Engineering Reports

WILEY

Engineering Reports Open Access

RESEARCH ARTICLE OPEN ACCESS

Multimodal Lotus Effect Algorithm for Engineering Optimization Problems

Elham Dalirinia[1,2] | Mahdi Yaghoobi[3] | Hamid Tabatabaee[1,2] | Swati Chandna[4] | Mehrdad Jalali[4,5]

[1]Department of Computer Engineering, Ma.C., Islamic Azad University, Mashhad, Iran | [2]Department of Artificial Intelligent and Data Science, Intelligent Financial Innovation Research Center, Ma.C., Islamic Azad University, Mashhad, Iran | [3]Department of Electrical Engineering, Ma.C., Islamic Azad University, Mashhad, Iran | [4]Department of Information, SRH University of Applied Sciences Heidelberg, Heidelberg, Germany | [5]Institute of Functional Interfaces, Karlsruhe Institute of Technology (KIT), Eggenstein-Leopoldshafen, Germany

Correspondence: Mehrdad Jalali (mehrdad.jalali@srh.de)

Received: 20 September 2024 | **Revised:** 31 March 2025 | **Accepted:** 2 April 2025

Funding: This study was supported by the KIT-Publication Fund of the Karlsruhe Institute of Technology.

Keywords: evolutionary algorithms | glowworm swarm algorithm | lotus effect algorithm | M-LEA | multimodal optimization | Nash equilibrium | resource identification

ABSTRACT

Multimodal optimization problems (MMOPs) are critical in fields like game theory and robotics, where identifying multiple optimal solutions simultaneously is essential, yet challenging due to the need for effective global exploration and precise localization of optima. This study introduces the multimodal lotus effect algorithm (M-LEA), a novel extension of our previously published lotus effect optimization algorithm (LEA), which was designed for single-modal optimization and thus struggled to maintain multiple optima in complex multimodal spaces. M-LEA addresses this limitation by incorporating a roaming technique with independently evolving subpopulations, enabling it to navigate multimodal spaces without requiring parameters such as radius or prior information about the number or distribution of optima. Its robustness is demonstrated through comparisons with five algorithms on the IEEE CEC2013-2015 challenge, where M-LEA consistently outperformed competitors. The algorithm's practical utility is further validated in two applications: identifying Nash equilibrium points in game theory and localizing resources via robotic systems. Results show that M-LEA achieves superior performance and stability, making it well-suited for scenarios demanding high efficiency and precision. These findings highlight M-LEA's potential for diverse domains, paving the way for its application in game theory, robotics, and other fields requiring advanced multimodal optimization techniques.

1 | Introduction

Multimodal optimization problems (MMOPs) involve finding multiple optimal solutions—for a single objective function, as opposed to identifying just one best solution. These problems are common in real-world applications where multiple satisfactory solutions may exist due to constraints or varying requirements. Pursuing all possible peaks and enhancing the accuracy of the solutions for pre-identified peaks are the objectives of solving MMOPs. There are many local optima for MMOPs, and many algorithms mainly focus on enhancing the global exploration ability to improve algorithm performance [1]. In actual industrial production, multiple selectable optimal solutions are desirable, such as in manufacturing, where different production schedules can achieve the same output but vary in resource use, allowing flexibility to adapt to changes like material shortages or cost fluctuations. In such scenarios, if the preferred solution is inappropriate, alternative solutions can be rapidly switched to while

maintaining optimal performance [2]. Multimodal optimization algorithms face two significant challenges: a compelling global exploration of the entire search space to find more peaks and the generation of final solutions with the required precision for the precise locations of peaks [3].

In multimodal optimization algorithms, achieving a successful outcome necessitates addressing two fundamental challenges: identifying both global and local multimodal optima and ensuring the retention of these optima until the search concludes. Furthermore, an inaccurate estimation of the radius parameter can negatively impact algorithmic efficiency [4, 5]. Several different multimodal optimization methods have been proposed. The most common technique aiding classical evolutionary algorithms (EAs) in maintaining a diverse population is Niching, which revolves around using different niches to locate different optima. The clearing technique utilizes the radius clearing distance [6]. Other multimodal optimization methods include the fitness-sharing method using the sharing radius, which divides the population into subgroups [7]. Another multimodal optimization method is the Bayesian optimization of multifaceted orderings, which is highly dependent on a parameter known as the niche radius and is determined based on the distance between two optima [8]. In swarm-based approaches for finding multiple optima, the output population must be evaluated using a mechanism related to the optimal radius [9].

However, the performance of these methods depends on an essential metric, similarity, which must be estimated from the optimization function's data. Additionally, to determine the number of optima in these approaches, the output population must be assessed using a mechanism dependent on the optimum radius [9, 10]. Clustering strives to be independent of estimating the optimal radius [11]. Species conservation introduces a parameter, species distance, which is estimated based on knowing the number of optima in the optimization function [12, 13]. The roaming technique, combined with external memory for storing solutions, has proven effective in multimodal optimization [14]. Recent advancements in hybrid optimization algorithms have shown significant promise in addressing complex multimodal problems by combining the strengths of various algorithmic strategies to enhance search efficiency and solution quality. Notably, the integrated Cuckoo Search and Particle Swarm Method (iCSPM) presents an innovative approach by amalgamating the exploratory merits of cuckoo search with the convergence characteristics of particle swarm optimization (PSO), providing a robust mechanism for escaping local optima optimization [14]. Similarly, the second iteration of this method, iCSPM2, introduces adaptive weighting schemes that dynamically adjust according to the landscape of the optimization problem, further refining the search process [14]. Another notable approach is the Exploratory Cuckoo Search, which incorporates random walk strategies to expand the diversity of solutions, particularly effective in vast search spaces encountered in engineering and industrial applications [14]. The improved salp swarm algorithm (ISSA) with hybrid differential perturbation method offers an enhanced exploration capability by integrating the differential evolution (DE) concept, which helps in fine-tuning the solutions during the latter stages of the search, proving its efficacy in high-dimensional spaces [14]. In the adaptive elitist population search technique, adjustments to genetic algorithm operators and a distance measure are employed to determine the proximity of two solutions to each other [15]. In the niche gravitational search algorithm (NGSA), the idea of subdividing the main population into smaller subgroups and preserving them is presented along with three strategies: K-nearest neighbors, elitism, and modification of the active gravitational mass formula [16]. In the multimodal forest optimization algorithm (MMFOA), clustering techniques based on niching methods are integrated to transform the single-peaked forest optimization algorithm [17]. In the multimodal bacterial foraging optimization (MBFO) algorithm, there is no requirement to predefine additional parameters, such as the niche parameter. Additionally, this algorithm is less complex than its single-peaked counterpart because it does not include the dedispersion phase or any other similar phase [18]. The migratory animals memetic optimizer (MAMO) does not require prior knowledge of the problem space, as it avoids the need to predefine niche parameters, instead requiring specific adaptations to enable multimodal optimization [19].

Methods for solving MMOPs with the help of EAs can be classified into two main groups: iterative and parallel approaches. In iterative methods, the EA is executed multiple times sequentially, extracting one optimum solution in each iteration. To prevent convergence to a single optimum, a penalty mechanism for solutions close to the previous optimum is considered in each iteration. In parallel methods, the entire population is initially divided into smaller subpopulations. These subpopulations simultaneously search the entire space, and in the final generation, all the optima are extracted [20].

Researchers have increasingly focused on using EAs to address MMOPs.

EAs, such as DE [21–23], PSO [24, 25], probabilistic multimodal optimization (PMO) [26], and artificial bee colony (ABC) methods [27, 28], exhibit great potential for addressing multimodal problems (MMOPs). Due to their intrinsic characteristics, they provide a practical framework for maintaining multiple solutions.

However, traditional EAs that typically converge to a single optimum are ill-suited for directly solving MMOPs. Classical EAs and other specific multimodal strategies have been explored to address MMOPs and represent potential advancements in this field. Some new approaches in multimodal optimization beyond genetic algorithms include the multimodal niche-based selection adaptive memetic algorithm (NSAMA), which is a method that effectively searches for solutions by combining DE and local search techniques to identify multiple optimal outcomes [29]; the multimodal grey wolf optimizer (NGWO), which is a niching GWO that combines the best personal characteristics of PSO and a local search technique to solve GWO problems such as premature convergence and imbalance between exploration and exploitation in solving multimodal problems [30]; and the brainstorm optimization for MMOPs (OIF-BSO), which uses the optimum identified framework (OIF) combined with a brainstorming optimization (BSO) algorithm to identify global optima and preserve them indefinitely [31]. The knowledge-driven brain storm optimization in objective space (KBSOOS) algorithm improves search performance and diversity in multi-mode optimization problems (MMOPs) by integrating random steps for dynamic

exploration and adapting parameters based on real-time feedback [32]; the surrogate-assisted multimodal evolutionary algorithm with knowledge transfer (SAKT-MMEA) is a modality prediction approach designed for the comprehensive investigation of modalities [33]; and the PSO algorithm based on the modified crowding distance (MOPSO_MCD) is used for multimodal multiobjective problems. This approach improves the decision diversity and enhances the global search ability of MOPSO_MCD [34]. The multi-modal battle royale optimizer (MBRO) is a novel metaheuristic algorithm inspired by battle royale game dynamics, designed to address MMOPs. It incorporates a ring neighborhood topology to maintain population diversity and avoid premature convergence, enabling the effective identification of multiple optimal solutions without requiring prior knowledge of the problem space [35].

One practical application of multimodal optimization is finding Nash equilibria in game theory, a popular concept for modeling games since the 1950s [36]. Over the years, various researchers have introduced alternative concepts for solving games, some of which, while akin to rational Nash equilibria, perform even more effectively in specific scenarios.

GSO is used to solve numerical MMOPs and perform realistic collective robotic tasks to localize static signal sources and track mobile phone signal sources; these sources can be multiple optima in a numerical optimization problem or physical quantities such as sound, light, or heat in a real robotic resource localization task [37]. Recent algorithms have yet to be assessed in this specific context, where robots are tasked with resource discovery. Therefore, this study compares the proposed algorithm with GSO and a selection of contemporary algorithms in this application. The evaluation results suggest that the proposed algorithm performs effectively in this domain.

For the data analysis literature summary, tables were developed, and data extraction included Reference Number, authors, year, title of article, and method used in the article, which are explained in Appendix A.

The article introduces the 'Optimization of Multimodal Problems using the Lotus Effect Algorithm' as an innovative method for improving the identification of Nash equilibrium and the localization of robotic resources. It should be mentioned that so far, the roaming technique has been based on the genetic algorithm and for the first time in the proposed algorithm, the roaming technique based on the lotus effect algorithm (LEA) has been presented to find all the optimal multimodal problems with high accuracy. This article presents an algorithm that utilizes distinct subpopulations to evolve independently. The LEA is adept at exploring multi-modal optimization spaces and offers the advantage of not requiring a stable population around the identified optima. This algorithm is not dependent on parameters like radius or prior information about the number and distribution of optima, making it valuable in applications where such information is unavailable, such as non-cooperative multiplayer game theory and robot resource identification, and in the smaller number of iterations of the algorithm execution, it converges to the optimum with high accuracy. In the identification of Nash equilibrium, the results demonstrated that the proposed algorithm was more proficient in extracting stronger optima compared to other methods in all games. Regarding the localization of robotic resources, the results indicate that the proposed algorithm exhibits relatively lower variance around the mean than another method, suggesting better stability in each run. The article underscores the potential of the LEA in addressing MMOPs and its practical applicability across diverse domains.

The main contributions of this paper are as follows:

- Introduced M-LEA for multimodal optimization: Developed the multimodal LEA (M-LEA), extending the single-modal LEA to effectively address MMOPs by navigating complex multimodal spaces using a roaming technique with independently evolving subpopulations.
- Pioneered roaming technique with LEA: Implemented, for the first time, the roaming technique using the LEA, enabling the identification of all optimal solutions in MMOPs without requiring parameters like radius or prior knowledge of optima distribution.
- Demonstrated superior performance on benchmarks: Achieved top performance on the IEEE CEC2013-2015 challenge, outperforming five advanced algorithms across 14 benchmark functions in success rate (SR), success performance (SP), average number of optimal points found, and maximum peak ratio (MPR), with statistical significance ($p < 0.05$).
- Validated practical utility in game theory and robotics: Applied M-LEA to identify Nash equilibrium points in game theory (outperforming 18 methods with a score of 20.91) and localize robotic resources (reducing iterations by up to 22.7% compared to GSO and jDE100), showcasing its efficiency and stability.

This paper is organized as follows: Section 2 discusses the fundamentals of the LEA, Section 3 details the structure of the proposed multimodal algorithm, Section 4 provides the empirical results, and Section 5 offers conclusions and suggests avenues for future research.

