Scope
Volume 14 Number 02 June 2024

Enhancing Slime Mould Algorithm Performance with Foraging Risk: Application to Constrained Engineering Optimization Problems

[1]Rajalakshmi Sakthivel & [2]Kanmani Selvadurai
[1]Department of Computer Science and Engineering, Puducherry Technological University, Puducherry, India
[2]Department of Information Technology, Puducherry Technological University, Puducherry, India

Abstract: Purpose : The primary objective of this research is to present the Slime Mould Algorithm with Foraging Risk (SMAFR), an enhanced version of the Slime Mould Algorithm (SMA). This development aims to address the inherent limitations of many metaheuristic algorithms, such as slow convergence and susceptibility to local optima. **Design/methodology/approach :** Drawing inspiration from the risk-related foraging behaviour observed in natural slime moulds, the SMAFR incorporates these behaviours into the algorithm's search mechanics. Through environmental parameters and mechanisms that mimic the slime mould's response to risk when seeking food, the algorithm is designed to balance exploration and exploitation. **Findings:** The SMAFR demonstrated improved efficiency over the original SMA. When rigorously assessed against three constrained real-world engineering problems, SMAFR exhibited superior performance in comparison to six other well-established metaheuristic algorithms. Statistical analyses of the results validated the effectiveness and robustness of the SMAFR in a variety of optimization scenarios. **Research limitations/implications :** While SMAFR exhibited strong performance in the tested scenarios, it's crucial to recognize the No Free Lunch (NFL) theorem's implications, suggesting that no single optimization technique is universally superior. The performance of the SMAFR, like all algorithms, may vary based on the specific optimization problem. **Practical implications :** The SMAFR offers an advanced solution for a range of optimization problems, especially those that require a balance between exploration and exploitation. Its design, influenced by real-world biological behaviours, lends it an edge in certain optimization scenarios, making it a viable option for professionals and researchers in relevant fields. **Originality/value:** The incorporation of foraging risk behaviour, inspired by actual slime mould behaviour, into a metaheuristic algorithm is the cornerstone of the SMAFR's originality. This unique approach not only enhances the algorithm's efficiency but also brings it closer to mimicking real-world adaptive behaviours. The presented work thus offers significant value by providing an innovative solution to tackle prevalent challenges in optimization algorithms.

Keywords: Slime Mould Algorithm, Foraging Risk, Metaheuristic Algorithm, Bio-Inspired Algorithm

1. Introduction

Optimisation algorithms are significant computational tools for solving complex problems, particularly when traditional mathematical methods prove inadequate. Among the wide variety of optimisation algorithms, metaheuristic algorithms (Yang, 2010, Kaveh, A., &IlchiGhazaan, M., 2017, Kaveh, A., &Dadras, A., 2017, Kaveh, A., &Zolghadr, A., 2017, Kaveh, A., Akbari, H., &Hosseini, S. M., 2020 and Kaveh, A., &Khayatazad, M., 2013) have gained substantial recognition due to their ability to operate without prior knowledge of the problem's structure. Utilising heuristic strategies, they efficiently search the solution space for optimal or near-optimal solutions. Consequently, they find extensive applications across diverse fields, such as engineering (Bozorg-Haddad et al., 2017), computer science (Yang, 2010), finance (Soler-Dominguez et al., 2017), and logistics (Zäpfel et al., 2010), offering valuable solutions when conventional methods fall short.Metaheuristic algorithms can be broadly classified into two categories: population-based metaheuristics (Beheshti and Shamsuddin, 2013) and single-solution-based metaheuristics (Abdel-Basset et al., 2018). Population-based metaheuristics (Beheshti and Shamsuddin, 2013) operate simultaneouslyon multiple potential solutions, making them especially effective for optimisation problems with extensive search space and multiple local optima. Two prominent types include Evolutionary Algorithms (EAs) (Yu and Gen, 2010) and Swarm Intelligence Algorithms (SIAs) (Chakraborty and Kar, 2017). EAs, inspired by natural selection and genetic recombination(Mirjalili and Mirjalili, 2019, Beyer and Schwefel, 2002, Koza, 1994), evolve a population of potential solutions to improve quality, thereby demonstrating an impressive capability in handling large search spaces and multiple local optimisations. However, they often require substantial computational resources and precise parameter optimisation. On the other hand, SIAs imitate the collective behaviour of self-organised systems, where simple behavioural rules allow for efficient search space exploration. Despite their advantages, SIAs also demand careful parameter tuning and significant computational resources.

Single solution-based metaheuristics(Abdel-Basset et al., 2018) are optimization methods that iteratively refine a single solution to achieve satisfactory results. Examples include Hill Climbing(Greiner, 1996), which makes small, iterative improvements; Simulated Annealing(Bertsimas and Tsitsiklis, 1993), inspired by metallurgical annealing, combining new solution exploration with refinement; and Tabu Search (Glover and Laguna, 1998), using memory structures to avoid revisiting solutions. Other emerging techniques like Variable Neighborhood Search (VNS) alter local search neighborhoods (Mladenović& Hansen, 1997), Iterated Local Search (ILS) applies local searches with intermittent perturbations (Lourenço, Martin, &Stützle, 2003), Guided Local Search (GLS) uses penalties to direct the search (Voudouris, Tsang, &Alsheddy, 2010), Greedy Randomized Adaptive Search Procedure (GRASP) constructs and iteratively refines randomized solutions (Feo&Resende, 1995), and

Harmony Search (HS) improves solutions based on a harmony memory consideration rate (Geem, Kim, &Loganathan, 2001). Each method offers distinct strengths, making them suitable for various optimization challenges, ranging from well-established approaches like Hill Climbing(Greiner, 1996) to innovative strategies like Harmony Search (Geem, Kim, &Loganathan, 2001).

