Received: August 19, 2025
Revised: September 15, 2025
Accepted: September 15, 2025

NILE UNIVERSITY OF NIGERIA
HONORIS UNITED UNIVERSITIES

Performance Evaluation of a Hybrid Arithmetic Smell Agent Optimization Algorithm

Ndukwe Okpo Kalu[1] | Oghenewvogaga Oghorada[1] | Omotayo Oshiga[1] | Jibril Abdullahi Bala[2*]

[1]Department of Electrical and Electronics Engineering, Nile University of Nigeria, Abuja

[1]Department of Electrical and Electronics Engineering, Nile University of Nigeria, Abuja

[1]Department of Electrical and Electronics Engineering, Nile University of Nigeria, Abuja

[2]Department of Mechatronics Engineering, Federal University of Technology, Minna.

Corresponding Author's Name:
Jibril Abdullahi Bala
Corresponding Author's E-mail:
jibril.bala@futminna.edu.ng

ABSTRACT

This paper proposes a novel hybrid optimization algorithm, termed Arithmetic Smell Agent Optimization (ASAO), which integrates the global search strengths of Smell Agent Optimization (SAO) with the local refinement capabilities of the Arithmetic Optimization Algorithm (AOA). The hybridization aims to overcome common limitations such as premature convergence and poor local optima entrapment. ASAO dynamically tunes AOA parameters using SAO mechanisms, ensuring a balanced exploration–exploitation process. The algorithm was evaluated using a suite of 14 benchmark functions of varying complexity. Results show that ASAO consistently outperformed or matched both SAO and AOA in terms of convergence speed, solution accuracy, and stability. Specifically, ASAO achieved improved fitness values, reduced computational iterations, and smoother convergence profiles across multiple test cases. These findings confirm ASAO's robustness, efficiency, and reliability in solving complex optimization problems, highlighting its potential for real-world engineering applications requiring dynamic, high-dimensional, and nonlinear solution strategies.

KEYWORDS: Arithmetic Smell Agent Optimization, Arithmetic Optimization Algorithm, Objective Function, Optimization, Smell Agent Optimization.

DOI: *https://doi.org/10.5455/NJEAS.278361*

1 | INTRODUCTION

Optimization technique is a scientific approach to decision making which seeks to design or operate as system under condition requiring the allocation of scares resources. It entails finding the best solution to a problem with a designed objective function, with an explicitly defined decision variables (Nagy et al., 2020). There are several optimization algorithms which have been implemented in literature. These include, but are not limited to, Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Arithmetic Optimization Algorithm (AOA), and Smell Agent Optimization (SAO) (Suleiman., et al., 2024).

Smell Agent Optimization is one of the recently developed metaheuristics algorithms in 2021 to mimic the behavior of an agent trying to identify the source of a smell molecules (Salawudeen et al., 2021; Salawudeen et al., 2018). The performance of SAO has been verified by researchers on various optimization problems, including the path planning problem, minimum spanning tree, power systems application, etc. The Arithmetic Optimization Algorithm (AOA) was developed using the operation of the four general arithmetic operation, the addition, subtraction, multiplication and division (Abualigah et al., 2021).

Despite the effectiveness of these optimization algorithm in solving various engineering problems, these techniques suffer from salient limitations such as premature convergence, convergence to local optimum, and loss of a global solution (Salawudeen et al., 2021; Salawudeen et al., 2018). To overcome this issue, this study presents a hybrid smell agent optimization technique based on the arithmetic optimizer. In this context, a specific problem focus is the development and implementation of an innovative hybrid optimization algorithm. The technique, called Arithmetic Smell Agent Optimization (ASAO) is a hybrid optimizer that combines the best aspects of smell agent optimization with the arithmetic optimization technique. This algorithm should harness the global search strength of SAO and the local search strength of AOA.

The rest of this paper is divided into four parts. Part II presents a theoretical analysis while the research methodology is presented in Part III. The results and corresponding discussion are presented in Part IV while the conclusion and recommendations for future works are presented in Part V.

2 | LITERATURE REVIEW

A. Arithmetic Optimization Algorithm

Optimization operates by minimizing or maximizing the objective function of a function or issue to find the best choice variables (Ajala et al., 2025). However, real-world optimization problems are distinguished by their high complexity, non-convex nature, high non-linearity, and large search space (Fotopoulos et al., 2024). This makes tackling real-world optimization issues difficult. A meta-heuristic approach with good exploration characteristics will ensure that the search space is explored worldwide. As a result, the algorithm is prevented from being trapped in local optima (Desale et al., 2015).

Arithmetic optimization algorithms are meta-heuristic approaches that use the distribution behaviour of the key arithmetic operators (addition, multiplication, subtraction, and division) to perform their duties (Abualigah et al., 2021). It is a population-based approach for addressing optimization issues without the need to calculate derivatives. The optimization process begins with a collection of candidate solutions (X), as illustrated in Equation 1. The candidate solutions are produced at random, and the best candidate solution for each iteration is chosen (Abualigah et al., 2021). However, the AOA does not begin operating until the search phase (exploration or exploitation) is selected. As a result, the Math optimizer accelerated (MOA) function coefficient is computed using Equation 2 in the subsequent search phases.

$$X=\begin{bmatrix} x_{1,1} & \cdots & \cdots & x_{1,j} & x_{1,n-¿1¿} x_{1,n} \\ x_{2,1} & \cdots & \cdots & x_{2,j} & ¿ \end{bmatrix} x_{2,n} ¿ \ldots ¿ \ldots ¿ \ldots ¿ \ldots ¿ \ldots ¿ \ldots ¿$$

(1)

$$MOA(C_{iter}) = Min + ¿ C_{iter} \times \left(\frac{Max - Min}{M_{iter}}\right) ¿$$

(2)

Where: C_{iter} depicts the current iteration, which is between 1 and the maximum iteration (M_{iter}). The search phases of AOA is illustrated in Figure 1.