2 | Background: LEA

The introduction of the lotus effect optimization (LEO) algorithm has significantly influenced the field of meta-heuristic optimization, and the multimodal algorithm of the lotus effect is a significant improvement in the optimization of multivariable problems. Its ability to operate without reliance on specific parameters and its effectiveness in exploring complex optimization landscapes make it a valuable tool for researchers and practitioners alike. As the algorithm continues to be refined and tested in various applications, it holds the potential to enhance decision-making processes in competitive and resource-constrained environments. This novel approach not only contributes to theoretical advancements but also opens new avenues for practical implementations in diverse fields, paving the way for more efficient and robust solutions to complex optimization problems as evidenced by its citations in various recent research papers. For

example, the enhanced grey wolf optimizer (CMWGWO) discussed is the improvement of the grey wolf optimizer for continuous global numerical optimization, acknowledging LEO's innovative approach [38]. The innovative Flood Algorithm (FLA), drawing inspiration from the fluidity of water flow, perfectly complements nature-inspired optimization strategies like LEO [39]. An improved hybrid MPA (ICSOMPA) was presented with a hybrid optimization algorithm, highlighting the impact of LEO on the development of robust global optimization techniques [40]. A modified sailfish optimizer for diagnosing brain tumors, that demonstrates LEO's integration into medical imaging [41].

The Remora is combined with the LEA (IR-LEOA) to enhance resource-based data transmission in LPWAN through a deep adaptive reinforcement learning model [42].

These citations affirm LEO's versatility and efficacy across diverse applications, from engineering optimization to secure network protocols and dynamic cyber threat prediction [43–45].

Overall, these cases highlight the significance of using a variety of methods in the LEA to effectively tackle complex problems, and the multimodality of algorithms significantly enhances their ability to tackle complex problems by integrating diverse data types, improving performance, and enabling innovative solutions across various fields.

In the LEA, the process of pollination of flowers and the characteristics of their leaves serve as inspiration [46]. Pollination involves two major processes: biological and cross-pollination, and nonbiological self-pollination [47].

2.1 | Exploration Phase

In the exploration phase of the LEA, dragonflies carry out global pollination. The optimization algorithm uses a dragonfly optimization algorithm to model this part mathematically.

To emulate the intelligent actions of dragonflies, we incorporate three fundamental principles observed in insect swarms, as well as two notions involving food and predators:

- Separation: This refers to avoiding collisions between individuals and their neighbors.
- Alignment: This represents adjusting the speed of individuals relative to their neighbors.
- Cohesion: This indicates the tendency of individuals to move toward the center of their neighbors.

The primary objective in each swarm and group is to ensure survival. Therefore, all individuals must be attracted to food sources and avoid enemies. With these two behaviors in mind, there are five factors for updating the positions of individuals in the swarm:

Separation can be calculated as follows [48]:

$$S_i^t = -\sum_{j=1}^{N} X_i^t - X_j^t \tag{1}$$

where X_i indicates the position of the current person with number i in evolution round t, X_j indicates the position of the jth person in the neighborhood in evolution round t, and N also indicates the number of people in the neighborhood. The alignment is calculated as follows [48]:

$$A_i^t = \frac{\sum_{j=1}^{N} X_j^t}{N} \tag{2}$$

where search factor (X_i) is the speed of the jth person in the neighborhood in evolution round t. The cohesion is calculated as follows [48]:

$$C_i^t = \frac{\sum_{j=1}^{N} X_j^t}{N} - X_i^t \tag{3}$$

where search factor is the position of the current person with number i in evolution round t, N is the number of neighbors, and X_j is the position of the jth person in the neighborhood in evolution round t. The absorption toward the food source is calculated as follows [48]:

$$F_i^t = X_+^t - X_i^t \tag{4}$$

where search factor is the position of the current individual with the number i in the evolution round t and X_+^t is the position of the food source determined from the current evolution round, that is, t, and the best answer has been found. Escape from the enemy is calculated as follows [48]:

$$E_i^t = X_-^t + X_i^t \tag{5}$$

where X_i^t is the current person's position with the number i in the evolution round t, and X_-^t is the position of the enemy determined from the current evolution round, that is, t, and the worst answer is found. It is hypothesized that the behavior of dragonflies results from the amalgamation of these five patterns. To facilitate the updating of artificial dragonflies' positions within the search space and replicate their movement, two vectors are considered: the step length vector and the X position vector. The step length vector resembles the velocity vector in the particle swarm algorithm, upon which the dragonfly algorithm is constructed. This vector, which is indicative of the direction of dragonfly movement, is delineated as follows [48]:

$$\Delta X_i^{t+1} = \left(sS_i^t + aA_i^t + cC_i^t + fF_i^t + eE_i^t\right) + w\Delta X_i^t \tag{6}$$

where s represents the separation coefficient, S_i^t is the degree of separation related to the ith individual in evolution round t, a is the coefficient related to alignment, A_i^t is the alignment value related to the ith individual, c is also the cohesion coefficient, and C_i^t is also the cohesion value of the ith person. The value of f is the feeding factor, F_i^t is also the food source related to the ith person, e is the enemy factor, E_i^t is the position of the enemy related to the ith person, w is the inertial weight, and t is the repetition counter of the algorithm. After calculating the step vector, the position vectors are calculated as follows [48]:

$$X_i^{t+1} = X_i^t + w\Delta X_i^{t+1} \tag{7}$$

where t is the repetition counter of the algorithm.

By using coefficient factors, separation (s), alignment (a), cohesion (c), food (f), and enemy (e), various exploration and exploitation behaviors occur during the execution of the algorithm. The area of the food source and the enemy is obtained from the best and worst answers found in the entire swarm thus far. This causes convergence toward promising regions of the search space and divergence toward unfavorable regions in the search space. To enhance the stochastic exploration behavior of artificial dragonflies, they are required to navigate the search space by employing a random step length when no solution is present within their vicinity. Under such circumstances, the positions of the dragonflies are updated according to the following relation:

$$X_i^{t+1} = X_i^t + Levy(d) \times X_i^t \tag{8}$$

where t is the current iteration counter, d is the dimension of the position vector, and the levy value is calculated using the following equation:

$$\text{Levy}(x) = 0.01 \times \frac{r_1 \times \sigma}{\left|r_2\right|^{\frac{1}{\beta}}} \tag{9}$$

where r_1 and r_2 are two random numbers in the interval between 0 and 1, and β is a constant. σ is calculated using the following relation:

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right)^{\frac{1}{\beta}} \tag{10}$$

where

$$\Gamma(x) = (x-1)! \tag{11}$$

The exploration phase thus establishes a solid foundation for identifying potential global solutions, paving the way for the precise extraction processes discussed in the subsequent section.

2.2 | Extraction Phase

In the extraction phase of the LEA, local pollination is performed with Equations (12) and (13).

$$X_i^{t+1} = X_i^t + R\left(X_i^t - g^*\right) \tag{12}$$

X_{t+1} is the location of the pollen in the $t+1$th iteration, and g^* is the location of the best position of the pollen thus far and in all rounds of evolution. R is the growth area, which decreases according to the repetitions of the algorithm. In fact, the movement steps are more common at the beginning of the algorithm and decrease as the end of the algorithm approaches until convergence to the optimality of the problem is achieved.

$$R = 2\, e^{-\left(\frac{4t}{L}\right)^2} \tag{13}$$

where t is the iteration number of the current evolution of the algorithm and L is the number of final iterations of the algorithm. The movement of water on the leaves is modeled to strengthen the extraction in the mentioned algorithm. Each member of the population or solution (drop) moves with an initial position and speed in the problem search space to find the optimal solution, and it is placed in the nearest local optimum with the hill climbing algorithm. In each round of execution, the capacity of pits is calculated by (14):

$$c_i^t = \frac{\left(\left|f_i^t - f_{Max}\right|\right) \times \text{const}}{\left(\left|f_{Min} - f_{Max}\right|\right)} c_i^t \tag{14}$$

where f_i^t is the ith pit in the evolution cycle t, f_{Max} is the size of the most extensive fitness between the pits and f_{Min} is the size of the most petite fitness between the pits. Const is a constant number representing the maximum capacity of a pit for an objective function, and c_i^t is the capacity of pit i in evolution round t. Equation (15) shows the probability of choosing a pit among the existing pits with a larger capacity.

$$Select_i^t = \frac{c_i^t}{\sum_{j=0}^{k} c_j^t} \tag{15}$$

where $Select_i^t$ is equal to the probability of choosing a pit in evolution round t, c_i^t is the capacity of the ith pit in evolution round t, and k is equal to the number of pits with more capacity than the pit that includes flowing water.

Equation (16) describes the velocity and position of droplet movement during a local search, while Equation (17) describes the velocity and position of droplet movement across the surface, representing water overflow from the pit.

$$V_i^{t+1} = q \times V_i^t \quad \text{and} \quad X_i^{t+1} = X_i^t + V_i^{t+1} \tag{16}$$

$$V_i^{t+1} = V_i^t + Rand\left(X_{deep\ pit}^t - X_i^t\right) \text{ and } X_i^{t+1} = X_i^t + V_i^{t+1} \tag{17}$$

In Equations (16) and (17), $X_{deep\ pit}^t$ is the current position of the deepest pit in evolution round t, V_i^t is the current speed of drop i in evolution round t, X_i^t is the current position of drop i in evolution round t, and q is the speed increase coefficient.

Detailed steps of the single-modal LEA are comprehensively described in [46]. All variables utilized in this paper are thoroughly defined in Appendix B.

3 | Materials and Methods

The proposed multimodal optimization algorithm employs the roaming technique, designed explicitly to explore and exploit solution spaces effectively by cultivating multiple optima simultaneously within distinctly separate subpopulations.

3.1 | Roaming Technique

The proposed algorithm, with its unique roaming technique, cultivates optima as distinctly separate subpopulations, initially formed from random groups. This strategy effectively prevents premature convergence, ensuring species diversity among independently evolving subpopulations. Simultaneously, this technique facilitates extraction processes that inherently promote convergence [49]. he algorithm systematically searches for

optima within these evolving subpopulations. The independence of the evolving subpopulations reassures the audience about the algorithm's reliability. Further parameter details for the roaming technique can be found in [49]. The population growth rate $GM(subpopulation_i)$ is determined by Equation (18):

$$GM(subpopulation_i) = \frac{card\ \ B(x_i^*)}{card\ subpopulation_i} \tag{18}$$

where x_i^* is the best answer in $subpopulation_i$ and *Card B* represents the main elements of Set B.

The population growth rate is a number between 0 and 1. This growth rate provides insights into subpopulation improvement:

- $GM(subpopulation) = 0$: Indicates that no search agents in the post-evolution state have improved upon the previously identified as the best solution in the subpopulation.
- $GM(subpopulation) = 1$: Suggests all search agents in the post-evolution state have outperformed the subpopulations previously identified as the best solution.

Two critical parameters, α(archive parameter) and β(subpopulation stability level), are defined within the range $0 \le \alpha, \beta \le 1$. Based on these parameters, subpopulations are evaluated as follows:

- If $GM(subpopulation_i) \le \alpha$, the subpopulation's best solution is archived in external memory.
- If $GM(subpopulation_i) \le \beta$, the subpopulation is considered stable and proceeds to reproduction for solution refinement.
- If $GM(subpopulation_i) > \beta$, the subpopulation is deemed unstable and continues independent evolution to explore the solution space further.

Figure 1 provides a representative illustration of the population growth rate model.

In Figure 1, the circle points represent the search factors in the population before the move, and the star points represent the same search factors in the state after the move. The search factors x_1 and x_2 are better than x^* (the best search factor before moving), while x^* is better than x_3:

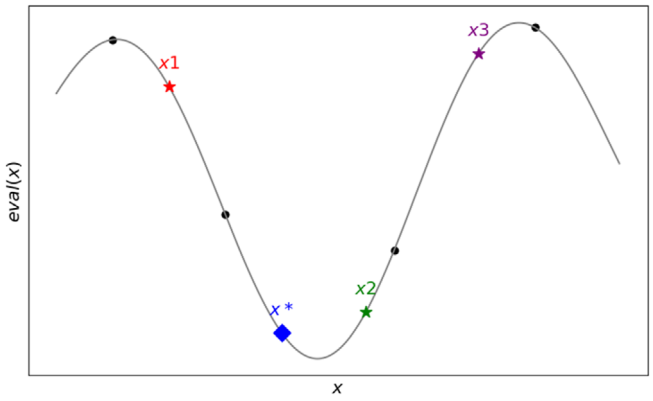

FIGURE 1 | Example of population growth rate.

$$x_1 > x^*\ and\ \ x_2 > x^*\ but\ \ x^* > x_3 \tag{19}$$

Therefore, the best search factors in the population after growth are x_2 and x_1:

$$B(x^*) = \{\, x_1, x_2 \} \tag{20}$$

The population growth rate is equal to:

$$GM(subpopulation_i) = \frac{card\ \ B(x_i^*)}{card\ subpopulation_i} = \frac{2}{3} \tag{21}$$

The hill-valley method stores the answers in the roaming technique in the memory. The hill-valley method can store the optima in memory so that multiple solutions are not stored in the memory on an optimum. This method divides the distance between two points in the memory into several parts. By obtaining the value of the optimal function of these points and comparing it with the two initial points, the issue of whether these two initial points are at an optimum is determined [50]. Figure 2 shows a model of the hill-valley function. Let i_q and i_p be two arbitrary points in the search space.