The main objective of metaheuristic algorithms (Rajalakshmi and Kanmani, 2022, Rajalakshmi et al., 2021, Dhanya et al., 2018, Rajalakshmi et al., 2021, Rajalakshmi et al., 2021) is to find optimal or near-optimal solutions for given optimisation problems by deploying heuristic strategies for effective search space exploration. In this paper, anovel nature-inspired metaheuristic algorithm called the Slime Mould Algorithm with Foraging Risk (SMAFR) is introduced. It employs the foraging behaviour of slime moulds and considers the associated risks, aiming to address the drawbacks of the existing metaheuristics. To validate the efficiency of the proposed SMAFR, it was rigorously test it against a set of three constrained engineering problems.

2. Literature Study

The field of nature-inspired metaheuristic algorithms (Zhang, Y., et al., 2015, Zhang & Chen, 2023) continues to experience significant developments. The literature reveals numerous algorithms inspired by various species' behaviours, exhibiting remarkable problem-solving abilities, particularly in optimization tasks (Sakthivel&Selvadurai, 2024). The Fennec Fox Optimization (FFA) algorithm is an important example of a nature-inspired algorithm. It models the behaviours of the Fennec Fox's to balance exploration and exploitation in the optimization process (Trojovská et al., 2022). Similarly, the Reptile Search Algorithm (RSA) takes its inspiration from the hunting strategies of crocodiles. It distinguishes itself from other algorithms by incorporating both encircling and foraging behaviours into its framework (Abualigah et al., 2022).The Aquila Optimizer (AO) is inspired by the hunting techniques of the Aquila, or eagle. This algorithm mimics the eagle's diverse predatory strategies, including the high-altitude soaring followed by a vertical stoop, and the low-altitude contour flight that culminates in a short glide to attack. By emulating these varied flight patterns and attack strategies, AO effectively navigates through complex optimization landscapes, adapting its search methodology to suit different problem scenarios. This unique approach has demonstrated effectiveness in solving a wide range of complex optimization problems (Abualigah et al., 2022).Also noteworthy is the Jellyfish Search (JS) optimizer, which encapsulates the behaviour of jellyfish, proving effective especially in solving mathematical benchmark functions of various dimensions (Chou and Truong, 2021).

A novel development is the Cat and Mouse-Based Optimizer (CMBO) that mimics the interactions between cats and mice. The CMBO's strength lies in its ability to provide quasi-optimal solutions closer to the global optimum, proving itself as an efficient tool in the optimization landscape (Dehghani et al., 2021). The Cheetah Optimizer (CO) (Akbari et al., 2022) is a novel algorithm inspired by cheetahs' hunting behaviour, known for its rapid convergence and dynamic adaptation strategy, effectively balancing exploration and exploitation in large, complex search spaces. Integrating both stochastic and deterministic elements, it mimics the cheetah's strategic movements, aiding in avoiding local optima and ensuring thorough solution space exploration.

The Artificial Gorilla Troops Optimizer (GTO) simulates the social intelligence of gorilla troops. The GTO has demonstrated superior performance, particularlyfor high-dimensional problems; thereby expanding the capabilities of existing metaheuristic optimization algorithms (Abdollahzadeh et al., 2021). Now focusing on the Slime Mould Algorithm (SMA)(Li et al., 2020), it has undergone multiple enhancements. The Chaotic Slime Mold Algorithm (CSMA) was proposed to improve the basic SMA's exploitation phase, resultingin superior performance in terms of solution accuracy (Altay, 2022). In another study, the enhanced version of SMA, named MSMA, enhanced the standard SMA by incorporating a chaotic opposition-based learning strategy and two adaptive parameter control strategies (Tang et al., 2021). Similarly, the Chaotic Slime Mould Optimization Algorithm (CSMOA) was developed to speed up SMA's global convergence and avoid local solutions (Dhawale et al., 2021).

Despite these improvements, it is essential to note that such enhancements often compromise the biological authenticity of the original algorithms. Therefore, proposed work focuses on developing an enhanced version of the SMA, ensuring that its essence - derived from the biological behaviour of slime moulds - remains intact. This approach is expected to balance the need for optimization efficiency and the preservation of natural principles at the core of these nature-inspired algorithms. The rest of this paper is structured as follows: Section 3 provides a detailed description of the SMA. Section 4 presents the proposed SMAFR. The comparison of SMAFR with six other metaheuristic algorithms across three engineering design problem is presented in Section 5. Finally, Section 6 concludes the study and discusses the potential implications of findings.

3. Slime Mould Algorithm (Li et al., 2020)

The Slime Mould Algorithm (SMA), a novel optimisation strategy, was first proposed by Li et al. (2020). The fundamental principle of SMA is inspired by the behaviour of a eukaryotic organism, slime mould, during in its plasmodium stage.

During this phase, the slime mould grows slowly in colder temperatures and focuses predominantly on the search and digestion of food. The organic component of the slime mould navigates towards food sources, secretes digestive enzymes, and effectively breaks down food particles.Li et al. (2020) provided a mathematical model encapsulating this foraging behaviour of slime mould. According to this model, the movement of the slime mould is triggered by the scent of the food source present in the environment. The model can be represented mathematically as follows:

$$\vec{X} = \begin{cases} \overrightarrow{X_b(t)} + \overrightarrow{v_b}.\left(W.\overrightarrow{X_A(t)} - \overrightarrow{X_B(t)}\right), r < p \\ \overrightarrow{v_c}.\overrightarrow{X(t)}, r \geq p \end{cases} \tag{3.1}$$