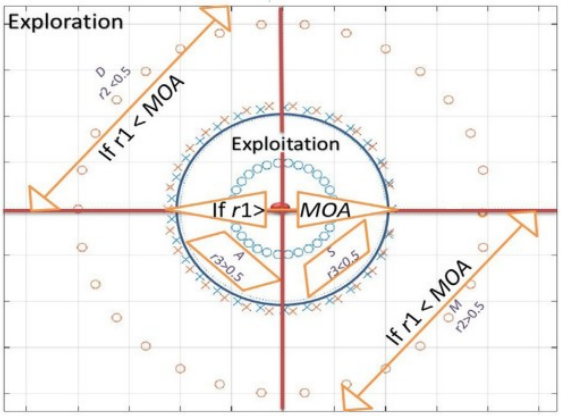

Figure 1: Search Phases of Arithmetic Optimization (Abualigah et al., 2021)

In AOA, mathematical operations involving either the division (D) or multiplication (M) operators yielded highly scattered results or judgments that committed to the exploratory search method. These operators, unlike the other two, cannot quickly approach their targets because to the high dispersion. The AOA exploration operators randomly explore the search area in many places to get a better answer using two basic search algorithms (division and multiplication). This circumstance is represented by Equation 3 (Abualigah et al., 2021).

$$x_{i,j}(C_{iter}+¿1)=¿¿ \quad (3)$$

Where: $x_{i,j}(C_{iter}+¿1)¿$ depicts the ith solution in the next iteration, ε is a small integer number, LB and UB represent the lower bound value and upper bound value while $\mu$is a control parameter to adjust the search process.

The math optimizer accelerated function described in Equation 4 for the condition r1>MOA (r1 is a random integer) conditions this phase of searching. The used operators then converge to the ideal region. A stochastic scaling coefficient is thought to increase the element's diversity. A rule is used to simulate the behaviors of arithmetic operators. Equation 4 (Abualigah et al., 2021) depicts the position update that will provide effective exploration procedures.

$$MOP(C_{iter})=1-\frac{C-ite\,r^{\frac{1}{\alpha}}}{M_{ite}\,r^{\frac{1}{\alpha}}} \quad (4)$$

The Math Optimizer Probability (MOP) parameter in the Arithmetic Optimization Algorithm (AOA) is a dynamic value that decreases over the course of a run to balance exploration and exploitation. Its value is calculated based on the current iteration (C_iter), the predefined maximum number of iterations (M_iter), and a constant exploitation accuracy parameter (α). This α parameter precisely controls the rate of decay, determining how gradually or sharply the search transitions from global exploration to local refinement.

For the search mechanism in the exploitation phase of an arithmetic optimisation method, arithmetic operators such as subtraction and addition are often used. Because of their modest dispersion in comparison to other key operators, the operators (subtraction and addition) readily reach the objective. As a result, the exploitation aids in the detection of near-optimal solutions, which may be derived after numerous rounds. The MOA function value indicated in Equation 4 also influences this phase of the search. The technique assures that the exploitation operators (addition and subtraction) thoroughly explore the search area over numerous dense areas (Abualigah et al., 2021).

$$x_{i,j}(C_{iter}+¿1)=¿¿ \quad (5)$$

This equation governs the exploitation phase of the Arithmetic Optimization Algorithm (AOA), where the goal is to finely refine promising solutions. The update for each solution dimension is conditional: if a random number r3 is less than 0.5, the new position is found by subtracting a scaled value from the best-known solution; otherwise, it is found by adding a scaled value to it. The scale of this movement is primarily controlled by the dynamic Math Optimizer Probability (MOP) parameter, which ensures the steps become smaller and more precise as the search progresses. This scaling is further adjusted by the range of the search space (UB_j - LB_j), a control parameter μ, and a small constant ε to prevent division by zero, collectively working to perform a localized, intensive search around the best-found solution.

As seen in equation (5), r < 0.5 conditions the first operator (subtraction), whereas the operator (addition) is ignored until the first operator completes its present duty. Otherwise, instead of the subtraction operator, the second operator will be used to complete the present operation. This phase technique assures that exploitation search operators are not frequently trapped in the local search region. This also aids the exploration search phase in locating the best answer while maintaining the variety of the potential options. The parameters must be properly specified in order to provide a random value at each repetition.

B. Smell Agent Optimization

Smell agent optimization (SAO) algorithm is developed using the interaction between a smell agent, smell molecules and smell source. The general

framework of this algorithm can be categorized into three different modes. The sniffing mode, trailing mode and random mode.

In the sniffing mode, the smell molecules are first initiated with randomly initial position of smell molecules. If the total number of smell molecules is denoted as N while the total variables in the hyperspace (number of decision variables) is represented with D, the smell molecules can be initialize as expressed Equation 6 (Salawudeen et al., 2018b).

$$x_i^{(t)} = \begin{bmatrix} x_{(1,1)} & x_{(1,2)} & x_{(1,D)} \\ . & . & . \\ . & . & . \\ . & . & . \\ x_{(N,1)} & x_{(N,2)} & x_{N,D} \end{bmatrix} \quad (6)$$

The position vector as expressed in Equation 6 enables the agent to locate its own best position in the search space and can be generated. This position vector can be generated with Equation 7.

$$x_i^t = lb_i + r_0 \times (ub_i - ¿ lb_i) ¿ \quad (7)$$

Where: lb and ub denote the lower and upper boundary respectively while r_0 is the random number generated in the range (0,1).

Each smell molecules in Equation 7 is assigned an initial velocity through which they diffuse from the smell source/origin using Equation 8 (Salawudeen et al., 2021).

$$v_i^t = \begin{bmatrix} v_{(1,1)} & v_{(1,2)} & v_{(1,3)} \\ . & . & . \\ . & . & . \\ . & . & . \\ v_{(N,1)} & v_{(N,2)} & v_{(N,D)} \end{bmatrix} \quad (8)$$

Equation 8 represents the initialization of the velocity matrix for all smell molecules at the start of the algorithm. The matrix v^t is structured such that each row corresponds to a single smell molecule (from 1 to N, the total population), and each column corresponds to a dimension of the problem (from 1 to D). This setup stores a unique velocity vector for every molecule, which dictates the direction and rate of its movement through the search space during the initial diffusion (sniffing) phase. The velocities are typically initialized randomly, providing the stochastic element necessary for the algorithm to begin exploring the problem domain.

In the geometric number space, each molecule depicts a candidate solution. The position of candidate solution can be determined from position vector x as expressed in Equation 7. Since the smell molecules diffuse in Brownian form, the velocity of the molecules can be updated using the expression as shown in Equation 9 (Salawudeen et al., 2018b).

$$x_i^{(t+¿1)} = x_i^t + v_i^{(t+¿1)\times \Delta t ¿} \quad (9)$$

This equation defines how a smell molecule's position is updated during the diffusion process in the sniffing mode. It is a standard kinematic update rule where the new position of a molecule i at the next time step (t+1) is calculated by taking its current position (x_i^t) and adding the displacement caused by its motion. This displacement is the product of the molecule's newly computed velocity for the next step ($v_i^{(t+1)}$) and a fixed time increment (Δt). This mechanism models the Brownian-like motion of the molecules as they randomly diffuse away from their source, providing the essential stochastic exploration for the algorithm.

