bat Algorithm (package—*microbats* [55]),
- GA—Genetic Algorithm (package—*GA* [43]).

The selection of algorithms for comparison was justified by their relevant characteristics and applications. The Genetic Algorithm (GA) was chosen because it is the oldest, most commonly used, and well-developed metaheuristic, which is highly suitable for both discrete and continuous problems. Particle Swarm Optimization (PSO) was also included due to its effectiveness in solving continuous problems, achieving good results even in cases of very high-dimensional functions. Although Artificial Bee Colony Algorithm (ABC) is typically used for comparisons, Bat Algorithm (BA) was chosen as an alternative, because it is very similar to ABC, while it yields very good results in seeking suboptimal solutions of functions. With such a well-grounded selection of algorithms, it was possible to conduct comprehensive comparisons, taking into account different types of optimization problems.

As noted, the current version of ADPO has been fully re-implemented in *R*. For the remaining approaches, packages from the CRAN repository were used. In this way, it can be claimed that "standard" realization of each method, against which ADPO was compared, has been applied. The IDE RStudio was used as the programming environment.

For the experiments, eight well-known standard benchamrking functions, each with multiple local extrema, and one global extremum, have been selected (see the next section). For each method, 50 experiments were completed. Next, the root-mean error values and the standard deviation were calculated for the found extrema. In what follows, average results are reported. Graphs illustrating the solution search process for each function, along

with charts depicting the values of the discovered minima in relation to the number of iterations, are also provided.

Population size is an important hyperparameter of the configuration of the nature-inspired algorithms. Table 1 summarizes used population sizes. These values were chosen on the basis of the analysis of the literature, where they were found to be the most commonly used for the functions in question. Here, let us recall that the population size parameters for the APDO method were chosen experimentally. However, in the reported experiments the same value was used in all eight benchmark cases. Hence, the selected value (s) can be seen as an overall "reasonable choice" (though, not necessarily the best one for each individual method and function). It can thus be claimed that the reported comparisons are "fair" because in each case the same popular/reasonable population size was used for all benchmark functions.

Table 1. The population size configuration for the algorithms.

	Algorithm	The Size of the Population
1	PSO	40
2	APDO	30
3	BAT	40
4	GA	40

In this context, it is important to stress again that the aim of this work was not to find the most optimal hyperparameters for each method in the case of each individual problem. Note also that the selected functions are very popular standard benchmarks used across the scientific literature, but they are not real-life, large-scale, and/or extremely difficult problems. However, the goal of the experiments reported in this contribution was to answer the following general question: is the APDO method competitive vis-a-vis other popular nature-inspired methods . If the answer was to be positive (which turns out to be the case), then further explorations would make sense and there would be a reason for them to be undertaken. This point needs to be kept in mind when reading what follows.

Benchmark Functions

As noted, the experiments involved the optimization of eight benchmark functions with two unknowns [56,57], aiming at identifying their global minimum within specified ranges of variable values [58,59]. Table 2 furnishes the function names, variable value interval limits, sought values of the two unknowns, and the desired global minimum value.

Table 2. List of benchmark functions used for the experiment.

No.	Function Name	Range of Values $x1$ and $x2$	Searched Values of $x1$ and $x2$	Searched Value of $f(x1, x2)$
1	Easom	$[-100, 100]$	(3.14, 3.14)	-1
2	Eggholder	$[-512, 512]$	(512, 404)	-959
3	Drop Wave	$[-5.12, 5.12]$	(0, 0)	-1
4	Levy N.13	$[-10, 10]$	(1, 1)	0
5	Matyas	$[-10, 10]$	(0, 0)	0
6	Rastrigin	$[-5.12, 5.12]$	(0, 0)	0
7	Zirilli	$[-10, 10]$	$(-1.04, 0)$	-0.35
8	Venter	$[-50, 50]$	(0, 0)	-400

Mathematical equations of the benchmark functions, used in the performed experiments, are as follows:

Eggholder:

$$f(x) = -(x_2 + 47)sin\sqrt{|\frac{x_1}{2} + (x_2 + 47)|} - x_1 sin\sqrt{|x_1 - (x_2 + 47)|} \quad (5)$$

Venter:

$$f(x) = x_1^2 - 100cos(x_1)^2 - 100cos(x_1^2/30) + x_2^2 - 100cos(x_2)^2 - 100cos(x_2^2/30) \quad (6)$$

Matyas:

$$f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2 \quad (7)$$

Zirilli:

$$f(x) = 0.25x_1^4 - 0.5x_1^2 + 0.1x_1 + 0.5x_2^2 \quad (8)$$

Easom:

$$f(x) = -cos(x_1)cos(x_2)exp(-((x_1 - \pi)^2 + (x_2 - \pi)^2)) \quad (9)$$

Rastrigin:

$$f(x) = -cos(x_1)cos(x_2)exp(-((x_1 - \pi)^2 + (x_2 - \pi)^2)) \quad (10)$$

Levy N.13:

$$f(x) = sin^2(3\pi x_1) + (x_1 - 1)^2(1 + sin^2(3\pi x_2)) + (x_2 - 1)^2(1 + sin^2(2\pi x_2)) \quad (11)$$

Drop Wave:

$$f(x) = -\frac{1 + cos(12\sqrt{x_1^2 + x_2^2})}{0.5(x_1^2 + x_2^2) + 2} \quad (12)$$

Finally, in Figures 6 and 7, graphical representations of the eight test functions are presented.