In this equation, $\overrightarrow{X_b(t)}$ represents the current state of the slime mould, while $\overrightarrow{X_A(t)}$and $\overrightarrow{X_B(t)}$ are random selections of slime mould. The variable $(\overrightarrow{v_c})$ gradually decreases from 1 to 0, and $(\overrightarrow{v_b})$ ranges between [-a, a]. The parameter 'a' is calculated as:

$$a = \operatorname{arctanh}\left(-\left(\frac{t}{max\_t}\right) + 1\right) \tag{3.2}$$

The parameter 'p' is defined by:

$$p = \tanh|S(i) - DF| \tag{3.3}$$

Where,S(i) represents the fitness of the slime mould and DF is the overall best fitness. The weight (W) of the slime is represented as:

$$\overrightarrow{W(smellIndex(\imath))} = \begin{cases} 1 + r.\log\left(\frac{bF - S(i)}{bF - wF} + 1\right), & \text{First half in } S(i) \\ 1 - r.\log\left(\frac{bF - S(i)}{bF - wF} + 1\right), & \text{others} \end{cases} \tag{3.4}$$

WheresmellIndex represents the sorting of S, bF stands for best fitness and wF for worst fitness, and 'r' is a random value between [0, 1]. The food attraction of the slime mould is given by:

$$\vec{X} = \begin{cases} rand.\ (UB - LB) + LB, & rand < z \\ \overrightarrow{X_b(t)} + \overrightarrow{vb}.\left(W.\overrightarrow{X_A(t)} - \overrightarrow{X_B(t)}\right), & r < p \\ \overrightarrow{vc}.\overrightarrow{X(t)}, & r \geq p \end{cases} \tag{3.5}$$

The variable z is set at 0.3, where LB and UB denote the lower and upper bounds, respectively, and both 'rand' and 'r' fall within the range of [0, 1]. This mathematical representation characterises the slime mould's food foraging behaviour can be effectively used for optimisation problems (Molga and Smutnicki, 2005, Yang, 2010). Despite the noteworthy performance exhibited by the Slime Mould Algorithm across various scenarios, it's not without its limitations. The algorithm frequently faces issues such as suboptimal convergence and susceptibility to entrapment in local optima. Numerous algorithms (Altay, 2022, Tang et al., 2021, Dhawale et al., 2021) have been proposed to rectify these problems, but these algorithms often fail to preserve the inherent biological functionality of the original algorithm. A novel enhanced proposed algorithm is the Slime Mould Algorithm with Foraging Risk, which is designed to address these issues while ensuring the preservation of biologically-inspired functionality.

4. Slime Mould Algorithm with Foraging Risk

This section introduces the Slime Mould Algorithm with Foraging Risk (SMAFR), a novel evolution of the original Slime Mould Algorithm (SMA) (Li et al., 2020). In its natural environment, the behaviour of slime mould (Vallverdú et al., 2018) a unicellular organism prevalent in a variety of conditions is significantly influenced by factors like temperature, humidity, light, and food supply (Takamatsu et al., 2009). This behavioural adaptation, intrinsic to slime mould's food-seeking process, serves as the inspiration for the SMAFR.One of the primary adaptive mechanisms of slime mould is its ability to assess risks when foraging for food. This behaviour not only retains the algorithm's biological functionality but also enhances its efficiency. Slime mould prefers humid environments and stays inactive during unsuitable environmental situations, waiting for optimal conditions before expanding. These characteristics of slime mould provide us with the opportunity to overcome the slow convergence and local optima adherence issues faced by the original SMA.

The environmental condition is represented by a parameter called 'C' in SMAFR is defined as:

$$\mathrm{C} = \sqrt{\sum_{\mathrm{i}=1}^{\mathrm{d}} (\mathrm{X(t)_i} - \mathrm{X_{GB}(t)})^2} \qquad (4.1)$$

WhereX(t) denotes the current slime, and $\mathrm{X}_{GB}(\mathrm{t})$ stands for the best slime. The minimum climatic conditions required for the growth of slime mould are represented by$\mathrm{C_{min}}$, which is given by:

$$\mathrm{C_{min}} = \frac{10\mathrm{E}^{-6}}{(365)^{\frac{\frac{\mathrm{iter}}{\mathrm{Max-iter}}}{2.5}}} \qquad (4.2)$$

In nature, slime moulds exhibit a foraging behaviour that is sensitive to environmental conditions like nutrient availability. The SMAFR algorithm mimics this by using 'C' to determine if the conditions are right for the slime mould to actively forage.In SMAFR, 'C' helps assess the 'risk' associated with continuing to explore the current region of the search space. A high 'C' value might indicate a higher 'risk', prompting the algorithm to explore new areas, while a low 'C' value indicates a lower 'risk' and hence, a preference to exploit the current area. 'C' reflects the environmental suitability or the favourability of conditions for the slime mould. A smaller value of 'C' suggests that the slime mould is closer to the optimal position, indicating a more favourable environment. Conversely, a larger value of 'C' implies that the slime mould is further from the optimal position, suggesting less favourable conditions.

This behaviour can be mathematically represented as follows:

$$X(t) = LB + Levy(n) \times (UB - LB),\ C > C_{min} \quad (4.3)$$

In this equation, the Levy flight is calculated as:

$$Levy(n) = 0.01 \times \frac{r_1 \times \sigma}{|r_2|^{\frac{1}{\beta}}} \quad (4.4)$$

and

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{\frac{1}{\beta}} \quad (4.5)$$

Where, $\Gamma(n) = (n-1)!$, r_1 & r_2 are random numbers in the range [0,1], β is set to 1.4. The pseudo-code of the proposed Slime Mould Algorithm with Foraging Risk (SMAFR) is detailed as follows.