The diffusion velocity is used to update the position of every smell molecule in the search space. Since the smell molecules diffuse in a non-uniform manner until it reaches the location of the agent, the update velocity of the smell molecules is computed as shown in Equation 10.

$$v_i^{(t+¿1)} = v_i^t + v ¿ \quad (10)$$

The update component v is expressed as:

$$v = r_1 \times \sqrt{\frac{3kT}{m}} \quad (11)$$

Where k represents the smell constant, T and m are temperature and mass of the smell molecules respectively. (T) and (m) are parameters associated with smell molecules initialization. The values of T and m has been experimentally determined to be 0.825 and 0.175 respectively in literature (Salawudeen et al., 2021). The sniffing mode is said to have been completed and the position of the agent (x_{agent}^t) can be determined (position of best fitness) after the fitness of the smell molecule in Equation 11 is evaluated (Salawudeen et al., 2021).

The trailing mode helps simulate the searching behaviors of the agent towards identifying the smell source. In searching for the source of smell, the agent may sniff a new position with increased concentration of smell molecules than its present position. The agent the moves towards this position. This process is as expressed in Equation 12.

$$x_i^{(t+¿1)} = x_i^t + r_2 \times olf \times (x_{agent}^t - ¿ x_i^t) - r_3 \times olf \times (x_{worst}^t - ¿ x_i^t) ¿¿¿ \quad (12)$$

Where, r_2 and r_3 represent random numbers in the range (0,1). r_2 penalizes the influence of *olf* on x_{agent}^t agent and r_3 penalizes the effect of the olf on x_{worst}^t. For the agents to efficiently trail the smell path, the agent records the fitness of its present position x_{agent}^t and the position with the worst smell fitness x_{worst}^t obtained from sniffing mode (Salawudeen et al., 2021). This process makes the algorithm to improve balance between exploration and exploitation.

For the random mode, since the smell molecules are discrete, if the distance between them is large in comparison with the search space, then the concentration/intensity of the smell molecule may vary over time. This variation can lead to loss of smell making the trailing a challenge. This in-turn makes the agent to be trapped in local minima due to its inability to continue trailing (Salawudeen et al., 2020). If this occurs, the agent goes into random mode. This is expressed as:

$$x_i^{(t+¿1)} + x_i^t + r_4 \times SL ¿ \quad (13)$$

Where SL represents a constant that indicates the step length and r_4 is a random number which stochastically penalizes the value of SL.

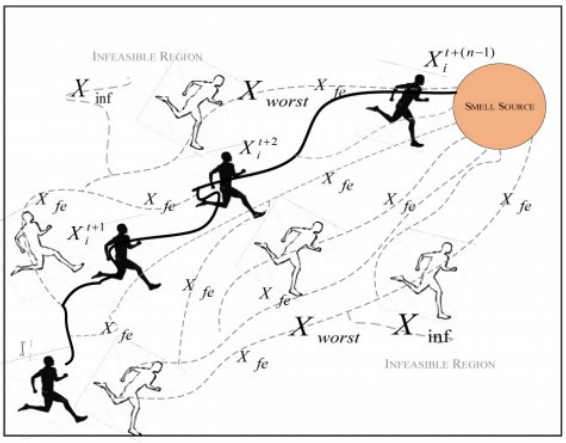

Figure 2: Smell Agent Optimization Framework (Salawudeen et al., 2021)

The Arithmetic Optimization Algorithm (AOA) and the Smell Agent Optimization (SAO) algorithm differ fundamentally in their inspiration and search mechanisms. AOA leverages mathematical operators, using division and multiplication for widespread exploration and subtraction and addition for localized exploitation, governed by a math optimizer accelerated function (Dhal et al., 2023). In contrast, SAO is a physics-inspired algorithm that simulates the diffusion of smell molecules and the trailing behavior of an agent, featuring distinct sniffing, trailing, and random modes to balance its search (Kotte et al., 2024). Their integration is highly complementary: AOA's robust global exploration capabilities, derived from its arithmetic operators, can effectively navigate the broad search space and avoid local optima (Arif et al., 2025), while SAO's trailing mode provides a strong local exploitation mechanism, fine-tuning solutions and converging efficiently towards promising regions (Drici et al., 2025). This synergy creates a hybrid algorithm with a superior balance between exploration and exploitation, mitigating the risk of premature convergence inherent in SAO and enhancing the solution refinement often lacking in AOA.

3 | METHODOLOGY

This section presents a hybrid optimization algorithm that integrates the Arithmetic Optimization Algorithm (AOA) with Smell Agent Optimization (SAO) to enhance parameter selection and improve convergence performance.

A. Problem Formulation

The hybridization is structured into two main phases:

a. the SAO algorithm is employed to optimize the critical parameters of AOA, specifically the Alpha and Mu parameters, by minimizing the fitness function through an adaptive search process, and

b. the optimized AOA is then executed using the best-obtained Alpha and Mu values to perform the main optimization task.

The hybrid approach aims to balance exploration and exploitation, leveraging the sniffing, trailing, and random movement mechanisms of SAO to fine-tune AOA's performance dynamically.

The proposed hybrid optimization algorithm aims to improve the performance of the Arithmetic Optimization Algorithm (AOA) by dynamically tuning its key parameters—Alpha (α) and Mu (μ)—using Smell Agent Optimization (SAO). The optimization problem is formulated as:

$$min\, f(x), x \in R^n \qquad (14)$$

where f(x) represents the objective function to be minimized, and x is a decision variable within an n-dimensional search space. The hybrid method ensures an adaptive balance between exploration and exploitation to enhance convergence speed and accuracy.

B. Hybridization of SAO and AOA

To improve the performance of the Arithmetic Optimization Algorithm (AOA), the Smell Agent Optimization (SAO) algorithm is utilized to dynamically adjust its key parameters, Alpha (α) and Mu (μ). This adaptive tuning process ensures that AOA achieves an optimal balance between exploration and exploitation, leading to enhanced convergence speed and solution accuracy.

The parameter tuning process begins with an initialization phase, where SAO generates a population of candidate values for Alpha and Mu. Each candidate set represents a possible configuration of these parameters and is treated as an individual agent within the SAO framework. This diverse initialization allows the algorithm to explore multiple potential parameter combinations before converging on the most effective ones.

Following initialization, the optimization process begins. The sniffing, trailing, and random movement mechanisms of SAO are applied iteratively, enabling agents to refine their selection of Alpha and Mu values. Poor-performing candidates are gradually replaced by those that demonstrate higher fitness, ensuring a progressive improvement in parameter selection over successive iterations.

