Swarm Intelligence Algorithms and Applications: An Experimental Survey

Anasse Bari, Robin Zhao, Jahnavi Swetha Pothineni, and Deepti Saravanan

Courant Institute of Mathematical Sciences
Computer Science Department
New York University, New York NY 10012, USA
abari@nyu.edu, bz1037@nyu.edu, jp5867@nyu.edu, ds6812@nyu.edu

Abstract. Swarm Intelligence draws inspiration from the collective intelligent behavior of animals such as birds, fish, and bees. It refers to the collective behavior of decentralized, self-organized systems composed of many simple agents that interact with each other and their environment. As the Swarm Intelligence (SI) field expands, more SI meta-heuristics are incorporated into developing new Artificial Intelligence (AI) tools. This study presents a comprehensive survey of state-of-the-art Swarm Intelligence algorithms and their applications. Seven SI algorithms are described – Artificial Bee Colony Optimization, Bat-inspired algorithm, Flock by Leader algorithm, Grey Wolf Optimizer, Flower Pollination Algorithm, Whale Optimization, and Moth Flame Optimization, with their applications in three diverse fields: Scheduling Problem, Data Science and Robotics, and Route and Layout Optimization. The performance of the algorithms is compared experimentally, and discussions about future directions in Swarm Intelligence research are outlined.

Keywords: Swarm Intelligence, Artificial Intelligence, Artificial bee colony optimization, Whale Optimization, Moth Flame Optimization, Flower Pollination Algorithm, Bat Algorithm, Grey Wolf Optimization, Bird Flocking, Flock by Leader, Swarm Intelligence Applications

1. Introduction

When it comes to inspiration for solutions to computer science problems, nature is the perfect place to look. Natural phenomena, from bird flocks searching for food to interactions in an ecosystem, demonstrate that nature can always find optimal, yet simple, solutions to extremely complex problems on a global scale. This analogy is relevant to the computer science ideology, which seeks solutions that offer fast execution and simple implementation. As a result of this relationship, the field of Swarm Intelligence emerged, and its bio-inspired algorithms are a heuristics approach that models nature's behaviors for solving problems like optimization.

For the past decade, much research has concentrated on Swarm Intelligence. "Nature-inspired Algorithms for Optimization" by Yang [1] provides a thorough overview of the state-of-the-art optimization techniques that have been inspired by

natural processes and phenomena. A Survey of Bio-inspired Optimization Algorithms by Binitha and Sathya [2] presents a comprehensive survey of various bio-inspired optimization algorithms, including their concepts, underlying principles, and applications in different domains. "A Review of Applications of Bio-Inspired Algorithms in Engineering" by Kar (2016) presents a review of bio-inspired computing algorithms and their applications in various domains.

The bio-inspired algorithms covered in this study are organized by the chronological order in which they were proposed. The algorithms covered are Artificial Bee Colony Optimization, Bat-inspired algorithm, Flock by Leader, Grey Wolf Optimizer, Flower Pollination Algorithm, Whale Optimization, and Moth Flame Optimization.

The organization of the paper is as follows: Section 2 explains, in detail, the above-mentioned Swarm Intelligence algorithms. Section 3 provides an experimental comparison. Section 4 discusses the algorithms' applications in diverse fields. Lastly, the discussion and conclusion explore the evolution and impact these algorithms have on solving many complex optimization problems in the future.

2. Swarm Intelligence Algorithms

2.1 Artificial Bee Colony Optimization (ABC)

Proposed by Karaboga et al. [4], artificial bee colony optimization is a swarm-based metaheuristic algorithm, inspired by the foraging behavior of honey bees. The objective of the problem is to find such rich food sources. Following the initialization phase, the algorithm iterates over the Employed Bees Phase, Onlooker Bees Phase, and Scout Bees Phase until the maximum number of iterations or maximum CPU time is achieved.

Initialization Phase: The algorithm generates a randomly distributed initial population of an 'x' number of solutions. Each solution x_i is a D-dimensional vector where D is the number of optimization parameters.

Employed Bees Phase: Now the iteration begins. Initially, the bees randomly select food sources and share their nectar information with the hive at the dance area. The employed bees go to the food areas from their personal experience and then choose new food sources via visual information in their neighborhood using equation (1).

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}) \quad (1)$$

where k ∈ {1, 2, ..., n}, is chosen randomly with n = number of employed bees, and j ∈ {1, 2, ..., D} randomly chosen indices, $\phi_{ij} \in$ [-1,1] and is a random number. As the search approaches an optimal solution, the step length adaptively reduces. The employed bees report their new nectar information to the onlookers at the dance area.

Onlooker Bees Phase: The probability of the food source being chosen by the onlookers is directly proportional to the nectar amount of the food source. The process repeats where the onlookers find new food sources based on personal and visual information in their neighborhood. In every iteration, the number of employed and onlooker bees are equal, which is equal to the number of solutions in the population.