Algorithm: Slime Mould Algorithm with Foraging Risk (SMAFR)
Input: Population of slime, Algorithm parameters, Maximum iterations
Output: Best_fitness

```
1: SMAFR(Population, Parameters, Max_iter)
2:    Initialize slime population X = {$X_1$, $X_2$, ..., $X_N$};
3:    Initialize iteration counter t = 0;
4:    while t <= Max_iter do
5:        for each slime i in population do
6:            Compute fitness F(i);
7:            Compute weight W(i);
8:        end for
9:        for each search agent i in population do
10:           Update p, $v_b$, and $v_c$ for agent i;
11:           Update the position of agent i, X(i, t+1);
12:       end for
13:       Compute environmental condition C for each agent;
14:       if C satisfies the environmental monitoring condition, then
15:           Update X(t) for each agent;
16:       end if
17:       Update minimum environmental condition $C_{min}$;
18:       Increment iteration counter t = t + 1;
19:   end while
```

```
20:    best_fitness = minimum fitness in the population;
21:   return best_fitness;
22: end
```

This pseudo-code outlines the main steps of the SMAFR, beginning with the initialization of the slime population and algorithm parameters. Subsequently enters a loop, which lasts for a predetermined maximum number of iterations. Within this loop, the algorithm computes the fitness and weight of slime, updates various parameters, and checks the environmental condition. If the condition is met,the algorithm will updatethe slime's position. In the SMAFR algorithm, the process iterates through slime mould populations and updates their positions until a predefined maximum number of iterations, Max_iter, is reached. An iteration counter, starting at zero, increments after each cycle of operations, including fitness evaluation and position updates. The algorithm concludes once this counter equals Max_iter, ensuring a finite and controlled execution, after which the optimal solution found is returned.The proposed modifications to the SMAFR enhance its resemblance to the biological behaviours of slime mould and improve its performance by addressing issues present in the original SMA. As a result, the SMAFR shows potential as a valuable tool for resolving complex optimisation problems.

5. Constrained Engineering Optimization using SMAFR Algorithm

Constraint management is a critical process in the optimization process, playing a vital role in distinguishing between feasible and infeasible solutions produced by heuristic algorithms. There are numerous strategies available for constraint management, including but not limited to penalty functions, special operators, repair algorithms, separation of objectives and constraints, and hybrid techniques (Coello, 2002, Sakthivel&Selvadurai, 2024). The penalty function is one of the simplest and most commonly used strategies. This method focuses on penalizing solutions that fall outside the feasible solution space, effectively transforming the constrained optimization problem into an unconstrained optimization problem. Constraints are efficiently managed within the context of the SMAFR algorithm through the implementation of a penalty function. To illustrate the proficiency of the SMAFR algorithm in managing constraints, this section investigates its performance using three typical constrained engineering optimization problems: gear train design, three-bar truss design, and the welded beam design problem. The comprehensive analysis and results of these application-based evaluations provide additional perspectives on the practical efficiency and versatility of the SMAFR algorithm(Sakthivel&Selvadurai, 2024).

5.1 Gear Train Design Problem

A common problem in the field of mechanical engineering is the optimization of gear ratios in a four-gear train set. The primary objective of this problem is to minimize the gear ratio for this specific set, where the number of teeth on each of the four gears represents the variables (Sandgren, 1990, CoelloCoello, 2000). While there are no explicit constraints in this problem, the range of variables, or in other words, the feasible number of teeth a gear can have, are implicitly considered as constraints that influence the gear ratio and consequently, the overall system efficiency.The general design of this gear system is depicted in Figure 1. The following section presents the mathematical formulation for the optimisation of the gear train design problem. This problem exemplifies the efficacy of the SMAFR algorithm in managing engineering constraints in a practical setting(Sakthivel&Selvadurai, 2024).

$$Min\ f(T) = \left(\frac{1}{6.931} - \frac{T_2\,T_3}{T_1\,T_4}\right)^2, where\ T = (T_1, T_2, T_3, T_4) \qquad (5.1)$$

$\exists\ 12 \leq T_i \leq 60, for\ i = 1,2,3,4.$

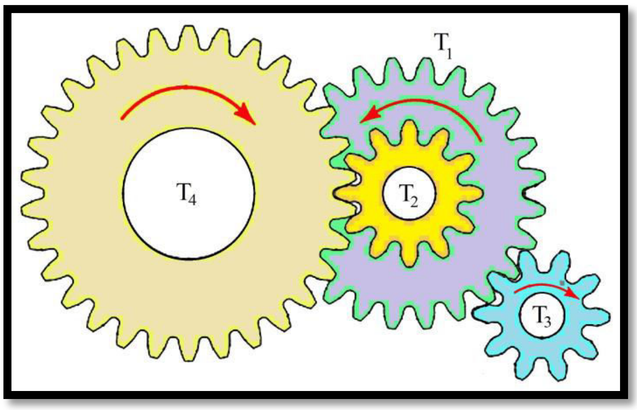

Figure 1. Gear Train Design Problem.

Table 1 presents the most optimal results produced by the SMAFR algorithm for the Gear Train Design Problem. In the evaluation of the Gear Train Design Problem, the SMAFR algorithm demonstrated notable superiority over other algorithms like PSO (Clerc, 2010), SMA (Li et al., 2020), ABC (Karaboga and Basturk, 2008), DE (Price, 2013), WOA (Mirjalili and Lewis, 2016), and GOA (Mirjalili et al., 2018) (Sakthivel &Selvadurai, 2024). SMAFR achieved the lowest optimum cost with a more efficient convergence rate, showcasing its ability to quickly identify the most effective gear configurations. Its performance edge is attributed to its unique foraging risk mechanism, inspired by slime mould behavior, which effectively balances exploration and exploitation in the search space. This adaptability, coupled with robustness against local optima and computational efficiency, makes SMAFR particularly effective for complex optimization tasks like the gear train design, where finding a global optimum is crucial amidst a landscape filled with numerous local optima.