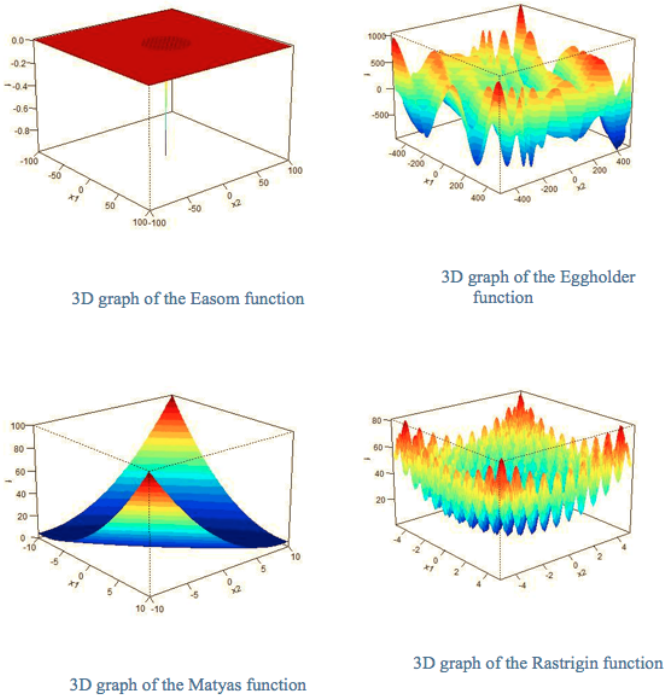

Figure 6. Graphical representation of the first four benchmark functions: Easom, Eggholder, Matyas, Rastrigin.

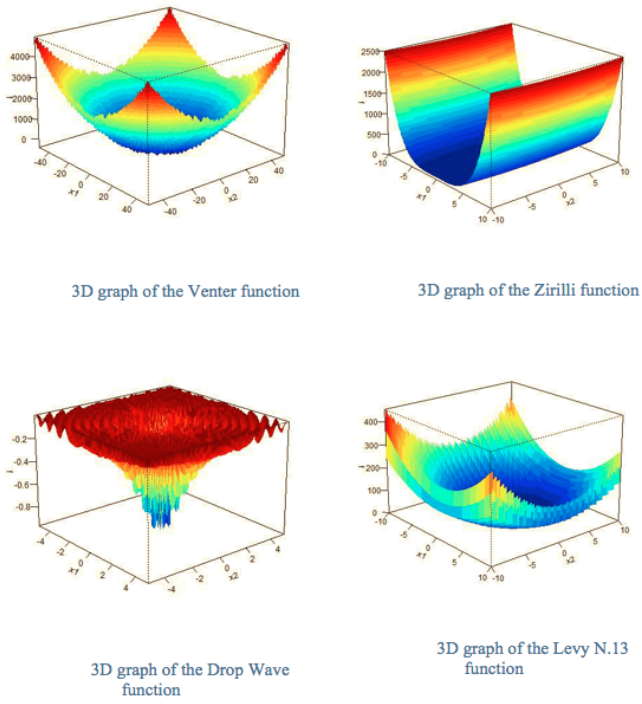

Figure 7. Graphical representation of the remaining benchmark functions: Drop Wave, Levy N.13, Venter, Zirilli.

5. Experimental Results

Let us now discuss the results obtained for the selected benchmarks, using ADPO and the remaining three classical approaches: Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Bat Algorithms (BA). Before proceeding to the discussion of the specific numerical results obtained for each of the benchmark functions, let us compare the times needed to find the solutions.

5.1. Execution Time Comparison

In the case of optimization algorithms, execution time is very important. The average time results from 50 runs of the four algorithms for the eight benchmarking functions are summarized in Table 3.

Table 3. Average algorithm time for functions.

No.	Function Name	PSO	APDO	BAT	GA
1	Eggholder	0.0996	0.06	0.1308	0.122
2	Venter	0.06	0.047	0.06	0.18
3	Matyas	0.0748	0.062	0.0816	0.058
4	Zirilli	0.072	0.066	0.082	0.060
5	Easom	0.07	0.04	0.079	0.098
6	Rastrigin	0.073	0.038	0.076	0.18
7	Levy N.13	0.076	0.04	0.084	0.187
8	Drop Wav	0.077	0.04	0.085	0.147

Overall, it can be observed that the APDO method, in the majority of cases, finds the solution in the shortest average time. Only for the Matyas and the Zirilli functions is the BAT faster. However, it has to be admitted that, in all cases, the computation times are relatively short. Hence, the main observation that can be formulated is as follows: for the standard 2D benchmark functions, for the typical hyperparameter settings, the APDO approach finds the solution in a "competitive time".

Let us now look, separately, into experimental results obtained for the individual benchmark functions.

5.2. Results for the Eggholder Function

Here, "good results" were achieved by the PSO and APDO algorithms, with $f(x)$ ranging from -897 to -959 for PSO, and from -928 to -957 for APDO. These should be compared to the targeted value of -959. Conversely, GA and BAT exhibited significant "instabilities" characterized by large MSE errors and standard deviations spanning several dozen units (see Figure 8). Figure 8 illustrates the performance differences. Here, the line representing the performance of GA, due to the very large error, is completely outside of the visible area (has not been included).