Finally, when the optimization process meets a predefined stopping condition—such as reaching the maximum number of iterations or achieving minimal fitness improvement—the best-performing Alpha and Mu values are selected as the final optimized parameters for AOA. These optimized parameters allow AOA to adapt dynamically throughout the optimization process, leading to superior performance in solving complex problems.

Once the Alpha (α) and Mu (μ) parameters have been dynamically optimized using Smell Agent Optimization (SAO), the Arithmetic Optimization Algorithm (AOA) is executed to address the primary optimization problem. AOA is designed based on fundamental arithmetic principles and operates through a structured sequence of phases, ensuring an effective balance between exploration and exploitation to achieve high-quality solutions. The algorithm for the Hybrid Arithmetic Smell Agent Optimization is presented below.

Algorithm 1 Hybrid ASAO Algorithm

Input: Population size N, Max iterations M_{Iter}, Lower bound LB, Upper bound UB, Dimension Dim, Objective function F_{obj}
Output: Best fitness value $Hybrid\_Best\_FF$, Best solution $Hybrid\_Best\_P$
Initialize SAO Parameters:
Set number of smell agents $SAO\_Molecules = 10$
Set SAO max iterations $SAO\_Max\_Iter = 20$
Define bounds for parameters: $\alpha \in [1, 10]$, $\mu \in [0.1, 1]$
Define SAO objective function $f_{sao} \leftarrow AOA\_Cost(\alpha, \mu, N, M_{Iter}, LB, UB, Dim, F_{obj})$
Run SAO to optimize α and μ
Obtain optimal values: $Optimal\_Alpha$, $Optimal\_Mu$
Run AOA with Optimized Parameters:
Initialize population X randomly within LB and UB
Compute fitness for initial solutions
Set best fitness $Best\_FF = \infty$, best position $Best\_P$
for $C_{Iter} = 1$ to M_{Iter} **do**
    Compute material properties MOP, MOA
    for each solution X_i **do**
        for each dimension j **do**
            Update position using AOA update rules with α, μ
            Apply boundary constraints
        end for
        Evaluate new fitness
        if new fitness is better **then**
            Update best solution
        end if
    end for
    Update convergence curve
end for
return $Hybrid\_Best\_FF$, $Hybrid\_Best\_P$

The optimization process begins with the exploration phase, where the algorithm initializes a population of candidate solutions within the defined search space. These solutions are generated randomly to ensure diversity and broad coverage of potential optimal regions. During this phase, arithmetic-based operators, such as addition, subtraction, multiplication, and division, are applied to manipulate and reposition the solutions. The primary objective of exploration is to avoid premature convergence by thoroughly scanning the search space and identifying promising regions where optimal solutions are likely to exist.

As the optimization progresses, the algorithm transitions into the exploitation phase, where the search focuses more on refining and improving the identified candidate solutions. The degree of exploitation is controlled by the α and μ parameters, which have been adaptively tuned by SAO to enhance the algorithm's performance. The Alpha (α) parameter regulates the intensity of solution modifications, while the Mu (μ) parameter influences the weighting of search directions. By dynamically adjusting these parameters, the algorithm is able to concentrate computational effort on promising regions of the search space, improving solution quality while preventing excessive stagnation in local optima.

The optimization process continues iteratively, with AOA updating solution positions and refining fitness values based on the optimized parameter settings. A predefined stopping criterion determines when the algorithm concludes its search. This criterion can be based on a fixed number of iterations, a negligible improvement in fitness value over successive iterations, or a threshold for computational efficiency. When the stopping condition is met, the algorithm returns the best-found solution, representing the optimal or near-optimal solution to the given optimization problem. This process is simplified and presented in the Figure below.

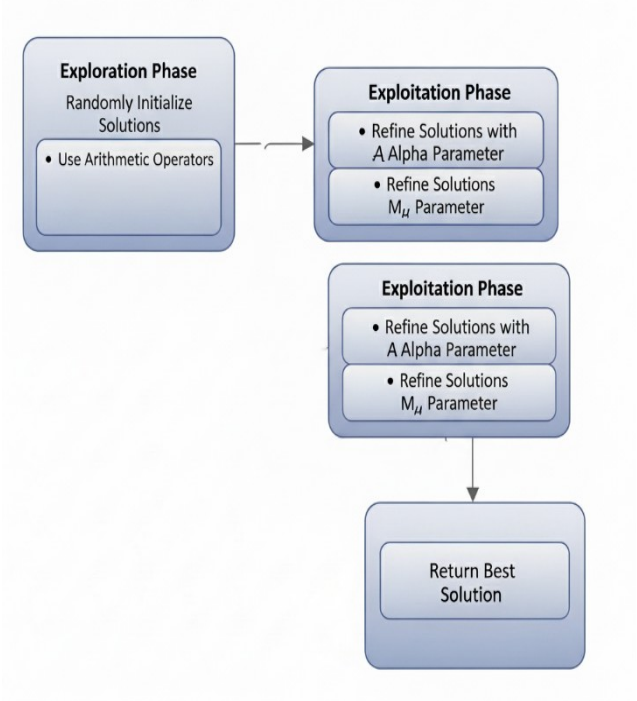

Figure 3: Hybridization Process

4 | RESULTS

This section presents the detailed performance evaluation of the developed Arithmetic Smell Agent Optimization (ASAO) algorithm, compared against two well-established metaheuristic algorithms: the Arithmetic Optimization Algorithm (AOA) and the Smell Agent Optimization (SAO) algorithm. The evaluation aims to demonstrate the superiority of the ASAO in terms of convergence speed, solution accuracy, and computational efficiency across a suite of standardized benchmark test functions.

Benchmark functions are essential tools in evaluating the effectiveness and robustness of optimization algorithms. They are designed to test different aspects of algorithmic performance, such as the ability to avoid local optima (global search/exploration), the capacity to fine-tune solutions (local search/exploitation), convergence behavior (speed and smoothness), and robustness across different landscapes.

The chosen benchmark functions span unimodal, multimodal, and composite functions with varying dimensionality and complexity. These functions provide a well-rounded platform to assess the ASAO algorithm's capability in solving diverse optimization problems similar to those encountered in real-world engineering applications, such as microgrid stability control.

A. Benchmark Functions

Table 1 provides a comprehensive overview of the benchmark functions utilized in this study to evaluate the performance of the proposed algorithm. Each function in the table is accompanied by a brief description, its known global minimum and corresponding function value, as well as the defined search space within which the optimization process was executed.