Table 1. Performance Analysis of Gear Train Design Problem

Algorithm	Optimum values for variables				Optimum cost
	T_1	T_2	T_3	T_4	
SMAFR	41	12	12	20	**3.2749E-18**
SMA	43	12	19	36	3.0372E-12
PSO	46	20	17	44	2.6008E-011
ABC	43	19	16	43	1.251E-10
DE	44	14	18	48	2.4008E-10
GOA	20	16	43	49	2.75E-10
WOA	34	14	18	48	1.263E-08

5.2 Three Bar Truss Design Problem

The three-bar truss design problem, a prevalent structural optimization problem in civil engineering, requires the manipulation of two variables in order to achieve the lowest possible weight, under the constraints of stress, deflection, and buckling. The search space of this problem is severely constrained, necessitating extensive research. The problem's representation (Sandgren, 1990, CoelloCoello, 2000) is illustrated in Figure 2. The mathematical expression of the three-bar truss design problem can be given as follows(Sakthivel&Selvadurai, 2024):

$$min\ f(X) = L \times \left(2\sqrt{2}x_1 + x_2\right) \quad (5.2)$$

$$\exists \quad \frac{\sqrt{2}x_1+x_2}{2x_1x_2+\sqrt{2}x_1^2}P \le \sigma \quad (5.3)$$

$$\frac{x_2}{2x_1x_2+\sqrt{2}x_1^2}P \le \sigma \quad (5.4)$$

$$\frac{1}{x_1+\sqrt{2}x_2}P \le \sigma \quad (5.5)$$

$0.01 \le x_i \le 1\ for\ i = 1,2.$

where, L=100 cm, P=2 km/cm^2 and σ=2 km/cm^2

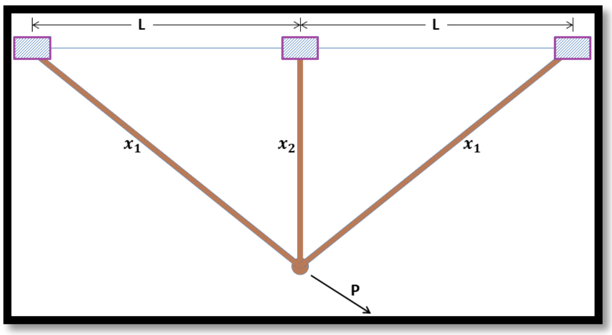

Figure 2. Three Bar Truss Design Problem.

Optimal solutions obtained by the SMAFR algorithm for the three-bar truss design problem are detailed in Table 2. In the assessment of the Three Bar Truss Design Problem, a critical challenge in civil engineering optimization, the SMAFR algorithm emerged as a particularly effective solution. This problem, focusing on minimizing the weight of a truss under specific stress, deflection, and buckling constraints, demands precise manipulation of two variables. SMAFR outperformed other algorithms such as PSO (Clerc, 2010), SMA (Li et al., 2020), ABC (Karaboga and Basturk, 2008), DE (Price, 2013), WOA (Mirjalili and Lewis, 2016), and GOA (Mirjalili et al., 2018), achieving the lowest optimum cost (Sakthivel &Selvadurai, 2024). Notably, SMAFR's success in this task can be attributed to its advanced search capabilities and adaptability, inspired by biological processes. It effectively navigated the constrained search space and balanced the competing demands of stress and weight minimization, demonstrating its potential as a robust tool for complex structural optimization problems in civil engineering.

TABLE 2. Performance Analysis of Three Bar Truss Design Problem.

Algorithm	**Decision variables**		**Optimum cost**
	x_1	x_2	
SMAFR	0.66579	0.33476	**186.6721**
SMA	0.76345	0.46321	215.6541
PSO	0.78997	0.66579	228.6549
DE	0.81815	0.36946	267.8173
ABC	0.74669	0.41526	265.9358
GOA	0.78967	0.41932	264.9815
WOA	0.79603	0.42945	261.8754

5.3 Welded Beam Design Problem

The welded beam design problem, a well-established problem in structural optimization (Tang et al., 2021), endeavors to minimize the production costs of a welded beam. Figure 3 depicts the Welded Beam Design Problem (Sandgren, 1990,CoelloCoello, 2000,Deb, 1991). Optimization constraints for this problem include shear stress (τ), bending stress in the beam (θ), buckling load (Pc), and beam end deflection (δ). The four variables considered during optimization include weld thickness (h), clamping bar length (l), bar height (t), and bar thickness (b). The mathematical formulation of the welded beam design problem can be outlined as follows(Sakthivel&Selvadurai, 2024):

Consider $x = (x_1, x_2, x_3, x_4) = [h\ l\ t\ b]$,

$$min\, f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) \quad (5.6)$$

$$\exists g_1(x) = \tau(x) - \tau_{max} \le 0, \quad (5.7)$$

$$g_2(x) = \sigma(x) - \sigma_{max} \le 0, \quad (5.8)$$

$$g_3(x) = \delta(x) - \delta_{max} \le 0, \quad (5.9)$$

$$g_4(x) = x_1 - x_4 \le 0, \quad (5.10)$$

$$g_5(x) = P - P_c(x) \le 0, \quad (5.11)$$

$$g_6(x) = 0.125 - x_1 \le 0 \quad (5.12)$$

$$g_7(x) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \le 0 \quad (5.13)$$

$0.1 \le x_1 \le 2$,
$0.1 \le x_2 \le 10$,
$0.1 \le x_3 \le 10$,
$0.1 \le x_4 \le 2$

$$\text{where}\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}, \quad (5.14)$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2}, \tau'' = \frac{MR}{J}, M = P\left(L + \frac{x_2}{2}\right), \quad (5.15)$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2}, \quad (5.16)$$

$$J = 2\left\{\sqrt{2}x_1x_2\left[\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2\right]\right\}, \quad (5.17)$$

$$\sigma(x) = \frac{6PL}{x_4x_3^2}, \quad (5.18)$$

$$\delta(x) = \frac{6PL^3}{Ex_3^2x_4}, \quad (5.19)$$

$$P_c(x) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)' \quad (5.20)$$

$P = 6000\text{lb}, L = 14\text{ in.}, \delta_{max} = 0.25\text{ in.}, E = 30 \times 1^6\text{psi}, G = 12 \times 10^6\text{psi}$,
$\tau_{max} = 13{,}600\text{ psi}, \sigma_{max} = 30{,}000\text{ psi}$.