Scout Bees Phase: When the nectar of a food source is abandoned using greedy selection, a scout bee chooses a new food source at random to replace the poor solution. In every iteration, at most one scout bee goes outside searching for a new food source.

The performance of ABC was compared against Particle Swarm Optimisation (PSO) and Differential Evolution (DE) in the original paper for 13 test functions over 30 independent runs. ABC achieved competitive results, finding a global minimum for seven test functions and close to global optima for five test functions. It was also found to be relatively insensitive to parameter change and was able to handle a wide range of constrained optimization problems.

2.2 Bat-Inspired Algorithm

Bat Algorithm is another metaheuristic algorithm introduced by Xin-She Yang based on the echolocation of microbats [5]. Naturally, a microbat emits loud sound pulses and listens to their echoes from surrounding objects. These pulses are sent at an increased rate when closer to the prey, but the sound gets quieter when they get very close.

In the Bat Algorithm, all bats b_i are initialized randomly at random positions x_i, velocity v_i and frequency f_i with a fixed frequency range $[f_{min}, f_{max}]$ and varying emission rate $r_i \in (0,1]$ and loudness A_i. Evaluate the fitness F_i of each bat b_i as $x*$ with fitness $F*$.

After every iteration t, the new positions, and velocities of the bats are given by equations (2), (3) and (4).

$$f_i = f_{min} + (f_{max} - f_{min})\beta \tag{2}$$

$$v_i^{t+1} = v_i^t + (x_i^t - x*) f_i \tag{3}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{4}$$

where $\beta \in [0, 1]$ is a random vector drawn from uniform distribution and $x*$ is the current best optimal position. The fitness of the new solutions is evaluated and the best optimal solution $x*$ is updated based on the new fitness values.

Once the best solution is selected, the x_{old} of all the bats is updated using equation (5).

$$x_{new} = x_{old} + \epsilon A^{(t)} \quad (5)$$

where ϵ is [-1, 1] and $A^{(t)}$ is the average loudness of all the bats at this iteration. In addition to updating the positions and velocities, BA uses frequency tuning and parameter control to strike a balance between exploration and exploitation.

The loudness A_i and pulse rate r_i of each bat is regulated after every iteration using equations (6) and (7).

$$A_i^{t+1} = \alpha A_i^{t} \quad (6)$$

$$r_i^{t+1} = r_i^{0} [1 - exp(-\gamma t)], \quad (7)$$

where $0 < \alpha < 1$ and $\gamma > 0$ are constants. This is continued until the maximum number of iterations or desired level of fitness is reached. The optimal solution is then returned.

Bat Algorithm was tested on benchmark functions such as Rosenbrock's Bat Algorithm in the original paper and produced competitive results with that of PSO and Genetic Algorithm in terms of convergence speed and quality of solutions. The Bat Algorithm was also found to be relatively insensitive to parameter change.

2.3 Flock by Leader: Bird Flocking Algorithm

In 2012, Bari et al. further elevated the Flock Clustering model [6] by introducing flock leaders into the algorithm [7]. Inspired by pigeon flocks, Flock by Leader (FBL) algorithm minimized the moves of agents for data clustering, and is parameter free, while the original model required predefined maximum neighboring distance and number of iterations. The components of the Swarm Clustering Framework are Swarm Metric Space, Swarm Virtual Space, Agents Position Graph, and Feature Similarity Graph.

A Swarm Metric Space M_S is defined as (X, ρ), with $X \in R^d$, $X\{x_1, x_2, ..., x_d\}$, and distance $\rho: X \times X \to R^d$. The distance ρ satisfies Reflexivity, Symmetry, and Triangle inequality. An instance of distance is given by equation (8).

$$d(X_i^t, X_b^t) = \sqrt{\sum_{d=1}^{D} (x_{d,i} - x_{d,j})^2} \quad (8)$$

where D is the total number of dimensions of a swarm agent and $x_{d,i}$ is the dth element in agent i's position vector.

The Swarm Virtual Space V_s is a set $X \in R^d$, $X\{x_1, x_2, ..., x_d\}$ defined in the Euclidean 2-dimensional space, where n swarm agents are deployed at random. It is used to visualize clusters in 2D space.

The Agent Position Graph G_p is defined as $G_p = \{V, E_p\}$, where V is the set of vertices from 1 to n, and E is the set of edges between two vertices. G_p maintains the position of each agent throughout the algorithm, and the distance between two agents is stored in A_p , the adjacent matrix of G_p. Therefore A_p has a size of $|v| \times |v|$ such that

$$a_{i,j} = ||p_i - p_j||_2 \tag{9}$$

The Feature Similarity Graph $G_s\{V, E_f\}$ stores the similarity value between agents calculated using distance formula specified in the Swarm Metric Space. Therefore, the Swarm Clustering Framework is defined as (M_s, V_s, G_p, G_s).

