Hawk-Dove Dynamics in Continuous Spaces: Escaping Local Optima via Game-Theoretic Swarm Regulation

Jakub Dras[1] and Piotr Pilszczek[1]

[1] *Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44100 Gliwice, POLAND*
Email: jd309975@student.polsl.pl, pp310219@student.polsl.pl

Abstract

Heuristic algorithms like Particle Swarm Optimization (PSO), Grey Wolf Optimizer (GWO) or Whale Optimization Algorithm (WOA) have always been widely studied to provide a sufficient solution for a variety of optimization problems. However, while they are accurate enough, their stability and achievements can be greatly improved for problems regarding functions that have a dynamic local optimum. In this article, we utilize results obtained from game theory -- specifically the Hawk-Dove game -- so that their application will eliminate randomization techniques, and instead, supply heuristics with well-tested and individual, function-dependent search mechanisms. Overall, the results prove accuracy and stability increases in all three modified algorithms, while keeping the complexities close to their original values.

Keywords

PSO, GWO, WOA, Hawk-Dove, Game Theory, Heuristics

1. Introduction

Heuristic algorithms like these have always been a reliable and efficient way to obtain local extrema without making multiple unnecessary calculations. There exist several variants, from basic ones like Particle Swarm Optimization, to more advanced and state-of-the-art solutions such as the Gray Wolf Optimizer and Whale Optimization Algorithm.

In this paper, we discuss multiple proposed game-theoretic modifications to the well-known algorithms.

All of the heuristic algorithms used as subjects of our modification have been modified with the same set of rules. The implementation of the Hawk-Dove game is an effect of an original method that treats separate objects coming from the algorithm's population as different types of entities, thus applying different strategies to each of them. Ultimately, at the end of each iteration, we utilize the objects' positions so that the game simulation occurs only where it is relevant.

As a consequence of the assumptions described below, each algorithm results not only in an extreme value, but it also trains the Hawk-Dove ratio within its population. The main motivation of this paper is based on the observations obtained while applying the described set of rules, and is to research how accurate and stable the algorithms can become and how much variety we can introduce to the examined functions.

*IVUS 2026: Information Society and University Studies 2026, May 12, Kaunas, Lithuania

jd309975@student.polsl.pl (J. Dras); pp3102l9@student.polsl.pl (P. Pilszczek)

Copyright © 2026 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0)

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-06

CEUR-WS.org/Vol-4317/paper4.pdf

2. Methodology and Algorithms

In this project, we focus on optimizing two-variable real functions (later referenced as cost functions):

$$f: [-a, a] \times [-a, a] \to \mathbb{R},$$

where $a > 0$. We will denote the cost function domain as D. We denote the number of tries as T, the number of iterations as I and the population size as P. We will also denote the hawk type as $\mathcal{H}$ and dove type as $\mathcal{D}$ so that the set of entity types is $\{\mathcal{H}, \mathcal{D}\}$.

2.1. The Hawk–Dove Game

As previously mentioned, the modifications are based on the Hawk–Dove game. Therefore, we use the winning value V and cost value C such that $0 < V < C$ to simulate interactions between entities. The Hawk–Dove game payoff function:

$$\text{HD}: \{\mathcal{H}, \mathcal{D}\} \times \{\mathcal{H}, \mathcal{D}\} \to \mathbb{R}^2$$

can be represented as the game solution given by Table 1 below.

Table 1. Payoff values for the Hawk-Dove game.

	Hawk	**Dove**
Hawk	((V-C)/2, (V-C)/2)	(V, 0)
Dove	(0, V)	(V/2, V/2)

Each game will be used to simulate the hawk ratio within the population over time. This ratio, after the i-th iteration, can be denoted as $r_{\mathcal{H}}(i) \in [0,1]$ where $i \in \{1, \dots, I\}$ is the iteration. The starting population will always have a set ratio of hawks $r_{\mathcal{H}}(0)$.