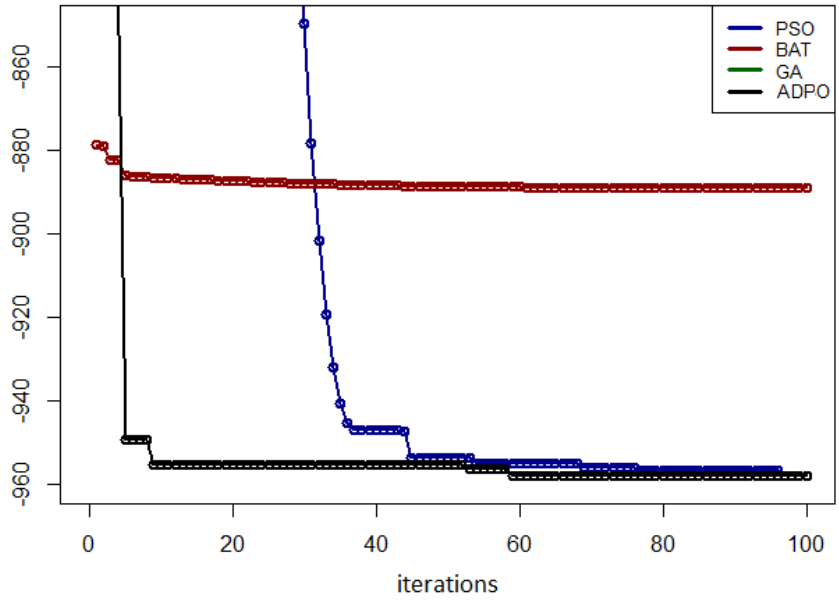

Figure 8. The results, obtained in subsequent iterations, for the Eggholder benchmark function.

5.3. Results for the Venter Function

The best numerical results for the Venter function have been obtained by the APDO method (see, Figure 9). Slightly worse results can be seen for GA and PSO. Clearly, the worst results have been produced by BAT. Here, the best result is $f(x) = -387$, when the actual value is -400. It should be noted, however, that some additional experiments have been performed. In each case, for the three non-APDO algorithms, with the increase in the size of the population (above that reported in Table 1, the results became better (became more accurate). Nevertheless, it was decided that, for the reasons already discussed, in this contribution only results for a single (popular in the literature) population size will be reported. Finally, for the reported population sizes, the APDO is the fastest, while the GA algorithm is the slowest (see Table 3). Obviously, with increases in the population sizes, the time needed to find the solution increases in all cases.

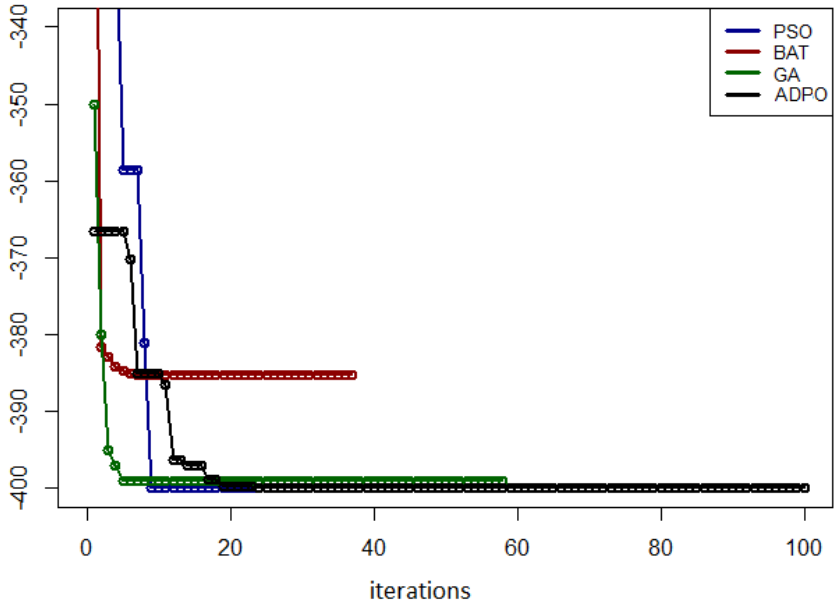

Figure 9. The Venter benchmark function values obtained in successive iterations.

5.4. Results for the Zirilli Function

As can be seen in Figure 10, for the Zirilli function, solutions found by all methods are quite good. Recall that in each case, the "basic" population sizes (as summarized in Table 1) have been used. Here, the only important difference is related to the times of execution. Interestingly, contrary to the case of the Venter function, the GA approach is the fastest (on average).

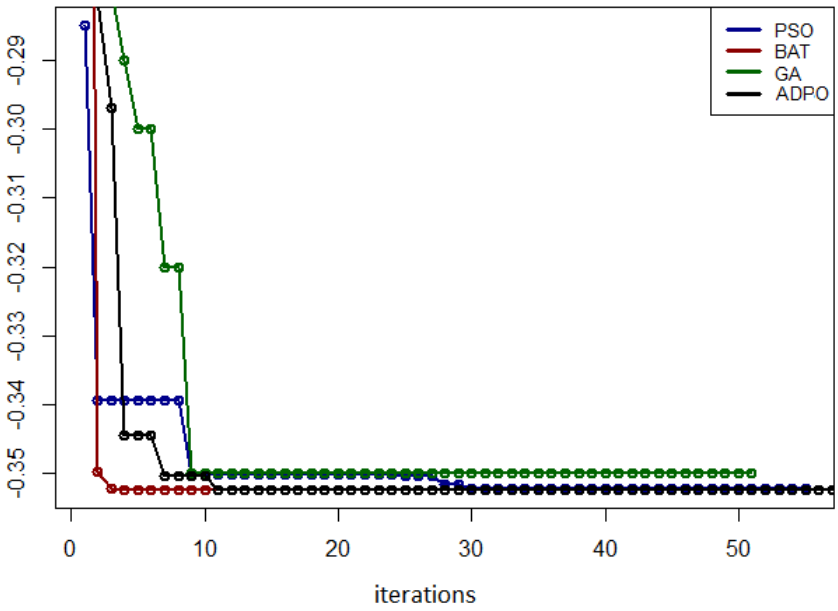

Figure 10. The Zirilli benchmark function values obtained in successive iterations.

5.5. Results for the Easom Function

Overall, again for this function the size of the population was significant for the non-ADPO approaches. For the "baseline populations", PSO, BAT, and GA were quite inaccurate. However, again with the increase in the population size, the quality of the results improved. Here, for the baseline hyperparamenters, GA was the slowest, while ADPO was the fastest approach. The results have been summarized in Figure 11.