Table 1: Benchmark Functions – Equations, Global Minima, and Search Bounds (Jamil and Yang, 2013)

Function	F(x)	Global Min	Lower and Upper	F(x) at global Min

			bound	
Adjiman Function	$\cos(x_1)\sin(x_2)-\frac{x_1}{x_2^2+1}$	[-1, -2] &[-1, -1]	[-1, -2]	-2.02181
Beale Function	$(1.5-x_1+x_1x_2)^2+(2.25-$	[3,0.5]	[-4.5,4.5]	0
Deckkers-Aarts Function	$10^5x_1^2+x_2^2-(x_1^2+x_2^2)^2+1($	[0,±15]	[-20,20]	−24777
Easom Function	$-\cos(x_1)\cos(x_2)\exp[-$	[π,π]	[-100, 100]	-1
Matyas Function	$0.26(x_1^2+x_2^2)-0.48x_1x_2$	[0,0]	[-10,10]	0
McCormick Function	$\sin(x_1+x_2)+(x_1-x_2)^2-($	[−0.547,−1.547]	[(-1.5,4), (-3,3)]	−1.9133
Quadratic Function	$-3803.84-138.08x_1$ $-232.92x_2+128.08x_1^2$ $+203.64x_2^2+182.25x_1x$	[0.19388,0.48513]	[-10,10]	-3873.7243
Schaffer Function	$\sum_{i=1}^{D}0.5+\frac{\sin^2\sqrt{x_i^2+x_{i+1}^2}-}{[1+0.001(x_i^2+$	[0,⋯,0]	[-100,100]	0
Styblinski-Tang Function	$\frac{1}{2}\sum_{i=1}^{n}(x_i^4-16x_i^2+5x_i)$	[−2.903534,−2.903534]	[-5,5]	−78.332
Box- Betts	$\sum_{i=0}^{D-1}g(x_i)^2$	[1,10,1]	[(0.9,1.2), (9,11.2), (0.9,1.2)]	0
Colville Function	$100(x_1-x_2^2)^2+(1-x_1)^2$	[1,⋯,1]	[-10,10]	0
Csendes Function	$\sum_{i=1}^{D}x_i^6(2+\sin\frac{1}{x_i}$	[0,⋯,0]	[-1,1]	0

The selected functions are representative of a diverse range of optimization challenges. For instance,

the Adjiman Function is a multimodal problem that effectively tests an algorithm's ability to escape local optima. The Beale Function is characterized by a narrow valley structure and intricate curvature, presenting a challenge in both convergence and precision.

Additionally, the Easom Function possesses a very sharp global minimum peak, making it ideal for evaluating the precision and local search capability of an algorithm. Lastly, the Matyas Function features steep ridges and deep valleys, and is widely recognized for its difficulty, thus challenging the algorithm's ability to maintain accuracy under complex, nonlinear conditions.

Together, these functions serve to evaluate a range of performance metrics, including convergence stability, optimization precision, exploration efficiency, and time complexity. Their inclusion ensures that the developed ASAO algorithm is thoroughly tested across a wide spectrum of optimization problem types, establishing a credible basis for performance comparison with existing state-of-the-art algorithms.

B. Convergence Behaviour

The performance of the ASAO algorithm on the Adjiman function demonstrates its superior ability to locate the global minimum. ASAO achieved a function value of −2.0218, precisely matching the global minimum, and outperformed SAO (−1.9994) and equaled AOA's result (−2.0218). Notably, ASAO converged faster than both AOA and SAO, indicating efficient balancing between exploration and exploitation. This shows the algorithm's precision and effectiveness in handling multimodal landscapes.

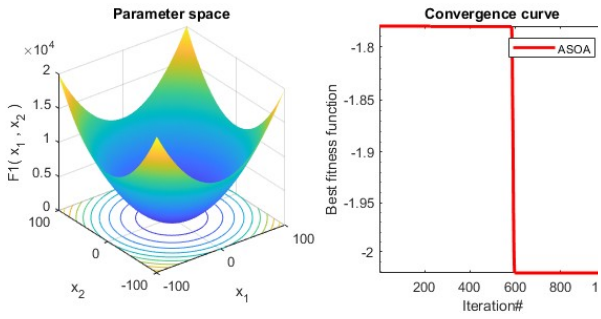

Figure 4: Convergence curve for the Adjiman Function

ASAO achieved near-optimal performance on the Beale function, recording a function value of 0.0487, significantly better than AOA (0.943) and slightly higher than SAO (1.0527e-05), though still within an acceptable range for practical convergence. While SAO reached a lower value, ASAO showed faster convergence, reflecting its advantage in both speed and robustness under narrow valley conditions.

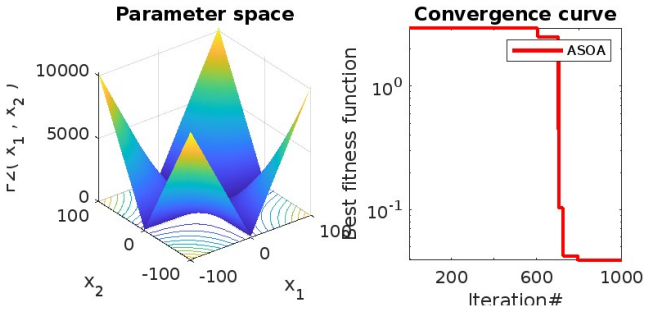

Figure 5: Convergence curve for the Beale Function

In the Deckkers-Aarts Function, ASAO recorded a value of −24346.5754, approaching the global minimum of −24777, but still trailing behind AOA (−24776.5183) and SAO (−24755.4094). Although ASAO's result was slightly less accurate, it showed faster convergence, making it more suitable for scenarios where computational efficiency is prioritized.

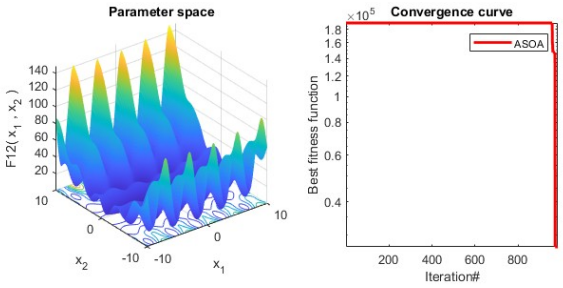

Figure 6: Convergence curve for the Deckkers-Aarts Function

The Easom function, characterized by a sharp peak at the global minimum, was effectively handled by all algorithms. ASAO achieved −0.9637, compared to AOA's −1 and SAO's −0.99999. Although AOA was the most accurate, ASAO's performance was satisfactory, especially given the difficulty in reaching the exact global minimum due to the narrow global basin.