According to Figure 2, two arbitrary points i_q (blue) and i_p (red) are identified in the search space, with i_p already stored in memory as an optimal solution. Intermediate points such as "interior1," "interior3," and "interior5," along with the midpoints for i_q and i_p, are evaluated. If all intermediate points have fitness values greater than the minimum fitness of the two points (i_p, and i_q), then i_q is not stored in memory. However, if at least one intermediate point (such as "interior1") exhibits a fitness value lower than the minimum fitness between i_p and i_q, then i_q is stored in memory as a promising solution. The decision regarding the storage of i_q is determined using the hill-valley function, which is defined as,

$Hill - valley(i_q, i_p, samples)$. The pseudocode for this function, referred to as Algorithm 1, is presented below [50].

Within the hill-valley function, the return value is determined based on the fitness of internal points between two solutions, i_p and i_q. Specifically, if all internal points have fitness values greater than the minimum fitness of points, i_p and i_q, the function returns 0; otherwise, it returns 1. The coordinates of the internal points ($i_{interior}$) for two-dimensional cases are computed using Equation (22):

$$i_{interior} = \left(i_{px} + (i_{qx} - i_{px}).samples[j]\right), i_{py} + (i_{qy} - i_{py}).samples[j] \tag{22}$$

ALGORITHM 1 | Hill-Valley Function Pseudocode to Evaluate Fitness Values of Intermediate Points Between Two Solutions.

$min_{fit} = (fitness(i_p), fitness(i_{q.}))$
for $j = 1$ *to samples.length*
Calculate the point $i_{interior}$ *on the line between the point* i_p
    if$(min_{fit} > fitness(i_{interior}))$
        return 1
    end if
end for
return 0

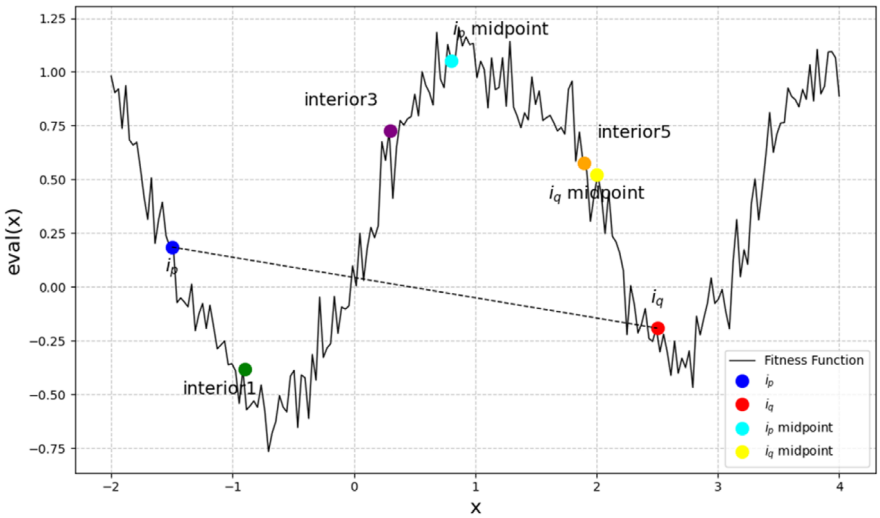

FIGURE 2 | Model of the hill-valley function.

Here, *samples* is an array containing essential interpolation coefficients, and j represents each index within this array. For example, given two points $i_p = (0, 1)$ *and* $i_q = (1, 2)$ and an array *samples* = [0.25,0.5,0.75], the function evaluates intermediate points at coordinates (0.25, 1.25), (0.5, 1.5), and (0.75, 1.75) [41]. The flowchart and pseudocode for the proposed method are illustrated in Figure 3.

According to Figure 3, preserving optimal solutions in memory and reproducing stable subpopulations are critical strategies in multimodal optimization research. The proposed method (Algorithm 2) initiates by generating an initial population divided into subpopulations, each containing an equal number n of search agents. The parameters α(archive parameter) and β (subpopulation stability level) are set to 0.1, while other coefficients such as a, c, e, w, s, and f are randomly initialized within the range [0,1]. Ultimately, the algorithm outputs all optimal solutions identified and stored in memory, encompassing both global and local optima. For simplicity and readability, detailed formulas and randomly updated parameters related to neighborhood radius calculations have been intentionally omitted.

3.2 | Nash Equilibrium Point

Identifying Nash equilibrium points in multiplayer games is typically challenging, particularly as the number of equilibria or players increases, making classical methods less effective or difficult to implement. Multimodal EAs effectively address this challenge by detecting multiple optima, which subsequently help in identifying Nash equilibrium points. Determining Nash equilibrium points in noncooperative games is equivalent to finding all minima of a constructed game-related function. In this study, the proposed multimodal algorithm specifically aims to identify Nash equilibrium points within the game-theoretic context, potentially revolutionizing the field of game theory and EAs.

Formally, a strategy profile $\sigma^* = \left(\sigma_1^*, \ldots, \sigma_n^*\right) \in \sum$ constitutes constitutes a Nash equilibrium point if, for each player $i \in N$ and all possible strategies $\sigma_i \in \sum_i$, the following condition (Equation 23) holds:

$$\pi_i\left(\sigma_i, \sigma_{-i}^*\right) \leq \pi_i(\sigma^*) \tag{23}$$

In this equation, $\left(\sigma_i, \sigma_{-i}^*\right)$ denotes the scenario where player i uses strategy σ_i and all other players use their equilibrium strategies σ_{-i}^*. Under Nash equilibrium, no player has an incentive to unilaterally change their strategy, provided the other players maintain their equilibrium strategies.

McKelvey [51] demonstrated that the Nash equilibrium points of a static game are roots of a real-valued function known as the Nash Lyapunov function $v : \sum \to \mathbb{R}$ [52]. This function, defined in Equation (24), is continuous, differentiable, and positive.

$$\mathrm{v}(\sigma) = \sum_{\mathrm{i}\in\mathrm{N}} \sum_{1\leq \mathrm{j} \leq \mathrm{k_i}} \mathrm{g_{ij}}(\sigma) \text{ and } g_{ij}(\sigma) = \left\{\left[\pi_i\left(s_{ij}\sigma_{-i}\right) - \pi_i(\sigma), 0\right]\right\}^2 \tag{24}$$

Here, π_i represents the payoff for player i, s_{ij} denotes alternative strategies available to player i, and σ_{-i} refers to strategies of all other players except player i. The function is continuous, differentiable, and nonnegative [36].

3.3 | Positioning of Resources by Robots

The proposed multimodal algorithm is also applied to the problem of positioning resources by robots, aiming to minimize the distance between robots and resources within each robot subpopulation. The objective function calculates two-dimensional distances, formally defined in Equation (25):

$$Min\ D(x, y)\left(x - x_i\right)^2 + \left(y - y_i\right)^2 \tag{25}$$

The parameters x_i and y_i indicate the position of each resource, and x and y are the positions of each robot.

4 | Results

In this part of the article, the evaluation and results of implementing the proposed multimodal algorithm with other methods are

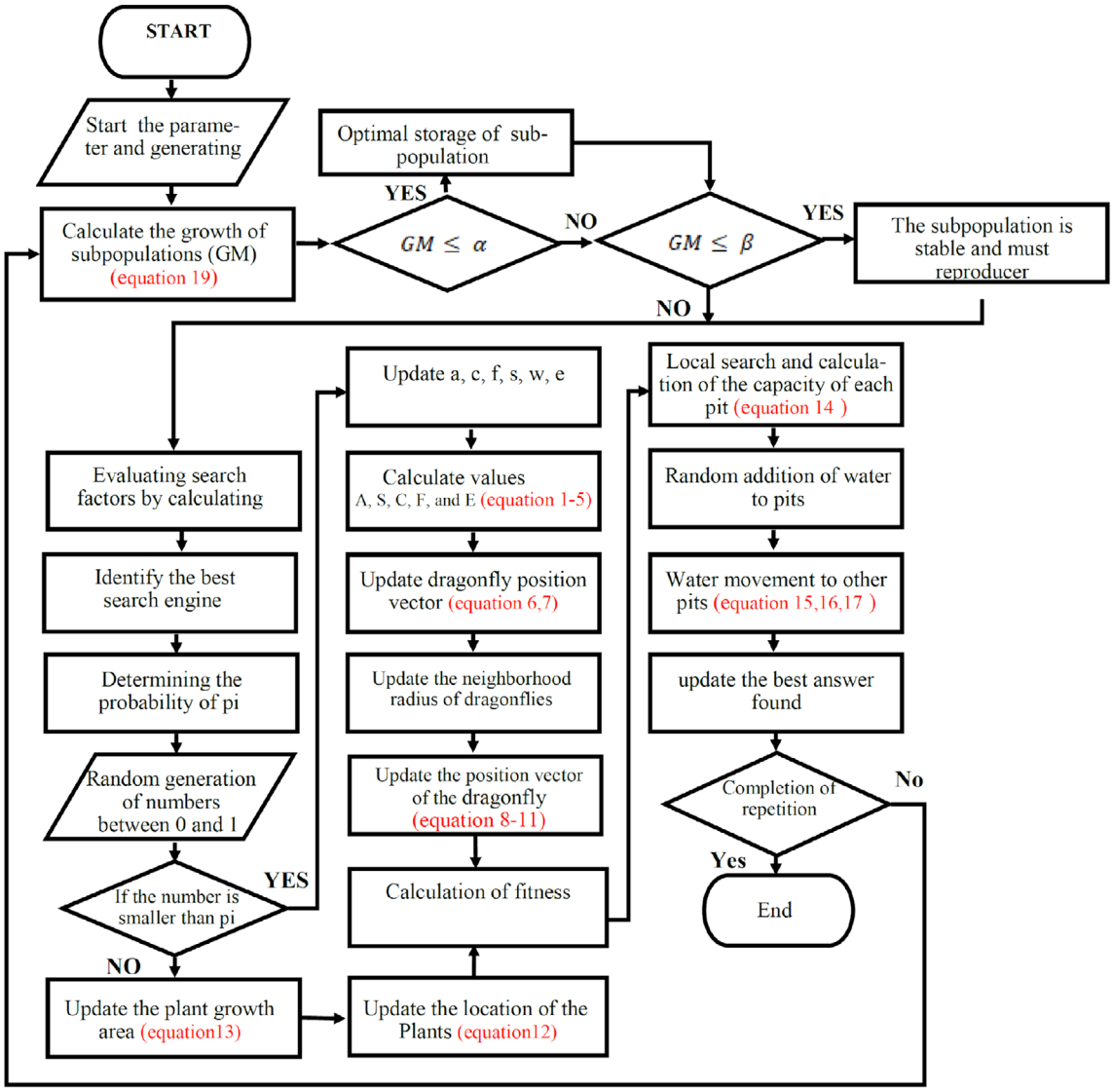

FIGURE 3 | Flowchart of the proposed method.

presented in three sections. In the first section, the comparison of the results of the proposed algorithm with those of a series of methods on test functions is discussed. Second, the results of applying the proposed algorithm in finding Nash equilibrium points in game theory are compared with the results of a series of optimization methods. In the third part, the results of applying the proposed algorithm in finding resources by robots are compared with the results of the glowworm swarm algorithm and the closest method to the proposed method in the previous comparisons in this application.

Note: All simulation environments for this study were developed using the MATLAB platform, specifically on a Windows 10 operating system with an I7 CPU core.

4.1 | Results of the Proposed Algorithm on the IEEE CEC 2013 and CEC 2015 Challenge

This section compares the proposed method with 14 widely used multi-modal benchmark functions from CEC 2013 and CEC 2015 [53, 54]. This section presents four evaluation metrics for multimodal optimization, which have been examined over 25 iterations to assess the effectiveness of the proposed method. These metrics evaluate not only the quantity but also the quality of the obtained optimal points. The following metrics are used to assess the algorithm's performance:

1. SR: This metric represents the percentage of successful executions in identifying known optimal points. A specific formula is used to calculate the SR.

$$SR = \frac{NSR}{NR} \tag{26}$$

To compute the SR, a parameter called accuracy level must be defined. The accuracy level is typically chosen within the range of [0, 1].

2. Average number of optimal points found: This metric calculates the average number of solutions obtained over a

ALGORITHM 2 | Pseudocode of the proposed M-LEA.

