Hindawi
Complexity
Volume 2023, Article ID 4577581, 22 pages
https://doi.org/10.1155/2023/4577581

Review Article

A Systematic Literature Review on Robust Swarm Intelligence Algorithms in Search-Based Software Engineering

Alam Zeb,[1] **Fakhrud Din**,[1] **Muhammad Fayaz**,[2] **Gulzar Mehmood**,[3] **and Kamal Z. Zamli**[4]

[1]*Faculty of Information Technology, Department of Computer Science & IT, University of Malakand, Lower Dir 18800, KPK, Pakistan*
[2]*Department of Computer Science, University of Central Asia, Naryn, Kyrgyzstan*
[3]*Department of Computer Science, IQRA National University, Swat Campus 19220, Peshawar, Pakistan*
[4]*Faculty of Science and Technology, Universitas Airlangga, C Campus JI. Dr. H. Soekamo, Mulyorejo, Surabaya 60115, Indonesia*

Correspondence should be addressed to Muhammad Fayaz; muhammad.fayaz@ucentralasia.org

Received 1 May 2022; Revised 4 September 2022; Accepted 7 October 2022; Published 23 February 2023

Academic Editor: Haitham Abdulmohsin Afan

Swarm intelligence algorithms are metaheuristics inspired by the collective behavior of species such as birds, fish, bees, and ants. They are used in many optimization problems due to their simplicity, flexibility, and scalability. These algorithms get the desired convergence during the search by balancing the exploration and exploitation processes. These metaheuristics have applications in various domains such as global optimization, bioinformatics, power engineering, networking, machine learning, image processing, and environmental applications. This paper presents a systematic literature review (SLR) on applications of four swarm intelligence algorithms i.e., grey wolf optimization (GWO), whale optimization algorithms (WOA), Harris hawks optimizer (HHO), and moth-flame optimizer (MFO) in the field of software engineering. It presents an in-depth study of these metaheuristics' adoption in the field of software engineering. This SLR is mainly comprised of three phases such as planning, conducting, and reporting. This study covers all related studies published from 2014 up to 2022. The study shows that applications of the selected metaheuristics have been utilized in various fields of software engineering especially software testing, software defect prediction, and software reliability. The study also points out some of the areas where applications of these swarm intelligence algorithms can be utilized. This study may act as a guideline for researchers in improving the current state-of-the-art on generally adopting these metaheuristics in software engineering.

1. Introduction

The goal of optimization is to find a solution that either minimizes or maximizes a criterion known as the objective function, fitness function, or cost function. A solution known as a feasible solution or admissible solution is required to solve the objective function under given constraints [1]. An optimization algorithm works iteratively until a specified number of iterations or when a specific amount of time is reached to enhance a given criterion [2].

Many real-life optimization problems in engineering, science, and business are complicated and, thus, no exact solutions can be provided in a fair amount of time. To tackle such problems, approximate algorithms are used. Approximate algorithms are further divided into specific heuristics and metaheuristics. Specific heuristics being problem-dependent solve specific problems, whereas metaheuristics are general-purpose strategies used to solve a huge number of optimization problems [3].

Generally, metaheuristics solve instances of hard problems through exploration of the search space where the solution is supposed to be found. There are mainly three objectives of metaheuristics: (1) providing a fast solution to a given problem; (2) tackling large problems; and (3) obtaining robust algorithms. Furthermore, metaheuristics are flexible and simple to design as well as implement [3].

In Search-Based Software Engineering (SBSE), metaheuristic search techniques are applied to optimization problems in software engineering. The main objective of SBSE is to move problems related to software engineering to machine-based search instead of human-based search. This is accomplished by using various techniques of metaheuristics as well as operations research [4]. Near-optimal solutions are acceptable in various engineering disciplines, and software engineering is no exception. It is due to this fact that metaheuristic techniques are applicable to solving software engineering problems. Thus, difficult problems can be solved using such search-based techniques with acceptable solutions where perfect solutions are infeasible or impossible [5].

Various swarm intelligence algorithms such as GWO, WOA, HHO, and MFO algorithms have been applied in numerous domains such as machine learning, networks, engineering optimization, environmental modeling, image processing, power dispatch problems, and medical and bioinformatics [6–9]. Some of the recent nature-inspired metaheuristic algorithms include monarch butterfly optimization (MBO) proposed by Wang et al. [10], slime mould algorithm (SMA) proposed by Li et al. [11], moth search (MS) algorithm proposed by Wang [12], marine predators algorithm (MPA) presented by Faramarzi et al. [13], hunger games search (HGS) presented by Yang et al. [14], and colony predation algorithm (CPA) developed by Tu et al. [15].

This paper presents a systematic literature review on applications of four swarm intelligence algorithms, namely, GWO, WOA, HHO, and MFO, in the domain of software engineering. Our motivation comes from the fact that, as per our knowledge, there is no such systematic review available. Also, a review study paper helps researchers with aggregated knowledge on a given topic in one place. Table 1 shows various fields in software engineering that utilize these swarm intelligence algorithms. It also points out various areas in software engineering where these algorithms could be utilized. The main contributions of this SLR are as follows:

(i) The SLR presents applications of four swarm-based metaheuristics i.e., GWO, WOA, HHO, and MFO in the field of software engineering.

(ii) The SLR analyzes 46 relevant studies from 2014 to 2022 according to several research questions.

(iii) This study points out new areas in software engineering where the selected metaheuristics could be utilized thus it identifies further research opportunities in the field.

Many swarm intelligence algorithms are available and are gaining prominence as they offer reasonable solutions to complex problems. These algorithms are mainly inspired by biological systems such as animal-based and insect-based. We have chosen four swarm intelligence algorithms, one from each group, i. e., GWO from animal-based, MFO from insect-based, HHO from bird-based, and WOA from fish-based groups. Also, the selected algorithms are relatively new, therefore, working on these will provide the current state-of-the-art. Moreover, these algorithms are popular among swarm intelligence algorithms, which is obvious from the number of their applications in various fields, including software engineering. Based on the above scenario, we have chosen these four algorithms for our SLR.

The Grey Wolf Optimizer is a metaheuristic algorithm that is inspired by grey wolves. It works by mimicking grey wolves in nature, such as their leadership hierarchy and hunting mechanism. Similarly, WOA is a nature-inspired metaheuristic optimization algorithm that mimics the social behavior of humpback whales. The bubble-net hunting strategy is the main inspiration for WOA. The third metaheuristic algorithm included in our study is HHO, which is a nature-inspired and population-based swarm intelligence algorithm. It is inspired by the chasing style and cooperative behavior of Harris' hawks in nature, known as surprise pounce. The fourth and last swarm intelligence algorithm included in our study is also a nature-inspired optimization algorithm called the MFO algorithm. Transverse orientation, the navigation method of moths in nature, is the main inspiration for this algorithm.

Recent applications of the GWO include Al-Momani et al. [16] used GWO to optimize the fixed parameters of the supercritical power plant (SCPP). Results were compared with GA which confirmed the superiority of GWO over GA for parameter identification of the SCPP model. Hao and Sobhani [17] utilized chaotic GWO to identify important functional parameters of solid oxide fuel cells. The adaptive GWO was applied to a 5 kW dynamic tubular stack. The result showed significant performance compared with existing well-known methods for optimization. Bardhan et al. [18] proposed a hybrid approach based on GWO and an artificial neural network (ANN) to find the load-carrying capability of concrete-filled steel tube (CFST) columns. The author also employed an enhanced version of GWO (EGWO) and created two hybrid models, ANN-AGWO and ANN-EGWO, to estimate CFST columns' load-carrying capacity. According to the results, the ANN-AGWO outperformed other methods. Dhar et al. [19] proposed a modified GWO for parameter tuning of the model of a direct deposition process of austenitic steel that utilizes the technique of extreme gradient boost. Results showed GWO's performance in tuning the parameters of the eth model was satisfactory. Yin and Sun [20] introduced distributed multiobjective GWO (DMOGWO) to solve the large-scale multiarea interconnected power systems (LMIPSs) problems. Results confirmed the better performance of the proposed method in solving LMIPS problems.

It is worth noting that these optimization methods do not guarantee the best solution, and since these are mostly used in NP-hard problems, they offer near-optimal solutions. Moreover, these metaheuristics come with their shortcomings, such as trapping in local optima. To avoid being trapped in local optima, various solutions have been proposed by offering variants of these metaheuristics, such as Hu et al. [21] tried to overcome GWO's shortcomings by developing a variant of GWO. The proposed GWOCMA-LOL uses CMAES, levy flight, and orthogonal learning strategies. According to the results, the proposed algorithm

Table 1: Swarm intelligence algorithms' areas of applications in software engineering.

Algorithms	Fields covered by the algorithm	Gaps to be covered by these algorithms
GWO	(i) Software testing (ii) Software bug/defect prediction (iii) System reliability (iv) Software requirements analysis (v) Agile software development (vi) Software effort estimation (vii) Software project scheduling problem (viii) Software usability (ix) Next release problem (x) Project cost estimation (xi) Software release planning (xii) Software risk management	(i) Software vulnerability prediction (ii) Software organization (iii) Software re-modularization (iv) Maintenance/evolution system integration (v) Software module clustering (vi) Finding good designs (vii) Reverse and re-engineering through transformation and re-factoring (viii) User-based fitness evaluation for aesthetic aspects of software engineering (ix) Avionics software (x) Automotive software (xi) Software configuration (xii) Software component allocation (xiii) Structural testing (xiv) Mutation testing (xv) Worst-and-best case execution time testing (xvi) Issue of boundary value analysis (xvii) Partition testing
WOA	(i) Software testing (ii) Software bug/defect prediction (iii) System reliability (iv) Software requirements analysis (v) Software effort estimation (vi) Software project scheduling problem (vii) Next release problem	
HHO	(i) Software testing (ii) Software bug/defect prediction	
MFO	(i) Software testing (ii) Software bug/defect prediction (iii) Software usability	

outperformed the original algorithm by resolving its shortcomings.

This paper is organized as follows: An overview of GWO is presented in the *Overview* section. The adopted research methodology is discussed in the *Methodology* section. Results obtained from the review study are presented in the *Results* section. Section *Discussion* summarizes the SLR and provides insights obtained from the study. Section *Threats to Validity* outlines the potential threats to validity. Finally, the *Conclusion* section concludes the literature review work.

2. Overview

This section presents an overview of the swarm intelligence algorithms included in our study. It explores sources of inspiration as well as mathematical models of these algorithms.

2.1. Grey Wolf Optimizer (GWO). The Grey Wolf Optimizer is an instance of a swarm intelligence technique that is inspired by the social hierarchy as well as the hunting behavior of the grey wolf (*Canis lupus*) of the Canidae family. They stay at the top of the food chain, being apex predators. They usually live in packs with an average size of 5 to 12 wolves. The group dictates a strict social dominant hierarchy [6, 22]. A multi-objective variant of the same algorithm, the Muli-Objective Grey Wolf Optimizer (MOGWO), is proposed by Mirjalili [23] for optimization of problems having more than one objective.

At the top of the hierarchy is the alpha, which leads the group. Beta, the subordinate wolves are next to alpha in the social hierarchy. They help the alphas with making decisions as well as other activities. In case the alpha becomes too old or passes away, the beta becomes the alpha. At the bottom of the hierarchy are the omega wolves, which are dominated by other dominant wolves. The fourth type in the hierarchy is delta, which has dominance over the omega but submits to alpha and beta. Delta wolves include caretakers, hunters, sentinels, scouts, and elders [22].

Like their social hierarchy, grey wolves also exhibit interesting social behavior during group hunting. They follow a set of efficient steps during group hunting, such as chasing, encircling, harassing, and attacking the prey. This behavior enables them to control big prey [6, 22]. Figure 1 shows the flowchart of GWO.

2.2. Mathematical Model of GWO. This section presents the mathematical models of the social behavior of grey wolves such as encircling and hunting the prey [22].