2.2. Particle Swarm Optimization + Hawk–Dove Game (PSO–HD)

The basic algorithm relies on three constants: inertia weight $w \in \mathbb{R}_{>0}$, cognitive coefficient $c_1 \in \mathbb{R}_{>0}$ and the social coefficient $c_2 \in \mathbb{R}_{>0}$. Its main idea is to adjust its population $\{p_1^{(j)}, \dots, p_P^{(j)}\}$ during the j-th iteration where $p_i^{(j)} = \left(x_i^{(j)}, v_i^{(j)}, \overline{x}_i^{(j)}\right)$ and where $x_i^{(j)} \in D$ is the position of entity i in the j-th iteration, $v_i^{(j)} \in \mathbb{R}^2$ is its velocity and $\overline{x}_i^{(j)} \in D$ is its best position. The adjustment is made using the following velocity update:

$$v_i^{(j)} = w \cdot v_i^{(j-1)} + c_1 \cdot r_1^{(j)} \cdot \left(\overline{x}_i^{(j-1)} - x_i^{(j-1)}\right) + c_2 \cdot r_2^{(j)} \cdot \left(g^{(j-1)} - x_i^{(j-1)}\right),$$

where $r_1^{(j)}, r_2^{(j)} \in (0,1)$ are random and $g^{(j)}$ is the global best position in the j-th iteration. Next, we calculate the norm:

$$\left\| v_i^{(j)} \right\| = \sqrt{\left(v_{i,1}^{(j)}\right)^2 + \left(v_{i,2}^{(j)}\right)^2}.$$

Then we limit the velocity based on a set parameter $v_{\max} \in \mathbb{R}_{>0}$ as follows:

$$v_i^{(j)} = \begin{cases} v_i^{(j)}, & \text{if } \| v_i^{(j)} \| \leq v_{\max}, \\ v_i^{(j)} \cdot \dfrac{v_{\max}}{\| v_i^{(j)} \|}, & \text{if } \| v_i^{(j)} \| > v_{\max}. \end{cases}$$

We then apply the velocity to the entity positions using the following formula:

$$x_i^{(j)} = x_i^{(j-1)} + v_i^{(j)}.$$

2.2.1. PSO–HD Modification

The modification introduces a new population $\{p'^{(j)}_1, \ldots, p'^{(j)}_P\}$ such that $p'^{(j)}_i = \left(x_i^{(j)}, v_i^{(j)}, \overline{x}_i^{(j)}, t_i^{(j)}\right)$ where $t_i^{(j)} \in \{\mathcal{H}, \mathcal{D}\}$ is the i-th entity type in the j-th iteration. It also transforms the constants c_1, c_2 and $v_{\max}$ into functions $c_1, c_2, v_{\max}: \{\mathcal{H}, \mathcal{D}\} \to \mathbb{R}_{>0}$ dependent on the type of a given entity. The population adjustment is a modification of the PSO velocity update:

$$v_i^{(j)} = w \cdot v_i^{(j-1)} + c_1\left(t_i^{(j-1)}\right) \cdot r_1^{(j)} \cdot \left(\overline{x}_i^{(j-1)} - x_i^{(j-1)}\right) + c_2\left(t_i^{(j-1)}\right) \cdot r_2^{(j)} \cdot \left(g^{(j-1)} - x_i^{(j-1)}\right).$$

The velocity is being corrected using:

$$v_i^{(j)} = \begin{cases} v_i^{(j)}, & \text{if } \left\|v_i^{(j)}\right\| \le v_{\max}\left(t_i^{(j-1)}\right), \\ v_i^{(j)} \cdot \dfrac{v_{\max}}{\left\|v_i^{(j)}\right\|}, & \text{if } \left\|v_i^{(j)}\right\| > v_{\max}\left(t_i^{(j-1)}\right). \end{cases}$$