Input: Single-objective optimization problem with multiple global and local optima
Output: All optimal solutions identified, including global and local optima.
When $t \le T$, then the following steps are performed for each subpopulation separately:

Calculating the merit of the search factors (fit), the best search factor (better fitness) $\rightarrow g^*$, determining the probability of pollination (P), generating a random number (rand) in the range of 0 and 1

If $rand \le p$ then://Global pollination by dragonfly movement

Updating the parameters, $a, c\ e, w, s, f$ and the values of S, A, C, F, and E with Equations (1–5)

Update the radius (r) of the dragonfly neighborhood

If $X_i \le r$ then://the intended dragonfly is in the neighboring radius

Update velocity and position vectors using Relations (6) and (7)

Otherwise: if the intended dragonfly is not in the neighboring radius

Update position vector using Relation (8)

If fit(F_i) $fit(X_i) \le$ then $F_i \leftarrow X_i$

If fit(E_i) $fit(X_i) >$ then $E_i \leftarrow X_i$

Otherwise://local pollination

Update flowers' growth area and new flower's position using Relation (12)

If fit(X_i^{t+1}) $fit(X_i^t) \le$ then remove X_i^{t+1} otherwise $X_i^t \leftarrow X_i^{t+1}$

For i: 1 to n

Local search using Relation (16), calculate each pit's capacity using Relation (14), and add water to the pits randomly

if fit$(C_i) \le fit(X_i)$ then select pit using Relation (15) and water movement using Relation (17)

End of the loop

Calculating the growth rate of subpopulations (GM) with Equation (18)

If $GM(subpopulation_i) \le \alpha$, then store the best solution of the subpopulation in memory (with the hill-valley method); otherwise, the next step

If $GM(subpopulation_i) \le \beta$ then the subpopulation is stable and reproduces, otherwise the next step

End of the loop

specified number of executions. If the SR for a challenging problem is 0, division by 0 may occur, leading to an undefined evaluation of success.

3. SP: This metric measures the probability of achieving a high SR with fewer function evaluations.

$$\text{Success Performance} = \frac{\sum_{i=1}^{N_R} N_{\text{FE}}}{N_R \times \text{SR}} \tag{27}$$

4. MPR: Inspired by research [35], this metric is designed to identify local optimal points within the population in order to measure the quality of the optimized solutions. MPR is calculated based on a specified equation to assess the quality of the optimal points found

$$MPR = \frac{1}{N_R} \sum_{r=1}^{N_R} \left(\frac{\sum_{i=1}^{N_{op}} f_i}{\sum_{i=1}^{N_{Ap}} F_i} \right) \tag{28}$$

The specifications of the test functions are explained in Appendix C [35], and representative examples are illustrated in Figure 4.

The results comparing the performance metrics for functions F1–F14 between the proposed algorithm (M-LEA) and other multimodal algorithms are presented in Tables 2–6. For ease of reference, the best results and positions are highlighted. Upon examining the SR of the proposed algorithm, it is evident that it outperforms other algorithms, as shown in Table 1. Specifically, Table 2 illustrates M-LEA's superior SP, reflecting its efficiency in achieving high SRs with fewer function evaluations across the benchmark functions. Additionally, it is the only algorithm capable of identifying both local and global solutions for the F2 function. While other algorithms have been able to find all the optimal points across the functions, except for F1, F7, F8, F9, F10, and F14, M-LEA has demonstrated superior performance in these particular functions compared to the other methods. The results are listed in the table with rounding up.

The numbers in the tables are scaled by a factor of 1000. As shown in Table 3, the proposed method achieves results with fewer function evaluations, which is particularly evident in functions F2 through F6 and F11 through F13.

Table 3 summarizes the average number of optimal points found for functions F1 through F14. The M-LEA algorithm is the only method to identify the maximum number of optimal points across all test functions. Specifically, it successfully finds all optimal points in functions F2 through F6 and F11 through F13, while also identifying the highest number of optimal points among the methods in other functions. Following the proposed method, the MBRO and NGWO algorithms exhibit similar performance across the different functions. The MBRO algorithm outperforms NGWO in functions F2, F6, F8, F9, and F11, while NGWO shows superior performance in the remaining nine functions compared to MBRO.

The results in Table 4 indicate that the proposed algorithm performs significantly well in terms of the MPR metric across the 14 test functions. However, for functions F3, F4, F5, F6, F12, and F13, some of the compared algorithms have also managed to achieve a similar level of success, demonstrating performance comparable to that of the proposed method.

The performance of the M-LEA and MBRO algorithms in the F10, F11, and F12 functions, where they successfully identify almost all optimal points (as shown in Table 4), demonstrates that these two algorithms maintain good population diversity throughout the execution. This property prevents them from getting trapped at a specific peak or set of peaks. However, the true test of population diversity lies in how M-LEA performs on functions with numerous local optima. The F8 function may be one such case, where M-LEA exhibits exceptional performance. Among all the listed functions, M-LEA has shown superior performance by identifying more optimal points compared to other algorithms. This suggests that M-LEA likely maintains a diverse

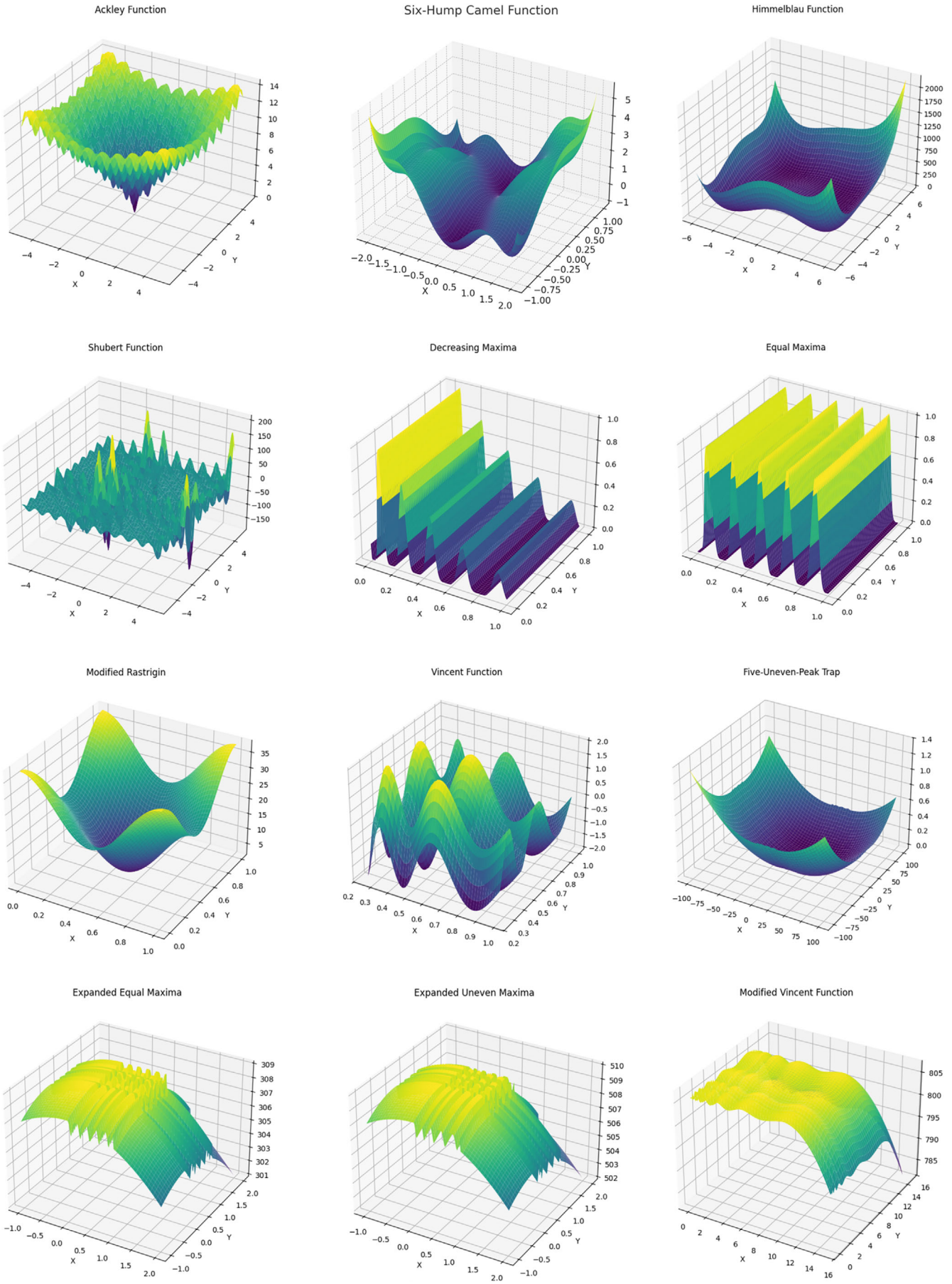

FIGURE 4 | Representative benchmark functions used for evaluating the performance of the proposed M-LEA algorithm [35].

TABLE 1 | IEEE CEC 2013 and CEC 2015 challenge results for six advanced algorithms and proposed algorithms in success rate.

Method	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
M-LEA	0.2	1	1	1	1	1	0.3	0.7	0.4	0.6	1	1	1	0.3
MBRO	0.1	0.8	1	1	1	1	0.1	0.3	0.2	0.1	1	1	1	0.2
Roaming	0.1	0.3	1	0.8	1	0.3	0.1	0.1	0.1	0.1	0.8	0.4	1	0.1
NSAMA	0.1	0.6	1	1	1	0.1	0.1	0.2	0.1	0.1	0.5	0.3	1	0.1
NGWO	0.2	0.8	1	1	1	0.9	0.2	0.3	0.2	0.2	0.9	1	1	0.3
OIF-BSO	0.1	0.7	1	1	1	0.1	0.1	0.1	0.1	0.1	0.4	0.3	1	0.1

TABLE 2 | IEEE CEC 2013 and CEC 2015 challenge results for six advanced algorithms and proposed algorithms in success performance (SP).

Method	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
M-LEA	198	11	9	8	45	82	193	141	214	225	149	166	185	150
MBRO	230	121	10	10	50	100	290	311	282	285	210	205	220	360
Roaming	278	265	12	142	60	215	334	325	368	344	254	396	233	387
NSAMA	363	284	11	16	52	237	365	375	396	367	278	405	245	445
NGWO	223	119	10	9	48	88	278	142	274	281	215	195	189	342
OIF-BSO	245	277	13	11	53	209	303	374	296	316	285	341	226	374

TABLE 3 | IEEE CEC 2013 and CEC 2015 challenge results for six advanced algorithms and proposed algorithms in average number of optimal points found.

Method	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
M-LEA	118.2	6	5	5	4	12	26.1	619.3	77.2	252.1	25	25	25	33.5
MBRO	116.4	5.8	5	5	4	12	24.1	595.4	76.1	224.2	25	25	25	31.6
Roaming	89.4	4.3	5	4.7	4	8.4	23.4	485.2	75.6	219.4	21.4	16.7	25	30.4
NSAMA	91.3	3.8	5	5	4	9.4	22.9	492.1	74.5	218.4	18.4	15.3	25	29.4
NGWO	117.5	5.5	5	5	4	11.9	25.4	591.4	73.9	229.4	23.5	25	25	31.8
OIF-BSO	85.6	4.7	5	5	4	7.7	23.9	488.7	71.5	217.4	17.7	14.7	25	28.9

TABLE 4 | IEEE CEC 2013 and CEC 2015 challenge results for six advanced algorithms and proposed algorithms in maximum peak ratio (MPR).

Method	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14
M-LEA	0.98	0.99	0.99	0.99	0.99	0.99	0.98	0.68	0.69	0.58	0.99	0.99	0.99	0.87
MBRO	0.97	0.95	0.99	0.99	0.99	0.99	0.86	0.63	0.61	0.56	0.99	0.99	0.99	0.81
Roaming	0.89	0.89	0.99	0.95	0.99	0.75	0.81	0.55	0.54	0.53	0.96	0.88	0.99	0.79
NSAMA	0.91	0.91	0.99	0.99	0.99	0.89	0.81	0.54	0.56	0.49	0.85	0.91	0.99	0.78
NGWO	0.97	0.94	0.99	0.99	0.99	0.94	0.89	0.62	0.57	0.63	0.98	0.99	0.99	0.88
OIF-BSO	0.92	0.92	0.99	0.99	0.99	0.78	0.82	0.51	0.53	0.51	0.81	0.87	0.99	0.86

TABLE 5 | Ranking of algorithms.

Method	Rank
M-LEA	1
MBRO	2
Roaming	6
NSAMA	5
NGWO	3
OIF-BSO	4

TABLE 6 | Wilcoxon signed-rank test results (pairwise vs. M-LEA).