2.2.1. Encircling the Prey. The encircling behavior of grey wolves is depicted mathematically by the following equations:

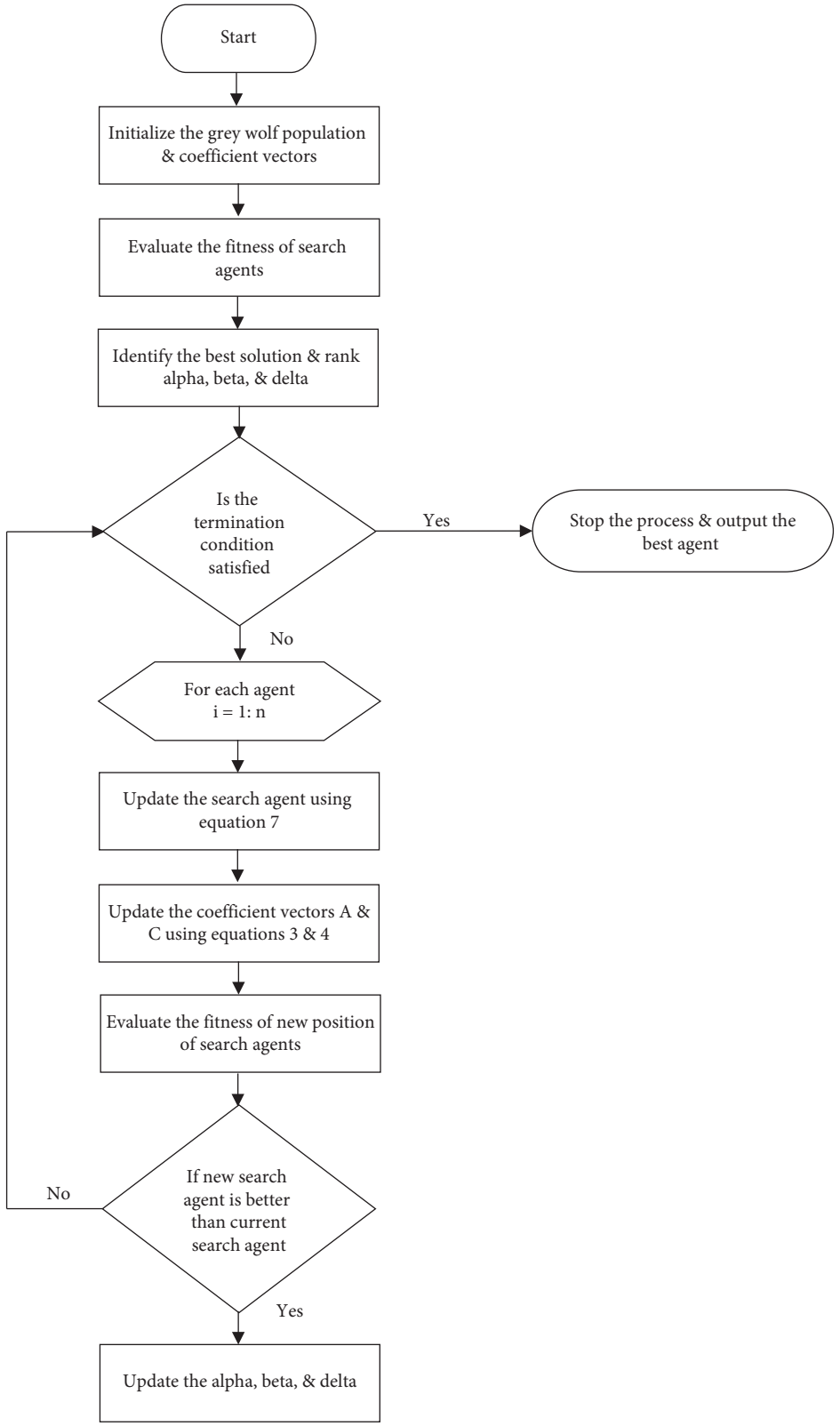

Figure 1: Flowchart of Grey Wolf Optimization (GWO) algorithm.

$$\vec{D} = \left|\vec{C}.\vec{X_p}(t) - \vec{X}(t)\right|, \quad (1)$$

$$\vec{X}(t+1) = \vec{X_p} - \vec{A}.\vec{D}, \quad (2)$$

where the current iteration is shown by t, grey wolf's position vector is indicated by $\vec{X}$ and position vector of prey is indicated by $\vec{X_p}$. $\vec{A}$ and $\vec{C}$ represent the coefficient vectors calculated as follows:

$$\vec{A} = 2\vec{a}.\vec{r_1} - \vec{a}, \quad (3)$$

$$\vec{C} = 2.\vec{r_2}, \quad (4)$$

where r_1 and r_2 are random vectors in the range [0, 1] while components of $\vec{a}$ are decreased from 2 to 0 linearly as iterations proceed.

2.2.2. Hunting. In a given abstract search space, the best candidate solution is supposed to be alpha whereas beta, and delta are supposed to know the prey's potential location i. e., the best solution. The top three solutions obtained are saved. Other search agents such as omegas update their positions according to the position of the best search agents. This concept is depicted mathematically by the following equations:

$$\begin{aligned}&\vec{D_\alpha} = \left|\vec{C_1}.\vec{X_\alpha} - \vec{X}\right|, \vec{D_\beta} = \left|\vec{C_2}.\vec{X_\beta} - \vec{X}\right|, \vec{D_\delta} = \left|\vec{C_3}.\vec{X_\delta} - \vec{X}\right|,\\ &\vec{X_1} = \vec{X_\alpha} - \vec{A_1}.\left(\vec{D_\alpha}\right), \vec{X_2} = \vec{X_\beta} - \vec{A_2}.\left(\vec{D_\beta}\right), \vec{X_3} = \vec{X_\delta} - \vec{A_3}.\left(\vec{D_\delta}\right),\end{aligned} \quad (5)$$

where $\vec{D_\alpha}, \vec{D_\beta}$ and $\vec{D_\delta}$ are vectors showing distance between prey's location $\vec{X}$ and alpha, beta, and delta, respectively.

$\vec{X_\alpha}, \vec{X_\beta}$ and $\vec{X_\delta}$represent the position vectors of alpha, beta and delta respectively. $\vec{X_1}, \vec{X_2}$ and $\vec{X_3}$ represent the next location of alpha, beta, and delta, respectively. Whereas, $\vec{C_1}, \vec{C_2}, \vec{C_3}, \vec{A_1}, \vec{A_2}$ and $\vec{A_3}$ are coefficient vectors defined by equations (3) and (4)

$$\vec{X}(t+1) = \frac{\vec{X_1} + \vec{X_2} + \vec{X_3}}{3}, \quad (6)$$

where $\vec{X}(t+1)$ represent the next location of the prey i.e., the average of the best three solutions so far to be obliged by other search agents such as omega for updating their position.

2.3. Whale Optimization Algorithm (WOA). WOA is a nature-inspired metaheuristic optimization algorithm proposed by Mirjalili and Lewis [24] in 2016 which mimics the social behavior of humpback whales. The bubble-net hunting strategy is the main inspiration for WOA. The Whale optimization algorithm is a nature-inspired metaheuristic optimization algorithm proposed by Mirjalili and Lewis [24] in 2016 which mimics the social behavior of humpback whales. Whales feel emotions and are considered intelligent animals. Humpback whales (Megaptera novaeangliae) are among the biggest baleen whales. An adult humpback whale almost grows to the size of a school bus.

The humpback whales follow an interesting special hunting method known as the bubble-net feeding method. Small fish and schools of krill are the preferred prey of humpback whales. They forage by creating distinctive bubbles along a "9"-shaped or circle. Their two maneuvers related to bubbles are named "upward-spirals" and "double-loops". In "upward-spirals" the humpback whales take dives and create bubbles in a spiral shape around the prey, whereas "double-loops" consists of three stages: coral loop, lobtail, and capture loop [24].

2.4. Mathematical Model of WOA. This section presents the mathematical model of the unique foraging behavior of humpback whales such as searching for prey, encircling prey, and spiral bubble-net feeding maneuver [24].

2.4.1. Encircling the Prey. The following equations mathematically depict the encircling behavior of humpback whales:

$$\begin{aligned}\vec{D} &= \left|\vec{C}.\vec{X^*}(t) - \vec{X}(t)\right|\\ \vec{X}(t+1) &= \vec{X^*}(t) - \vec{A}.\vec{D},\end{aligned} \quad (7)$$

where the current iteration is shown by t, $\vec{X}$ represents the position vector, and X^* represents the the position vector of the optimal solution.

The following equations calculate vectors $\vec{A}$ and $\vec{C}$ as

$$\vec{A} = 2\vec{a}.\vec{r} - \vec{a}, \quad (8)$$

$$\vec{C} = 2.\vec{r}, \quad (9)$$

where $\vec{r}$ is a random vectors in the range [0, 1] while $\vec{a}$ decreases from 2 to 0 linearly as iterations proceed.

2.4.2. Bubble-Net Attacking Method. The bubble-net behavior represents the exploitation phase. It is modeled mathematically by using two approaches as follows:

(1) *Shrinking Encircling Mechanism.* To achieve this behavior, the value of $\vec{a}$ in equation (8) is decreased which further decreases the fluctuation range of $\vec{A}$.

(2) *Spiral Updating Position.* In this approach, the distance between the whale and prey is calculated first. Positions of

whale and prey are represented by (X, Y) and $,(X^*, Y^*)$, respectively. Then the helix-shaped movement of humpback whales is mimicked using the spiral equation as follows:

$$\overrightarrow{X}(t+1) = \overrightarrow{D'}.e^{bl}.\cos(2\pi l) + \overrightarrow{X^*}(t), \tag{10}$$

where $\overrightarrow{D'} = |\overrightarrow{X^*}(t) - \overrightarrow{X}(t)|$ shows the i^{th} whale's distance to the prey i.e., the optimal solution so far, b defines the logarithmic spiral's shape, l indicates a random number in the range $[-1, 1]$, whereas "." indicates element-wise multiplication.

The following equation models the foraging behavior of humpback whales by assuming a 50% probability each for the shrinking encircling mechanism and the spiral model to update the whales' position during the optimization process:

$$\overrightarrow{X}(t+1) = \begin{cases} \overrightarrow{X^*}(t) - \overrightarrow{A}.\overrightarrow{D}, & \text{if } p < 0.5, \\ \overrightarrow{D'}.e^{bl}.\cos(2\pi l) + \overrightarrow{X^*}(t), & \text{if } p \geq 0.5, \end{cases} \tag{11}$$

where p indicates a random number in the range [0, 1].

2.4.3. Search for Prey. The humpback whales also search randomly for prey. This random search represents the exploration phase. Unlike the exploitation phase, the position of a search agent in the exploration phase is updated randomly as opposed to the optimal search agent found so far. This mechanism as well as $|\overrightarrow{A}| > 1$ enables the WOA algorithm for a global search. The following equation presents the mathematical model:

$$\begin{aligned} \overrightarrow{D} &= \left|\overrightarrow{C}.\overrightarrow{X_{\text{rand}}} - \overrightarrow{X}\right|, \\ \overrightarrow{X}(t+1) &= \left|\overrightarrow{X_{\text{rand}}} - \overrightarrow{A}.\overrightarrow{D}\right|, \end{aligned} \tag{12}$$

where $\overrightarrow{X_{\text{rand}}}$ represents a random position vector i.e., a random whale out of the current population.

2.5. Harris Hawks Optimizer (HHO). The next metaheuristic algorithm included in our study is HHO, which is a nature-inspired and population-based swarm intelligence algorithm proposed by Heidari et al. [25] in 2019. It is inspired by the chasing style and cooperative behavior of Harris' hawks (Parabuteo unicinctus) in nature, known as surprise pounce. The Harris' hawks are considered intelligent birds and are found in the southern half of Arizona, USA.

To capture prey, Harris' hawks use "surprise pounce" as the main strategy. This tactic is also called the "seven kills" strategy. Using this strategy, the detected rabbit is attacked from different directions, and the hawks concurrently converge on the escaping rabbit. The "seven kills" approach may take seconds or several minutes to capture prey depending on its escaping capabilities. Harris' hawks demonstrate various chasing styles based on the escaping patterns of prey as well as the circumstances [25].

2.6. Mathematical Model of HHO. This section mathematically models the exploratory as well as exploitative phases of the HHO algorithm inspired by various hunting strategies of Harris' hawks [25].

2.6.1. Exploration Phase. The following equation models the exploration mechanism of HHO. The perch behavior in HHO is based on two strategies (1) perch based on the positions of other members, which is modeled in equation (13) [ref] for the condition of $q < 0.5$, or (2) random perch on tall trees, which is modeled in Equation (13) for the condition of $q \geq 0.5$.

$$X(t+1) = \begin{cases} X_{\text{rand}}(t) - r_1\left|X_{\text{rand}}(t) - 2r_2X(t)\right|, & \text{if } q \geq 0.5, \\ \left(X_{\text{rabbit}}(t) - X_m(t)\right) - r_3\left(LB + r_4(UB - LB)\right), & \text{if } q < 0.5, \end{cases} \tag{13}$$

where $X(t+1)$ represents hawks' position vector in the next iteration t, $X_{\text{rabbit}}(t)$ represents rabbit's position, $X(t)$ represents hawks' current position vector, r_1, r_2, r_3, r_4, and q represent random numbers inside in the range (0,1), LB and UB represent variables' upper and lower bounds, $X_{\text{rand}}(t)$ shows a hawk randomly selected out of the current population, and average position of hawks' current population is represented by X_m which is defined by equation 14:

$$X_m(t) = \frac{1}{N}\sum_{i=1}^{N} X_i(t), \tag{14}$$

where $X_i(t)$ represents each hawks' location in iteration t and the total number of hawks is represented by N.