Afterwards, we update the entity positions. We then run a Hawk–Dove game simulation for every entity $p'^{(j)}_i$ and its neighbors in the j-th iteration with a radius $n_0 = n_0(D)$ with which it has not yet interacted $\mathcal{N}_0^{(j)}(i)$ where:

$$\mathcal{N}_0^{(j)}(i) \subset \mathcal{N}^{(j)}(i) = \{k: \left\|x_i^{(j)} - x_k^{(j)}\right\| < n_0\} \backslash \{i\}$$

and obtain a Hawk–Dove game score for those entities:

$$S_i^{(j)} = \sum_{k \in \mathcal{N}_0^{(j)}(i)} \mathrm{HD}_{i,k}^{(j)}, \quad \text{where } \mathrm{HD}\left(t_i^{(j)}, t_k^{(j)}\right) = \left[\mathrm{HD}_{i,k}^{(j)}, \mathrm{HD}_{k,i}^{(j)}\right].$$

Now, for each i-th entity, we adopt the strategies of the winning entities for at most $l_0 \in \mathbb{N}_0$ random pairs $\left(p'^{(j)}_i, p'^{(j)}_{k_l}\right)$ where $k_l \in \mathcal{N}^{(j)}(i)$:

$$t_i^{(j)} = \begin{cases} t_i^{(j-1)}, & S_i^{(j)} > S_k^{(j)}, \\ t_k^{(j-1)}, & S_i^{(j)} \le S_k^{(j)}. \end{cases}$$

2.3. Gray Wolf Optimizer + Hawk–Dove Game (GWO–HD)

The main concept of the base algorithm is to adjust its population of wolves $\{w_1^{(j)}, \ldots, w_P^{(j)}\}$ during the j-th iteration where entity $w_i^{(j)} = \left(x_i^{(j)}\right) \in D$ represents a position. The adjustment is made by first selecting three dominating entities $\alpha^{(j)}, \beta^{(j)}, \delta^{(j)}$ (meaning the entities with the best global positions in the j-th iteration), then calculating vectors $X_{e,i}^{(j)}$ for each wolf and finally updating the wolf position with:

$$x_i^{(j)} = \frac{X_{\alpha,i}^{(j)} + X_{\beta,i}^{(j)} + X_{\delta,i}^{(j)}}{3}.$$

Both modifications described below introduce a new population $\{w'^{(j)}_1, \ldots, w'^{(j)}_P\}$ such that $w'^{(j)}_i = \left(x_i^{(j)}, t_i^{(j)}\right)$ where $t_i^{(j)} \in \{\mathcal{H}, \mathcal{D}\}$ is the i-th entity type in the j-th iteration. We will also introduce the concept of ω-wolves such that each ω-wolf is an entity that is not one of the three dominating wolves (α, β or δ).

2.3.1. Alpha–Noise Variation

In this variation, wolves will have two different position update strategies. Let $w_i^{(j)}$ be an ω-wolf with type $t_i^{(j)} = \mathcal{H}$. We will update its position based on the following equation:

$$x_i^{(j)} = X_{\alpha,i}^{(j)} + \xi_i^{(j)},$$

where $\xi_i^{(j)} \in \mathbb{R}^2$ is a random component of the i-th entity in the j-th iteration. All remaining wolves will have their positions updated using the standard GWO formula. After each iteration, we run the Hawk–Dove game simulation.

2.3.2. Scout Variation

This variation also introduces two different ways of updating entity positions. This time, the update of an ω-wolf $w_i^{(j)}$ of type $t_i^{(j)} = \mathcal{H}$ will be based on:

$$x_i^{(j)} = x_r^{(j-1)} - A_{\text{scout}}^{(j)} D_{\text{scout}}^{(j)},$$

where $r \in \{1, \ldots, P\}$ is a random index, $A_{\text{scout}}^{(j)} \in (-2,2)$ is a random value and $D_{\text{scout}}^{(j)} = \left|2r_1 x_r^{(j-1)} - x_i^{(j-1)}\right|$ where $r_1^{(j)} \in (0,1)$ is random. All remaining wolves will have their positions updated using the standard formula. After each iteration, we run the Hawk–Dove game simulation.