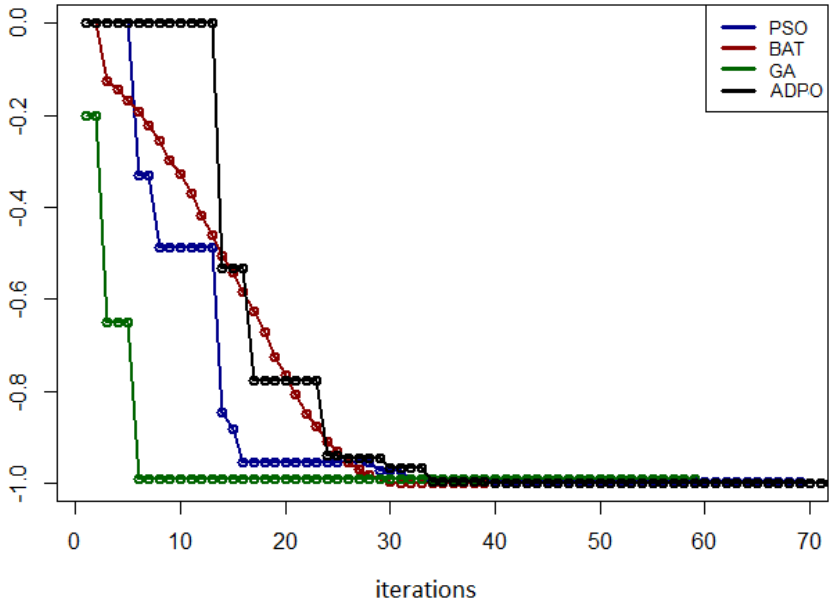

Figure 11. Values of the Easom benchmark function obtained in successive iterations.

5.6. Results for the Drop Wave, Levy N.13, and Rastrigin Functions

The correctness of the Drop Wave (Figure 12), Levy N.13 (Figure 13), and Rastrigin (Figure 14) optimization results is similar for all tested methods. In additional experiments, it was noted that minor errors and noticeable standard deviation were evident only with small population sizes of specifically 20 individuals. Furthermore, these supplementary experiments also revealed the poor quality of BAT optimization, particularly evident in the case of the Rastrigin function, where the reported minimum was 1 with 200 individuals, contrary to the expected value of 0.

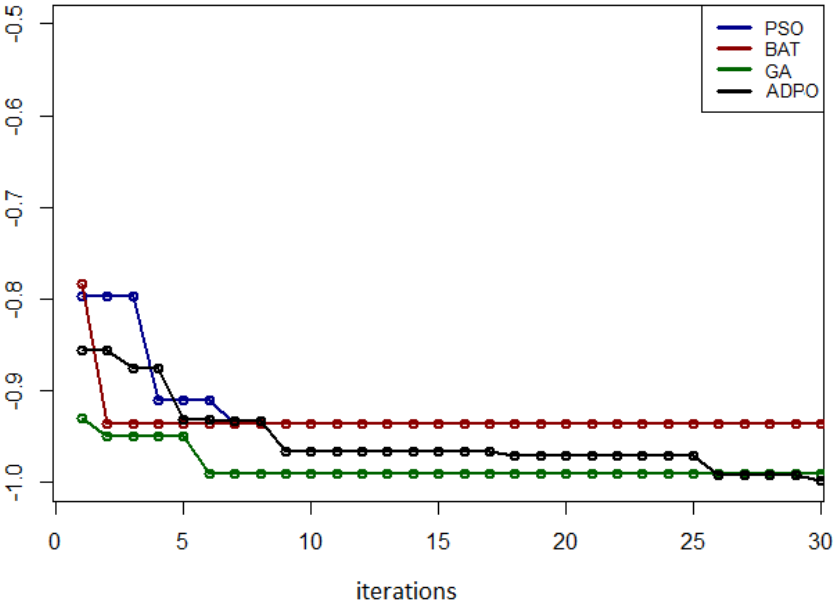

Figure 12. Values of the Drop Wave benchmark function obtained in successive iterations.

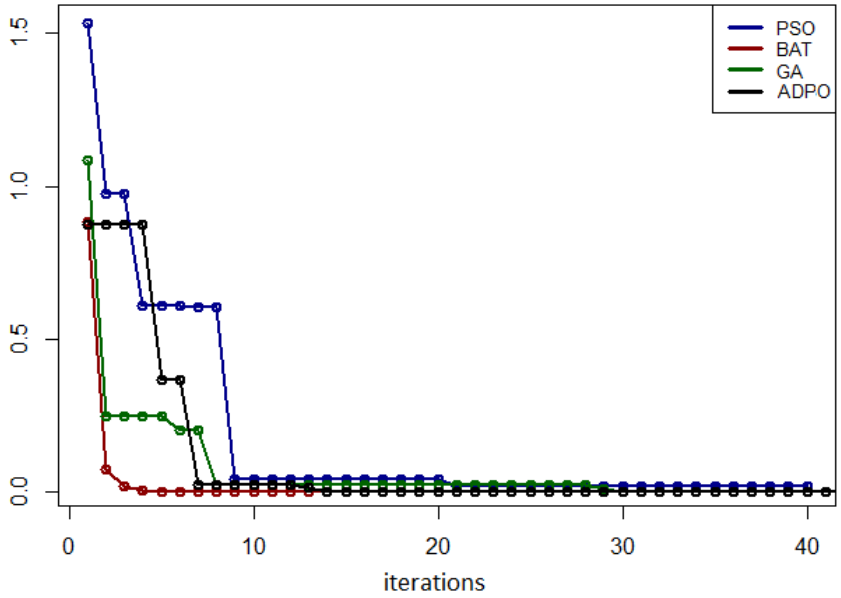

Figure 13. Values of the Levy N.13 benchmark function obtained in successive iterations.