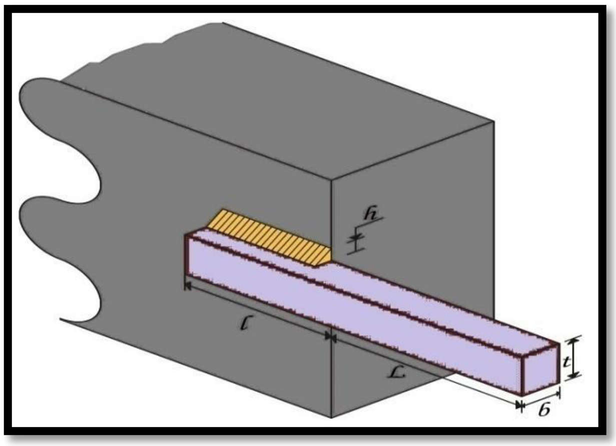

Figure 3. Welded Beam Design Problem

Table 3. Performance Analysis of Welded Beam Design Problem

Algorithm	Optimum values for variables				Optimum cost
	h	L	t	b	
SMAFR	0.2034	3.2651	9.0462	0.2156	**1.67542**
SMA	0.2046	3.2874	9.0321	0.2086	1.69752
PSO	0.2047	3.3702	9.0462	0.2046	1.76454
DE	0.2035	3.4611	9.0375	0.2081	1.72997
ABC	0.2059	3.4785	9.0158	0.2034	1.72328
GOA	0.2028	3.3603	9.0563	0.2054	1.71492
WOA	0.2043	3.3725	9.0453	0.2145	1.72523

The SMAFR algorithm's performance in addressing the Welded Beam Design Problem, a notable challenge in structural optimization, is presented in Table 3. This comparison demonstrates SMAFR's superiority over other contemporary algorithms such as PSO (Clerc, 2010), SMA (Li et al., 2020), ABC (Karaboga and Basturk, 2008), DE (Price, 2013), WOA (Mirjalili and Lewis, 2016), and GOA (Mirjalili et al., 2018). This problem involves minimizing the production costs of a welded beam while adhering to constraints like shear stress, bending stress, buckling load, and beam end deflection(Sakthivel&Selvadurai, 2024). SMAFR achieved the most cost-effective design with optimal variable values (h, l, t, b) yielding the lowest cost at 1.67542. Its success can be attributed to its advanced search and optimization capabilities, which efficiently navigated the complex interplay of variables and constraints. SMAFR's performance in this context underscores its potential as an effective tool for intricate structural optimization tasks, where precise balancing of multiple factors is crucial for optimal design.

6. Conclusions and Future Works

The novel SMAFR algorithm has demonstrated exceptional efficiency in exploring and exploiting promising search regions, as evidenced by its performance in three constrained engineering problems. When compared with well-established algorithms like SMA, PSO, ABC, DE, GOA, and WOA, SMAFR has shown its capability to adeptly navigate complex problems, proving its robustness in various domains. However, in line with the No Free Lunch (NFL) theorem, it's important to recognize that no single optimization technique excels universally. SMAFR, in this respect, has emerged as a powerful optimizer in most explored scenarios.Looking forward, there is substantial potential to expand SMAFR's scope. The exploration of a binary iteration of the algorithm could address multi-objective issues, enhancing its applicability and effectiveness across diverse optimization scenarios. Additionally, its adaptability and efficiency position SMAFR as a prime algorithm for application in larger-scale, real-world problems (Kaveh, 2014).

The future research directions and expanded applications of SMAFR are

i. **Advanced Adaptation Mechanisms:** Future research could delve into more complex adaptation mechanisms inspired by slime mould behaviour, enhancing efficiency in dynamic optimization scenarios.
ii. **Hybridization and Integration:** Combining SMAFR with other robust optimization algorithms could create synergistic effects for more complex problems.
iii. **Expanding to New Domains:** Testing SMAFR in fields like renewable energy optimization, bioinformatics, and complex network analysis could demonstrate its versatility.
iv. **Dynamic Environmental Simulation:** Incorporating simulations of dynamic environmental factors in future SMAFR versions could mirror the natural adaptability of slime mould, benefiting rapidly changing conditions like financial markets or logistics.
v. **Customization for Specific Constraints:** Tailoring SMAFR to tackle special constraints, such as multi-objective or highly non-linear problems, could lead to more efficient solutions in fields like aerospace engineering.

The exploration of SMAFR's applications extends its potential, contributing substantially to the evolution of optimization techniques in engineering and other fields. As SMAFR transitions from a conceptual framework to a practical tool, it presents ongoing opportunities for future research and application, showcasing its relevance and adaptability in various domains. This progression highlights the significance of continuous development and adaptation in optimization methodologies.

Statements and Declarations

Acknowledgments:

We express deep appreciation to the faculty, peers, and staff at Puducherry Technological University for their unwavering support and valuable insights, which have greatly enriched our research. Our thanks also extend to all individuals who contributed their feedback and guidance throughout various phases of this project.