2.4. Whale Optimization Algorithm + Hawk–Dove Game (WOA–HD)

The main idea of the base algorithm is to adjust its population of whales $\{w_1^{(j)}, \ldots, w_P^{(j)}\}$ during the j-th iteration where entity $w_i^{(j)} = \left(x_i^{(j)}\right) \in D$ represents a position. The adjustment is made by first calculating appropriate coefficients $A_i^{(j)}$ and $C_i^{(j)}$ for each i-th entity and then updating its position:

$$x_i^{(j)} = \begin{cases} g^{(j-1)} - A_i^{(j)} \left|C_i^{(j)} g^{(j-1)} - x_i^{(j-1)}\right|, & \text{if } p_i^{(j)} < 0.5 \text{ and } \left|A_i^{(j)}\right| < 1, \\ x_r^{(j-1)} - A_i^{(j)} \left|C_i^{(j)} x_r^{(j-1)} - x_i^{(j-1)}\right|, & \text{if } p_i^{(j)} < 0.5 \text{ and } \left|A_i^{(j)}\right| \geq 1, \\ g^{(j-1)} + \left|g^{(j-1)} - x_i^{(j-1)}\right| e^{bl^{(j)}} \cos\left(2\pi l^{(j)}\right), & \text{if } p_i^{(j)} \geq 0.5, \end{cases}$$

where $r \in \{1, \ldots, P\}$ is a random index, $p_i^{(j)} \in (0,1)$ and $l^{(j)} \in \left(-1 - \frac{j}{I}, 1\right)$ are random values, $b \in \mathbb{R}$ is a parameter and $g^{(j)} \in D$ is the global best position in the j-th iteration.

Both modifications described below introduce a new population $\{w'^{(j)}_1, \ldots, w'^{(j)}_P\}$ such that $w'^{(j)}_i = \left(x_i^{(j)}, t_i^{(j)}\right)$ where $t_i^{(j)} \in \{\mathcal{H}, \mathcal{D}\}$ is the i-th entity type in the j-th iteration.

2.4.1. p-Strategy Variation

In this modification, we eliminate the random component by replacing its usage with the appropriate strategy check. The updated formula receives the following form:

$$x_i^{(j)} = \begin{cases} g^{(j-1)} - A_i^{(j)} \left|C_i^{(j)} g^{(j-1)} - x_i^{(j-1)}\right|, & \text{if } t_i^{(j-1)} = \mathcal{D} \text{ and } \left|A_i^{(j)}\right| < 1, \\ x_r^{(j-1)} - A_i^{(j)} \left|C_i^{(j)} x_r^{(j-1)} - x_i^{(j-1)}\right|, & \text{if } t_i^{(j-1)} = \mathcal{D} \text{ and } \left|A_i^{(j)}\right| \geq 1, \\ g^{(j-1)} + \left|g^{(j-1)} - x_i^{(j-1)}\right| e^{bl^{(j)}} \cos\left(2\pi l^{(j)}\right), & \text{if } t_i^{(j-1)} = \mathcal{H}. \end{cases}$$

After each iteration, we run the Hawk–Dove game simulation.

2.4.2. Scout Variation

In this variation, we update all entities with type $t_i^{(j)} = \mathcal{D}$ as described in the WOA formula. However, entities of type $t_i^{(j)} = \mathcal{H}$ will be updated according to:

$$x_i^{(j)} = x_r^{(j-1)} - A_{\text{scout}}^{(j)} \left|C_i^{(j)} x_r^{(j-1)} - x_i^{(j-1)}\right|,$$

where $r \in \{1, \ldots, P\}$ is a random index and $A_{\text{scout}}^{(j)} \in (-2,2)$ is a random value. After each iteration, we run the Hawk–Dove game simulation.