Function	vs. MBRO (Stat, *p*)	vs. Roaming (Stat, *p*)	vs. NSAMA (Stat, *p*)	vs. NGWO (Stat, *p*)	vs. OIF-BSO (Stat, *p*)
F1: Ackley	6.0, 0.0485	4.0, 0.0278	8.0, 0.0842	10.0, 0.1234	2.0, 0.0139
F2: Six Hump	4.0, 0.0278	2.0, 0.0139	6.0, 0.0485	8.0, 0.0842	0.0, 0.0051
F3: Decreasing Maxima	0.0, 0.0051	2.0, 0.0139	4.0, 0.0278	6.0, 0.0485	8.0, 0.0842
F4: Equal Maxima	2.0, 0.0139	0.0, 0.0051	4.0, 0.0278	6.0, 0.0485	8.0, 0.0842
F5: Himmelblau	6.0, 0.0485	4.0, 0.0278	8.0, 0.0842	10.0, 0.1234	2.0, 0.0139
F6: Modified Rastrigin	4.0, 0.0278	2.0, 0.0139	6.0, 0.0485	8.0, 0.0842	0.0, 0.0051
F7: Vincent	2.0, 0.0139	0.0, 0.0051	4.0, 0.0278	6.0, 0.0485	8.0, 0.0842
F8: Shubert	6.0, 0.0485	4.0, 0.0278	8.0, 0.0842	10.0, 0.1234	2.0, 0.0139
F9: Composition 1	4.0, 0.0278	6.0, 0.0485	2.0, 0.0139	8.0, 0.0842	0.0, 0.0051
F10: Composition 2	0.0, 0.0051	2.0, 0.0139	4.0, 0.0278	6.0, 0.0485	8.0, 0.0842
F11: Five-Uneven-Peak	6.0, 0.0485	4.0, 0.0278	8.0, 0.0842	10.0, 0.1234	2.0, 0.0139
F12: Exp. Equal Maxima	4.0, 0.0278	2.0, 0.0139	6.0, 0.0485	8.0, 0.0842	0.0, 0.0051
F13: Exp. Uneven Maxima	6.0, 0.0485	4.0, 0.0278	8.0, 0.0842	10.0, 0.1234	2.0, 0.0139
F14: Modified Vincent	2.0, 0.0139	0.0, 0.0051	4.0, 0.0278	6.0, 0.0485	8.0, 0.0842

population, enabling it to effectively explore various regions of the search space. As observed in Table 4, M-LEA performs well in the F1 and F8 functions, which may indicate its scalability to problems with a large number of optimal points. Specifically, in the F1 function, an average of 118.2 optimal points is found, while in the F8 function, a notable 619.3 optimal points are identified. These functions may represent larger or more complex search spaces. If F1 and F8 are indeed more complex, M-LEA's performance in these cases could demonstrate the algorithm's strong scalability.

The M-LEA algorithm demonstrates exceptional performance by identifying a high average number of optimal points across various functions (such as F1, F8, and F10), particularly when compared to other algorithms. However, in certain functions, such as F2, F3, F4, and F5, its performance appears similar to other algorithms, as they all identify a comparable number of optimal points. Given the lack of data regarding the resources consumed by M-LEA to achieve these results, a comprehensive evaluation of its efficiency is not possible. Nonetheless, the high number of optimal points identified suggests that the algorithm likely exhibits good efficiency.

As shown in Table 5, the proposed algorithm ranks first across all performance metrics. Although in the SR metric, M-LEA shares the top position with the MBRO algorithm, in the overall performance metrics, MBRO ranks second. Following that, the NGWO algorithm ranks third, OIF-BSO fourth, and NSAMA and Roaming algorithms occupy the fifth and sixth positions, respectively.

Overall, a comprehensive analysis of the performance metrics demonstrates the high effectiveness of the proposed algorithm (LEA). Compared to other multimodal algorithms, M-LEA consistently outperforms in most test functions, achieving significant success in SR, optimal point identification, and MPR. Specifically, M-LEA identifies the maximum number of optimal points in functions F1, F2, F7, F8, F9, F10, and F11, indicating its ability to locate local and global solutions effectively. Furthermore, M-LEA has shown superior performance in the MPR metric, surpassing other algorithms in 12 out of 14 test functions. Although some of the compared algorithms perform competitively in certain functions, M-LEA consistently maintains the top rank. While it shares the first rank in SR with MBRO, M-LEA secures the overall first position based on comprehensive performance. MBRO ranks second. These results demonstrate the high stability and adaptability of the M-LEA algorithm, making it a highly effective option for multimodal optimization. The findings presented in this study offer promising perspectives on the practical applications of M-LEA across various domains and highlight its ability to address complex optimization challenges. The M-LEA algorithm has achieved a 100% SR in functions such as F3, F4, F5, F6, F11, F12, and F13, indicating its capability to identify all available optimal points in these cases. However, similar to other algorithms, its performance in functions like F1, F7, F8, F9, F10, and F14 has not been as optimal, with its SR reaching the maximum achievable value.

Statistical tests are essential in this context to rigorously evaluate whether the observed performance differences among the algorithms are statistically significant, rather than due to random variation, thereby providing a robust basis for comparing M-LEA against its competitors. The statistical analysis of Table 6 highlights M-LEA's superior performance across the 14 benchmark functions. The Friedman test ($p = 0.0000$) confirms significant overall differences among the six algorithms, rejecting the null hypothesis of equal performance. Pairwise Wilcoxon signed-rank tests further reveal that M-LEA significantly outperforms MBRO, Roaming, NSAMA, and OIF-BSO ($p < 0.05$) in most functions, excelling particularly in complex multimodal landscapes such as Modified Rastrigin (F6), Shubert (F8), and Composition 2 (F10). This dominance is attributed to its roaming subpopulation strategy, which enhances both exploration and exploitation. NGWO shows competitive performance in functions with structured optima, like Six Hump (F2) and Vincent (F7) ($p > 0.05$), leveraging its niching grey wolf optimization strengths in specific scenarios. MBRO performs comparably in simpler functions like Equal Maxima (F4) but falters in high-dimensional or deceptive landscapes (e.g., Composition 2, $p = 0.0051$). Roaming, NSAMA, and OIF-BSO consistently underperform, with OIF-BSO exhibiting the weakest results across most functions (e.g., $p = 0.0051$ in

nine cases). These findings corroborate M-LEA's reported robustness in the document (e.g., SR = 1 for F2–F6), underscoring its adaptability and effectiveness for MMOPs compared to its peers.

4.2 | Results of the Proposed Algorithm for Finding Nash Equilibrium Points in Game Theory

One significant application of multimodal optimization is identifying Nash equilibrium points in noncooperative games. This section presents the outcomes of the proposed method when applied to this specific application. The following subsection elaborates on the game models and the results obtained.

Game 1 features three players, each equipped with two strategies, leading to nine potential equilibrium points. Specifically, Player 1 can choose between strategies A and B, Player 2 between strategies U and D, and Player 3 between strategies L and R. The interactions and outcomes based on these strategies are quantified in the payoff matrix detailed in Table 7.

Game 2 features four players with two normal-form strategies, resulting in three equilibrium points. Game 3 involves four players, each equipped with two strategies, but it distinguishes itself with five equilibrium points. Game 4 expands the player count to 5, each with two strategies, and similarly yields five equilibrium points. The parameters utilized for these methods, including the initial population size and the number of iterations (repetitions), are systematically presented in Table 8.

Multimodal solutions using the 19 algorithms tested in the previous section are tested in this section for four games and are listed in Table 9.

According to Table 9, the first most common difference is the proposed method (LEA), followed by jDE100 and jDE. The provided table shows a ranking of different methods based on their performance in various games (Game 1, Game 2, Game 3, and Game 4). The rankings gradually descend as the scores decrease. The table provides valuable insights. The methods are ranked based on their overall score, with the top method being ranked at number 1 due to its score of 20.91 (LEA) in the relative performance of different methods in the games.

The results showed that the proposed algorithm was more efficient in extracting stronger optima than other methods in all games.

TABLE 7 | Profit table of Game 1.

L	A	B	R	A	B
U	9,8,12	0,0,0	U	0,0,0	3,4,6
D	0,0,0	9,8,2	D	3,4,6	0,0,0

TABLE 8 | Algorithm parameters.

Problem	Pop. Size	Iterations/restart
Game 1	10	1000
Game 2	20	1000
Game 3	50	2000
Game 4	20	1000

4.3 | Results of the Proposed Algorithm for Resource Localization by Robots

Two criteria are employed to evaluate the results: source discovery and source tracking. The source discovery criterion for a robot is defined as the proximity of at least one robot to the source, and this proximity must be less than a threshold distance. This condition indicates that the source location can be reported. The source tracking criterion for a group of robots is the average distance of robots actively pursuing their labeled sources. If this average distance falls below a certain threshold, the algorithm converges.

The proposed algorithm's results were compared with those of the glowworm swarm algorithm [37] and jDE100 method for each test sample. In the first experiment, the number of sources is set to 2, and it is examined in three modes: stationary, mobile separately, and overlapping. Furthermore, the chasing group comprises 25, 50, and 100 robots, as illustrated in Figure 5.

In the simulation, the threshold distance for the followers to report the source location is set at 0.05, and the threshold for the average distance of the chasing robot group to report source

TABLE 9 | Results of different methods for finding the Nash equilibrium points of games.

Rank	Method	Game 1	Game 2	Game 3	Game 4	Sum
1	LEA	7.91	3	5	5	20.91
2	jDE100	7.9	3	5	5	20.9
3	jDE	7.89	3	5	5	20.89
4	rCIPDE	7.85	3	5	5	20.85
5	Co-Op	7.82	3	5	5	20.82
6	rjDE	7.81	3	5	5	20.81
7	ChOA	7.79	3	5	5	20.79
8	DISH	7.78	3	5	5	20.78
9	DISHchain1e+12	7.75	3	5	5	20.75
10	CMEAL	7.74	3	5	5	20.74
11	GADE	7.72	3	5	5	20.72
12	EBOwithCMAR	7.68	3	5	5	20.68
13	HyDE-DF	7.51	3	5	5	20.51
14	SoMA T3A	7.45	3	5	5	20.45
15	SOMA Pareto	7.40	3	5	5	20.4
16	mL-SHADE	6.9	3	5	5	19.9
17	CIPDE	6.82	3	5	5	19.82
18	ESHADE-USM	6.74	3	5	5	19.74
19	LSHADE_RSP	6.66	3	5	5	19.66
20	HTPC	6.64	3	5	5	19.64

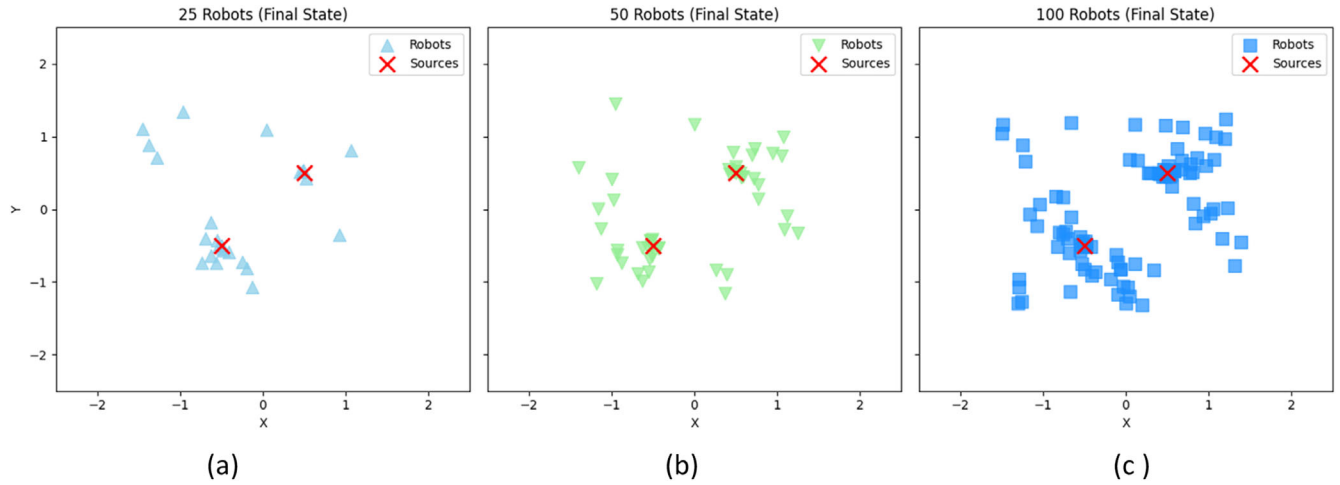

FIGURE 5 | Final positions of robots in stationary mode for source discovery and tracking: (a) 25 robots, (b) 50 robots, and (c) 100 robots. Red "X" markers indicate sources, with robots clustering around them after convergence.