During the escape, the prey's energy decreases considerably which can be modeled mathematically as

$$E = 2E_0\left(1 - \frac{t}{T}\right), \tag{15}$$

where E shows the prey's escaping energy, T represents the maximum number of allowed iterations, and E_0 represents the initial energy.

2.6.2. Exploitation Phase. Harris' hawks attack by performing a surprise pounce on the prey. However, prey often succeeds in escaping, which gives rise to various chasing

styles. Based on the chasing strategies as well as escaping behaviors, HHO has four strategies to model the attacking stage. Suppose r represents the prey's escape chance, $(r < 0.5)$ means successful escape while $(r \geq 0.5)$ shows unsuccessful escape before surprise pounce. A hard or soft besiege is performed by the hawks to catch the prey. Here, the parameter E is utilized to decide between soft and hard besiege processes. The soft besiege is performed when $|E| \geq 0.5$ and when $|E| \leq 0.5$, the hard besiege takes place.

(1) *Soft Besiege*. The behavior is performed when $r \geq 0.5$ and $|E| \geq 0.5$ as modeled by the following rules:

$$\begin{aligned} X(t+1) &= \Delta X(t) - E|JX_{\text{rabbit}}(t) - X(t)|, \\ \Delta X(t) &= X_{\text{rabbit}}(t) - X(t), \end{aligned} \tag{16}$$

where $\Delta X(t)$*represents* the difference between the rabbit's position vector and the current location in iteration t, r_5 represents a random number in the range (0,1), and $J = 2(1 - r_5)$ randomly presents the rabbit's jumping strength during the escaping procedure.

(2) *Hard Besiege*. When $r \geq 0.5$ and $|E| < 0.5$, the prey has low escaping energy. In this scenario, the following equation updates the current positions:

$$X(t+1) = X_{\text{rabbit}}(t) - E|\Delta X(t)|. \tag{17}$$

(3) *Soft Besiege with Progressive Rapid Dives*. When $|E| \geq 0.5$ and $r < 0.5$, the rabbit's energy level is still high enough for a successful escape. Thus, prior to surprise pounce, a soft besiege is constructed. The following equation updates the hawks' position in the soft besiege phase:

$$X(t+1) = \begin{cases} Y, & \text{if} F(Y) < F(X(t)), \\ Z, & \text{if} F(Z) < F(X(t)), \end{cases} \tag{18}$$

where Y and Z are obtained using equations (19) and (20) respectively as follows:

$$Y = X_{\text{rabbit}}(t) - E\left|JX_{\text{rabbit}}(t) - X(t)\right|, \tag{19}$$

$$Z = Y + S \times LF(D), \tag{20}$$

where D represents the dimension of the problem and S represents a random vector by size $1 \times D$ and LF represents the levy flight function defined by equation (21).

$$LF(x) = 0.01 \times \frac{u \times \sigma}{|v|^{1/\beta}}, \sigma = \left(\frac{\Gamma(1+\beta) \times \sin(\pi\beta/2)}{\Gamma(1+\beta/2) \times \beta \times 2^{\beta-1/2}}\right)^{1/\beta}, \tag{21}$$

where u, v represents random values in the range (0,1) and β represents a default constant with the value 1.5.

(4) *Hard Besiege with Progressive Rapid Dives*. When $|E| < 0.5$ and $r < 0.5$, a hard besiege is constructed before the surprise pounce as the rabbit lacks the energy to escape. The following equation models the hard besiege condition:

$$X(t+1) = \begin{cases} Y, & \text{if} F(Y) < F(X(t)), \\ Z, & \text{if} F(Z) < F(X(t)), \end{cases} \tag{22}$$

where Y and Z are defined by equations (23) and (24), respectively,

$$Y = X_{\text{rabbit}}(t) - E\left|JX_{\text{rabbit}}(t) - X_m(t)\right|, \tag{23}$$

$$Z = Y + S \times LF(D), \tag{24}$$

where $X_m(t)$ is defined by equation (14).

2.7. Moth-Flame Optimizer (MFO). Another swarm intelligence algorithm included in our study is also a nature-inspired optimization proposed by Mirjalili [26] in 2015 called the MFO algorithm. Transverse orientation, the navigation method of moths in nature, is the main inspiration for this algorithm. Moths are butterfly-like insects with over 160,000 different species.

Moths have an interesting navigation method at night. They utilize moonlight when flying at night using a mechanism known as transverse orientation. Using this method, a moth flies in a straight path by maintaining a fixed angle with respect to the moon. This is an effective method that guarantees flying in a straight line [26].

However, moths are observed flying spirally around the lights. This is because the transverse orientation only works when the light is far away, like the moon. Using artificial lights, they want to keep a similar angle with the light to fly in a straight line. However, this behavior only creates a useless and deadly spiral fly path for moths. In the end, they converge toward the light. The Mothflame optimization algorithm is inspired by this behavior [26].

2.8. Mathematical Model of MFO. MFO is consisted of moths and flames. Moths as search agents are represented in a matrix as follows:

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & \cdots & m_{1,d} \\ m_{2,1} & m_{2,2} & \cdots & \cdots & m_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{n,1} & m_{n,2} & \cdots & \cdots & m_{n,d} \end{bmatrix}, \tag{25}$$

where n represents the number of moths and d represents the dimension.

Also, an array is used to store the fitness of the corresponding moth as follows:

$$OM = \begin{bmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_n \end{bmatrix}, \tag{26}$$

where n represents the number of moths.

Furthermore, the best positions discovered by moths are represented as flames which are stored matrix F and their fitness value is stored in an array OF.

The spiral moment of moths around flames is modeled as follows:

$$S\left(M_i, F_j\right) = D_i.e^{bt}.\ \cos(2\pi t) + F_j, \tag{27}$$

where D_i shows the distance of the i^{th} moth for the j^{th} flame, b is a constant which define the logarithmic spiral's shape, and t indicates a random number in the range $[-1, 1]$.

D is defined by the following equation:

$$D_i = \left|F_j - M_i\right|, \tag{28}$$

where M_i represents the i^{th} moth and F_j represents the jth flame.

For exploitation as well as convergence, the flames decrease in number as depicted in the following equation:

$$\text{flameno} = \text{round}\left(N - l * \frac{N-1}{T}\right), \tag{29}$$

where the current number of iteration is represented by l, maximum number of flames are shown by N, and the maximum number of iterations is shown by T.

3. Research Methodology

This Systematic Literature Review (SLR) follows the proposed methodology of Kitchenham and Charters [27]. Various discrete activities comprise our work. These activities can be divided into three phases, such as planning, conducting, and reporting the SLR. The first phase, namely, planning of the study is comprised of the research questions addressed by the SLR, identifying the data sources, forming the search strategy, data extraction process, and synthesis.

3.1. Research Questions. This systematic literature review explores the applications of GWO, WOA, HHO, and MFO in software engineering. Specification of research questions (RQs) is an essential part of any SLR. For the accomplishment of the SLR, the following RQs are included:

RQ1: What is the evaluation of studies published on the GWO, WOA, HHO, and MFO algorithms in software engineering?

RQ2: What are the applications of GWO, WOA, HHO, and MFO in software engineering?

RQ3: Which individuals, organizations, and countries are actively involved in searched-based software engineering using GWO, WOA, HHO, and MFO?

RQ4: What are the potential applications of GWO, WOA, HHO, and MFO in software engineering?

3.2. Data Sources and Search Strategy. Based on [28], our search string incorporates two essential search terms, namely, "Creature Name" and "Domain". The first term mentions algorithms such as GWO, WOA, HHO, and MFO inspired by the social behavior of grey wolves, humpback whales, Harris' hawks, and moths, respectively. The second term describes the fields where the social behavior of grey wolves has been mimicked, such as software engineering. This term is accompanied by appropriate synonyms.

The Boolean AND has been used to combine the major search terms of our search string, whereas the Boolean OR has been used to incorporate the synonyms as well as related terms. The following is the search string used in our SLR.

("grey wolf optimizer," OR, GWO, OR "whale optimization algorithm," OR, WOA, OR "Harris hawks optimization," OR, HHO, OR "moth-flame optimization" OR MFO) AND ("software engineering" OR "software development" OR "software testing" OR "requirements engineering" OR "usability engineering" OR software OR testing).

For the accumulation of related publications, academic data sources such as IEEExplore, ACM Digital Library, SpringerLink, and ScienceDirect have been searched. Most of our publications of interest came from using Google Scholar. Publications and contributions came from journals, books, and conferences.

This SLR includes full-text studies citing GWO, WOA, HHO, and MFO in software engineering and was published from 2014 until 2022. Major search terms must be part of each study. Furthermore, the written language of these publications must be English. The titles and abstracts of these papers are investigated to ensure they fit the SLR. Personal blogs, web pages, and studies that do not cite the mentioned swarm intelligence algorithms are excluded from the SLR. The exclusion and inclusion criteria for the selection of studies are summarized in Figure 2.

3.3. Data Extraction and Synthesis. To address the research questions, this section explores the strategy of extracting and analyzing data from selected studies. A list comprised of various data items is obtained from the selected studies. Table 2 presents the recorded information which will be used for providing answers to the research questions (RQs). Also, the individual data items associated with the relevant research questions are shown in Table 2.

We applied the search string to various digital libraries such as IEEExplore, ACM Digital Library, SpringerLink, and ScienceDirect and obtained 68, 24, 275, and 2270 studies, respectively. We also used Google Scholar and using a search string, 2446 studies were obtained. Our search provided a total of 5083 studies, as shown in Table 3. Primary studies were obtained from the search results by using the PRISMA flow diagram as shown in Figure 3.

The PRISMA flow diagram consists of various steps such as identification, screening, eligibility, and inclusion, as depicted in Figure 3. All studies are identified through electronic databases by using the search string in the first step, i. e., identification. A total of 5083 studies have been identified. In the second step, namely, screening, irrelevant and grey studies are removed and outcomes in 103 studies are based on the inclusion and exclusion criterion 1. Duplicate studies were removed based on exclusion criterion 2, which resulted in 68 studies. More studies are excluded based on exclusion criteria 3 and 4 and we are left with 54 studies. In the eligibility step, 48 studies are found to be eligible. After reading these 48 studies, they are found to be relevant and are considered primary studies for this SLR. However, Section *Quality assessment criteria* further assesses the selected papers based on quality assessment criteria.

3.4. Quality assessment Criteria. After the implementation of inclusion and exclusion criteria, a quality assessment for each study is performed to determine its credibility as well as relevance. Being a supplementary criterion, it helps in excluding studies having low quality based on some given

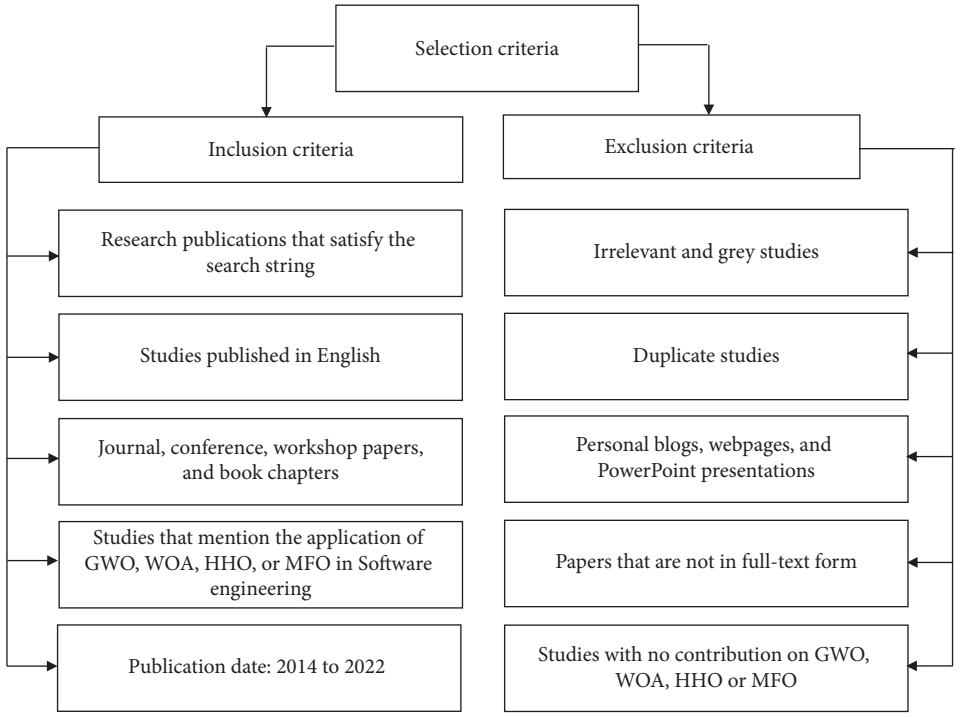

Figure 2: Summary of selection criteria.