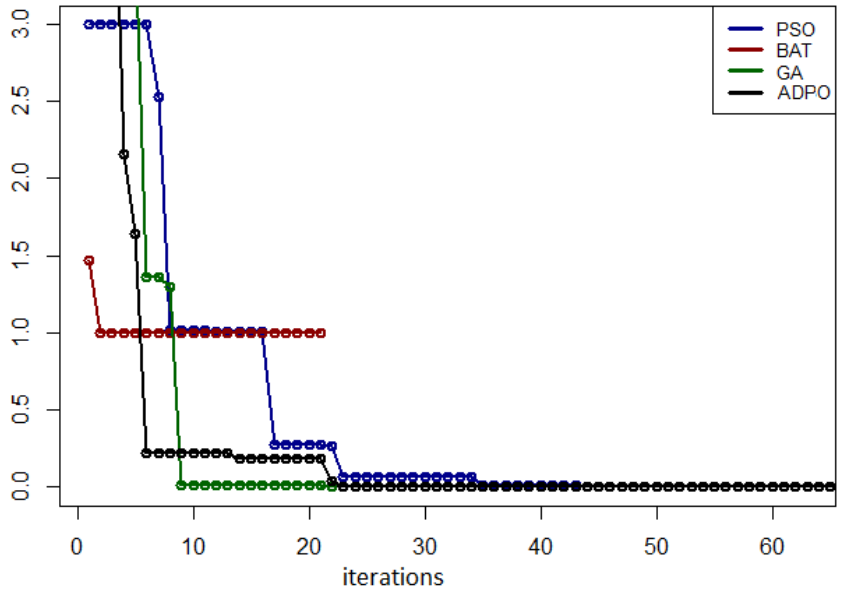

Figure 14. Values of the Rastrigin benchmark function obtained in successive iterations.

5.7. Summary of Experimental Results

Let us now provide the summary of all performed experiments, including those that have not been reported above in the form of tables and figures. Overall, as expected, the quality of the results of the performed optimizations depended on the tested method, the specific benchmark function, and the size of the population. The latter was studied in supplementary experiments, but was not reported (other than the selected observations above), to deliver a focused fair assessment. Naturally, as the population size increased, the accuracy of the obtained results improved in all cases. This improvement was particularly pronounced when utilizing swarm methods with the Eggholder or Venter functions. However, it is important to note that as the number of swarms increased to enhance accuracy, the execution time also significantly escalated (for obvious reasons).

Another interesting factor that influenced the quality of the obtained results was the complexity of the test function "combined" with the range in which the extremes have been sought. Functions with particularly broad ranges of values, e.g., the Eggholder function for

[−512, 512], were the most difficult to optimize. On the other hand, small-scale functions, with a minimum value of 0, or close to it, were often successfully solved even for small populations (e.g., 20). This also suggests that the Eggholder function could be used in subsequent studies, in which population size (and solution finding time) would play a more pronounced role.

It should be noted that, for the standard benchmarks used in the performed experiments, the difficulty of solving the given functions seemed to affect the algorithm's time, particularly in the case of the BAT algorithm. Its performance worsened with the difficulty of the benchmark function. Perhaps this is due to the way of implementing this algorithm, e.g., because of the library from which it was taken. However, investigating this effect is clearly out of the scope of this work.

Experiments have shown that, in most cases, the genetic algorithm (GA) is characterized by a good quality of results when executed for 70 or more iterations. However, typically, this is the slowest method. Moreover, its results for the difficult optimization of the Eggholder function were of very low quality. On the other hand, for this task (the Eggholder function), the particle swarm algorithm (PSO) was the best. However, an adequate accuracy was obtained only with swarms of 100 and larger.

Against this background, the proposed ADPO method distinguished itself by providing reasonably correct results, in a short (or, often, the shortest) time. Here, particularly interesting is the case of the Eggholder function, where ADPO is faster than the PSO method. Most likely, it is related to the fact that in ADPO there is no distinction between global and local neighborhood (as opposed to PSO). In other words, the search is performed by passing information about the best solution, resulting in the herd "moving" in its direction, while also searching the neighborhood. At the same time, for the APDO, each specimen can become the source of the best solution. Random start of specimens from scattered places enables searching larger space solutions. At the same time, if the best solution is found in a given cycle, the remaining specimens can be sent to search all points for the currently optimal solution. In this context, the novelty of the ADPO method is in its use of the Ordered Fuzzy Notation, combined with GR defuzzification, as its inherent part. This allows for a smooth description of the transition of a specimen from one solution to another. Thanks to the use of OFN, the population reaches the desired extreme functions earlier.

6. Concluding Remarks

In the conducted experiments, a new optimization method based on the swarming behavior of Duroc pigs, combined with the use of OFN for modeling numbers and GR defuzzification, yielded satisfactory results. These results are very promising, especially considering that the ADPO population was smaller, while yielding better (or at least similar) results, when compared with the competing classical algorithms. This is a significant finding, as it challenges the conventional belief that larger populations are necessary for better optimization results. The fact that the proposed method performed well with a small population size is noteworthy and could have implications for optimizing computational resources in practical applications.

Furthermore, the experiments considered also the execution time. It was observed that the proposed method yielded good results in terms of speed, indicating its efficiency in producing quality solutions within a reasonable timeframe. This combination of effectiveness and efficiency is a valuable characteristic for any optimization method, as it speaks to its potential practical utility in real-world scenarios, where time and computational resources are often limited. It may also mean that the ADPO can solve more complex problems with resources smaller than in the case of other well-known optimization approaches.

The experiments were conducted in a 2D space, and the results were found to be very promising. This sets the stage for further research, where the space will be expanded to a greater number of dimensions. This expansion is crucial as it will allow for a more comprehensive understanding of the method's performance across different problem spaces, potentially uncovering specific limitations of its versatility and robustness.