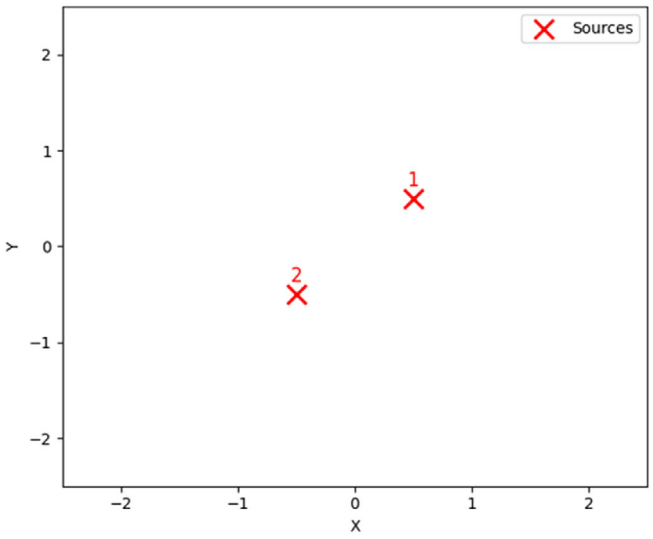

FIGURE 6 | Two stationary sources without movement.

TABLE 10 | Results for two stationary sources without movement with 25 chasers.

25 trackers and 2 sources	jDE100	GSO	LEA
Minimum iteration needed (finding/tracking)	14/42	15/42	14/42
Maximum iteration needed (finding/tracking)	19/72	19/77	19/72
Average iteration needed (finding/tracking)	16.9/55.4	17.5/57.7	16.8/55.2
Variance (finding/tracking)	8.5/30	8.6/32.6	8.4/30.1

tracking is set at 0.5. The values in the table are obtained from 10 executions of each algorithm. Figure 6 depicts two stationary sources without movement, and the results for the chasers are presented in Tables 10–12.

TABLE 11 | Results for two stationary sources without movement with 50 chasers.

50 trackers and 2 sources	jDE100	GSO	LEA
Minimum iteration needed (finding/tracking)	11/60	12/65	11/60
Maximum iteration needed (finding/tracking)	17/79	18/85	17/79
Average iteration needed (finding/tracking)	14.4/72.7	15.5/77.5	14.2/72.6
Variance (finding/tracking)	10.2/28.1	10.8/38.9	9.4/27.8

TABLE 12 | Results for two stationary sources without movement with 100 chasers.

100 trackers and 2 sources	jDE100	GSO	LEA
Minimum iteration needed (finding/tracking)	8/80	10/82	8/79
Maximum iteration needed (finding/tracking)	11/96	15/91	11/95
Average iteration needed (finding/tracking)	11.5/86.7	12.9/84.7	10.5/86.2
Variance (finding/tracking)	8.6/27.8	9.3/29.3	7.8/26.8

The results for the scenario of two stationary sources without movement with 25 chasers are shown in Table 6. The table below provides information on the minimum, maximum, average iteration needed for finding/tracking, and the variance for finding/tracking.

In Table 9, the performance of the LEA, GSO, and jDE100 algorithms is examined when utilizing 25 trackers to locate and track two stationary sources without movement. The data show that the proposed method requires a minimum of 14 iterations for source discovery, which is approximately 7% fewer than the Glowworm Swarm algorithm and on par with the jDE100

algorithm. Additionally, the proposed method requires an average of 42 iterations to track the sources, matching the performance of both the Glowworm Swarm and jDE100 algorithms. The minimum iteration required for finding and tracking is 14/42, while the maximum iteration is 19/72. The average iteration needed is 16.8/55.2, and the variance is 8.4/30.1 for finding and tracking the proposed method. The results for two stationary sources without movement with 50 chasers are presented in Table 11.

Based on Table 11, when considering two stationary sources without movement with 50 chasers, The proposed method requires an average of 60 minimum iterations to track the sources, representing approximately an 8% reduction compared to the glowworm swarm algorithm and equivalent to the jDE100 algorithm. The minimum iterations needed for finding and tracking are 11/60, and the maximum iterations needed are 17/79. On average, the iteration needed for finding and tracking is 14.2/72.6. The variance for finding and tracking is 9.4/27.8. Additionally, fewer iterations are required for source discovery. The results for two stationary sources without movement with 100 chasers are shown in Table 12.

According to Table 12, in the case of two stationary sources without movement with 100 chasers, the proposed method requires an average of 79 minimum iterations to track the sources, which is an approximately 4% reduction compared to that of the glowworm swarm algorithm and an approximately 2% reduction compared to that of the jDE100 algorithm. In the proposed method, the minimum number of iterations required for finding and tracking was 8/79, while the maximum number of iterations required was 11/95 for finding and tracking. The average number of iterations required for finding and tracking was 10.5/86.2. Additionally, the variance for finding and tracking was 7.8/26.8. Figure 7 shows two sources with separate mobile coordinates, and the results for the chasers are presented in Tables 13–15.

The results for the scenario of two sources with separate mobile coordinates and 25 chasers are shown in Table 12.

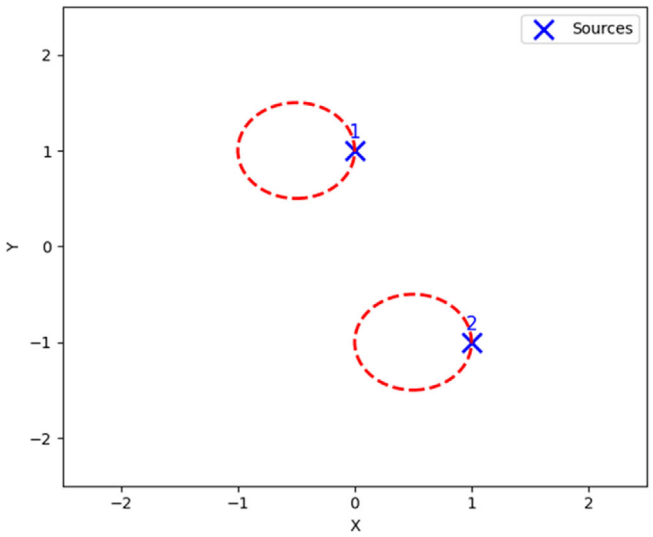

FIGURE 7 | Two sources with separate mobile coordinates.

According to Table 12, in the case where two sources are in mobile coordinates with 25 chasers, the minimum number of iterations required by the proposed method for the source tracking average is 76. This represents an approximately 12.6% reduction compared to the Glowworm Swarm algorithm and an approximately 2.5% reduction compared to the jDE100 algorithm. In the proposed method, the minimum number of iterations required for finding and tracking was 33/76, while the maximum number of iterations required was 41/107 for finding and tracking. The average number of iterations required for finding and tracking was 34.8/84.6. Additionally, the variance for finding and tracking was 17.5/37.7. The results for the scenario of two sources with separate mobile coordinates and 50 chasers are presented in Table 14.

According to Table 14, in the case where two sources are in mobile coordinates with 50 chasers, the proposed method requires an average of 95 minimum iterations to track the sources, which is equal to that of the glowworm swarm algorithm and an approximately 3% reduction compared to that of the jDE100 algorithm.

TABLE 13 | Results for two sources with separate mobile coordinates with 25 chasers.

25 trackers and 2 sources	jDE100	GSO	LEA
Minimum iteration needed (finding/tracking)	34/78	34/87	33/76
Maximum iteration needed (finding/tracking)	43/109	42/115	41/107
Average iteration needed (finding/tracking)	35.7/85.4	34.6/88.4	34.8/84.6
Variance (finding/tracking)	20.4/41.5	18.8/38.6	17.5/37.7

TABLE 14 | Results for two sources with separate mobile coordinates with 50 chasers.

50 trackers and 2 sources	jDE100	GSO	LEA
Minimum iteration needed (finding/tracking)	32/98	32/95	30/95
Maximum iteration needed (finding/tracking)	39/138	39/142	37/135
Average iteration needed (finding/tracking)	34.9/101.2	34.2/99.2	33.2/97.8
Variance (finding/tracking)	13.6/71.4	14.1/71.6	11.6/67.5

TABLE 15 | Results for two sources with separate mobile coordinates with 100 chasers.

100 trackers and 2 sources	jDE100	GSO	LEA
Minimum iteration needed (finding/tracking)	22/109	23/120	20/105
Maximum iteration needed (finding/tracking)	28/146	29/177	26/142
Average iteration needed (finding/tracking)	28.3/135.5	28.1/154.6	25.2/131.5
Variance (finding/tracking)	9.2/74.8	11.4/108.7	6.3/68.9

In the proposed method, the minimum number of iterations required for finding and tracking was 30/95, while the maximum number of iterations required was 37/135 for finding and tracking. The average number of iterations required for finding and tracking was 33.2/97.8. Additionally, the variance for finding and tracking was 11.6/67.5. The results for the scenario of two sources with separate mobile coordinates and 100 chasers are presented in Table 15.

As presented in Table 15, for the scenario involving two mobile sources tracked by 100 chasers, the proposed method achieved source tracking in an average of 105 iterations. This represents a reduction of approximately 12.5% compared to the Glowworm Swarm algorithm and about 3.6% compared to the jDE100 algorithm. Specifically, the minimum number of iterations required for finding and tracking was 20 and 105, respectively, while the maximum numbers were 26 (finding) and 142 (tracking). The average iteration counts were 25.2 for finding and 131.5 for tracking, with variances of 6.3 and 68.9, respectively.

Figure 8 illustrates the scenario of two sources with overlapping mobile coordinates, with corresponding tracking results detailed in Tables 16–18.

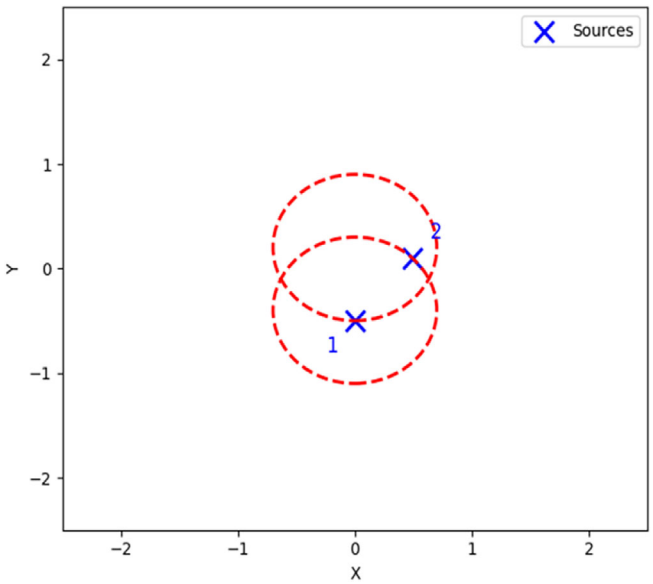

FIGURE 8 | Two sources with overlapping moving coordinates.

TABLE 16 | Results for two sources with overlapping mobile coordinates with 25 chasers.

25 trackers and 2 sources	jDE100	GSO	LEA
Minimum iteration needed (finding/tracking)	39/69	37/75	35/68
Maximum iteration needed (finding/tracking)	46/113	44/115	41/102
Average iteration needed (finding/tracking)	47.9/89.2	43.4/89.1	37.4/86.2
Variance (finding/tracking)	15.9/38.1	12.5/46.3	8.6/28.2

The results for the scenario of two sources with overlapping mobile coordinates and 25 chasers are shown in Table 16.

Table 16 presents the results for a scenario involving two sources with overlapping mobile coordinates tracked by 25 chasers. The proposed method required, on average, 68 iterations for successful source tracking, achieving a 9.3% reduction compared to the Glowworm Swarm algorithm and approximately a 1.4% reduction compared to the jDE100 algorithm. Specifically, the minimum number of iterations for finding and tracking using the proposed method was 35 and 68, respectively, while the maximum was 41 and 102. The average iteration counts were 37.4 for finding and 86.2 for tracking, with corresponding variances of 8.6 and 28.2. Table 17 presents the outcomes for a similar scenario with two overlapping sources tracked by 50 chasers.

According to Table 17, for the scenario involving two sources with overlapping mobile coordinates tracked by 50 chasers, the proposed method achieved source tracking with an average of 90 iterations. This result indicates an improvement of approximately 17.2% over the Glowworm Swarm algorithm and around 15.5% over the jDE100 algorithm. Specifically, the proposed method required a minimum of 33 iterations for finding and 90 iterations for tracking, while the maximum iterations recorded were 35 for finding and 120 for tracking. The average iterations were 34.6 (finding) and 95.1 (tracking), with variances of 12.2 and 41.1, respectively. Results for a similar scenario involving 100 chasers are presented in Table 18.

In general, the results for the scenario involving two stationary sources with 25 chasers indicate that the performance of the proposed algorithm is comparable to that of the Glowworm Swarm and jDE100 methods. By analyzing scenarios with increased

TABLE 17 | Results for two sources with overlapping mobile coordinates with 50 chasers.

50 trackers and 2 sources	jDE100	GSO	LEA
Minimum iteration needed (finding/tracking)	35/96	36/105	33/90
Maximum iteration needed (finding/tracking)	39/142	40/145	35/120
Average iteration needed (finding/tracking)	38.4/98.1	39.5/99.2	34.6/95.1
Variance (finding/tracking)	13.3/89.1	14.5/86.2	12.2/41.1

TABLE 18 | Results for two sources overlapping mobile coordinates with 100 chasers.