Using the same set of flocking rules, Bird Flocking by Leader Algorithm implements dynamic maximum distance between agents. First, the leaders of boids and their maximum distances are found by neighborhood and reverse neighborhood analysis. Then for each leader, its followers within its maximum distance start moving according to separation, cohesion, and alignment. Each follower is marked as visited, and the algorithm ends when there are no more unvisited agents.

To find the leaders and their maximum distances in the swarm, neighborhood and reverse neighborhood analysis is done. Let $X \in R^d$, $X\{x_1, x_2, ..., x_d\}$ to be the swarm population, $\rho(x_i, x_j)$ to be the distance between agent i and agent j The k-neighborhood of x_i in iteration t ($kNB_t(x_i)$) is defined as all agents within a circle with x_i being the center and radius d_{max}^{Li} such that

$$d_{max}^{Li} = max_{o \in Knn(x_i)} \rho(x_i, o) \tag{10}$$

where $Knn(x_i)$ is the set of k nearest neighbors of x_i.

The reverse neighborhood of x_i, denoted as ($Dr - kNB_t(x_i)$), is the set of data points whose kNB contains x_i at iteration t. In order to determine the density of swarm agents, Agent Role Factor $ARF_t(x_i)$ is introduced where

$$ARF_t(x_i) = \frac{|DR - kNB_t(x_i)|}{|DR - kNB_t(x_i)| + |DkNB_t(x_i)|} \tag{11}$$

The larger the $|DR - kNB_t(x_i)|$, the larger the $ARF_t(x_i)$ and the denser x_i situated in the swarm. In particular, if $ARF_t(x_i) \geq 0.5$, x_i is a flock leader. If $ARF_t(x_i)$ is close to 0, then x_i is an outlier.

Experiments were conducted using two datasets, one with 100 news articles from cyberspace and one being the benchmark iris dataset. The Flock by Leader Algorithm achieves 98.66% reduction in the number of iterations, a 7.5%, 16.5% increase in accuracy in terms of F-measure than other Flocking algorithms and K-means algorithms respectively.

2.4 Grey Wolf Optimization

Grey wolf optimizer (GWO), inspired by the leadership and hunting mechanism of grey wolves, is a new metaheuristic algorithm proposed by Mirjalili et al [8]. There are three levels of ranking in a wolf pack – alpha(α), beta(β), and omega(δ). The alpha is the leader, the beta aids the alpha in various pack activities, and the omega is like the scapegoat. The rest of the pack that does not fall under any of the above ranks is the delta.

The algorithm performs initialization of the grey wolf population $X_i\ (i = 1,\ldots, n)$ where n is the size of the population. The goal of the algorithm is to find the best solution (alpha). The wolves encircle the prey while hunting, which is mathematically expressed at iteration t as equations (12) and (13).

$$D = |C\,X^t_p - X^t| \tag{12}$$

$$X^{t+1} = X^t_p - AD \tag{13}$$

where X^t_p is the position of the prey at iteration t, X^t is the position vector of the wolf at iteration t and A, C values are given by equations (14) and (15)

$$A = 2ar_1 - a \tag{14}$$

$$C = 2r_2 \tag{15}$$

with r_1, r_2 as random numbers in [0,1] and the components of a linearly decreased from 2 to 0 over the cycles. The values of a, A and C are initialized before the iteration begins. The fitness of each search agent is then calculated.

The iteration now begins. For every search agent, the position of the current search agent is updated using equations (16), (17), (18) and (19).

$$X^{t+1} = (X_1 + X_2 + X_3)/3 \tag{16}$$

$$X_1 = X_\alpha - A_1(C_1X_\alpha - X^t) \tag{17}$$

$$X_2 = X_\beta - A_2(C_2X_\beta - X^t) \tag{18}$$

$$X_3 = X_\delta - A_3(C_3X_\delta - X^t) \tag{19}$$

Then, update the values of the vectors a, A and C. Repeat this till the maximum number of iterations is attained. The final value X_α is the best solution obtained.

In the original paper, the algorithm was tested on 29 benchmark minimization functions and gave competitive results compared to PSO, DE, Gravitational Search

Algorithm (GSA), and Fast Evolutionary Programming (FEP). It was particularly effective in solving problems with multiple local optima and high-dimensional search spaces.

2.5 Flower Pollination Algorithm

Flower Pollination Algorithm is a nature-inspired metaheuristic algorithm proposed by Yang in 2012 that mimics the behavior of pollination of flowering plants [9]. Pollination in plants occurs in two forms – abiotic and biotic. In abiotics, the pollen is transferred by a pollinator like insects and animals. In biotic, they do not need any pollinators. Instead, the wind and diffusion act as a medium to transfer pollen. Pollination can be achieved either by self-pollination or cross-pollination.