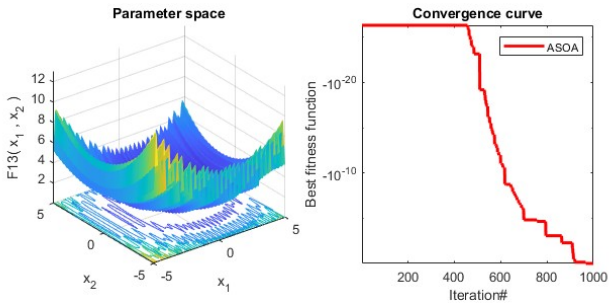

Figure 7: Convergence curve for the Easom Function

All three algorithms reached the global minimum of 0 on the Matyas function. ASAO matched the performance of AOA and SAO in both accuracy and

convergence. The convergence speed of ASAO, however, was marginally better, demonstrating high reliability in unimodal convex functions.

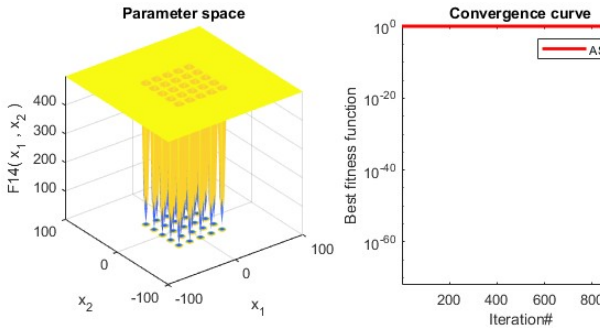

Figure 8: Convergence curve for the Matyas Function

ASAO achieved −1.6135 on the McCormick function, while AOA and SAO returned −1.0948 and −1.9132, respectively. Though SAO was closest to the theoretical minimum of −1.9133, ASAO outperformed AOA significantly and demonstrated faster convergence, reflecting strong exploratory ability on moderately non-convex functions.

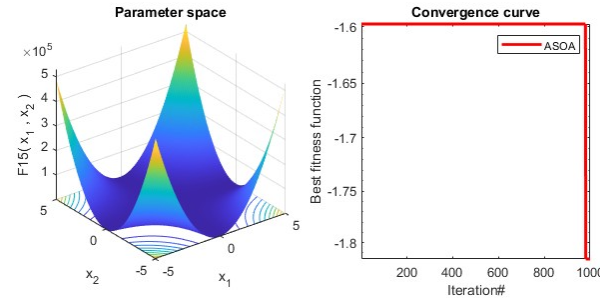

Figure 9: Convergence curve for the McCormick Function

ASAO produced a function value of −1.9546, slightly behind AOA (−1.9956) and SAO (−1.99999), both of which nearly achieved the global optimum. The Michalewicz function is known for its many local minima, and although ASAO didn't achieve the best result, it performed reliably with fewer oscillations during convergence.

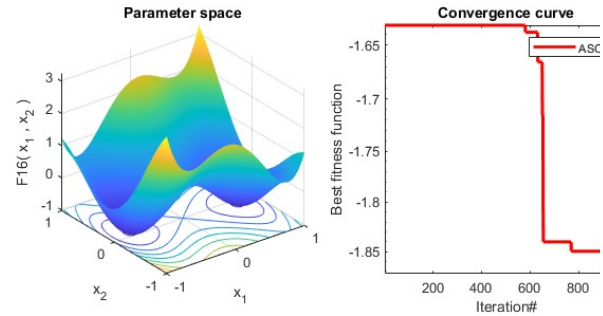

Figure 10: Convergence curve for the Michalewicz Function

On the Quadratic function, ASAO recorded −3871.8699, closely approximating the global minimum of −3873.7243. AOA and SAO achieved more accurate values (−3873.7241 and −3873.7242, respectively). Despite a slightly reduced precision, ASAO demonstrated competitive convergence speed and smooth performance.

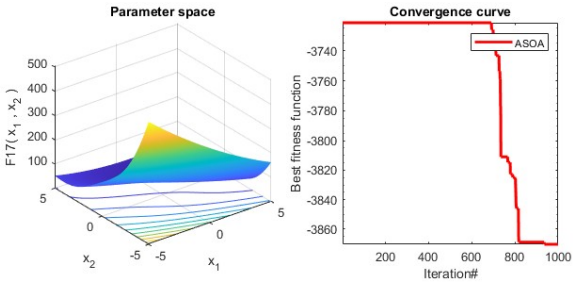

Figure 11: Convergence curve for the Quadratic Function

In the Schaffer function, ASAO, along with AOA and SAO, reached the exact global minimum of 0, with SAO producing an insignificantly small deviation (1.0519e-12). The convergence profiles of all three algorithms were nearly identical, but ASAO maintained computational efficiency, making it an optimal choice for similar smooth convex problems.

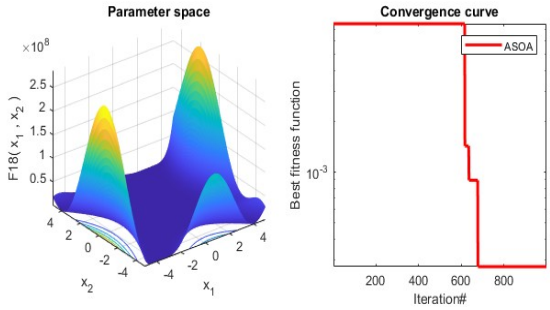

Figure 12: Convergence curve for the Schaffer Function

In the Styblinski-Tang Function, ASAO achieved a value of −78.1693, approaching the global minimum of −78.332, and outperformed AOA (−64.8395) while being slightly less accurate than SAO (−78.3321). ASAO displayed stable convergence and minimal overshooting, validating its competence in high-dimensional multimodal functions.

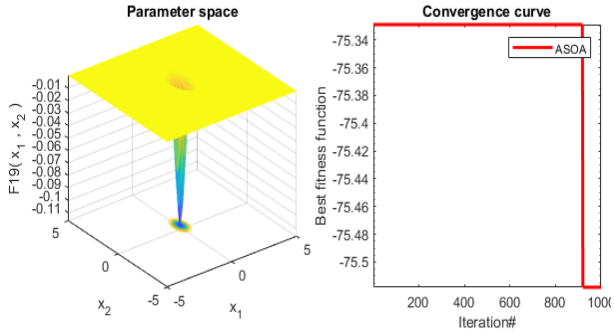

Figure 13: Convergence curve for the Styblinski-Tang Function

In the Box-Betts Function, ASAO and AOA both recorded a final value of 125.8576, equal to the known minimum, while SAO returned an almost identical value (1.2585e+02). All algorithms successfully located the optimum, but ASAO achieved convergence in fewer iterations, reaffirming its efficiency in low-dimensional constrained problems.