Table 2: Data items from selected studies for recording information.

ID	Data item	Description	Relevant RQ
D1	Year of publication	What is publication year of the article?	RQ1
D2	Name of author	Who are the authors of the article?	RQ1
D3	Title of paper	What is the article's title?	RQ1
D4	Subject area	The published article explores which knowledge area?	RQ1
D5	Document type	The article belongs to which document type?	RQ1
D6	Source	What is the article's source such as journal, conference, or book?	RQ1
D7	Swarm based intelligence	What swarm-based intelligence algorithms are involved in search-based software engineering?	RQ2
D8	Case study	Which case study is explored in the paper?	RQ2
D9	Affiliation	The study is published in which affiliated institutes/organizations?	RQ3
D10	Country	Which country has the research institute that has published the study?	RQ3
D11	Application	What are the potential applications of GWO, WOA, HHO, and MFO in software engineering?	RQ4

Table 3: Databases with number of published studies.

ID	Database	Obtained studies	Electronic database's web link
DB1	IEEExplore	68	https://ieeexplore.ieee.org
DB2	ACM digital library	24	https://dl.acm.org
DB3	SpringerLink	275	https://link.springer.com
DB4	ScienceDirect	2270	https://www.sciencedirect.com
DB5	Google scholar	2446	https://scholar.google.com

weights and certain thresholds. The following questions are used to determine the quality assessment criteria:

Q1. Does the research state its aims clearly?

Q2. Does the research compare its proposed estimation method with existing methods?

Q3. Does the study analyze its limitations explicitly?

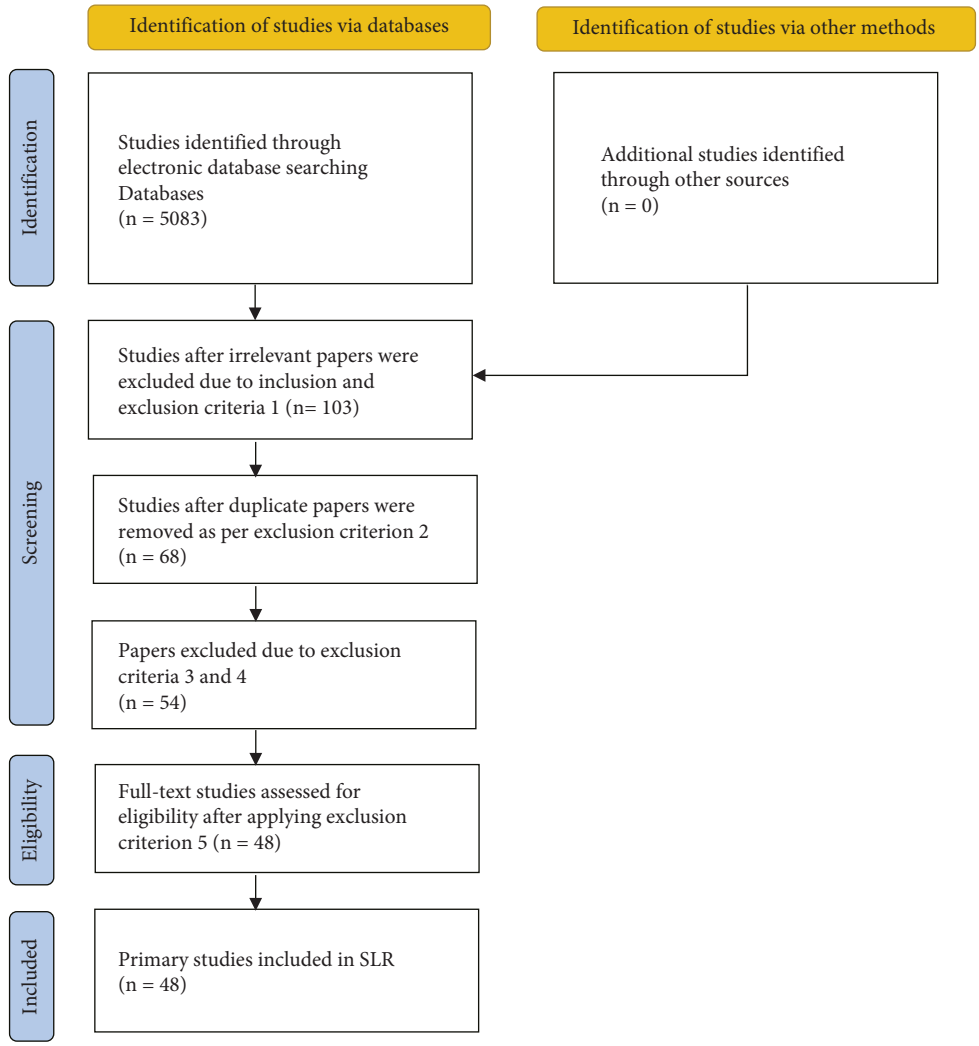

Figure 3: PRISMA flow diagram for selection of primary studies.

Q4. Does the study have citations?

Each question is assigned a weight of 0.25. Since there are a total of four questions, therefore, the combined weight is 1. Thus the total score for a single study can be given as:

0.00 (No)

0.25 (Rarely)

0.50 (Partly)

0.75 (Mostly)

1.00 (Yes)

Based on the abovementioned criterion, a study is included if it scores 0.5 or above, which means out of the four given assessment criteria, it must fulfill at least two.

As a result of the application of this quality assessment, 02 more studies are discarded with a 0.25 (rarely) score each. In other words, all the studies with an assigned score of "partly, mostly, and yes" are included in the SLR. Finally, 46 studies were selected for this SLR. Scores based on quality assessment criteria for the selected studies are given in Table 4. The authors shared the same opinion throughout the selection process.

4. Results

This section provides answers to the RQs by analyzing the selected studies. Detailed outcomes of the literature analysis are presented in this section.

Frequency of published studies based on GWO, WOA, HHO, and MFO in software engineering (RQ1).

The identified studies are analyzed to find out the frequency as well as the advancement of publications based on the selected metaheuristics in software engineering. Figure 4 shows year-wise publication of the selected studies with an average of 05 studies per year. The figure also shows the increased number of publications in recent years. Many research studies indicate that variants of the selected metaheuristics have been used in software engineering, as depicted in Table5. Figure 5 illustrates the type of selected publications. According to the figure, 76% are articles, 22% are inproceeding, and 2% are incollection of the identified studies. Figure 6 illustrates various fields in software engineering in the identified publications. Table 6 shows various journals and conferences associated with the identified research work. It is evident from Table 6 that 4 papers have

Table 4: Selected papers with quality assessment scores.

Study	Score
S1	Mostly
S2	Mostly
S3	Partly
S4	Yes
S5	Partly
S6	Mostly
S7	Mostly
S8	Mostly
S9	Mostly
S10	Mostly
S11	Mostly
S12	Partly
S13	Mostly
S14	Mostly
S15	Yes
S16	Mostly
S17	Mostly
S18	Partly
S19	Mostly
S20	Partly
S21	Partly
S22	Mostly
S23	Partly
S24	Partly
S25	Mostly
S26	Partly
S27	Mostly
S28	Mostly
S29	Yes
S30	Mostly
S31	Mostly
S32	Mostly
S33	Yes
S34	Partly
S35	Mostly
S36	Mostly
S37	Mostly
S38	Mostly
S39	Mostly
S40	Mostly
S41	Mostly
S42	Mostly
S43	Mostly
S44	Mostly
S45	Mostly
S46	Mostly

been published in "IEEE Access", 2 papers each have been published in Springer and "International Journal of Intelligent Engineering and Systems," while one paper each has been published in the rest of the journals and conferences.

4.1. Applications of GWO, WOA, HHO, and MFO in Software Engineering (RQ2). The identified studies discuss various applications of the selected metaheuristics, i.e., GWO, WOA, HHO, and MFO, in software engineering. Figure 7 shows the selected algorithms used by our study. It is clear from the figure that GWO tops the list with 51%, WOA comes second with 30%, while MFO and HHO are at third and fourth positions with 13% and 6% respectively. Table 7 summarizes these applications. The following paragraphs discuss these applications in detail.

Altaie et al. [47] generated a test suite to increase the coverage of paths based on GWO and PSO. According to the results, the PSO algorithm performed better than the GWO algorithm. Jyoti and Sharma [44] proposed an enhanced version of WOA (Ambha_WOA) using 12 test case suites obtained from SIR and GitHub repositories. The new algorithm was compared with WOA using various performance metrics such as fault detection ratio metric, precision and recall, and classification accuracy. NP-statistical tests were conducted for result validation. Wang and Zhao [56] used WOA with chaos initialization for the automatic test case generation method. By using the improved WOA, the method aimed at one path at a time for optimizing the population as well as finding the optimal value. Kaur [51] utilized HHO for extracting the optimal test cases. Sharma and Saha [39] combined MFO and the Firefly Algorithm (FA) for test path generation. The authors used five object-oriented benchmark applications for verification. The full coverage of the path was confirmed by the results. Agrawal et al. [43] used a hybrid WOA for the regression test case selection approach. The proposed approach improved the fault detection ability as per the obtained results. Hassan et al. [55] utilized WOA for t-way test suite generation with constraint support. Results confirmed competitive outcomes for WOA as compared with existing metaheuristics. Harikarthik et al. [54] used the regression test case prioritization mechanism for generating test cases. Then, they used the kernel fuzzy c-means (KFCM) clustering technique for clustering the test cases. They utilized WOA for the weight optimization process. Rastogi [30] used the GWO and FA (GWFF) methods to optimize the Test Case (TC) and generate path coverage in less execution time. The GWFF showed better performance in terms of fitness values when compared with Particle Swarm Optimization (PSO), Bee Colony Algorithm (BCA), and Cuckoo Search (CS). Sunitha [49] proposed a technique that makes use of optimal test prioritization in the Cloud using KFCM to overcome the shortcomings of various algorithms and to find an accurate optimization of existing software applications. The technique has two steps. Firstly, the KFCM technique was applied based on a similarity feature in organizing the test cases in software applications. Secondly, GWO was applied to achieve an optimal solution. Sharma and Saha [53] used MFO for model-based software testing based on object orientation. According to the results, for large software applications, MFO outperformed other metaheuristics, i.e., FA and ACO in creating optimized test cases. Metwally [52] used MFO to create a technique for automatic test data generation which in a single run could generate the whole test suite. According to the results, the new technique outperformed the random generator. Badanahatti and Murthy [48] used cloud-based regression testing. The proposed technique has three stages, namely, test case generation, clustering, and test case prioritization. As per the results, the proposed method preceded the minimum implementation time of the available technique. Gupta and

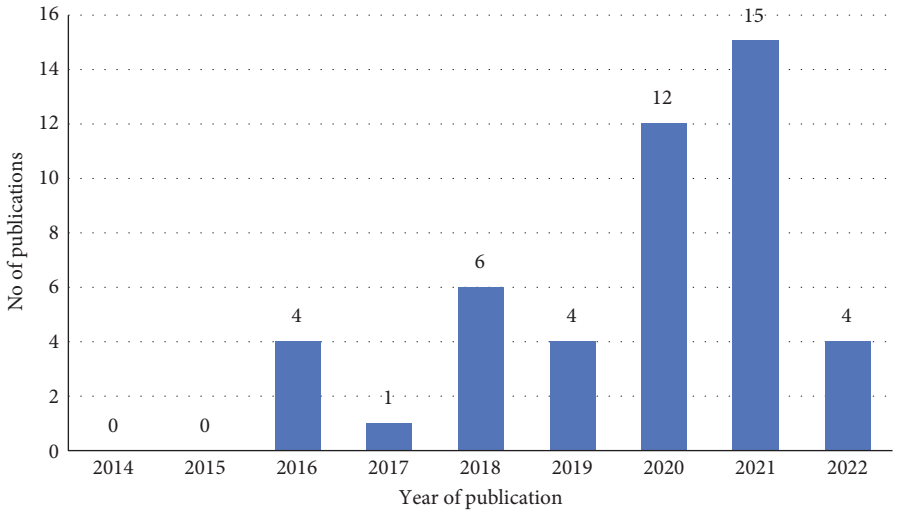

Figure 4: Frequency of published study by year.