The Flower Pollination Algorithm is governed by the following rules:

Rule 1: Biotic and cross-pollination by pollinators obeying Levy flight is considered a global pollination process.

Rule 2: Abiotic and self-pollination are considered to be local pollination.
Rule 3: Flower constancy can also be treated as reproduction probability which is proportional to the similarity of the flowers.

Rule 4: Local or global pollination can be managed by selecting switch probability which lies between 0 and 1.

To optimize the objective function, a population of n flowers is initialized with random solutions and a switch probability p. Then, the best solution from the initial population is selected as g^*. For every iteration, every x_i of the flowers is updated as follows:

1. If the randomly selected probability value is less than p, global pollination is chosen. Since the insects can fly over long distances, global pollination ensures the survival of the fittest. At every iteration $t + 1$, the updation the x_i with global pollination is given by equation (20).

$$x_i^{t+1} = x_i^t + L(x_i^t - g^*) \tag{20}$$

   where L is a step vector that follows Levy distribution.

2. If the randomly selected probability value is greater than p, then we opt for local pollination. At any iteration $t + 1$, the updation of x_i with local pollination is given by equation (21).

$$x_i^{t+1} = x_i^t + \epsilon(x_j^t - x_k^t) \tag{21}$$

   where ϵ is drawn from the uniform distribution [0,1] and j, k are random numbers from all the solutions.

The population is updated if they are better than the existing ones. The current best solution g* is updated accordingly in every iteration. This entire process is repeated till the maximum number of iterations is reached.

The performance of the Flower Pollination Algorithm (FPA) was compared with PSO and Genetic Algorithm in the original paper in terms of the number of iterations. 100 independent runs were carried out for a population size of 25. FPA outperformed both algorithms with an exponential convergence rate. The algorithm was also successfully applied to the pressure vessel design problems.

2.6 Whale Optimisation Algorithm

The Whale Optimization Algorithm (WOA) is a metaheuristic optimization algorithm that was first proposed by Seyedali Mirjalili in 2016 [10]. It is inspired by the hunting techniques and social behavior of humpback whales. The algorithm begins by randomly initializing a population of candidate solutions, "whales". These are iteratively updated using Exploration, Exploitation, and Encircling.

Exploration (*Bubble-net attacking method)* is used to search for new candidate solutions by randomly moving some whales in the search space. The position of each whale is updated using the spiral equation (22).

$$\vec{X}_{t+1} = \vec{D}.e^{bl}.cos(2\Pi l) + \vec{X}^{*}_{t} \tag{22}$$

where D is the distance of the i^{th} whale to the prey (best solution obtained so far), b is constant of definition of the logarithmic spiral shape and l is a random number in [-1, 1], imitating the helix-shaped movement of humpback whales and shrinking encircling mechanism.

Exploitation (*Search for prey)* focuses the search around the best candidate solution found so far to improve its quality where the position is updated using equation (23).

$$\vec{X}_{new} = \vec{X}_{rand} - \vec{A}.\vec{D} \tag{23}$$

where X_{rand} is a randomly chosen whale from the population, A is a coefficient that decreases linearly from 2 to 0 and D is the distance between the current position of the whale and X_{rand}.

Encircling forces other whales to converge towards the best candidate solution found so far. The position is updated using equations (24) and (25).

$$\vec{D} = |\vec{C}.\vec{X}^{*}_{t} - \vec{X}_{t}| \tag{24}$$

$$\vec{X}_{t+1} = \vec{X}^{*}_{t} - \vec{A}.\vec{D} \tag{25}$$

where t = current iteration, X_t* is the position vector of the best candidate solution, A and C represent coefficient vectors and D is the distance between the current position of the whale and X_t*. A and C are computed as in equations (26) and (27).

$$\vec{A} = 2.\vec{a}.\vec{r} - \vec{a} \quad (26)$$

$$\vec{C} = 2.\vec{r} \quad (27)$$

where a is a vector that decreases linearly from 2 to 0 and r is a random vector in [0,1].

The performance of Whale Optimization Algorithm (WOA) was tested on benchmark functions and compared with PSO, GSA, DE, and FEP in the original paper. The WOA showed high exploration ability, local optima avoidance, and convergence speed.

2.7 Moth Flame Optimisation Algorithm

Moth Flame optimization is a population-based meta-heuristic algorithm proposed by Mirajalili [11]. They navigate in the night using moonlight to travel in a straight line. But with the artificial lights, the moths fly in spiral paths. In this algorithm, both moths and flames act as solutions. Moths are actual search agents that go around the flame to explore the search area and the flames are the best positions of moths obtained so far.