Overall, the use of swarming behavior of pigs, as a basis for optimization, turned out to be an intriguing concept. The successful application of this approach, in the context of the performed experiments, underscores the potential of bio-inspired approaches in solving complex optimization problems. Moreover, utilization of OFN for modeling numbers is another noteworthy aspect of the proposed approach. This indicates a departure from traditional numerical modeling techniques and suggests that alternative methods can be effective in optimization. The successful integration of OFN in the proposed method opens possibilities for exploring unconventional mathematical tools in the realm of optimization, potentially leading to innovative approaches and solutions.

In conclusion, the experiment has provided valuable insights into the effectiveness, efficiency, and potential of the proposed new optimization method. The promising results, especially in the context of small population sizes and faster computation, indicate its viability for practical applications. Furthermore, joint exploration of nature-inspired and unconventional mathematical concepts in optimization opens new avenues for future research and innovation in the field. As the experiment moves towards higher dimensions, it is poised to uncover further nuances of the method and its applicability across diverse problem spaces.

Author Contributions: Conceptualization, D.E. and J.M.C.; methodology, J.M.C.; software, D.E.; validation, M.P., S.F. and M.G.; formal analysis, M.P., S.F. and M.G.; investigation, S.F., M.G. and M.P.; resources, D.E.; data curation, J.M.C.; writing—original draft preparation, J.M.C. and D.E.; writing—review and editing, S.F., M.G. and M.P.; visualization, D.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author due to (specify the reason for the restriction).

Acknowledgments: We would like to acknowledge how the involvement of M.P., S.F. and M.G. is a result of a bilateral collaboration between the Polish Academy of Sciences and the Bulgarian Academy of Sciences, through a project titled "Practical Aspects of Scientific Computing".

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Jakšić, Z.; Devi, S.; Jakšić, O.; Guha, K. A Comprehensive Review of Bio-Inspired Optimization Algorithms Including Applications in Microelectronics and Nanophotonics. *Biomimetics* **2023**, *8*, 278. [CrossRef] [PubMed]
2. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
3. Fister, I.; Fister, I., Jr.; Yang, X.S.; Brest, J. A comprehensive review of firefly algorithms. *Swarm Evol. Comput.* **2013**, *13*, 34–46. [CrossRef]
4. Jiang, Y.; Wu, Q.; Zhu, S.; Zhang, L. Orca predation algorithm: A novel bio-inspired algorithm for global optimization problems. *Expert Syst. Appl.* **2022**, *188*, 116026. [CrossRef]
5. Golilarz, N.A.; Gao, H.; Addeh, A.; Pirasteh, S. ORCA optimization algorithm: A new meta-heuristic tool for complex optimization problems. In Proceedings of the 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 18–20 December 2020; IEEE: New York, NY, USA, 2020; pp. 198–204.
6. Drias, H.; Drias, Y.; Khennak, I. A new swarm algorithm based on orcas intelligence for solving maze problems. In Proceedings of the Trends and Innovations in Information Systems and Technologies, Budva, Montenegro, 7–10 April 2020; Springer: Berlin/Heidelberg, Germany, 2020; Volume 1, pp. 788–797. [CrossRef]
7. Cavagna, A.; Cimarelli, A.; Giardina, I.; Parisi, G.; Santagati, R.; Stefanini, F.; Viale, M. Scale-free correlations in starling flocks. *Proc. Natl. Acad. Sci. USA* **2010**, *107*, 11865–11870. [CrossRef] [PubMed]
8. Chu, H.; Yi, J.; Yang, F. Chaos particle swarm optimization enhancement algorithm for UAV safe path planning. *Appl. Sci.* **2022**, *12*, 8977. [CrossRef]
9. Zamani, H.; Nadimi-Shahraki, M.H.; Gandomi, A.H. Starling murmuration optimizer: A novel bio-inspired algorithm for global and engineering optimization. *Comput. Methods Appl. Mech. Eng.* **2022**, *392*, 114616. [CrossRef]
10. Oftadeh, R.; Mahjoob, M.; Shariatpanahi, M. A novel meta-heuristic optimization algorithm inspired by group hunting of animals: Hunting search. *Comput. Math. Appl.* **2010**, *60*, 2087–2098. [CrossRef]
11. Adhirai, S.; Mahapatra, R.P.; Singh, P. The Whale Optimization Algorithm and Its Implementation in MATLAB. *Int. J. Comput. Inf. Eng.* **2018**, *12*, 815–822.