Table 5: Summary of the selected metaheuristics' variants used in software engineering.

Variant	Short name	Effectiveness compared with	Efficiency/performance evaluated/validated by	Case study	Publication year	Authors [ref.]
Chaotic grey wolf optimization algorithm	CGWO	GWO, PSO, GA, and WO	Pham-Zhang (PZ) model and testing effort functions (TEFs)	Parameter optimization of software reliability growth model	2021	Dhavakumar and gopalan [29]
Grey Wolf-FireFly method	GWFF	BCA, PSO, and CS	Mean time between failures (MTBF)	Software test case mechanism	2019	Rastogi [30]
Agile risk prioritization–grey wolf optimization	ARP-GWO	Existing agile	Usability goals achievement metric (UGAM) and index of integration (IoI)	Risk prioritization in agile software development	2021	Prakash and viswanathan [31]
Modified grey wolf optimization	MGWO	GWO, MBBAT, MWOA, and MMFO	Software development life cycle (SDLC) models	Software usability	2021	Jain [32]
Multi-objective grey wolf optimizer	MOGWO	NSGA2, MOTLBO, and SPEA2	8 quality indicators, i.e., fairness indicators, uncertainty indicators, and multi-objective optimizations indicators (MOO)	Multi objective next realease problem (MONRP)	2021	Ghasemi [33]
Improved grey wolf optimizer	IGWO	GWO	Real world failure datasets, statistical comparison metrics	Parameter estimation of software reliability growth models (SRGMs)	2021	Musa [34]
Hybrid grey wolf optimizer	HGWO	PSO, ABC, DABC, CGA, and MGA	Two groups of datasets	Parameter estimation of software reliability growth models (SRGMs)	2019	Alneamyand dabdoob [35]
Whale and grey wolf optimization algorithm	WGW	Replacement access, library and ID card project (RALIC) dataset	RALIC dataset	Prioritization of software requirements	2018	Masadeh et al. [36]
Enhanced binary harris hawk optimization algorithm	EBHHO	BGOA, ESBHHO, BALO, BWOA, BBA, BGSA, and GA	Datasets in the field of software fault prediction (SFP)	Software fault prediction	2020	Thaher and arman [37]

Table 5: Continued.

Variant	Short name	Effectiveness compared with	Efficiency/performance evaluated/validated by	Case study	Publication year	Authors [ref.]
Island binary moth flame optimization	IsBMFO	BMFO	21 public software datasets	Software defect prediction	2021	Khurma et al. [38]
Moth firefly algorithm	MFA	MFO and FA	5 object-oriented benchmark applications	Path testing	2020	Sharma and saha [39]
Enhanced binary moth flame optimization	EBMFO	BGOA, BGSA, WOA, BBA, and BALO	15 different software fault projects obtained from PROMISE public software engineering repository	Software fault prediction	2020	Tumar et al. [40]
Modified mothflame optimization	MMFO	MFO, BBAT, and MBBAT	The dataset with 23 features	Usability feature selection	2020	Gupta et al. [41]
Modified whale optimization algorithm	MWOA	WOA, GWO, PSO, and DABC	34 benchmark function	Parameter estimation of SRGMs	2021	Lu and ma [42]
Hybrid whale optimization algorithm	HWOA	BAT and ACO	Subject programs retrieved from the software artifact infrastructure repository	Regression test case selection	2020	Agrawal et al. [43]
Ambha_Whale optimization algorithm	Ambha_WOA	WOA	12 test case suites taken from GitHub and SIR repositories	Test case suite selection	2022	Jyoti and sharma [44]
Tournament selection method with binary WOA	TBWOA	BGWO, BGSA, BPSO, BALO, BBA, and BSSA	17 SFP datasets from the PROMISE repository	Software fault prediction	2021	Hassouneh et al. [45]
Multi-objective whale optimization algorithm	MOWOA	NSGA2, MOTLBO, and SPEA2	8 quality indicators i.e., fairness indicators, uncertainty indicators, and multi-objective optimizations indicators (MOO)	Multi objective next release problem (MONRP)	2021	Ghasemi [33]
Modified whale optimization algorithm	MWOA	WOA	3 real measured test/debug data sets	Parameter estimation of SRGMs	2018	Lu and ma [46]

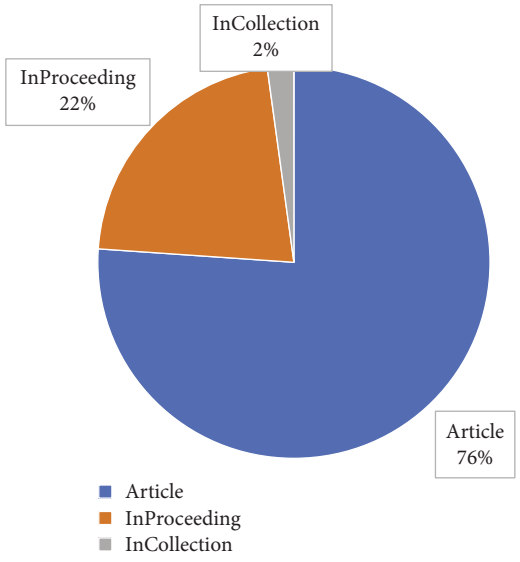

Figure 5: Document type of identified studies.

Gupta [50] utilized Biographical Based Optimization (BBO) and GWO in test suite prioritization as well as minimization. The performance was then evaluated with other meta-heuristic algorithms.

Rhmann [57] utilized GWO and random forest (RF) for predicting software vulnerability. According to the results, the proposed technique, i.e., GW-RF, outperformed machine learning techniques for software vulnerability prediction. Mohammad et al. [59] proposed a multi-objective HHO for binary classification problems using the Adaptive Synthetic Sampling (ADASYN) Technique. The authors used a healthcare dataset. Results confirmed the significant performance of the proposed model. Almayyan [58] evaluated different feature selection algorithms such as GWO, CS, Bat, and PSO for the prediction of software defects. The NASA dataset benchmarks were analyzed using three clustering algorithms such as X-Means, Farthest First, and Self Organizing Map (SOM). The proposed clustering model was used to build an efficient predictive model with an acceptable number of features as well as a good detection

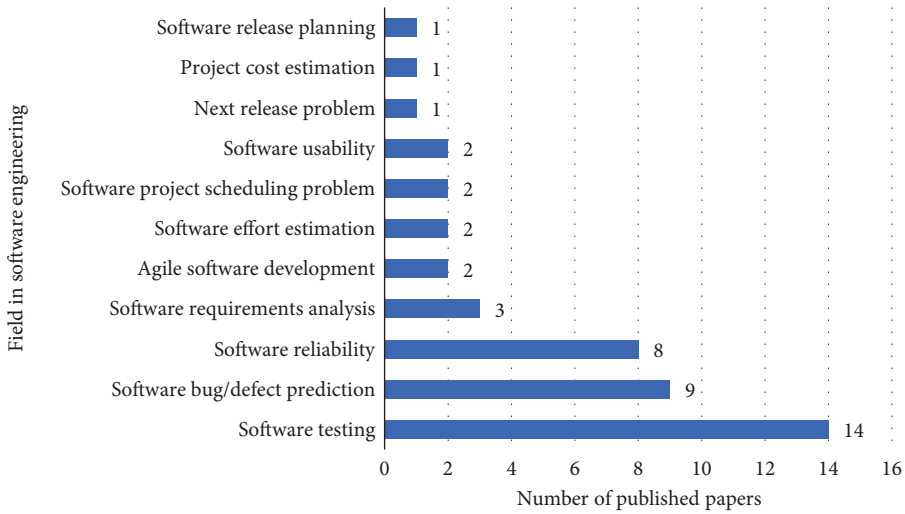

Figure 6: Ratio of identified studies by various fields in software engineering.

rate. Khurma et al. [38] proposed a binary variant of MFO (BMFO) by using the island BMFO (IsBMFO) model for the software defect prediction (SDP) problem. Results showed that IsBMFO, followed by support vector machine (SVM) classification, outperformed other models for the SDP problem, with an average G-mean of 78%. Alsghaier and Akour [60] developed an approach for software fault prediction by combining the genetics algorithm (GA) with the SVM classifier and WOA. It was then applied to 24 datasets, and the results confirmed the improved performance of the software fault prediction process. Hassouneh et al. [45] utilized WOA to present an efficient wrapper Feature Selection (FS) approach. The WOA was enhanced with nature selection operators to improve its efficacy in dealing with FS tasks. Zhu et al. [61] developed an enhanced metaheuristic search-based feature selection algorithm named EMWS by utilizing WOA and simulated annealing (SA). The results confirmed the superiority of EMWS. Thaher and Arman [37] proposed an enhanced binary version of HHO (EBHHO) for the FS problem. The authors used the Adaptive Synthetic (ADASYN) oversampling technique to reduce the dimensionality of the dataset using the FS technique. Results showed the superiority of EBHHO over the basic HHO. Tumar et al. [40] proposed an enhanced binary MFO (EBMFO) as a wrapper feature selection with ADASYN for software fault prediction. According to the results, the EBMFO improved the overall performance of classifiers.

Dhavakumar and Gopalan [29] used the Chaotic GWO algorithm (CGWO) in the Software Reliability Growth Model (SRGM). SRGM utilizes optimization algorithms to advance the parameters by dividing them into stages. However, by using CGWO, the technique is upgraded by using all the parameters simultaneously. Lu and Ma [42] proposed a modified WOA (MWOA) for predicting software reliability. Using software fault historical data, MWOA is used to predict the faults during software testing. Also, for the accurate estimation of software faults, the authors developed a modified sigmoid model (MSM). Musa [34] adopted the Improved GWO (IGWO) to estimate the optimum parameters for SRGMs. The proposed method utilizes seven real-world failure datasets. As per the results, the proposed method of IGWO performs better than the existing GWO. Gupta et al. [64] applied Random Walk GWO (RW-GWO) to determine the optimal redundancies to optimize the system reliability and the optimum cost of two distinct complex systems with constraints imposed on system reliability. The comparison proved the efficiency of the RW-GWO. Alneamy and Dabdoob [63] proposed the binding of the GA and the GWO algorithms for the estimation of the parameters of SRGMs. According to the results, the proposed binding GWO-RGA performed better than the previous binding HGWO in terms of accuracy of parameter estimation as well as performance using the same datasets. Lu and Ma [46] proposed a modified WOA (MWOA) and a three-stage SRGM. The parameters of three-stage SRGMs are estimated using MWOA. Results showed that using MWOA with the three-stage model is more helpful in estimating the software faults. Salahaldeen et al. [35] estimated the parameters of SRGMs depending on failure data. A hybrid GWO (HGWO) was used by combining the GWO with the Real Coded Genetic Algorithm (RGA). According to the results, all other algorithms, such as PSO, ABC, Classic GA (CGA), the Dichotomous Artificial Bee Colony (DABC), and the Modified GA (MGA), were outperformed by the GWO. Sheta and Abdel-Raouf [62] explored the GWO algorithm to estimate the SRGM's parameters to minimize the difference between the actual and estimated number of failures of the software. Three different SRGM models, namely, the Power Model (POWM), the Exponential Model (EXPM), and the Delayed S-Shaped Model (DSSM), were evaluated. The study was conducted using three different datasets.

Alzaqebah et al. [66] utilized the WOA to prioritize the software requirements. The proposed technique was compared with the analytical hierarchy process (AHP). According to the results, the proposed technique outperformed the AHP technique with an approximate margin of 40%. Masadeh et al. [36] proposed a hybrid approach

Table 6: Number of papers published in various journals or conferences.