The MFO algorithm starts by randomly initializing the position of moths M moving in d-dimensional space. The fitness function of each moth is evaluated using equation (28).

$$M(i,j) = (ub(i) - lb(i)) * rand() + lb(i) \quad (28)$$

where i represents the i^{th} moth and j represents the current dimension of the space, ub and lb are the upper and lower bounds of the i^{th} variable. The MFO algorithm also randomly initializes the flames F_j.

During every iteration, the positions of flames are updated and sorted based on the fitness values of moths. The moths then update their positions based on their corresponding flames using equations (29) and (30).

$$M_i = S(M_i, F_j) \quad (29)$$

$$S(M_i, F_j) = D_i e^{bt} cos(2\pi t) + F_j \quad (30)$$

where M_i being the i^{th} moth, F_j being the j^{th} flame, S being the spiral function, D_{i^i} being the distance of the i^{th} moth from the j^{th} flame, b being a constant for defining the shape of the logarithmic spiral, and t being a random number in [−1, 1].

This allows the moths to explore in a higher space, which potentially leads to better solutions. But updating the moths' positions around n flames may decrease the chance

of exploitation around the best flame. Hence, the number of flames decreases after every iteration as in equation (31).

$$Flame\ nums\ =\ round(N\ -\ I\ *\frac{N-1}{T}) \qquad (31)$$

where N is the maximum number of flames, I is the current iteration number and T indicates the maximum iteration numbers. The flame with maximum fitness after the maximum number of iterations is reached or the convergence criteria are met is the best optimal solution found.

The Moth Flame Optimization (MFO) Algorithm was tested on 19 benchmarked functions and the performance was compared with other algorithms such as PSO and GA in terms of balancing exploration and exploitation in the original paper. MFO Algorithm produced competitive results with high exploitation ability. The algorithm was also successfully applied to constraint optimization problems.

Table 1 below summarizes the parameters and properties of the seven algorithms discussed.

Table 1. SI Algorithms - Parameters and Properties

SI Algorithm	Parameters	Properties
Artificial Bee Colony Algorithm	Colony Size, Number of employed, Onlooker and scout bees, Limit parameter, Neighborhood size, Maximum number of iterations.	Balances exploration and exploitation, Solution evaluation using an objective function, Parameter Tuning, Convergence overtime
Bat Algorithm	Population size, Loudness, Pulse rate, Minimum and maximum frequency, Alpha (influence of best bat on movement of other bats), Gamma (Rate of loudness reduction overtime), Maximum number of iterations.	Balances exploration and exploitation, Echolocation behavior using Frequency and Loudness, Solution evaluation using an objective function, Parameter Tuning, Convergence overtime
Bird Flocking: Flock by Leader Algorithm	Number of birds, Minimum Distance.	Adaptive, Self-organization, Emergent behavior, Decentralized decision making.
Grey Wolf	Population size, Alpha (best	Mimics social and hunting

Optimizer	wolf), Beta (second-best wolf), Delta (third-best wolf), Initial search range, Convergence criterion, Search and Randomness, Maximum number of iterations.	behavior of wolves, Parameter Tuning, Convergence overtime.
Flower Pollination Algorithm	Population size, Pollination rate, Levy exponent, Initial step size, Scaling factor, Convergence criterion, Maximum number of iterations.	Mimics the pollination process of flowers, Balances exploration and exploitation, Adaptive Search Strategy, Parameter Tuning, Convergence overtime.
Whale Optimization Algorithm	Population size, A (Search agent's encircling behavior), C (balance the exploration and exploitation ability), B (improve exploitation ability), I (improve exploration ability), Maximum number of iterations.	Combines exploration and exploitation, Applied in continuous search space, Global Optimization, Fast convergence.
Moth Flame Algorithm	Population size, Light intensity, Attraction and Mutation coefficients, Convergence criterion, Maximum number of iterations.	Balances exploration and exploitation, Mimics moths behavior of attraction to light, Stochasticity, Global Optimization, Fast convergence.

3. Experimental Analysis

This section provides a comparison of the performance of the seven algorithms in identifying the optimal hyperparameters for a random forest classifier for the MNIST data. The MNIST database is a popular dataset of handwritten digits that is widely used for training image processing systems. It contains 60,000 examples for training and 10,000 examples for testing. The dataset is a subset of a larger set available from the National Institute of Standards and Technology (NIST). The digits in the dataset have been standardized in size and centered in a fixed-size image.

The following range of values are considered for each of the five hyperparameters of the Random Forest algorithm:

1. n_estimators (number of decision trees in the forest) was optimized over the range of 10 to 100.
2. max_depth (maximum depth of each decision tree) was optimized over the range of 5 to 50.
3. min_samples_split (minimum number of samples required to split an internal node) was optimized over the range of 2 to 11.
4. min_samples_leaf (minimum number of samples required to be at a leaf node) was optimized over the range 1 to 11.
5. max_features (the maximum number of features to consider when looking for the best split) was optimized over the range of 1 to 64.