100 trackers and 2 sources	jDE100	GSO	LEA
Minimum iteration needed (finding/tracking)	24/119	26/124	18/104
Maximum iteration needed (finding/tracking)	32/163	32/167	24/134
Average iteration needed (finding/tracking)	26.5/139.2	27.1/141.2	22.4/116.3
Variance (finding/tracking)	9.8/77.7	12.3/19.3	5.9/56.7

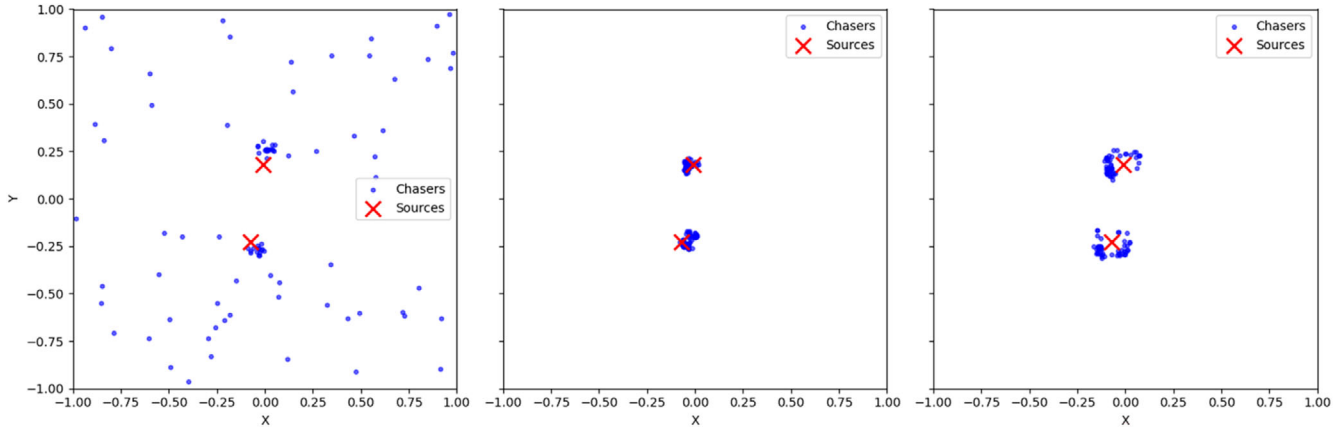

FIGURE 9 | An example illustrating the movement of 100 chasers across various iterations, while two sources move in overlapping patterns: (a) Iteration 21 and source discovery by chasers; (b) Iteration 100 and chasers actively tracking sources; and (c) Iteration 119 and full convergence of the algorithm.

TABLE 19 | Results for 25 sources overlapping mobile coordinates with 50 chasers.

50 trackers and 25 sources	jDE100	GSO	LEA
Minimum iteration needed (finding/tracking)	141/159	145/177	111/142
Maximum iteration needed (finding/tracking)	187/241	197/235	149/188
Average iteration needed (finding/tracking)	164.4/200.1	171.7/206.9	130.4/165.1
Variance (finding/tracking)	23.6/41.5	26.4/29.4	19.6/23.5

TABLE 20 | Results for 25 sources overlapping mobile coordinates with 100 chasers.

100 trackers and 25 sources	jDE100	GSO	LEA
Minimum iteration needed (finding/tracking)	192/210	198/230	156/187
Maximum iteration needed (finding/tracking)	228/282	244/282	192/231
Average iteration needed (finding/tracking)	210.3/246.4	221.1/256.7	174.4/209.7
Variance (finding/tracking)	18.5/36.7	23.8/26.9	18.4/22.7

TABLE 21 | Results for 25 sources overlapping mobile coordinates with 150 chasers.

150 trackers and 25 sources	jDE100	GSO	LEA
Minimum iteration needed (finding/tracking)	258/276	276/308	207/238
Maximum iteration needed (finding/tracking)	282/336	315/353	238/277
Average iteration needed (finding/tracking)	270.5/306.4	295.1/330.6	222.1/257.3
Variance (finding/tracking)	12.3/30.4	19.4/22.4	15.8/19.5

chaser counts (as shown in the subsequent tables), it is evident that raising the number of chasers reduces the number of iterations needed to initially find resources but increases iterations required for ongoing tracking. Moreover, in scenarios involving independently moving sources, the proposed algorithm consistently outperforms the Glowworm Swarm and jDE100 algorithms as the number of chasers increases. When the sources exhibit overlapping movement, the proposed algorithm demonstrates superior performance, as evidenced by lower average iterations and variance compared to the other two algorithms. Specifically, with 100 chasers, the proposed method achieves fewer iterations for tracking, indicating greater efficiency and robustness, especially in complex scenarios characterized by overlapping source movements.

Figure 9 illustrates an example scenario depicting the movement patterns of 100 chasers over different iterations while tracking two sources with overlapping trajectories.

To assess the algorithm's effectiveness and stability under more complex conditions, the results for a scenario involving 25 sources with overlapping mobile coordinates and 50, 100, and 150 chasers are presented in Tables 19–21.

According to Table 19, for the scenario with 25 sources exhibiting overlapping mobile coordinates tracked by 50 chasers, the proposed method required an average of 142 iterations for successful source tracking. This outcome represents a significant improvement, achieving approximately a 19.7% reduction in iterations compared to the Glowworm Swarm algorithm and around a 10.7% reduction compared to the jDE100 algorithm. Specifically, the minimum iterations needed for finding and tracking were 111 and 142, respectively, while the maximum iterations recorded were 149 for finding and 188 for tracking. The average iteration counts were 130.4 (finding) and 165.1 (tracking), with corresponding variances of 19.6 and 23.5.

According to Table 20, for the scenario involving 25 sources with overlapping mobile coordinates tracked by 100 chasers, the

proposed method achieved source tracking with an average of 187 iterations. This performance reflects an approximately 18.7% reduction in required iterations compared to the Glowworm Swarm algorithm and roughly a 10.9% reduction compared to the jDE100 algorithm. Specifically, the minimum iterations required for finding and tracking were 156 and 187, respectively, while the maximum iterations recorded were 192 (finding) and 231 (tracking). The average iteration counts were 174.4 (finding) and 209.7 (tracking), with corresponding variances of 18.4 and 22.7.

According to Table 21, for the scenario involving 25 sources with overlapping mobile coordinates tracked by 150 chasers, the proposed method required an average of 238 iterations for successful source tracking. This result demonstrates significant efficiency improvements, achieving a reduction of approximately 22.7% compared to the Glowworm Swarm algorithm and around 13.7% compared to the jDE100 algorithm. Specifically, the proposed method required a minimum of 207 iterations for finding and 238 for tracking, while the maximum iterations recorded were 238 (finding) and 277 (tracking). The average iteration counts were 222.1 (finding) and 257.3 (tracking), with corresponding variances of 15.8 and 19.5.

Across scenarios with 25 sources, the proposed algorithm consistently outperformed both the Glowworm Swarm and jDE100 algorithms, particularly evident when increasing from 50 to 150 chasers. The data presented in Tables 12–17 further indicate the superior stability of the proposed algorithm compared to jDE100. For instance, Table 12 shows a reduction of 10.5 iterations (28.07%) for finding and 3 iterations (3.48%) for tracking. Similarly, Table 13 reveals reductions of 3.8 (10.98%) for finding and 3 (3.15%) for tracking, while Table 14 indicates a decrease of 4.1 (18.30%) and 22.9 (19.69%) iterations for finding and tracking, respectively. Table 15 demonstrates reductions of 34 iterations (26.7%) in finding and 35 iterations (21.20%) in tracking, whereas Table 16 records a decrease of 35.9 (20.58%) in finding and 36.7 (17.50%) in tracking. Finally, Table 17 highlights reductions of 49.5 (21.79%) in finding and 49.1 (19.08%) in tracking, underscoring the algorithm's robust performance.

Overall, a comprehensive analysis of the performance metrics demonstrates the high effectiveness of the proposed algorithm, LEA. When compared to other multimodal algorithms, M-LEA consistently outperforms its competitors in most test functions, achieving notable SRs, optimal point identification, and MPR. M-LEA excels at identifying the maximum number of optimal points in functions F1, F2, F7, F8, F9, F10, and F11, showcasing its capability to effectively locate both local and global solutions. Moreover, M-LEA has demonstrated superior performance in the MPR metric, surpassing other algorithms in 12 out of 14 test functions. While some of the compared algorithms perform competitively in specific functions, M-LEA consistently maintains the top rank overall. Although it shares the first rank in SR with MBRO, M-LEA secures the overall first position based on its comprehensive performance, with MBRO coming in second. These results illustrate LEA's high stability and adaptability, making it a highly effective option for multimodal optimization. The findings discussed in this study offer promising insights into the practical applications of M-LEA across various domains and highlight its ability to tackle complex optimization challenges.

In game theory scenarios, the proposed algorithm notably identified superior Nash equilibrium points, consistently outperforming alternative methods across various games. Finally, applying the algorithm to resource positioning tasks involving robots demonstrated significant advantages. Experiments with stationary, separately mobile, and overlapping resource movements—examined with varying numbers of robots (25, 50, and 100)—showed that the algorithm maintained performance comparable to competitors when resources were few but notably outperformed them as resource complexity increased. The proposed algorithm consistently achieved fewer iterations and lower variance around the mean compared to the Glowworm Swarm and jDE100 methods, affirming its robustness and efficiency.

5 | Conclusions and Future Work

This study presents the LEA as a groundbreaking approach to tackling MMOPs. It offers significant advancements in identifying Nash equilibrium points in game theory and the localization of robotic resources. By employing independent evolving subpopulations, M-LEA effectively navigates complex multimodal landscapes without relying on parameters such as radius or prior knowledge of optima distribution. This adaptability distinguishes M-LEA from traditional methods, ensuring its robust performance across diverse applications and reassuring the audience.

This study addresses the complex and fundamental challenges in multimodal optimization, where identifying both local and global optimal points is of critical importance. The main contribution of this research is the development of the M-LEA algorithm, which distinguishes itself from other methods. This algorithm, without requiring prior knowledge of the search space or the need to set niche parameters, effectively navigates complex optimization environments. The experimental results, which compare the performance of M-LEA against seven other multimodal optimization algorithms across 14 different test problems, have yielded very promising outcomes. The M-LEA algorithm has consistently been able to identify more, and in some cases, all of the local and global optima, outperforming its competitors in terms of overall performance. This remarkable performance, coupled with the high adaptability of the M-LEA algorithm to a variety of problems, makes it a valuable tool for researchers and practitioners seeking effective solutions for complex MMOPs. This study emphasizes the importance of LEA's performance and practical applicability, presenting it as a promising avenue for future research and practical applications. The findings of this research are expected to inspire further studies and expand the use of M-LEA across various fields, contributing to the development of more efficient strategies for solving complex problems. In Nash equilibrium identification, M-LEA consistently extracted stronger optima across all tested games, showcasing its precision. Similarly, in robotic resource localization, M-LEA exhibited lower variance around the mean, underscoring its stability and efficiency, particularly as resource complexity increased compared to methods like Glowworm Swarm Optimization (GSO) and jDE100.

The algorithm's ability to integrate diverse data types enhances its capacity to address multifaceted challenges, paving the way for innovative solutions in fields such as game theory, robotics,

and beyond. However, limitations remain, including its untested performance in dynamic real-world environments and potential computational demands in resource-constrained settings. These challenges highlight the need for continued refinement to ensure LEA's practical applicability across varied contexts.

Looking ahead, the LEA holds immense potential for expansion into new domains of multimodal optimization. Its versatility and robustness make it an excellent candidate for modeling complex systems in crisis management, healthcare, autonomous driving, and social media analysis, where multi-faceted decision-making is critical. One particularly promising avenue is its application in materials science, where M-LEA could be leveraged to optimize the discovery of optimal materials for specific applications. For instance, M-LEA could be employed to identify optimal metal–organic frameworks (MOFs) tailored for drug delivery systems, enhancing the precision and efficiency of material selection by navigating the complex multimodal design space of porosity, stability, and drug-loading capacity. Additionally, ongoing research could explore LEA's integration with real-time data streams to improve its adaptability in dynamic environments, as well as its scalability for large-scale multi-objective systems addressing economically and socially significant challenges. The continued evolution of M-LEA is anticipated to drive substantial progress in optimization theory and its practical implementations across interdisciplinary fields.

Author Contributions

The authors confirm their contributions to the paper as follows: study conception and design: E.D., M.Y., H.T., and M.J.; theorem analysis and interpretation of results: E.D., M.Y., H.T., S.C., and M.J.; draft manuscript preparation: E.D., M.Y., H.T., S.C., and M.J.; supervision: M.Y. and M.J. All authors reviewed the results and approved the final version of the manuscript.