Name of journal or conference	No of published studies
IEEE Access	4
International Journal of Intelligent Engineering & Systems	2
Springer	2
2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)	1
2018 17th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)	1
2018 9th International Conference on Information and Communication Systems (ICICS)	1
2020 10th International Conference on Cloud Computing, Data Science & Engineering	1
2020 11th International Conference on Information and Communication Systems (ICICS)	1
2020 6th International Engineering Conference "Sustainable Technology and Development"(IEC)	1
2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)	1
2021 16th International Conference on Emerging Technologies (ICET)	1
2021 5th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence	1
Advances in Systems Science and Applications	1
Applied Intelligence	1
Cluster Computing	1
Computing	1
International Journal for Research in Applied Science and Engineering Technology	1
International Journal of Advanced Computer Science and Applications	1
International Journal of Applied Metaheuristic Computing (IJAMC)	1
International Journal of Artificial Intelligence and Applications (IJAIA)	1
International Journal of Computer Applications	1
International Journal of Computer Applications in Technology	1
International Journal of Distributed Systems and Technologies (IJDST)	1
International Journal of Recent Research and Review	1
International Journal of Recent Technology and Engineering (IJRTE)	1
IOP Conference Series: Materials Science and Engineering	1
Journal of Algorithms & Computational Technology	1
Journal of Ambient Intelligence and Humanized Computing	1
Journal of Information and Optimization Sciences	1
Journal of Intelligent & Fuzzy Systems	1
Journal of Systems and Software	1
Mathematics	1
Modern Applied Science	1
Soft Computing	1
Soft computing for problem solving	1
Soft Computing Methods for System Dependability	1
Software: practice and experience	1
Specialusis Ugdymas	1
TELKOMNIKA (Telecommunication Computing Electronics and Control)	1
Webology	1
Wireless Communications and Mobile Computing	1

using WOA and GWO algorithms (WGW) for the prioritization of software requirements. The RALIC dataset was used to evaluate the proposed method. Compared with the RALIC dataset, the proposed method showed 91% accuracy in requirements prioritization. In [65], Masadeh et al. applied GWO to prioritize the requirements of a software project in an ordered list. Results were compared and evaluated with the AHP technique based on average running time and dataset size. According to the results, the RP-GWO performed better than the AHP mechanism by approximately 30%.

Prakash and Viswanathan [31] proposed a technique based on Agile Risk Prioritization and GWO (ARP–GWO) for risk prioritization. By using the technique, risk factors were prioritized during the development of agile software. The effectiveness of ARP-GWO was analyzed using performance metrics such as Usability Goals Achievement Metric and Index of Integration. The results showed enhancements in comparison with the available agile process. Also, the author [67] performed a comparative study of five effective metaheuristic algorithms, namely, Ant Colony Optimization (ACO), PSO, GA, GWO, and AHP, to prioritize the risks in agile environments. As per results, GWO performed better in the prioritization of risks in an agile environment based on accuracy, reliability, running time, and error rate.

Khan et al. [68] presented a Deep Neural Network (DNN) model based on GWO and StrawBerry Algorithm (SBA) for software effort estimation. Nine benchmark functions were applied to validate the performance of the

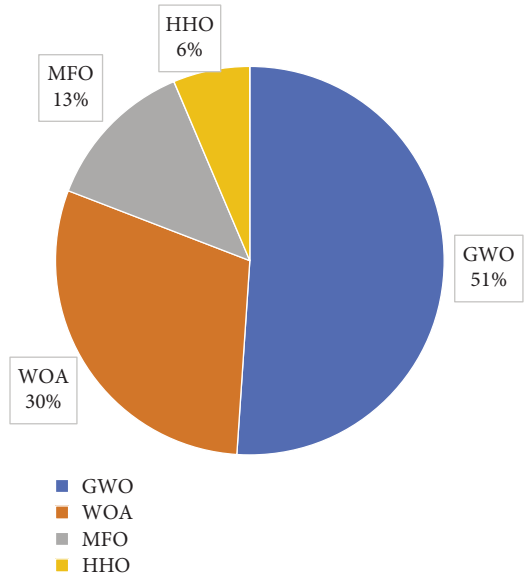

Figure 7: Swarm intelligence algorithm used by the identified studies.

Table 7: Summary of selected metaheuristics' applications.

Application area	No. of publications	[Ref.]
Software testing	14	[30, 39, 43, 44, 47–56]
Software bug/defect prediction	9	[37, 38, 40, 45, 57–61]
System reliability	8	[29, 34, 35, 42, 46, 62–64]
Software requirements analysis	3	[36, 65, 66]
Agile software development	2	[31, 67]
Software effort estimation	2	[68, 69]
Software project scheduling problem	2	[70, 71]
Software usability	2	[32, 41]
Next release problem	1	[33]
Software release planning	1	[72]
Project cost estimation	1	[73]
Software release planning	1	[72]
Software risk management	1	[74]

two algorithms. The proposed DNN model (GWDNNSB) produced better results for learning rate selection and initial weights as compared with the available work. Fadhil and Alsarraj [69] applied the WOA with the Constructive Cost Model II (COCOMO II) to solve the problem of ideal transactions. The NASA 93 dataset was used by the algorithm. The proposed approach based on the Mean Magnitude of Relative Error (MMRE) metric showed the best results.

Alabajee et al. [70] explored GWO for solving the software project scheduling problem (SPSP) by seeking an optimum solution. The authors compared results with the intelligent water drop algorithm (IWD), max-min ant system hyper cube framework (MMAS-HC), ACO, FA, intelligent water drop autonomous search (IWDAS), and intelligent water drop algorithm standard version (IWDSTD). According to the results, GWO showed better optimizing performance than the rest of the algorithms. Also, Alreffaee and Alabajee [71] proposed WOA to resolve SPSP by utilizing three datasets. The algorithm gave good results.

Jain [32] proposed a modified GWO (MGWO) algorithm for the selection of crucial features in a hierarchical software model. As per the results, MGWO outperformed other relevant optimizers in terms of accuracy. Gupta et al. [41] introduced a modified MFO (MMFO) for usability feature selection. Results confirmed the better performance of the proposed algorithm.

Ghasemi [33] proposed a multi-objective version of GWO and WOA by solving the bi-objective next release problem (NRP). The two algorithms with three other evolutionary algorithms were used to solve instances of the NRP problem from four different datasets. To satisfy the constraints of the NRP problem, a roulette wheel, and a cost-to-score ratio were used. According to the results, MOWOA outperformed others whereas, with reduced budget constraints, MOGWO performed better.

Hassan and Khan [73] implemented GWO, the Harmony Search Algorithm (HSA), and the SBA for software cost estimation. The NASA dataset was used, and to find a better algorithm, MMRE was utilized. GWO outperformed other algorithms regarding MMRE reduction.

Verma et al. [72] presented GWO's application in software release planning, considering warranty based on the proposed mathematical model used to measure software systems' reliability growth. The software cost model is based on fault reduction factor, fault removal efficiency, and error generation. The fault dataset of ERP systems was used to validate the model.

Prakash and Viswanathan [74] used the GWO algorithm to prioritize the risks involved in software development. The proposed method of software risk prioritization (SRP-GWO) was compared with other techniques such as PSO, AHP, average ranking, categorizing scale, and Delphi. Five attributes, namely, adaptability, simplicity, accuracy, consistency, and running time, were used for evaluation. According to the results, the proposed method outperformed other techniques.

4.2. Countries, Organization and Active Individuals Involved in Searched-Based Software Engineering Using GWO, WOA, HHO, and MFO (RQ3). This study finds active researchers, affiliated organizations and countries involved in research regarding the selected metaheuristics in software engineering. According to the top researchers' ranking, Abdullah Alzaqebah, Amjad Hudaib, Prakash Balasubramanian, Raja Masadeh, and Viswanathan Vadivel are the topmost researchers with 3 publications each.

Anju Saha, Ankur Choudhary, Arun Prakash Agrawal, Deepak K Gupta, Jamal Salahaldeen Alneamy, Kezhong Lu, Marrwa Abd-AlKareem Alabajee, Marwah M. A. Dabdoob, Rashmi Sharma, Taghreed Riyadh Alreffaee and Zongmin Ma have 2 publications each, as shown in Figure 8. Top organizations affiliated by authors are also ranked which shows that.

"Vellore Institute of Technology, Chennai, India," "The University of Jordan, Amman, Jordan" and "The World Islamic Sciences and Education University, Amman, Jordan" are at the top with 3 publications each.

"University of Mosul, Iraq", "Maharaja Agrasen Institute of Technology, Delhi, India", "Nanjing University of Aeronautics and Astronautics, Jiangsu, China" and "Chizhou University, Anhui, China" have 2 publications each as illustrated in Figure 9.

Lastly, active countries having the highest number of published articles on selected metaheuristics in software engineering are ranked as per details given in Figure 10. According to the ranking, India is the most active country with 20 publications; Iraq is in the 2nd position with 6 publications; Jordan has 5 publications; China and Saudi Arabia have 4 publications each; Pakistan, Egypt, and Palestine have 3 publications each; Iran, Malaysia, and Vietnam have 2 papers each, whereas Brazil, Kuwait, Libya, Nepal, Poland, Portugal, Russia, and the USA have 1 publication each.

4.3. Potential Applications of GWO, WOA, HHO, and MFO in Software Engineering (RQ4). Various metaheuristic algorithms are used in different fields of software engineering. Ranichandra [75] utilizes ACO for enhancing estimation accuracy in analogy-based software cost estimation by proposing the non-orthogonal space distance (NoSD) technique. Prajapati and Kumar [76] address the issues related to multiple objective optimization of software remodularization using a customized version of PSO known as PSO-based multi-objective software remodularization (PSO-MoSR). In [77], Al-Azzoni and Iqbal use GA and ACO to demonstrate a software framework for solving instances of software component allocation problems. In [78], Wang propose a novel hybrid metaheuristic algorithm, CVTMaker, which may help publishers identify ideal crowdsourced virtual teams (CVTs). Sun et al. [79] utilize PSO with a reverse edge known as REPSO for multi-objective software module clustering problem (MOSMCP), which divides the complex software system into subsystems to obtain a perfect structure. Prajapati and Chhabra [80] use PSO-based module clustering (PSOMC) for Software Module Clustering problem. Tang et al. [81] propose two metaheuristic search algorithms, i. e., coordinate descent and genetic algorithms, for exploring the configuration space of Hadoop for high-performing configurations which perform significantly better than the default configuration settings. In [82], Haraty et al. use simulated annealing (SA) for state-based software testing, especially web applications. Mann et al. in [83] propose a path-specific approach for automatic test case generation (ATCG) using GA, PSO, and ABC. In the same paper, another approach, i. e., a test case prioritization (TCP), is also proposed using PSO.

Likewise, the selected metaheuristics i. e., GWO, WHOA, HHO, and MFO are used in most areas of software engineering due to their usefulness in terms of simplicity, flexibility, and scalability. They lead to the desired convergence by keeping a good balance between exploration and exploitation, which are the basic search behaviors of metaheuristic algorithms.

Applications of GWO, WOA, HHO, and MFO, in software engineering discusses applications of these swarm intelligence algorithms in various fields of software engineering, such as software testing, software reliability, next release problem, agile software development, and software requirements analysis, as given in Table 7. Metaheuristics algorithms are useful in optimization; therefore, problems that come under the realm of optimization problems could be solved using metaheuristics such as GWO. The following are some of the areas of software engineering where the applications of the selected swarm intelligence algorithms can potentially be used:

(i) Software vulnerability prediction

(ii) Software organization

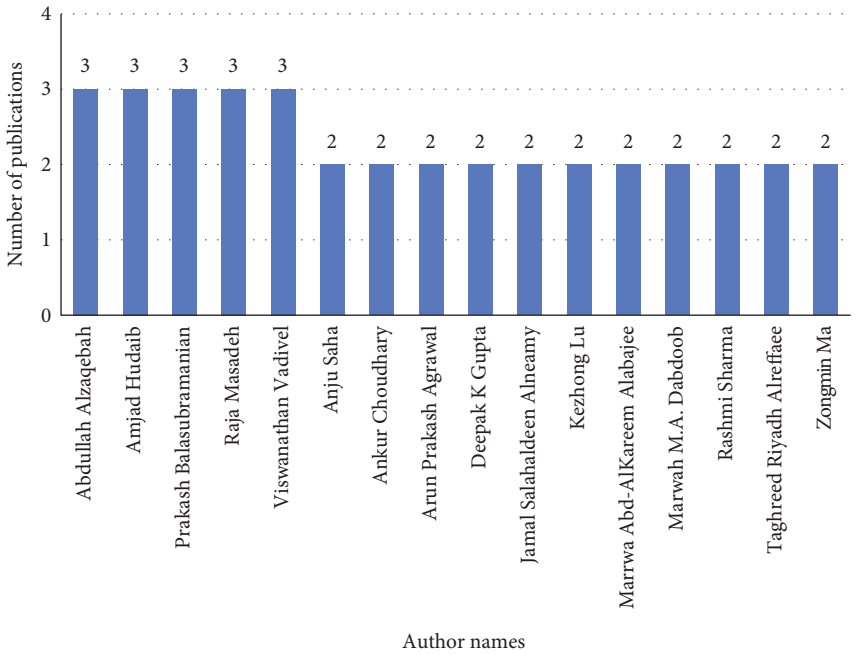

Figure 8: The top researchers using GWO in software engineering.