We evaluated the seven algorithms for 100 iterations to find the optimal parameters for the Random Forest Classification on the above-mentioned dataset. Table 2 provides the optimal hyperparameters identified by the seven algorithms and the accuracy attained by the random forest classifier with the corresponding hyperparameters.

Table 2. Random Forest Classification Parameters Estimation using SI Algorithms

	ABC	**Bat**	**BFL**	**GWO**	**FPA**	**WOA**	**MFO**
n_estimators	75	51	59	87	40	20	27
max_depth	48	17	25	49	29	9	18
min_samples_split	4	9	11	3	2	3	9
min_samples_leaf	1	8	1	1	2	1	3
max_features	23	29	21	13	14	5	28
ACCURACY	98.3	95.8	96.9	97.7	**98.6**	98.3	98.0

As can be seen from table 2, Flower Pollination Algorithm (FPA) performed the best with an accuracy of 98.6%. ABC and WOA performed the second best with an accuracy difference of 0.3.

4. Applications of Swarm Intelligence Algorithms

In this section, three major fields of applications are discussed in Table 3 where each of the Swarm Intelligence algorithms was employed for optimization purposes.

Table 3. SI Algorithms and their applications

Swarm	**Scheduling and**	**Data Science,**	**Route, Network and**

Intelligence Algorithm	Assignment Problem	ML, and Robotics	Layout Optimization
Artificial Bee Colony Algorithm (ABC)	**Task Scheduling [12]** Hybrid of ABC and PSO was used to maximize resource utilization and maintain load balance in cloud computing.	**Image Contrast Enhancement [13]** ABC was used to replace grey levels of the input image with new enhanced values.	**Vehicle Routing [14]** An improved ABC algorithm to solve the Vehicle Routing Problem (VRP) using combinatorial optimization.
Bat Algorithm (BA)	**Quadratic assignment problem [15]** Discrete BA with 2-exchange neighborhood and a modified uniform crossover mechanism were used for quadratic assignment problems.	**Satellite Image Enhancement [16]** Histogram equalized image was contrast enhanced where control parameters are optimized using BA.	**Energy Consumption in Wireless Sensor Networks [17]** BA was used to select the optimal sensor and route to reduce energy consumption.
Bird Flocking: Flock by Leader (FBL)	**Load Balancing [18]** Bird swarm optimization was applied for load balancing in cloud computing.	**Diabetes Disease Classification [19]** Flock Optimization was used to tune hyperparameters to classify diseases.	**Traveling Salesman Problem [20]** Traveling salesman problem was solved by an improved multi flock optimization.
Grey Wolf Optimizer (GWO)	**Scheduling Workflow of Applications [21]** Distributed GWO was used to map dependent workflow tasks for low total execution cost.	**Robot Path Planning [22]** GWO was used for robot path planning to avoid 3 different circular obstacles.	**Aerial Vehicle Path Planning [23]** GWO was used to find optimal 2-D path for unmanned combat aerial vehicles.
Flower Pollination Algorithm (FPA)	**Task scheduling in cloud computing environment [24]** Exploration Enhanced FPA was proposed in	**Multi-level Image thresholding [25]** Local pollination with Euclidean distance ratio and	**Urban transit routing [26]** Improved population generative method and FPA with average

	the cloud model to allocate tasks to the virtual machines with reduced makespan.	global pollination with random position vector for image segmentation.	travel time and the number of transfers as the optimization objective.
Whale Optimization Algorithm (WOA)	**Job shop scheduling [27]** Hybrid WOA enhanced with Levy flight and differential evolution was proposed to solve the job shop scheduling problem.	**Multi-robot space exploration [28]** Integration of deterministic coordinated multi robot exploration and Frequency Modified WOA.	**Traffic-aware routing in urban VANET [29]** Multi-objective auto-regressive WOA for traffic-aware routing in urban VANET.
Moth Flame Algorithm (MFO)	**Flow shop scheduling [30]** Improvised MFO was developed to solve green reentrant hybrid flow shop scheduling that minimizes the maximum completion time and reduces the comprehensive impact of resources and the environment.	**Feature selection for medical diagnosis [31]** Enhanced MFO integrating Levy flight operators into its structure with transfer functions was developed as a search strategy within the wrapper feature selection.	**Fault resilient routing: underwater wireless sensors [32]** A fault resilient routing based on MFO with a novel fitness function was proposed to transfer packets towards base stations through autonomous underwater vehicles.

5. Discussion

Nature-inspired algorithms can effectively solve complex optimization problems that are difficult, or impossible, to solve with traditional methods. These algorithms do not require a large amount of problem-specific knowledge, can be implemented relatively easily, and can provide the ability to balance exploration and exploitation during the optimization process.