Funding Information:

No specific grants from any external funding bodies in the public, commercial, or not-for-profit sectors were received to support this research. Nonetheless, the internal resources and backing provided by Puducherry Technological University were instrumental in facilitating our study.

Author's Contributions:

Rajalakshmi Sakthivel: Led the conceptualization and design of the methodology, developed and implemented the slime mould algorithm with foraging risk, carried out the experimental evaluations, and authored the initial manuscript draft.

Kanmani Selvadurai: Oversaw the optimization techniques and metaheuristic algorithms, enhanced the methodology design, and contributed to analyzing and interpreting the experimental data. Additionally, revised and refined the manuscript to ensure clarity and quality.

Ethics:

This research was performed in strict accordance with ethical standards and adhered to the highest ethical guidelines set by Puducherry Technological University. There were no ethical conflicts or concerns during the study.

Conflict of Interest:

The authors declare no competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Abdollahzadeh, B., Gharehchopogh, F. S. and Mirjalili, S. (2021). "Artificial gorilla troops optimizer: a new nature-inspired metaheuristic algorithm for global optimization problems". International Journal of Intelligent Systems, 36(10), 5887-5958.

[2] Abualigah, L., AbdElaziz, M., Sumari, P., Geem, Z. W. and Gandomi, A. H. (2022). "Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer". Expert Systems with Applications, 191, 116158.

[3] Abualigah, L., Yousri, D., AbdElaziz, M., Ewees, A. A., Al-Qaness, M. A. and Gandomi, A. H. (2021). "Aquila optimizer: a novel meta-heuristic optimization algorithm". Computers & Industrial Engineering, 157, 107250.

[4] Akbari, M. A., Zare, M., Azizipanah-Abarghooee, R., Mirjalili, S. and Deriche, M. (2022). "The cheetah optimizer: A nature-inspired metaheuristic algorithm for large-scale optimization problems". Scientific reports, 12(1), 10953.

[5] Altay, O. (2022). "Chaotic slime mould optimization algorithm for global optimization". Artificial Intelligence Review, 55(5), 3979-4040.

[6] Beheshti, Z. and Shamsuddin, S. M. H. (2013). "A review of population-based meta-heuristic algorithms". Int. J. Adv. Soft Comput. Appl, 5(1), 1-35.

[7] Bertsimas, D. and Tsitsiklis, J. (1993). “Simulated annealing”. Statistical science, 8(1), 10-15.

[8] Beyer, H.-G. andSchwefel, H.-P. (2002). “Evolution strategies–a comprehensive introduction”. Natural computing, 1, 3-52.

[9] Bozorg-Haddad, O., Solgi, M. and Loáiciga, H. A. (2017). Meta-heuristic and evolutionary algorithms for engineering optimization. John Wiley & Sons.

[10] Chakraborty, A. and Kar, A. K. (2017). “Swarm intelligence: A review of algorithms”. Nature-inspired computing and optimization: Theory and applications, 475-494.

[11] Chou, J.-S. and Truong, D.-N. (2021). “A novel metaheuristic optimizer inspired by behavior of jellyfish in ocean”. Applied Mathematics and Computation, 389, 125535.

[12] CoelloCoello, C. A. (2000). “Constraint-handling using an evolutionary multiobjective optimization technique”. Civil Engineering Systems, 17(4), 319-346.

[13] Coello, C. A. C. (2002). “Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art”. Computer methods in applied mechanics and engineering, 191(11-12), 1245-1287.

[14] Deb, K. (1991). “Optimal design of a welded beam via genetic algorithms”. AIAA journal, 29(11), 2013-2015.

[15] Dehghani, M., Hubálovský, Š. and Trojovský, P. (2021). “Cat and mouse based optimizer: A new nature-inspired optimization algorithm”. Sensors, 21(15), 5214.

[16] Dhanya, K. M., Rajalakshmi, S., Kanmani, S. and Saraswathi, S. (2018, December). “Comparative Analysis Of Nature Inspired Insect Algorithms”. In proceedings of the national conference on machine learning techniques.

[17] Dhawale, D., Kamboj, V. K. and Anand, P. (2021). “An effective solution to numerical and multi-disciplinary design optimization problems using chaotic slime mold algorithm”. Engineering with Computers,1-39.

[18] Feo, T. A., &Resende, M. G. (1995). Greedy randomized adaptive search procedures. Journal of global optimization, 6, 109-133.

[19] Geem, Z. W., Kim, J. H., &Loganathan, G. V. (2001). A new heuristic optimization algorithm: harmony search. simulation, 76(2), 60-68.

[20] Glover, F. and Laguna, M. (1998). Tabu search (pp. 2093-2229). Springer US.

[21] Greiner, R. (1996). “PALO: A probabilistic hill-climbing algorithm”. Artificial Intelligence, 84(1-2), 177-208.

[22] Karaboga, D. and Basturk, B. (2008). “On the performance of artificial bee colony (ABC) algorithm”. Applied soft computing, 8(1), 687-697.

[23] Kaveh, A. (2014). Advances in metaheuristic algorithms for optimal design of structures (pp. 9-40). Switzerland: Springer International Publishing.

[24] Kaveh, A., &Dadras, A. (2017). A novel meta-heuristic optimization algorithm: thermal exchange optimization. Advances in engineering software, 110, 69-84.

[25] Kaveh, A., &IlchiGhazaan, M. (2017). A new meta-heuristic algorithm: vibrating particles system. ScientiaIranica, 24(2), 551-566.

[26] Kaveh, A., &Khayatazad, M. (2013). Ray optimization for size and shape optimization of truss structures. Computers & Structures, 117, 82-94.