3. Experiments

All the experiments were run using two selected cost functions:

1. the Schwefel static function:

$$\text{Schwefel}(x_1, x_2) = 418.9829 \cdot 2 - \left(x_1 \sin\left(\sqrt{|x_1|}\right) + x_2 \sin\left(\sqrt{|x_2|}\right)\right)$$

defined on the domain $D = [-500,500]^2$ and having the following global minimum: $\text{Schwefel}(420.9687,420.9687) = 0$,

2. the Chaos dynamic function (a modification of the Eggholder function):

$$\text{Chaos}^{(i)}(x_1, x_2) = \frac{x_1^2 + x_2^2}{100} - (x_2 + i)\sin\left(\sqrt{|x_1 + x_2 + i|}\right) - (x_1 - i)\sin\left(\sqrt{|x_1 - x_2 - i|}\right)$$

defined on the domain $D = [-100,100]^2$, where i is the current iteration of the heuristic algorithm.

We adopted the following parameters for all the performance tests: $T = 25$, $I = 200$, $P = 200$, $V = 40$, $C = 70$, thus the expected result for the dynamic Chaos function is the following minimum in the last iteration: $\text{Chaos}^{(200)}(-49.3788,49.2427) = -449.99$.

We also used the following Hawk–Dove game parameters: $l_0 = 5$ and neighbor radius $n_0(D) = n_0([-a, a]^2) = \frac{a^2}{100}$.

3.1. PSO & PSO–HD

The additional parameters used in this experiment for the basic PSO algorithm are: $c_1 = c_2 = 1.5$, $w = 1.0$, $v_{\max} = 2$. For the PSO–HD modification we used: $c_1(\mathcal{H}) = c_1(\mathcal{D}) = 0.5$, $c_2(\mathcal{H}) = 4.0$, $c_2(\mathcal{D}) = 5.0$, $v_{\max}(\mathcal{H}) = 50.0$, $v_{\max}(\mathcal{D}) = 2.0$.

The performance analysis of the basic and modified PSO-based algorithms is shown in Figure 1.

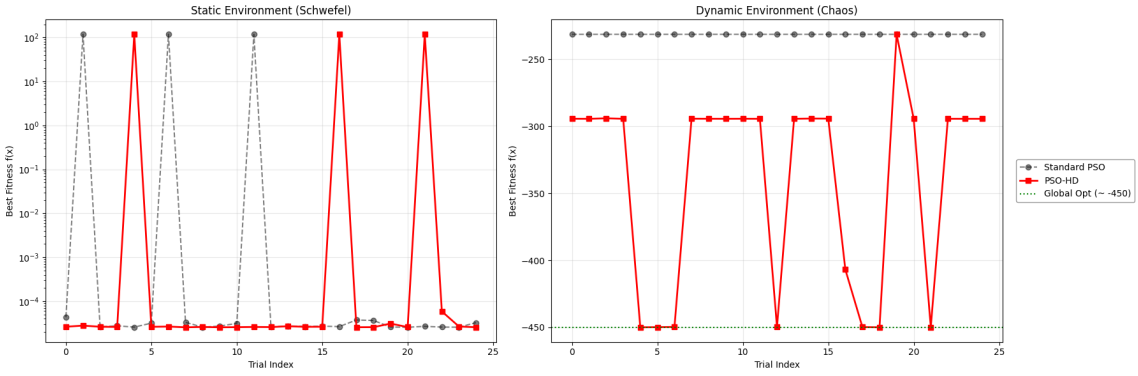


Figure 1: Performance analysis of the PSO algorithms

While the modification does not bring much improvement for the static Schwefel function (the mean values over all tries for both the modified and standard variants are the same – $\overline{y} \approx 14.2126$), it stabilizes the algorithm (PSO–HD found a better result 16 times out of $T = 25$).