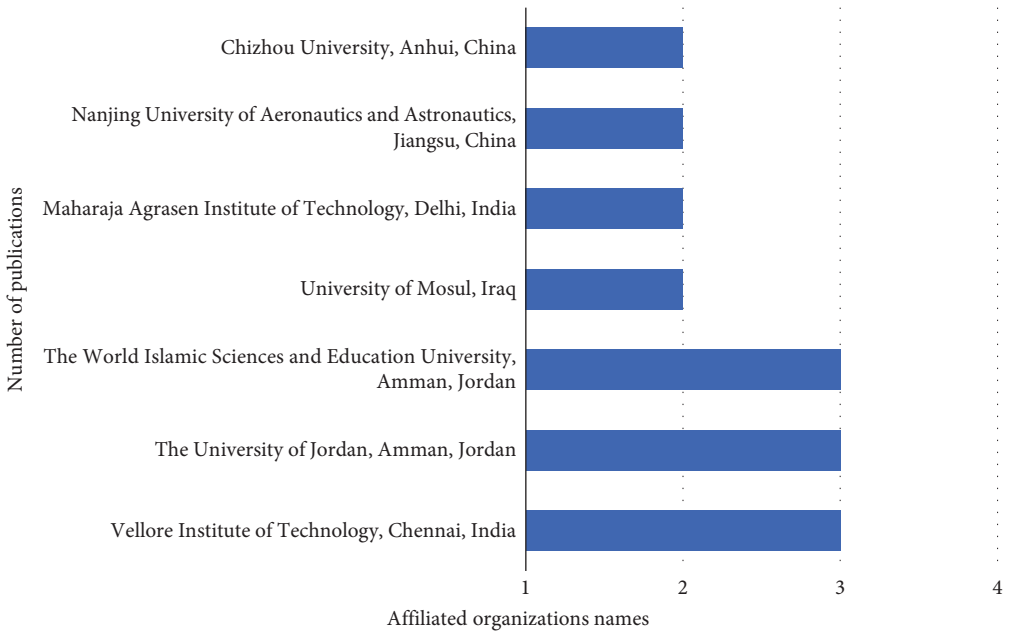

Figure 9: The top affiliated organizations using GWO in software engineering.

(iii) Software re-modularization
(iv) Maintenance/evolution system integration
(v) Software module clustering
(vi) Finding good designs
(vii) Reverse and re-engineering through transformation and re-factoring
(viii) User-based fitness evaluation for aesthetic aspects of software engineering
(ix) Avionics software
(x) Automotive software
(xi) Software configuration
(xii) Software component allocation
(xiii) Software testing
    (1) Structural testing
    (2) Mutation testing
    (3) Worst-and-best case execution time testing
    (4) Issue of boundary value analysis
    (5) Partition testing

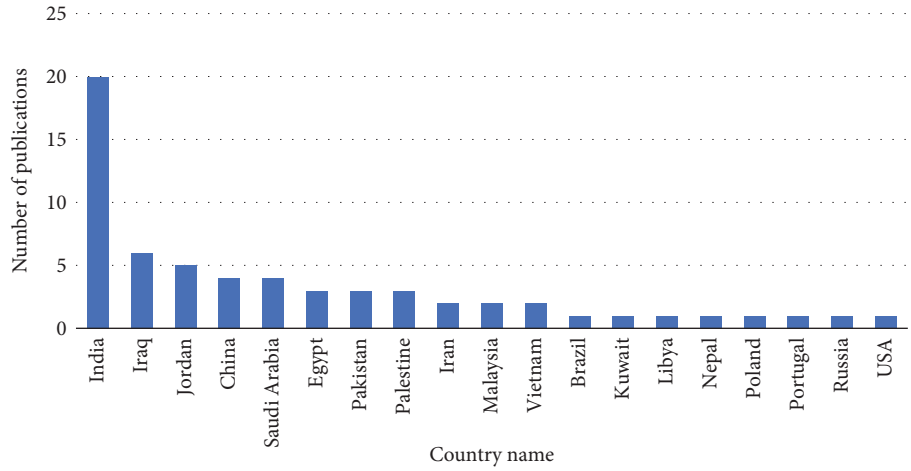

Figure 10: Active countries using GWO in software engineering.

5. Discussion

In this paper, we selected four swarm intelligence algorithms, i.e., GWO, WOA, HHO, and MFO, to study their applications in the field of software engineering. Our study presented an in-depth analysis of the applications of these metaheuristics in software engineering from 2014 up to 2022.

Our SLR is comprised of three phases, such as planning, conducting, and reporting. In the planning phase, we composed research questions that laid the foundation of our SLR. Data sources were then identified, such as IEEEXplore, ACM Digital Library, Springer Link, and ScienceDirect. We then formed a search strategy for extracting relevant literature. The PRISMA was utilized to obtain the primary studies for the SLR. The PRISMA flow diagram consists of various steps such as identification, screening, eligibility, and inclusion, as shown in Figure 3. Using PRISMA, 48 studies were found to be eligible. The identified studies were further subjected to quality assessment criteria and as a result, 2 more papers were dropped. Thus, a total of 46 papers were finally selected for our SLR.

According to the identified studies, GWO is the most used metaheuristic algorithm with 51% usage, WOA has 30% usage, MFO has 13% usage, and HHO has 6% usage, as shown in Figure 7. Also, our analysis showed that the selected metaheuristics have mostly been used in software testing, software defect prediction, and software reliability as shown in 6. Our study also revealed an increase in recent years in the usage of metaheuristic algorithms in software engineering, as evident in Figure 4. Furthermore, the top journals where the identified studies have been published are "IEEE Access," "Springer," and "International Journal of Intelligent Engineering and Systems."

As per the No Free Lunch theorem, no metaheuristic could solve all problems but a specific range of problems. Moreover, metaheuristics are easy to use and implement but they do not always guarantee the solution. To overcome some of the drawbacks of metaheuristics such as being trapped in local optima, researchers have developed variants of the existing metaheuristics as shown in detail in Table 5. Despite their shortcomings, metaheuristics are still popular as they provide near-optimal solutions to some of the hard problems such as NP-hard.

5.1. Threats to Validity. Various challenges can affect the validity of literature mapping or review studies. The following recommendations and guidelines can be used as compensation for the threats of this study:

(i) Coverage of relevant literature: This study includes publications that cite the selected metaheuristics, i.e., GWO, WOA, HHO, and MFO in software engineering from the time of GWO's invention to the present time, as GWO is the oldest among the selected swarm intelligence algorithms. New studies using the selected metaheuristics in software engineering published after our submission are beyond the scope of this study.

(ii) Coverage of research questions: The research questions may not cover all the aspects of state-of-the-art research on the selected metaheuristics in software engineering. Brainstorming is used to address this threat by optimally identifying the collection of research questions to cover the current research in this study.

6. Conclusion

This paper presented an in-depth literature study based on various applications of the selected metaheuristics, i.e., GWO, WOA, HHO, and MFO in the field of software engineering. The review study analyzed 46 papers published from 2014 to 2022 mentioning applications of the selected metaheuristics in the field of software engineering. Variants of the selected metaheuristics were also included in this study. This paper analyzed the selected papers according to the research questions of our SLR. This paper has mentioned countries, organizations, and authors actively involved in applying these metaheuristics in the field of software

engineering. The paper has also presented a detailed discussion on various applications of the selected metaheuristics in software engineering which revealed that these algorithms have mostly been used in software testing, software defect prediction, and software reliability. The study also pointed out areas of software engineering where these algorithms could be utilized. Hopefully, the current study may be helpful for new researchers using these swarm intelligence algorithms in software engineering for further enhancements.

Data Availability

This is a review article and all the data are discussed in the paper.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] B. Chopard and M. Tomassini, *An Introduction to Metaheuristics for Optimization*, Springer, Berlin/Heidelberg, Germany, 2018.

[2] J. K. Mykel and A. W. Tim, *Algorithms for Optimization*, Mit Press, Cambridge, Massachusetts, 2019.

[3] T. El-Ghazali, *Metaheuristics: from design to implementation*, Vol. 74, John Wiley & Sons, , Hoboken, New Jersey, U.S, 2009.

[4] M. Harman, S. Afshin Mansouri, and Y. Zhang, "Search-based software engineering: trends, techniques and applications," in *ACM Computing Surveys (CSUR)*vol. 45, no. 1, , pp. 1–61, ACM, 2012.

[5] M. Harman and B. F. Jones, "Search-based software engineering," in *Information and Software Technology*vol. 43, no. 14, , pp. 833–839, Elsevier, 2001.

[6] H. Faris, "Grey wolf optimizer: a review of recent variants and applications," in *Neural Computing & Applications*vol. 30, no. 2, , pp. 413–435, Springer, 2018.

[7] F. S. Gharehchopogh and H. Gholizadeh, "A comprehensive survey: whale Optimization Algorithm and its applications," *Swarm and Evolutionary Computation*, vol. 48, pp. 1–24, 2019.

[8] H. M. Alabool, D. Alarabiat, L. Abualigah, and A. A. Heidari, "Harris hawks optimization: a comprehensive review of recent variants and applications," *Neural Computing & Applications*, vol. 33, pp. 8939–8980, 2021.

[9] M. Shehab, L. Abualigah, H. Al Hamad, H. Alabool, M. Alshinwan, and A. M. Khasawneh, "Moth–flame optimization algorithm: variants and applications," *Neural Computing & Applications*, vol. 32, pp. 9859–9884, 2020.

[10] G.-G. Wang, S. Deb, and Z. Cui, "Monarch butterfly optimization," *Neural Computing & Applications*, vol. 31, pp. 1995–2014, 2019.

[11] S. Li, H. Chen, M. Wang, A. A. Heidari, and S. Mirjalili, "Slime mould algorithm: a new method for stochastic optimization," *Future Generation Computer Systems*, vol. 111, pp. 300–323, 2020.

[12] G. G. Wang, "Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems," *Memetic Computing*, vol. 10, pp. 151–164, 2018.

[13] A. Faramarzi, M. Heidarinejad, S. Mirjalili, and A. H. Gandomi, "Marine predators algorithm: a nature-inspired metaheuristic," *Expert Systems with Applications*, vol. 152, Article ID 113377, 2020.

[14] Y. Yang, H. Chen, A. A. Heidari, and A. H. Gandomi, "Hunger games search: visions, conception, implementation, deep analysis, perspectives, and towards performance shifts," *Expert Systems with Applications*, vol. 177, Article ID 114864, 2021.

[15] J. Tu, H. Chen, M. Wang, and A. H. Gandomi, "The colony predation algorithm," *Journal of Bionics Engineering*, vol. 18, pp. 674–710, 2021.

[16] A. Al-Momani, O. Mohamed, and W. Abu Elhaija, "Multiple processes modeling and identification for a cleaner supercritical power plant via Grey Wolf Optimizer," *Energy*, vol. 252, Article ID 124090, 2022.

[17] P. Hao and B. Sobhani, "Application of the improved chaotic grey wolf optimization algorithm as a novel and efficient method for parameter estimation of solid oxide fuel cells model," *International Journal of Hydrogen Energy*, vol. 46, no. 73, Article ID 36454, 2021.

[18] A. Bardhan, R. Biswas, N. Kardani et al., "A novel integrated approach of augmented grey wolf optimizer and ann for estimating axial load carrying-capacity of concrete-filled steel tube columns," *Construction and Building Materials*, vol. 337, no. 2022, Article ID 127454, 2022.

[19] A. R. Dhar, D. Gupta, S. S. Roy, A. K. Lohar, and N. Mandal, "Covariance matrix adapted grey wolf optimizer tuned eXtreme gradient boost for bi-directional modelling of direct metal deposition process," *Expert Systems with Applications*, vol. 199, Article ID 116971, 2022.

[20] L. Yin and Z. Sun, "Distributed multi-objective grey wolf optimizer for distributed multi-objective economic dispatch of multi-area interconnected power systems," *Applied Soft Computing*, vol. 117, Article ID 108345, 2022.

[21] J. Hu, H. Chen, A. A. Heidari et al., "Orthogonal learning covariance matrix for defects of grey wolf optimizer: insights, balance, diversity, and feature selection," *Knowledge-Based Systems*, vol. 213, Article ID 106684, 2021.

[22] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," in *Advances in Engineering Software*vol. 69, , pp. 46–61, Elsevier, 2014.

[23] S. Mirjalili, "Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization," in *Expert Systems with Applications*vol. 47, , pp. 106–119, Elsevier, 2016.

[24] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.