However, they may suffer from slow convergence rates and sensitivity to parameter selection. Due to their heuristic nature, these algorithms often lack theoretical guarantees of their performance or convergence, but they have been shown to be highly effective in solving complex optimization problems. It is important to have a

comprehensive understanding of the algorithms, as well as the problem statement, in order to determine which algorithm is most suitable given the constraints. Hybrid versions of these algorithms are currently being created that aim to overcome some of these weaknesses, given an objective and application area.

6. Conclusion

Swarm Intelligence models the behavior of nature to work with different kinds of data and machine learning problems from diverse fields. The capability to apply these algorithms in various cross-disciplinary fields makes Swarm Intelligence a promising area of research in the field of Artificial Intelligence. These algorithms offer a powerful and flexible approach to problem-solving, and their ability to handle non-linear and non-differentiable objective functions makes them particularly useful in practical applications where other optimization techniques may be limited. This review provided a comprehensive overview of seven Swarm Intelligence algorithms developed over the years which took inspiration from various animals and birds around us. Several potential application areas for these algorithms were also discussed and appreciated. All algorithms discussed (Artificial Bee Colony Optimization, Bat-inspired algorithm, Flock by Leader, Grey Wolf Optimizer, Flower Pollination Algorithm, Whale Optimization, and Moth Flame Optimization) were well-explored and applied for scheduling, machine learning, robotics, and route optimization problems, providing effective results and future directions. Currently, several ongoing projects are focused on creating a hybrid of these algorithms that could be applied in more diverse and complex domains.

7. Acknowledgements

This study was partially supported by the 2023 Courant Institute Suzanne McIntosh Research Fellowship. Special thanks to Marie Bogdanoff for her valuable insights. Marie is a member of the Predictive Analytics and AI Research Lab at New York University, led by Prof. Anasse Bari.

References

1. X. S. Yang, Nature-inspired algorithms for optimization. In: Springer Science & Business Media, 2010.
2. Binitha, S., & Sathya, S. S. (2018). A survey of bio-inspired optimization algorithms. In: Proceedings of the Second International Conference on Intelligent Computing and Communication (pp. 633-640). Springer.
3. Kar, A. K. (2016). Bio-inspired computing – A review of algorithms and scope of applications. In: Expert Systems with Applications, 59, 20-32. https://doi.org/10.1016/j.eswa.2016.04.018.
4. Karaboga, D., Basturk, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. In: J Glob Optim 39, 459–471 (2007). https://doi.org/10.1007/s10898-007-9149-x

5. X. S. Yang. A new metaheuristic bat-inspired algorithm. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), Springer, Berlin, Heidelberg, 2010, pp. 65-74. doi: 10.1007/978-3-642-12538-6_6.
6. Reynolds, C. W. (1987) Flocks, herds, and schools: a distributed behavioral model, in computer graphics, In: 21(4) (SIGGRAPH '87 Conference Proceedings) pages 25-34.
7. Bellaachia, A., Bari, A. (2012). Flock by leader: a novel machine learning biologically inspired clustering algorithm. In: Tan, Y., Shi, Y., Ji, Z. (eds) Advances in Swarm Intelligence. ICSI 2012. Lecture Notes in Computer Science, vol 7332. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-31020-1_15
8. S. Mirjalili, S. M. Mirjalili, and A. Lewis. Grey wolf optimizer. In: Advances in Engineering Software, vol. 69, pp. 46–61, 2014.
9. X. S. Yang. Flower pollination algorithm for global optimization. In: 2012 International Conference on Unconventional Computing and Natural Computation (UCNC), Orléans, France, 2012, pp. 240-249, doi: 10.1007/978-3-642-31525-1_27.
10. S. Mirjalili, A. H. Gandomi, and S. M. Mirjalili. Whale optimization algorithm. In: Advances in Engineering Software, vol. 95, pp. 51-67, 2016, doi: 10.1016/j.advengsoft.2016.01.008.
11. Seyedali Mirjalili. Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. In: Knowledge-Based Systems,Volume 89,2015,Pages 228-249,ISSN 0950-7051.
12. P. Maheswari, B. Edwin, and R. Thanka. A hybrid algorithm for efficient task scheduling in cloud computing environment. In: International Journal of Reasoning-based Intelligent Systems, vol. 11, p. 134, 01 2019.
13. A. Draa and A. Bouaziz. An artificial bee colony algorithm for image contrast enhancement. In: Swarm and Evolutionary Computation, vol. 16, pp. 69–84, 2014.
14. Yao B, Yan Q, Zhang M, Yang Y. Improved artificial bee colony algorithm for vehicle routing problem with time windows. In: PLoS One. 2017 Sep 29;12(9):e0181275. doi: 10.1371/journal.pone.0181275. PMID: 28961252; PMCID: PMC5621664.
15. M. E. Riffi, Y. Saji, and M. Barkatou. Incorporating a modified uniform crossover and 2-exchange neighborhood mechanism in a discrete bat algorithm to solve the quadratic assignment problem. In: Egyptian Informatics Journal, vol. 18, no. 3, pp. 221–232, 2017.
16. A. Asokan, D. E. Popescu, J. Anitha, and D. J. Hemanth. Bat algorithm based non-linear contrast stretching for satellite image enhancement. In: Geosciences, vol. 10, no. 2, p. 78, Feb. 2020, doi: 10.3390/geosciences10020078.
17. A. K. Sangaiah, M. Sadeghilalimi, A. A. R. Hosseinabadi and W. Zhang. Energy consumption in point-coverage wireless sensor networks via bat algorithm. In: IEEE Access, vol. 7, pp. 180258-180269, 2019, doi: 10.1109/ACCESS.2019.2952644.
18. Mishra, Kaushik and Majhi, Santosh Kumar. A binary bird swarm optimization based load balancing algorithm for cloud computing environment. In: Open Computer Science, vol. 11, no. 1, 2021, pp. 146-160. https://doi.org/10.1515/comp-2020-0215
19. Balasubramaniyan, D., Husin, N. A., Mustapha, N., Sharef, N. M., & Mohd Aris, T. N. (2023). Flock optimization induced deep learning for improved diabetes disease classification. In: *Expert Systems*, e13305. https://doi.org/10.1111/exsy.13305
20. Vahit Tongur and Erkan Ülker. 2019. PSO-based improved multi-flocks migrating birds optimization (IMFMBO) algorithm for solution of discrete problems. In: Soft Comput. 23, 14 (July 2019), 5469–5484. https://doi.org/10.1007/s00500-018-3199-5
21. B. H. Abed-alguni and N. A. Alawad. Distributed grey wolf optimizer for scheduling of workflow applications in cloud environments. In: Applied Soft Computing, vol. 102, p. 107113, 2021.
22. L. Doğan and U. Yüzgeç. Robot path planning using gray wolf optimizer', 05 2018.