However, when examining the results for the dynamic environment – the Chaos function – we can clearly see the improvements. The standard variant did not find the expected minimum even once (succeeding only 1 out of $T = 25$ tries) and obtained a mean value of $\overline{y} \approx -231.3151$, whereas the modified PSO–HD version found it 7 times (winning 24 out of $T = 25$ tries) and obtained a mean value of $\overline{y} \approx -339.8203$.

Another crucial part of the algorithms introduced in this paper is their ability to dynamically regulate populations based on the given environments. We used the following parameters in the analysis: $P = 180$, $C \in \{50,100,200,500\}$. Figure 2 shows the sensitivity analysis representing the hawk ratio within the algorithm's population. It clearly shows that while the algorithm does not need multiple scouting entities (such as hawks) in the Schwefel function ($r_{\mathcal{H},\text{Schwefel}}(200) \approx 30\%$), it drastically increases the ratio in a dynamic environment such as the Chaos function ($r_{\mathcal{H},\text{Chaos}}(200) \approx 95\%$) which enables it to search faster and more accurately.

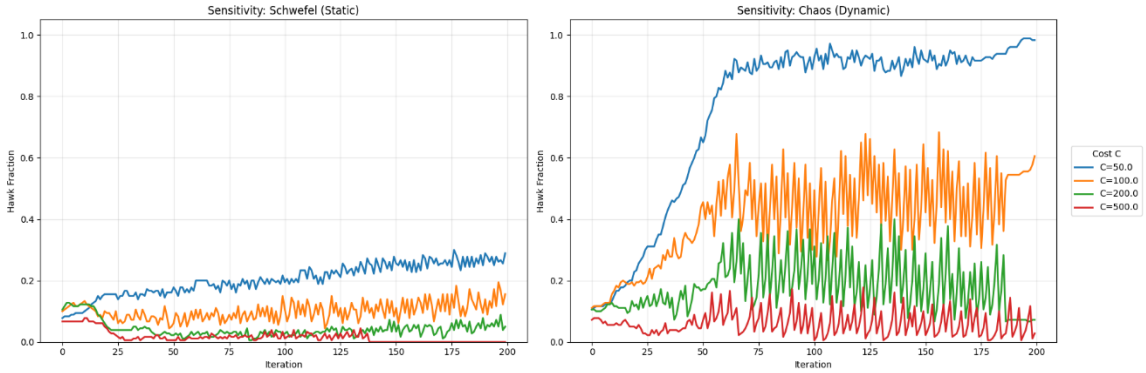


Figure 2: Sensitivity analysis of the PSO algorithms

3.2. GWO & GWO–HD

The performance analysis of the basic and modified GWO-based algorithms is shown in Figure 3.