[25] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: algorithm and applications," *Future Generation Computer Systems*, vol. 97, pp. 849–872, 2019.

[26] S. Mirjalili, "Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm," *Knowledge-Based Systems*, vol. 89, pp. 228–249, 2015.

[27] B. Kitchenham and S. Charters, *Guidelines for Performing Systematic Literature Reviews in Software Engineering*Elsevier, Amsterdam, Netherlands, 2007.

[28] M. Kader, "A systematic review on emperor penguin optimizer," in *Neural Computing and Applications*, pp. 1–21, Springer, Berlin/Heidelberg, Germany, 2021.

[29] P. Dhavakumar and N. P. Gopalan, "An efficient parameter optimization of software reliability growth model by using chaotic grey wolf optimization algorithm," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 3177–3188, 2021.

[30] P. Rastogi, "An optimal software test case mechanism using grey wolf-FireFly method," *International Journal of Intelligent Engineering and Systems*, vol. 12, no. 2, pp. 22–32, 2019.

[31] B. Prakash and V. Viswanathan, "ARP–GWO: an efficient approach for prioritization of risks in agile software development," *Soft Computing*, vol. 25, no. 7, pp. 5587–5605, 2021.

[32] R. Jain, *Feature Selection Algorithm for Usability Engineering: A Nature Inspired Approach"*Springer, Berlin/Heidelberg, Germany, 2021.

[33] M. Ghasemi, "Multi-objective whale optimization algorithm and multi-objective grey wolf optimizer for solving next release problem with developing fairness and uncertainty quality indicators," *Applied Intelligence*, vol. 51, pp. 1–30, 2021.

[34] A. A. Musa, "Parameter estimation of software reliability growth models: a comparison between grey wolf optimizer and improved grey wolf optimizer," in *Proceedings of the 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pp. 611–617, IEEE, Amity University, India, January, 2021.

[35] J. Salahaldeen and M. Marwan, "The use of original and hybrid grey wolf optimizer in estimating the parameters of software reliability growth models," *International Journal of Computer Application*, vol. 167, pp. 12–21, 2017.

[36] R. Masadeh, A. Hudaib, and A. Alzaqebah, "Wgw: A hybrid approach based on whale and grey wolf optimization algorithms for requirements prioritization," *Advances in Systems Science and Applications*, vol. 18, pp. 63–83, 2018.

[37] T. Thaher and N. Arman, "Efficient multi-swarm binary Harris hawks optimization as a feature selection approach for software fault prediction," in *Proceedings of the 2020 11th International Conference on Information and Communication Systems (ICICS)*, pp. 249–254, IEEE, Irbid, Jordan, April, 2020.

[38] R. A. Khurma, H. Alsawalqah, I. Aljarah, M. A. Elaziz, and R. Damasevicius, "An enhanced evolutionary software defect prediction method using island moth flame optimization," *Mathematics*, vol. 9, p. 1722, 2021.

[39] R. Sharma and A. Saha, "An integrated approach of class testing using firefly and moth flame optimization algorithm," *Journal of Information and Optimization Sciences*, vol. 41, pp. 599–612, 2020.

[40] I. Tumar, Y. Hassouneh, H. Turabieh, and T. Thaher, "Enhanced binary moth flame optimization as a feature selection algorithm to predict software fault prediction," *IEEE Access*, vol. 8, pp. 8041–8055, 2020.

[41] D. Gupta, A. K. Ahlawat, A. Sharma, and J. J. P. C. Rodrigues, "Feature selection and evaluation for software usability model using modified moth-flame optimization," *Computing*, vol. 102, pp. 1503–1520, 2020.

[42] K. Lu and Z. Ma, "A modified whale optimization algorithm for parameter estimation of software reliability growth models," *Journal of Algorithms & Computational Technology*, vol. 15, Article ID 174830262110344, 2021.

[43] A. P. Agrawal, A. Choudhary, and A. Kaur, "An effective regression test case selection using hybrid whale optimization algorithm," *International Journal of Distributed Systems and Technologies*, vol. 11, pp. 53–67, 2020.

[44] B. Jyoti and A. K. Sharma, "An enhanced Whale Optimisation Algorithm for test case suite selection: ambha_WOA," *Specialusis Ugdymas*, vol. 1, pp. 4356–4372, 2022.

[45] Y. Hassouneh, H. Turabieh, T. Thaher, I. Tumar, H. Chantar, and J. Too, "Boosted whale optimization algorithm with natural selection operators for software fault prediction," *IEEE Access*, vol. 9, Article ID 14239, 2021.

[46] K. Lu and Z. Ma, "Parameter estimation of software reliability growth models by a modified whale optimization algorithm," in *Proceedings of the 2018 17th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, pp. 268–271, IEEE, Wuxi, China, October, 2018.

[47] A. M. Altaie, T. M. Tawfeeq, and M. G. Saeed, "Automated test suite generation tool based on GWO algorithm," *Webology*, vol. 19, pp. 3835–3849, 2022.

[48] S. Badanahatti and Y. S. S. R. Murthy, "Optimal test case prioritization in cloud based regression testing with aid of KFCM," *International Journal of Intelligent Engineering and Systems*, vol. 10, no. 3, pp. 96–105, 2017.

[49] B. Sunitha, "Prioritization of software applications in cloud using GWO algorithm," *International Journal for Research in Applied Science and Engineering Technology*, vol. 6, no. 5, pp. 2070–2075, 2018.

[50] D. Gupta and V. Gupta, *Test Suite Prioritization Using Nature Inspired Meta-Heuristic Algorithms*Springer, Berlin/Heidelberg, Germany, 2016.

[51] A. Kaur, "An approach to extract optimal test cases using AI," in *Proceedings of the 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pp. 649–654, Noida, India, January, 2020.

[52] A. S. Metwally, "WAP: a novel automatic test generation technique based on moth flame optimization," in *Proceedings of the 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 59–64, IEEE, Ottawa, ON, Canada, October, 2016.

[53] R. Sharma and A. Saha, "Optimal test sequence generation in state based testing using moth flame optimization algorithm," *Journal of Intelligent and Fuzzy Systems*, vol. 35, pp. 5203–5215, 2018.

[54] S. K. Harikarthik, V. Palanisamy, and P. Ramanathan, "Optimal test suite selection in regression testing with testcase prioritization using modified Ann and Whale optimization algorithm," *Cluster Computing*, vol. 22, no. S5, Article ID 11425, 2019.

[55] A. A. Hassan, S. Abdullah, K. Z. Zamli, and R. Razali, "Combinatorial test suites generation strategy utilizing the whale optimization algorithm," *IEEE Access*, vol. 8, Article ID 192288, 2020.

[56] J. Wang and W. Zhao, "Automatic test case generation method based on improved whale optimization algorithm," in *Proceedings of the 2021 5th International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence*, pp. 7–16, Victoria Seychelles, April, 2021.

[57] W. Rhmann, "Software vulnerability prediction using grey wolf-optimized random forest on the unbalanced data sets," *International Journal of Applied Metaheuristic Computing*, vol. 13, pp. 1–15, 2022.

[58] W. Almayyan, "Towards predicting software defects with clustering techniques," *International Journal of Artificial Intelligence and Applications (IJAIA)*, vol. 12, p. 1, 2021.

[59] U. G. Mohammad, S. Imtiaz, M. Shakya, A. Almadhor, and F. Anwar, "An optimized feature selection method using ensemble classifiers in software defect prediction for healthcare systems," *Wireless Communications and Mobile Computing*, vol. 202214 pages, 2022.

[60] H. Alsghaier and M. Akour, "Software fault prediction using whale algorithm with genetics algorithm," *Software: Practice and Experience*, vol. 51, pp. 1121–1146, 2021.

[61] K. Zhu, S. Ying, N. Zhang, and D. Zhu, "Software defect prediction based on enhanced metaheuristic feature selection optimization and a hybrid deep neural network," *Journal of Systems and Software*, vol. 180, Article ID 111026, 2021.

[62] A. F. Sheta and A. Abdel-Raouf, "Estimating the parameters of software reliability growth models using the grey wolf optimization algorithm," *International Journal of Advanced Computer Science and Applications*, vol. 7, p. 4, 2016.

[63] J. S. Alneamy and M. M. A. Dabdoob, "Proposed binding between genetic algorithm and grey wolf optimizer to estimate the parameters of software reliability growth models," *International Journal of Recent Research and Review*, 2019.

[64] S. Gupta, K. Deep, and A. Assad, "Reliability–redundancy allocation using random walk gray wolf optimizer," in *Soft Computing for Problem Solving*, pp. 941–959, Springer, Berlin/Heidelberg, Germany, 2020.

[65] R. Masadeh, A. Alzaqebah, and A. Hudaib, "Grey Wolf algorithm for requirements prioritization," *Modern Applied Science*, vol. 12, p. 54, 2018.

[66] A. Alzaqebah, R. Masadeh, and A. Hudaib, "Whale optimization algorithm for requirements prioritization," in *Proceedings of the 2018 9th International Conference on Information and Communication Systems (ICICS)*, pp. 84–89, IEEE, Irbid, Jordan, April, 2018.

[67] B. Prakash and V. Viswanathan, "A comparative study of meta-heuristic optimisation techniques for prioritisation of risks in agile software development," *International Journal of Computer Applications in Technology*, vol. 62, pp. 175–188, 2020.

[68] M. S. Khan, F. Jabeen, S. Ghouzali, Z. Rehman, S. Naz, and W. Abdul, "Metaheuristic algorithms in optimizing deep neural network model for software effort estimation," *IEEE Access*, vol. 9, Article ID 60309, 2021.

[69] A. A. Fadhil and R. G. Alsarraj, "Exploring the whale optimization algorithm to enhance software project effort estimation," in *Proceedings of the 2020 6th international engineering conference âĂIJSustainable Technology and development(IEC)*, pp. 146–151, Victoria Seychelles, March, 2020.

[70] M. A. A. Alabajee, D. R. Ahmed, and T. R. Alreffaee, "Solving software project scheduling problem using grey wolf optimization," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 19, pp. 1820–1829, 2021.

[71] T. R. Alreffaee and M. A. A. Alabajee, "Solving software project scheduling problem using whale optimization algorithm," in *IOP Conference Series: Materials Science and Engineering*vol. 928, IOP Publishing, Article ID 032084, 2020.

[72] V. Verma, N. Neha, and A. G. Aggarwal, "Software release planning using grey wolf optimizer," in *Soft Computing Methods for System Dependability*IGI Global, Hershey, Pennsylvania, 2020.

[73] C. A. Hassan and M. S. Khan, "An effective nature inspired approach for the estimation of software development cost," in *Proceedings of the 2021 16th International Conference on Emerging Technologies (ICET)*, pp. 1–6, IEEE, Islamabad, Pakistan, December, 2021.

[74] B. Prakash and V. Viswanathan, "Risk prioritization for software development using grey wolf optimization," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 7, 2019.

[75] S. Ranichandra, "Optimizing non-orthogonal space distance using ACO in software cost estimation," *Mukt Shabd J*, vol. 9, pp. 1592–1604, 2020.

[76] A. Prajapati and S. Kumar, "PSO-MoSR: a PSO-based multi-objective software remodularisation," *International Journal of Bio-Inspired Computation*, vol. 15, pp. 254–263, 2020.

[77] I. Al-Azzoni and S. Iqbal, "Meta-heuristics for solving the software component allocation problem," *IEEE Access*, vol. 8, Article ID 153067, 2020.

[78] H. Wang, "Solving team making problem for crowdsourcing with hybrid metaheuristic algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 318-319, Boston, MA, USA, August, 2018.

[79] J. Sun, Y. Xu, and S. Wang, "PSO with reverse edge for multi-objective software module clustering," *International Journal of Performability Engineering*, vol. 14, p. 2423, 2018.

[80] A. Prajapati and J. K. Chhabra, "A particle swarm optimization-based heuristic for software module clustering problem," *Arabian Journal for Science and Engineering*, vol. 43, pp. 7083–7094, 2018.

[81] C. Tang, K. Sullivan, and B. Ray, "Searching for high-performing software configurations with metaheuristic algorithms," in *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, pp. 354-355, Gothenburg, Sweden, December, 2018.

[82] R. A. Haraty, N. Mansour, and H. Zeitunlian, "Metaheuristic algorithm for state-based software testing," *Applied Artificial Intelligence*, vol. 32, pp. 197–213, 2018.

[83] M. Mann, P. Tomar, and O. P. Sangwan, "Bio-inspired metaheuristics: evolving and prioritizing software test data," *Applied Intelligence*, vol. 48, pp. 687–702, 2018.