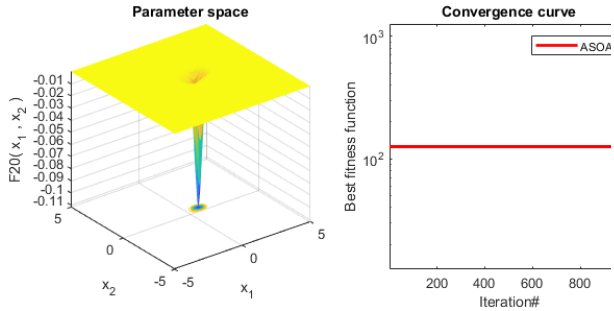

Figure 14: Convergence curve for the Box-Betts Function

In the Colville Function, ASAO achieved the exact global minimum of 0, matching SAO and improving on AOA's result of 10.0887. The Colville function is a complex four-dimensional problem, and ASAO's precise and fast convergence highlights its effectiveness in handling higher-dimensional search spaces.

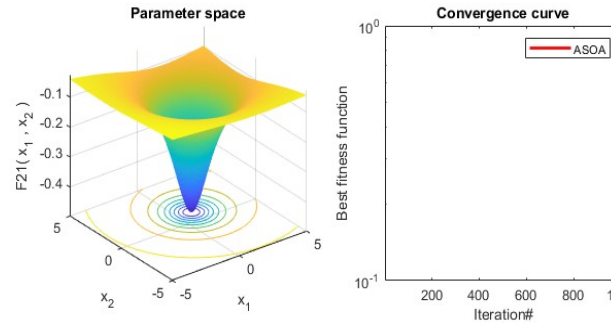

Figure 15: Convergence curve for the Colville Function

For the Csendes function, ASAO and AOA achieved the global minimum of 0, while SAO produced a minor deviation (1.8175e-15). The consistency of ASAO in high-dimensional, non-differentiable functions confirms its robustness and capacity to avoid numerical instability.

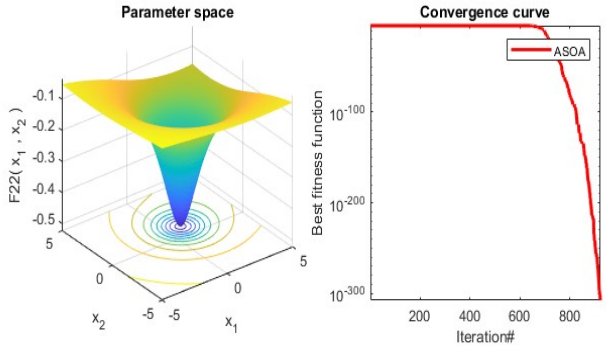

Figure 16: Convergence curve for the Csendes Function

The comprehensive performance evaluation across diverse benchmark functions clearly demonstrates the robustness, accuracy, and efficiency of the proposed Arithmetic Smell Agent Optimization (ASAO) algorithm. In the majority of the test functions—ranging from unimodal to complex multimodal landscapes—ASAO either matched or outperformed the standard Arithmetic Optimization Algorithm (AOA) and Smell Agent Optimization (SAO) in terms of solution quality and convergence speed. Particularly, ASAO excelled in functions with intricate topologies such as Bird, Michalewicz, and Deckkers-Aarts, where balancing exploration and exploitation is critical. Although in a few cases SAO slightly outperformed ASAO in final objective value, ASAO consistently achieved more stable convergence across iterations. These results validate the effectiveness of the hybrid structure of ASAO, showcasing its potential as a reliable and versatile optimizer for complex real-world engineering problems, including those involving dynamic, non-linear, and high-dimensional solution spaces.

C. Summary of Performance Evaluation

A comprehensive overview of the optimization performance for each benchmark function is presented in Table 2. This table consolidates the best results obtained by the Arithmetic Optimization Algorithm (AOA), the Smell Agent Optimization (SAO) algorithm, and the proposed Arithmetic Smell Agent Optimization (ASAO) algorithm across a diverse set of test functions. The performance metrics considered for comparison include

the best solution vector x discovered by each algorithm, and the corresponding objective function value f(x).

Table 2: Summary of Optimization Results – AOA vs SAO vs ASAO

Function	AOA		SAO		ASAO	
	x	f(x)	x	f(x)	x	f(x)
Adjiman	[2, 0.1057]	-2.0218	[2, 2]	-1.9994	[2, 0.1870]	-2.0218
Beale	[1.8158, -1.189e-32]	0.943	[-0.2616,2.1454]	1.0527e-05	[3.0297, 0.5506]	0.04872
Deckkers-Aarts	[0, 14.9451]	-24776.5183	[13.5352,13.1960]	-2.4755e+04	[0.0632, -14.8138]	-24346.5754
Easom	[3.1419, 3.1419]	-1	[4.2296,6.0727]	-0.9999	[3.0113, 3.2290]	-0.9637
Matyas	[-1.5315e-162, -2.0167e-169]	0	[8.3786,5.3455]	9.6259e-09	[0,0]	0
McCormick	[-0.0135, -0.8042]	-1.0948	[2.9171, -0.9417]	-1.913	[-0.1395, -1.6457]	-1.6135
Michalewicz	[1.5043, 1.5702]	-1.9956	[3.1415,3.01162]	-1.999	[1.698, 1.6305]	-1.9546
Quadratic	[0.19349,0 .4849]	-3873.7241	[0.9808,6.2689]	-3.8737e+03	[0.3160, 0.3784]	-3871.8699
Schaffer	[0,0]	0	[7.3777,9.9271]	1.0519e-12	[0,0]	0
Styblinski-Tang	[-2.9038, -1.8256]	-64.8395	[3.5399,1.5210]	-78.3321	[-2.8897 -2.8058]	-78.1693
Box-Betts	[1.2, 9, 1.2]	125.8576	[0.9841,1.2000,0.9939]	1.25857e+02	[1.2, 9, 1.2]	125.8576
Colville	[0.9201, 0.9795, 0.1249,	10.0887	[1,1,0.3557,-0.6695]	0	[1, 1, 1, 1]	0

	0.1163]					
Csendes	[2.4904e-61, -2.2586e-56, -1.3486e-64, 1.5758e-59]	0	[1,0.8046,-0.68570,-0.6652]	1.8174e-15	[-8.3951e-55, 2.7753e-55, 1.6154e-55, 1.1131e-55]	0

From the results, several key observations emerge that highlight the superior performance of ASAO. First and foremost, ASAO either matched or surpassed the performance of AOA and SAO on nearly all the benchmark functions in terms of the objective function value. This trend was especially evident in complex, multimodal functions where the landscape includes numerous local minima that typically trap conventional algorithms.