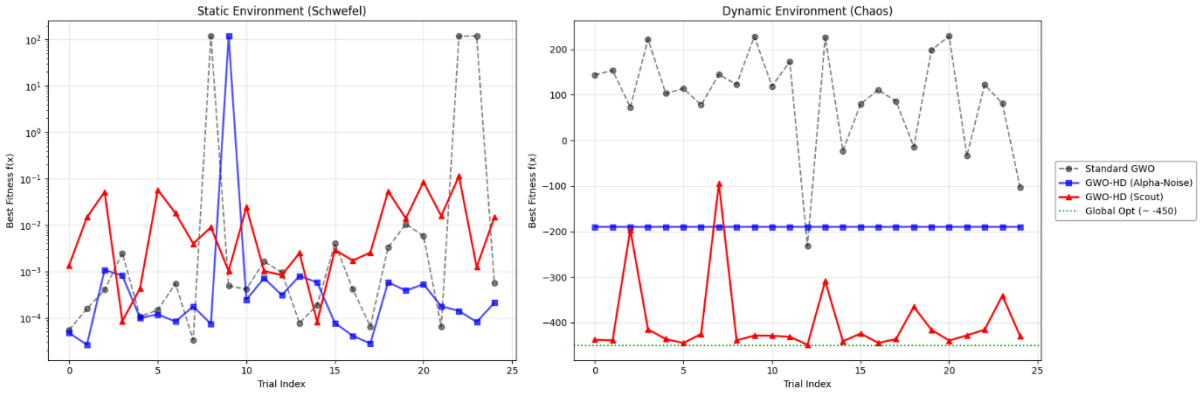


Figure 3: Performance analysis of the GWO algorithms

First, let us examine the Schwefel function. The standard variant obtains the worst mean of $\overline{y} \approx 14.2139$ and is the least stable algorithm. It finds the best value 5 times out of $T = 25$. The Scout variation has the best average value of $\overline{y} \approx 0.0196$ but it reached the best outcome only 2 out of $T = 25$ times. The best performing Alpha–Noise variation obtained the second best mean value of $\overline{y} \approx 4.7378$ and it is the most stable algorithm winning 18 out of $T = 25$ tries.

When it comes to the dynamic environment defined by the Chaos function, both basic and Alpha–Noise variations reached unsatisfactory scores not getting close to the expected local minimum even once (reaching mean values of $\overline{y} \approx 96.0565$ and $\overline{y} \approx -189.7816$). However, the Scout variation won 24 out of $T = 25$ tries, obtaining an average of $\overline{y} \approx -398.0749$ and solving the problem with a great stability.

3.3. WOA & WOA–HD

The additional parameter used for the WOA-based algorithms is $b = 1.0$.

The performance analysis of the basic and modified WOA-based algorithms is shown in Figure 4.

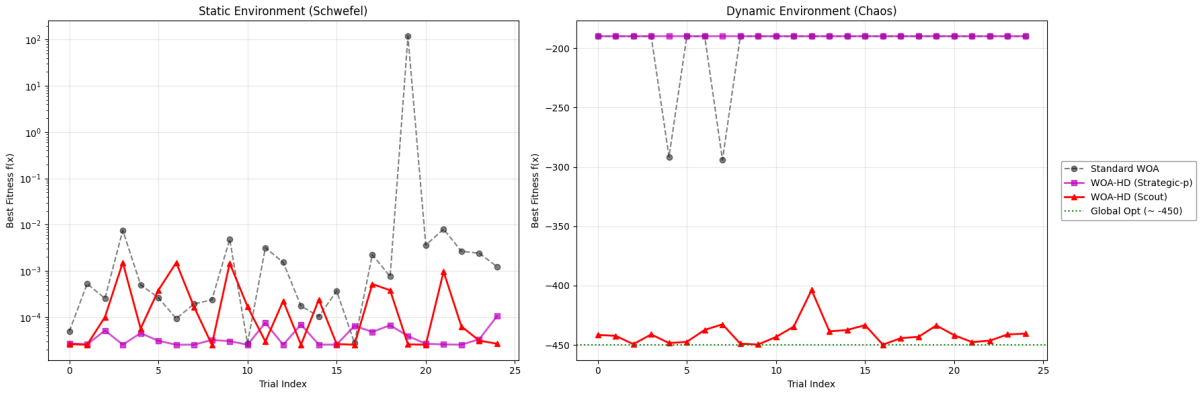


Figure 4: Performance analysis of the WOA algorithms

First, let us examine the Schwefel function. The standard variant obtains the worst mean of $\overline{y} \approx 4.7392$ and is the most unstable algorithm – it did not find the best value even once. The Scout variation has the second best average value of $\overline{y} \approx 0.0003$ and it reached the best outcome 10 times out of $T = 25$. The best performing p-Strategy variation obtained the best mean value of $\overline{y} \approx 0.0000$ and it is the most stable algorithm winning 15 times out of $T = 25$.