12. Rohani, M.R.; Shafabakhsh, G.A.; Asnaashari, E. The Workflow Planning of Construction Sites Using Whale Optimization Algorithm (WOA).The Turkish Online Journal of Design, Art and Communication-TOJDAC November 2016 Special Edition 2016. Available online: http://www.tojdac.org/tojdac/VOLUME6-NOVSPCL_files/tojdac_v060NVSE207.pdf (accessed on 22 January 2024).
13. Yang, X.S. Firefly Algorithms for Multimodal Optimization. In Proceedings of the Stochastic Algorithms: Foundations and Applications, Sapporo, Japan, 26–28 October 2009; pp. 169–178.
14. Shadravan, S.; Naji, H.; Bardsiri, V. The Sailfish Optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. *Eng. Appl. Artif. Intell.* **2019**, *80*, 20–34. [CrossRef]
15. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [CrossRef]
16. Suganthi, M.; Madheswaran, M. An improved medical decision support system to identify the breast cancer using mammogram. *J. Med. Syst.* **2012**, *36*, 79–91. [CrossRef] [PubMed]
17. Wang, L.; Shen, J.; Yong, J. A survey on bio-inspired algorithms for web service composition. In Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Wuhan, China, 23–25 May 2012; pp. 569–574. [CrossRef]
18. Shalamov, V.; Filchenkov, A.; Shalyto, A. Heuristic and metaheuristic solutions of pickup and delivery problem for self-driving taxi routing. *Evol. Syst.* **2019**, *10*, 3–11. [CrossRef]
19. Ewald, D.; Czerniak, J.M.; Paprzycki, M. A New OFNBee Method as an Example of Fuzzy Observance Applied for ABC Optimization. In *Theory and Applications of Ordered Fuzzy Numbers*; Springer: Berlin/Heidelberg, Germany, 2017; p. 223.
20. Ewald, D.; Zarzycki, H.; Apiecionek, L.; Czerniak, J.M. Ordered fuzzy numbers applied in bee swarm optimization systems. *J. Univers. Comput. Sci.* **2020**, *26*, 1475–1494. [CrossRef]
21. Ragmani, A.; Elomri, A.; Abghour, N.; Moussaid, K.; Rida, M. An improved Hybrid Fuzzy-Ant Colony Algorithm Applied to Load Balancing in Cloud Computing Environment. In Proceedings of the 10th International Conference on Ambient Systems, Networks and Technologies (ANT 2019), Leuven, Belgium, 29 April 29–2 May 2019; Volume 151, pp. 519–526. [CrossRef]
22. Grandin, T.; Curtis, S.; Greenough, W. Effects of rearing environment on the behaviour of young pigs. *J. Anim. Sci.* **1983**, *57*, 137.
23. McGlone, J.; Curtis, S.E. Behavior and Performance of Weanling Pigs in Pens Equipped with Hide Areas. *J. Anim. Sci.* **1985**, *60*, 20–24. [CrossRef] [PubMed]
24. Mrozek, D.; Dabek, T.; Małysiak-Mrozek, B. Scalable Extraction of Big Macromolecular Data in Azure Data Lake Environment. *Molecules* **2019**, *24*, 179. [CrossRef] [PubMed]
25. Grandin, T.; Curtis, S. Toy preferences in young pigs. *J. Anim. Sci.* **1984**, *59*, 85.
26. Pettigrew, J.E. Essential role for simulation models in animal research and application. *Anim. Prod. Sci.* **2018**, *58*, 704–708. [CrossRef]
27. de Silva Ramos-Freitas, L.C.; Torres-Campos, A.; Schiassi, L.; Yanagi-Júnior, T.; Cecchin, D. Fuzzy index for swine thermal comfort at nursery stage based on behavior. *DYNA* **2017**, *84*, 201–207. [CrossRef]
28. Harris, A.; Patience, J.; Lonergan, S.; Dekkers, J.; Gabler, N. Improved nutrient digestibility and retention partially explains feed efficiency gains in pigs selected for low residual feed intake. *J. Anim. Sci.* **2013**, *90*, 164–166. [CrossRef] [PubMed]
29. Held, S.; Mason, G.; Mendl, M. Using the Piglet Scream Test to enhance piglet survival on farms: Data from outdoor sows. *Anim. Welf.* **2007**, *16*, 267–271. [CrossRef]
30. Patel, B.; Chen, H.; Ahuja, A.; Krieger, J.F.; Noblet, J.; Chambers, S.; Kassab, G.S. Constitutive modeling of the passive inflation-extension behavior of the swine colon. *J. Mech. Behav. Biomed. Mater.* **2017**, *77*, 176–186. [CrossRef] [PubMed]
31. Dyczkowski, K. A Less Cumulative Algorithm of Mining Linguistic Browsing Patterns in the World Wide Web. In Proceedings of the 5th EUSFLAT Conference, Ostrava, Czech Republic, 11–14 September 2007.
32. Stachowiak, A.; Dyczkowski, K. A Similarity Measure with Uncertainty for Incompletely Known Fuzzy Sets. In Proceedings of the 2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), Edmonton, AB, Canada, 24–28 June 2013; pp. 390–394.
33. Marszalek, A.; Burczynski, T. Modeling and forecasting financial time series with ordered fuzzy candlesticks. *Inf. Sci.* **2014**, *273*, 144–155. [CrossRef]
34. Prokopowicz, P.; Czerniak, J.; Mikolajewski, D.; Apiecionek, L.; Slezak, D. *Theory and Applications of Ordered Fuzzy Numbers. A Tribute to Professor Witold Kosiński*; Studies in Fuzziness and Soft Computing; Springer International Publishing: Cham, Switzerland, 2017; Volume 365.
35. Kosinski, W.; Słysz, P. Fuzzy Numbers and Their Quotient Space with Algebraic Operations. *Bull. Pol. Acad. Sci. Math.* **1993**, *41*, 285–295.
36. Kosinski, W.; Prokopowicz, P.; Slezak, D. Fuzzy Reals with Algebraic Operations: Algorithmic Approach. In Proceedings of the IIS 2002 Symposium, Sopot, Poland, 3–6 June 2002; pp. 311–320.
37. Kosinski, W. On fuzzy number calculus. *Int. J. Appl. Math. Comput. Sci.* **2006**, *16*, 51–57.
38. Kosinski, W.; Prokopowicz, P.; Slezak, D. Ordered Fuzzy Numbers. *Bull. Pol. Acad. Sci. Math.* **2003**, *51*, 327–338.
39. Kosinski, W.; Frischmuth, K.; Wilczyńska-Sztyma, D. A New Fuzzy Approach to Ordinary Differential Equations. In Proceedings of the ICAISC 2010, Zakopane, Poland, 13–17 June 2010; Part I; Volume 6113, pp. 120–127.