[27] Kaveh, A., &Zolghadr, A. (2017). Cyclical parthenogenesis algorithm for layout optimization of truss structures with frequency constraints. Engineering Optimization, 49(8), 1317-1334.

[28] Kaveh, A., Akbari, H., &Hosseini, S. M. (2020). Plasma generation optimization: a new physically-based metaheuristic algorithm for solving constrained optimization problems. Engineering Computations, 38(4), 1554-1606.

[29] Koza, J. R. (1994). Genetic programming II: automatic discovery of reusable programs. MIT press.

[30] Li, S., Chen, H., Wang, M., Heidari, A. A. and Mirjalili, S. (2020). "Slime mould algorithm: A new method for stochastic optimization". Future Generation Computer Systems, 111, 300-323.

[31] Lourenço, H. R., Martin, O. C., &Stützle, T. (2003). Iterated local search. In Handbook of metaheuristics (pp. 320-353). Boston, MA: Springer US.

[32] Mirjalili, S. and Lewis, A. (2016). "The whale optimization algorithm". Advances in engineering software, 95, 51-67.

[33] Mirjalili, S. and Mirjalili, S. (2019). "Genetic algorithm". Evolutionary Algorithms and Neural Networks: Theory and Applications, 43-55.

[34] Mirjalili, S. Z., Mirjalili, S., Saremi, S., Faris, H. and Aljarah, I. (2018). "Grasshopper optimization algorithm for multi-objective optimization problems". Applied Intelligence, 48, 805-820.

[35] Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. Computers & operations research, 24(11), 1097-1100.

[36] Molga, M. and Smutnicki, C. (2005). "Test functions for optimization needs". Test functions for optimization needs, 101, 48.

[37] Price, K. V. (2013). Differential evolution. In Handbook of optimization: From classical to modern approach (pp. 187-214). Berlin, Heidelberg: Springer Berlin Heidelberg.

[38] Rajalakshmi, S. and Kanmani, S. (2022). "A comprehensive review on recent intelligent metaheuristic algorithms". International Journal of Swarm Intelligence, 7(2), 182-205.

[39] Rajalakshmi, S., Kanmani, S. and Saraswathi, S. (2021). "Improved Dragonfly Algorithm with Neighbourhood Structures". International Journal of Scientific Research in Science and Technology, 8, 303-309.

[40] Rajalakshmi, S., Kanmani, S., Varsha, C. and Auxilia, E. D. (2021). "Enhancement of Quadratic Assignment Problem Using Slime Mould Algorithm". International Journal of Scientific Research in Science and Technology, 9, 565-576.

[41] Sakthivel, R. &Selvadurai, K. (2024). Slime Mould Reproduction: A New Optimization Algorithm for Constrained Engineering Problems. Journal of Computer Science, 20(1), 96-105.

[42] Sandgren, E. (1988, September). Nonlinear integer and discrete programming in mechanical design. In International design engineering technical conferences and computers and information in engineering conference (Vol. 26584, pp. 95-105). American Society of Mechanical Engineers.

[43] Soler-Dominguez, A., Juan, A. A. and Kizys, R. (2017). “A survey on financial applications of metaheuristics”. ACM Computing Surveys (CSUR), 50(1), 1-23.

[44] Takamatsu, A., Takaba, E. and Takizawa, G. (2009). “Environment-dependent morphology in plasmodium of true slime moldPhysarumpolycephalum and a network growth model”. Journal of theoretical biology, 256(1), 29-44.

[45] Tang, A. D., Tang, S. Q., Han, T., Zhou, H. and Xie, L. (2021). “A modified slime mould algorithm for global optimization”. Computational Intelligence and Neuroscience, 2021.

[46] Trojovská, E., Dehghani, M. and Trojovský, P. (2022). “Fennec Fox Optimization: A New Nature-Inspired Optimization Algorithm”. IEEE Access, 10, 84417-84443.

[47] Vallverdú, J., Castro, O., Mayne, R., Talanov, M., Levin, M., Baluška, F. and Adamatzky, A. (2018). “Slime mould: the fundamental mechanisms of biological cognition”. Biosystems, 165, 57-70.

[48] Voudouris, C., Tsang, E. P., &Alsheddy, A. (2010). Guided local search. In Handbook of metaheuristics (pp. 321-361). Springer, Boston, MA.

[49] Yang, X.-S. (2010). Engineering optimization: an introduction with metaheuristic applications. John Wiley & Sons.

[50] Yang, X.-S. (2010). Test problems in optimization. arXiv preprint arXiv:1008.0549.

[51] Yu, X. and Gen, M. (2010). Introduction to evolutionary algorithms. Springer Science & Business Media.

[52] Zhang, J. and Sanderson, A. C. (2009). “JADE: Adaptive differential evolution with optional external archive”. IEEE Transactions on Evolutionary Computation, 13(5), 945-958.

[53] Zhang, Y., & Chen, H. (2023). Bio-Inspired Optimization in Engineering and Sciences. CMES-COMPUTER MODELING IN ENGINEERING & SCIENCES.

[54] Zhang, Y., Wang, S., &Ji, G. (2015). A comprehensive survey on particle swarm optimization algorithm and its applications. Mathematical problems in engineering, 2015.

[55] Zheng, Y. L., Ma, L. H., Zhang, L. Y. and Qian, J. X. (2003, November). “On the convergence analysis and parameter selection in particle swarm optimization”. In Proceedings of the 2003 international conference on machine learning and cybernetics (IEEE Cat. No. 03EX693) (Vol. 3, pp. 1802-1807). IEEE.

[56] Zitzler, E. and Thiele, L. (1999). "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach". IEEE transactions on Evolutionary Computation, 3(4), 257-271.