When it comes to the dynamic environment defined by the Chaos function, both basic and p-Strategy variations reached unsatisfactory scores not getting close to the expected local minimum even once (reaching mean values of $\overline{y} \approx -198.0408$ and $\overline{y} \approx -189.8006$). However, the Scout variation won 25 times out of $T = 25$, obtaining an average of $\overline{y} \approx -440.6830$ and solving the problem with great stability.

Overall, it is worth noting why the p-Strategy approach worked flawlessly in a static environment while completely failing in a dynamic one. The reason for such a result is that the p-Strategy mechanism is responsible for choosing an appropriate agent type between exploration and exploitation, and not for the behavior of the entity itself.

3.4. Summary

The summary of all the results described in our paper can be seen in Table 2 below.

Table 2. Summary of performance results.

Algorithm	Variant	Schwefel Mean	Wins	Chaos Mean	Wins
PSO	Basic	14.2126	9	-231.3151	1
PSO-HD	HD	**14.2126**	**16**	**-339.8203**	**24**
GWO	Basic	14.2139	5	96.0565	0
GWO-HD	Alpha-Noise	**4.7378**	**18**	-189.7816	1
GWO-HD	Scout	0.0196	2	**-398.0749**	**24**
WOA	Basic	4.7392	0	-198.0408	0
WOA-HD	p-Strategy	**0.0000**	**15**	-189.8006	0
WOA-HD	Scout	0.0003	10	**-440.6830**	**25**

4. Conclusion

The proposed modifications of swarm optimization algorithms (PSO, GWO, WOA) incorporating game-theoretic mechanics demonstrated that splitting entities into exploration-based (Hawks) and exploitation-based (Doves) agents results in an autonomous self-regulation system. This system efficiently replaces classical stochastic parameters with a deterministic response to population density in the local area, thus mitigating the problems associated with early convergence.

Our research revealed a correlation between depth of the modification and the type of optimization problem. In static environments (Schwefel), soft variants that preserve the swarm's social structure such as Alpha–Noise in GWO or p-Strategy in WOA provide the highest stability and best precision. On the other hand, in dynamic environments (Chaos), where information from the leader may become outdated during subsequent iterations, the Scout strategy proved essential. This enables the algorithm to understand that the social structure of the swarm is ineffective in the current environment and allows for aggressive "unfreezing" of the swarm, allowing escape from local optima, as evidenced by the dominance of this variant in dynamic tests.

In conclusion, implementation of Hawk–Dove dynamics in swarm based algorithms acts as a universal fuse against stagnation and premature convergence, significantly increasing the probability of finding the global optimum in changing conditions while maintaining computational complexity comparable to the original algorithms.

5. Declaration on Generative AI

The authors have not employed any Generative AI tools.

6. References

[1] SwarmReview - Swarm Intelligence Review
[2] J. Kennedy, R. Eberhart, Particle Swarm Optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, vol. 4, IEEE, 1995, pp. 1942-1948.
[3] M. Daviran, et al., Optimized Particle Swarm Optimization Algorithm, 2025.
[4] S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey Wolf Optimizer, Advances in Engineering Software 69 (2014) 46-61.
[5] M. Prokop, et al., Heuristic Methods in Engineering, 2024.
[6] S. Mirjalili, A. Lewis, The Whale Optimization Algorithm, Advances in Engineering Software 95 (2016) 51-67.
[7] R. Arjunan, et al., Optimizing with WOA, 2025.
[8] J. Maynard Smith, G. R. Price, The Logic of Animal Conflict, Nature 246 (1973) 15-18.
[9] H.-P. Schwefel, Evolution and Optimum Seeking, Wiley, New York, 1995.
[10] Dynamic Optimization Reference - Dynamic Optimization Benchmark.
[11] Eggholder Function Reference - Benchmark Functions for Optimization.
[12] Exploration vs Exploitation - Exploration-Exploitation Tradeoff in Swarm Optimization.
[13] Premature Convergence - Premature Convergence in Evolutionary Computation.