40. Kosinski, W.; Prokopowicz, P.; Slezak, D. Algebraic Operations on Fuzzy Numbers. In Proceedings of the IIS 2003, Zakopane, Poland, 2–5 June 2003; Kłopotek, M.A., Wierzchoń, S.T., Trojanowski, K., Eds.; Advances in Soft Computing; Springer: Berlin/Heidelberg, Germany, 2003; pp. 353–362.
41. Kosinski, W.; Prokopowicz, P.; Slezak, D. Calculus with Fuzzy Numbers. In *Intelligent Media Technology for Communicative Intelligence*; Bolc, L., Michalewicz, Z., Nishida, T., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3490, pp. 21–28. [CrossRef]
42. Kosinski, W.; Prokopowicz, P.; Kacprzak, D. Fuzziness—Representation of Dynamic Changes by Ordered Fuzzy Numbers. In *Views on Fuzzy Sets and Systems from Different Perspectives: Philosophy and Logic, Criticisms and Applications*; Seising, R., Ed.; Studies in Fuzziness and Soft Computing; Springer: Berlin/Heidelberg, Germany, 2009; Volume 243, pp. 485–508.
43. Kosinski, W. Evolutionary algorithm determining defuzzyfication operators. *Eng. Appl. Artif. Intell.* **2007**, *20*, 619–627. [CrossRef]
44. Kosinski, W.; Prokopowicz, P.; Slezak, D. On Algebraic Operations on Fuzzy Reals. In *Neural Networks and Soft Computing, Proceedings of the Sixth International Conference on Neural Networks and Soft Computing, Zakopane, Poland, 11–15 June 2002*; Rutkowski, L., Kacprzyk, J., Eds.; Physica-Verlag HD: Heidelberg, Germany, 2003; pp. 54–61. [CrossRef]
45. Czerniak, J.M.; Zarzycki, H.; Ewald, D.; Augustyn, P. Application of OFN Numbers in the Artificial Duroc Pigs Optimization (ADPO) Method. In Proceedings of the Uncertainty and Imprecision in Decision Making and Decision Support: New Challenges, Solutions and Perspectives, Warsaw, Poland, 12–14 October 2018; pp. 310–327.
46. Zarzycki, H.; Apiecionek, Ł.; Czerniak, J.M.; Ewald, D. The Proposal of Fuzzy Observation and Detection of Massive Data DDOS Attack Threat. In Proceedings of the Uncertainty and Imprecision in Decision Making and Decision Support: New Challenges, Solutions and Perspectives, Warsaw, Poland, 24–28 October 2021; pp. 363–378.
47. Mikolajewska, E.; Mikolajewski, D. The prospects of brain—Computer interface applications in children. *Cent. Eur. J. Med.* **2014**, *9*, 74–79. [CrossRef]
48. Mikolajewska, E.; Mikolajewski, D. Wheelchair Development from the Perspective of Physical Therapists and Biomedical Engineers. *Adv. Clin. Exp. Med.* **2010**, *19*, 771–776.
49. Chwastyk, A.; Pisz, I. OFN Capital Budgeting Under Uncertainty and Risk. In *Theory and Applications of Ordered Fuzzy Numbers: A Tribute to Professor Witold Kosiński*; Prokopowicz, P., Czerniak, J., Mikołajewski, D., Apiecionek, Ł., Ślęzak, D., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 157–169. [CrossRef]
50. Kacprzak, D. Input-Output Model Based on Ordered Fuzzy Numbers. In *Theory and Applications of Ordered Fuzzy Numbers: A Tribute to Professor Witold Kosiński*; Prokopowicz, P., Czerniak, J., Mikołajewski, D., Apiecionek, Ł., Ślezak, D., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 171–182. [CrossRef]
51. Kacprzak, M.; Starosta, B. Two Approaches to Fuzzy Implication. In *Theory and Applications of Ordered Fuzzy Numbers: A Tribute to Professor Witold Kosiński*; Prokopowicz, P., Czerniak, J., Mikołajewski, D., Apiecionek, Ł., Ślęzak, D., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 133–154. [CrossRef]
52. Dobrosielski, W.; Czerniak, J.; Szczepanski, J.; Zarzycki, H. Two New Defuzzification Methods Useful for Different Fuzzy Arithmetics. In *Uncertainty and Imprecision in Decision Making and Decision Support: Cross-Fertilization, New Models and Applications. IWIFSGN 2016*; Advances in Intelligent Systems and Computing; Springer: Berlin/Heidelberg, Germany, 2018; Volume 559, pp. 83–101.
53. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the Neural Networks, Perth, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
54. Shi, Y.; Eberhart, R. A modified particle swarm optimizer. In Proceedings of the 1998 IEEE International Conference on Evolutionary Computation Proceedings, Anchorage, AK, USA, 4–9 May 1998; pp. 69–73. [CrossRef]
55. Eberhart, R.C.; Shi, Y.; Kennedy, J. Swarm Intelligence. In *Proceedings of the Morgan Kaufmann Series on Evolutionary Computation*, 1st ed.; Morgan Kaufman: Burlington, MA, USA, 2001.
56. Szmidt, E.; Kacprzyk, J. Distances between intuitionistic fuzzy sets. *Fuzzy Sets Syst.* **2000**, *114*, 505–518. [CrossRef]
57. Kacprzyk, J.; Wilbik, A. Using Fuzzy Linguistic Summaries for the Comparison of Time Series: An application to the analysis of investment fund quotations. In Proceedings of the IFSA/EUSFLAT Conference, Lisbon, Portugal, 20–24 July 2009; pp. 1321–1326.
58. Piegat, A.; Pluciński, M. Computing with words with the use of inverse RDM models of membership functions. *Int. J. Appl. Math. Comput. Sci.* **2015**, *25*, 675–688. [CrossRef]
59. Zadrozny, S.; Kacprzyk, J. On the use of linguistic summaries for text categorization. In Proceedings of the IPMU, Perugia, Italy, 1 July 2004; pp. 1373–1380.