23. S. Zhang, Y. Zhou, Z. Li, and W. Pan. Grey wolf optimizer for unmanned combat aerial vehicle path planning. In: Advances in Engineering Software, vol. 99, pp. 121–136, 2016.
24. Bezdan, T., Zivkovic, M., Antonijevic, M., Zivkovic, T., Bacanin, N. (2021). Enhanced flower pollination algorithm for task scheduling in cloud computing environment. In: Joshi, A., Khosravy, M., Gupta, N. (eds) Machine Learning for Predictive Analysis. Lecture Notes in Networks and Systems, vol 141. Springer, Singapore. https://doi.org/10.1007/978-981-15-7106-0_16
25. L. Shen, C. Fan and X. Huang. Multi-level image thresholding using modified flower pollination algorithm. In: IEEE Access, vol. 6, pp. 30508-30519, 2018, doi: 10.1109/ACCESS.2018.2837062.
26. Fan, L., Chen, H. & Gao, Y. An improved flower pollination algorithm to the urban transit routing problem. In: Soft Comput 24, 5043–5052 (2020). https://doi.org/10.1007/s00500-019-04253-3
27. M. Liu, X. Yao, and Y. Li. Hybrid whale optimization algorithm enhanced with Lévy flight and differential evolution for job shop scheduling problems. In: Applied Soft Computing, vol. 87, p. 105954, 2020.
28. F. Gul, I. Mir, W. Rahiman and T. U. Islam. Novel implementation of multi-robot space exploration utilizing coordinated multi-robot exploration and frequency modified whale optimization algorithm. In: IEEE Access, vol. 9, pp. 22774-22787, 2021, doi: 10.1109/ACCESS.2021.3055852.
29. D. Rewadkar and D. Doye. Multi-objective auto-regressive whale optimisation for traffic-aware routing in urban VANET. In: IET Inf. Secur., vol. 12, no. 4, pp. 293-304, 2018. https://doi.org/10.1049/iet-ifs.2018.0002
30. F. Xu et al. Research on green reentrant hybrid flow shop scheduling problem based on improved moth-flame optimization algorithm. In: Processes, vol. 10, no. 12, 2022.
31. Abu Khurmaa, R., Aljarah, I. & Sharieh, A. An intelligent feature selection approach based on moth flame optimization for medical diagnosis. In: Neural Comput & Applic 33, 7165–7204 (2021). https://doi.org/10.1007/s00521-020-05483-5
32. Kumari, S., Mishra, P.K. & Anand, V. Fault resilient routing based on moth flame optimization scheme for underwater wireless sensor networks. In: Wireless Netw 26, 1417–1431 (2020). https://doi.org/10.1007/s11276-019-02209-x