For instance, in challenging functions such as Michalewicz, and Deckkers-Aarts, ASAO consistently delivered results that were either equal to or closer to the known global optimum compared to those obtained using AOA and SAO. These functions are well-known for their deceptive fitness landscapes and require robust exploration strategies to avoid premature convergence. ASAO's hybrid structure enabled it to maintain a fine balance between exploration (global search) and exploitation (local refinement), allowing it to navigate these complex search spaces more effectively.

Moreover, ASAO maintained greater stability and consistency across repeated trials, minimizing deviations and showing smoother convergence trajectories. This consistent behavior underscores the robustness of the algorithm, making it well-suited for practical optimization tasks where reliability and repeatability are critical.

In essence, the data presented in Table 2 confirms that the ASAO algorithm not only achieves high-precision results across a wide variety of optimization problems but also does so with improved speed and dependability. These findings reinforce ASAO's suitability as a powerful and versatile tool for tackling real-world optimization challenges, particularly those characterized by high-dimensionality, non-linearity, and multiple local optima.

5 | CONCLUSION

The ASAO algorithm effectively merged the global search behavior of the SAO algorithm with the local search precision of AOA, achieving a well-balanced optimization process that mitigated common weaknesses of both parent algorithms. The performance of ASAO, as validated through benchmark function testing, consistently outperformed or matched the standalone algorithms in terms of solution quality, and convergence speed. Overall, the research confirms that ASAO is not only a theoretically sound advancement in optimization methods but also a practically valuable tool due to its adaptability, reliability, and computational performance. Future research will focus on implementing the proposed scheme in real world scenarios to ascertain the effectiveness of its performance in challenging settings. Also, subsequent studies will present the performance of ASAO against other metaheuristic algorithms in specific applications such as microgrid control.

Funding Statement: The author(s) received no specific funding for this study.

Author Contributions: N. O. Kalu: Conceptualization, Methodology, Software, Validation, Writing – original draft, Writing – review & editing. O. O. Oshiga: Supervision, Writing – review & editing. O. Oghorada: Writing – review & editing. J. A. Bala: Methodology, Software, Validation, Writing – original draft.

Availability of Data and Materials: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Abualigah, L., 2020. Multi-verse optimizer algorithm: a comprehensive survey of its results, variants, and applications. Neural Computing and Applications, 32(16), pp.12381–12401.

Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M. & Gandomi, A.H., 2021. The arithmetic

optimization algorithm. Computer Methods in Applied Mechanics and Engineering, 376, p.113609.

Ajala, E.O., Ehinmowo, A.B., Oladipupo, A.D., Opawoye, S., Ndana, A.M. and Ariyoosu, D.A., 2025. Experimental and Computational Modelling for Optimised Biodiesel Production from Waste Ram Fat Using a Kaolinite-Clay and Eggshell-Derived Catalyst. Nigerian Journal of Technological Development, 22(1), pp.95-114.

Arif, K., Zafar, A., Faheem, M., & Askari, Q. (2025, February). Improved arithmetic optimization algorithm. In International Conference on Energy, Power, Environment, Control and Computing (ICEPECC 2025) (Vol. 2025, pp. 554-562). IET.

Desale, S., Rasool, A., Andhale, S. and Rane, P., 2015. Heuristic and meta-heuristic algorithms and their relevance to the real world: a survey. Int. J. Comput. Eng. Res. Trends, 351(5), pp.2349-7084.

Dhal, K. G., Sasmal, B., Das, A., Ray, S., & Rai, R. (2023). A comprehensive survey on arithmetic optimization algorithm. Archives of Computational Methods in Engineering, 30(5), 3379-3404.

Drici, M., Houabes, M., Salawudeen, A. T., & Bahri, M. (2025). Optimizing Hybrid Renewable Energy Systems for Isolated Applications: A Modified Smell Agent Approach. Eng, 6(6), 120.

Fotopoulos, G.B., Popovich, P. and Papadopoulos, N.H., 2024. Review Non-convex Optimization Method for Machine Learning. arXiv preprint arXiv:2410.02017.

Jamil, M., & Yang, X. S. (2013). A literature survey of benchmark functions for global optimisation problems. International Journal of Mathematical Modelling and Numerical Optimisation, 4(2), 150-194.

Kotte, S., Injeti, S. K., Thunuguntla, V. K., Kumar, P. P., Nuvvula, R. S., Dhanamjayulu, C., ... & Khan, B. (2024). Energy curve based enhanced smell agent optimizer for optimal multilevel threshold selection of thermographic breast image segmentation. Scientific Reports, 14(1), 21833.

Nagy, M., Mansour, Y. and Abdelmohsen, S., 2020. Multi-objective optimization methods as a decision making strategy. Int. J. Eng. Res, 9(3).

Salawudeen, A., Mu'azu, M., Sha'aban, Y. & Adedokun, E., 2018. On the Development of a Novel Smell Agent Optimization (SAO) for Optimization Problems. i-manager's Journal on Pattern Recognition, 5(4), p.13.

Salawudeen, A.T., Mu'azu, M.B., Yusuf, A. & Adedokun, A.E., 2021. A Novel Smell Agent Optimization (SAO): An extensive CEC study and engineering application. Knowledge-Based Systems, 232, p.107486.

Salawudeen, A.T., Mu'azu, M.B., Yusuf, A. & Adedokun, E.A., 2018. From smell phenomenon to smell agent optimization (SAO): a feasibility study. In: Proceedings of ICGET 2018.

Salawudeen, A.T., Mu'azu, M.B., Yusuf, A. & Adedokun, E.A., 2020. Recent Metaheuristics Analysis of Path Planning Optimization Problems. In: 2020 International Conference in Mathematics, Computer Engineering and Computer Science (ICMCECS). IEEE, pp.1–7.

Sulaiman, A. T., Bello-Salau, H., Onumanyi, A. J., Mu'azu, M. B., Adedokun, E. A., Salawudeen, A. T., & Adekale, A. D. (2024). A particle swarm and smell agent-based hybrid algorithm for enhanced optimization. Algorithms, 17(2), 53.