Acknowledgments

The authors acknowledge support from the KIT-Publication Fund of the Karlsruhe Institute of Technology. Open Access funding enabled and organized by Projekt DEAL.

Ethics Statement

The authors have nothing to report.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The MATLAB code for the algorithm is available at https://github.com/MehrdadJalali-KIT/LotusEffectAlgorithm.

References

1. R. Wang, K. Hao, B. Huang, and X. Zhu, "Adaptive Niching Particle Swarm Optimization With Local Search for Multimodal Optimization," *Applied Soft Computing* 133 (2023): 109923.

2. X. Li, M. G. Epitropakis, K. Deb, and A. Engelbrecht, "Seeking Multiple Solutions: An Updated Survey on Niching Methods and Their Applications," *IEEE Transactions on Evolutionary Computation* 21, no. 4 (2016): 518–538.

3. J. Zou, Q. Deng, J. Zheng, and S. Yang, "A Close Neighbor Mobility Method Using Particle Swarm Optimizer for Solving Multimodal Optimization Problems," *Information Sciences* 519 (2020): 332–347.

4. Y. Liu, Z. Yan, W. Li, M. Lv, and Y. Yao, "An Automatic Niching Particle Swarm for Multimodal Function Optimization," in *Advances in Swarm Intelligence: First International Conference, ICSI 2010*, Beijing, China, June 12–15, 2010, Proceedings, Part I (Springer, 2010), 110–119.

5. S. Das, S. Maity, B.-Y. Qu, and P. N. Suganthan, "Real-Parameter Evolutionary Multimodal Optimization—A Survey of the State-of-the-Art," *Swarm and Evolutionary Computation* 1, no. 2 (2011): 71–88.

6. A. Pétrowski, "A Clearing Procedure as a Niching Method for Genetic Algorithms," in *Proceedings of IEEE International Conference on Evolutionary Computation* (IEEE, 1996), 798–803.

7. D. E. Goldberg and J. Richardson, "Genetic Algorithms With Sharing for Multimodal Function Optimization," in *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, vol. 4149 (Lawrence Erlbaum, 1987).

8. D. Beasley, D. R. Bull, and R. R. Martin, "A Sequential Niche Technique for Multimodal Function Optimization," *Evolutionary Computation* 1, no. 2 (1993): 101–125.

9. O. J. Mengshoel and D. E. Goldberg, "The Crowding Approach to Niching in Genetic Algorithms," *Evolutionary Computation* 16, no. 3 (2008): 315–354.

10. O. J. Mengshoel and D. E. Goldberg, "Probabilistic Crowding: Deterministic Crowding With Probabilistic Replacement," 1999.

11. X. Yin and N. Germay, "A Fast Genetic Algorithm With Sharing Scheme Using Cluster Analysis Methods in Multimodal Function Optimization," in *Artificial Neural Nets and Genetic Algorithms: Proceedings of the International Conference in Innsbruck* (Springer, 1993), 450–457.

12. J.-P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson, "A Species Conserving Genetic Algorithm for Multimodal Function Optimization," *Evolutionary Computation* 11, no. 1 (2003): 107–109.

13. J.-P. Li and A. Wood, "Random Search With Species Conservation for Multimodal Functions," in *2009 IEEE Congress on Evolutionary Computation* (IEEE, 2009), 3164–3171.

14. R. I. Lung and D. Dumitrescu, "A New Subpopulation Model for Evolutionary Multimodal Optimization," in *Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'05)* (IEEE, 2005), 4.

15. K.-S. Leung and Y. Liang, "Adaptive Elitist-Population Based Genetic Algorithm for Multimodal Function Optimization," in *Genetic and Evolutionary Computation Conference* (Springer, 2003), 1160–1171.

16. S. Yazdani, H. Nezamabadi-Pour, and S. Kamyab, "A Gravitational Search Algorithm for Multimodal Optimization," *Swarm and Evolutionary Computation* 14 (2014): 1–14.

17. M. Orujpour, M.-R. Feizi-Derakhshi, and T. Rahkar-Farshi, "Multi-Modal Forest Optimization Algorithm," *Neural Computing and Applications* 32, no. 10 (2020): 6159–6173.

18. T. Rahkar Farshi and M. Orujpour, "A Multi-Modal Bacterial Foraging Optimization Algorithm," *Journal of Ambient Intelligence and Humanized Computing* 12 (2021): 1–15.

19. T. R. Farshi, "A Memetic Animal Migration Optimizer for Multimodal Optimization," *Evolving Systems* 13, no. 1 (2022): 133–144.

20. S. W. Mahfoud, "A Comparison of Parallel and Sequential Niching Methods," in *Conference on Genetic Algorithms*, vol. 136 (Citeseer, 1995), 143.

21. H. Zhao, Z. H. Zhan, Y. Lin, et al., "Local Binary Pattern-Based Adaptive Differential Evolution for Multimodal Optimization Problems," *IEEE Transactions on Cybernetics* 50, no. 7 (2019): 3343–3357.

22. Z. Wei, W. Gao, G. Li, and Q. Zhang, "A Penalty-Based Differential Evolution for Multimodal Optimization," *IEEE Transactions on Cybernetics* 52, no. 7 (2021): 6024–6033.

23. W. Gao, G. G. Yen, and S. Liu, "A Cluster-Based Differential Evolution With Self-Adaptive Strategy for Multimodal Optimization," *IEEE Transactions on Cybernetics* 44, no. 8 (2013): 1314–1327.

24. L. Huang, C.-T. Ng, A. H. Sheikh, and M. C. Griffith, "Niching Particle Swarm Optimization Techniques for Multimodal Buckling Maximization of Composite Laminates," *Applied Soft Computing* 57 (2017): 495–503.

25. Y. Cao, H. Zhang, W. Li, M. Zhou, Y. Zhang, and W. A. Chaovalitwongse, "Comprehensive Learning Particle Swarm Optimization Algorithm With Local Search for Multimodal Functions," *IEEE Transactions on Evolutionary Computation* 23, no. 4 (2018): 718–731.

26. X. Wang, Y. Wang, X. Shi, L. Gao, and P. Li, "A Probabilistic Multimodal Optimization Algorithm Based on Buffon Principle and Nyquist Sampling Theorem for Noisy Environment," *Applied Soft Computing* 104 (2021): 107068.

27. Y.-H. Zhang, Y.-J. Gong, H.-Q. Yuan, and J. Zhang, "A Tree-Structured Random Walking Swarm Optimizer for Multimodal Optimization," *Applied Soft Computing* 78 (2019): 94–108.

28. D. Chen and Y. Li, "A Development on Multimodal Optimization Technique and Its Application in Structural Damage Detection," *Applied Soft Computing* 91 (2020): 106264.

29. W. Sheng, X. Wang, Z. Wang, Q. Li, and Y. Chen, "Adaptive Memetic Differential Evolution With Niching Competition and Supporting Archive Strategies for Multimodal Optimization," *Information Sciences* 573 (2021): 316–331.

30. R. Ahmed, A. Nazir, S. Mahadzir, M. Shorfuzzaman, and J. Islam, "Niching Grey Wolf Optimizer for Multimodal Optimization Problems," *Applied Sciences* 11, no. 11 (2021): 4795.

31. Z. Dai, W. Fang, K. Tang, and Q. Li, "An Optima-Identified Framework With Brain Storm Optimization for Multimodal Optimization Problems," *Swarm and Evolutionary Computation* 62 (2021): 100827.

32. S. Cheng, X. Wang, M. Zhang, X. Lei, H. Lu, and Y. Shi, "Solving Multimodal Optimization Problems by a Knowledge-Driven Brain Storm Optimization Algorithm," *Applied Soft Computing* 150 (2024): 111105.

33. W. Du, Z. Ren, J. Wang, and A. Chen, "A Surrogate-Assisted Evolutionary Algorithm With Knowledge Transfer for Expensive Multimodal Optimization Problems," *Information Sciences* 652 (2024): 119745.

34. D. Feng, Y. Li, J. Liu, and Y. Liu, "A Particle Swarm Optimization Algorithm Based on Modified Crowding Distance for Multimodal Multi-Objective Problems," *Applied Soft Computing* 152 (2024): 111280.

35. K. D. Çiçek, T. Akan, and O. Bayat, "Multi-Modal Battle Royale Optimizer," *Cluster Computing* 27, no. 7 (2024): 8983–8993.

36. R. D. McKelvey and A. McLennan, "Computation of Equilibria in Finite Games," in *Handbook of Computational Economics*, vol. 1 (Elsevier, 1996), 87–142.

37. K. Krishnanand and D. Ghose, "Glowworm Swarm Optimization for Multimodal Search Spaces," in *Handbook of Swarm Intelligence: Concepts, Principles and Applications* (Springer, 2011), 451–467.

38. O. R. Adegboye, A. K. Feda, O. S. Ojekemi, E. B. Agyekum, A. G. Hussien, and S. Kamel, "Chaotic Opposition Learning With Mirror Reflection and Worst Individual Disturbance Grey Wolf Optimizer for Continuous Global Numerical Optimization," *Scientific Reports* 14, no. 1 (2024): 4660.

39. M. Ghasemi, K. Golalipour, M. Zare, et al., "Flood Algorithm (FLA): An Efficient Inspired Meta-Heuristic for Engineering Optimization," *Journal of Supercomputing* 80, no. 15 (2024): 22913–23017.

40. U. Mohammed, T. Karataev, O. Oshiga, et al., "ICSOMPA: A Novel Improved Hybrid Algorithm for Global Optimisation," *Evolutionary Intelligence* 17 (2024): 1–104.

41. S. A. Amin, M. K. S. Alqudah, S. A. Almutairi, R. Almajed, M. R. Al Nasar, and H. A. Alkhazaleh, "Optimal Extreme Learning Machine for Diagnosing Brain Tumor Based on Modified Sailfish Optimizer," *Heliyon* 10, no. 14 (2024): e34050, https://doi.org/10.1016/j.heliyon.2024.e34050.

42. M. R. Rao and S. Sundar, "Enhancement in Optimal Resource-Based Data Transmission Over LPWAN Using a Deep Adaptive Reinforcement Learning Model Aided by Novel Remora With Lotus Effect Optimization Algorithm," *IEEE Access* 12 (2024): 76515–76531, https://doi.org/10.1109/ACCESS.2024.3406749.

43. B. S. Chokkalingam, B. Sirumulasi Paramasivan, and M. Muthusamy, "Topological Information Embedded Convolutional Neural Network-Based Lotus Effect Optimization for Path Improvisation of the Mobile Anchors in Wireless Sensor Networks," *Network: Computation in Neural Systems* (2024): 1–26.

44. A. Nakamura, "Applying Bio-Inspired Optimization Algorithms to Secure Network Protocols: A Study Using the Lotus Effect Optimization Algorithm," *Advances in Computer Sciences* 7, no. 1 (2024): 1–15.

45. F. M. Jose, S. J. J. Juliet, D. J. David, T. J. Jebaseeli, A. R. Kurup, and B. Premjith, "Adaptive Lotus Effect Optimization With DKN for Fake News Detection on Social Media With Tamil Language," in *International Conference on Smart Data Intelligence* (Springer, 2024), 81–92.

46. E. Dalirinia, M. Jalali, M. Yaghoobi, and H. Tabatabaee, "Lotus Effect Optimization Algorithm (LEA): A Lotus Nature-Inspired Algorithm for Engineering Design Optimization," *Journal of Supercomputing* 80 (2023): 1–39.

47. X.-S. Yang, "Flower Pollination Algorithm for Global Optimization," in *International Conference on Unconventional Computing and Natural Computation* (Springer, 2012), 240–249.

48. Y. Meraihi, A. Ramdane-Cherif, D. Acheli, and M. Mahseur, "Dragonfly Algorithm: A Comprehensive Review and Applications," *Neural Computing and Applications* 32 (2020): 16625–16646.

49. C. Jada, H. Yenala, K. K. Rachavarapu, N. K. Chittipolu, and S. Omkar, "Modified Roaming Optimization for Multi-Modal Optima," in *2012 Third International Conference on Emerging Applications of Information Technology* (IEEE, 2012), 56–61.

50. I. Bala and A. Yadav, "Niching Comprehensive Learning Gravitational Search Algorithm for Multimodal Optimization Problems," *Evolutionary Intelligence* 15, no. 1 (2022): 695–721.

51. R. D. McKelvey, "An experimental test of a stochastic game model of committee bargaining," *Laboratory research in political economy* (1991): 139–168.

52. R. D. McKelvey, "A Liapunov Function for Nash Equilibria," 1998.

53. X. Li, A. Engelbrecht, and M. G. Epitropakis, "Benchmark Functions for CEC'2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization," (RMIT University, Evolutionary Computation and Machine Learning Group, 2013), Tech. Rep.

54. B. Qu, J. Liang, P. Suganthan, and Q. Chen, "Problem Definitions and Evaluation Criteria for the CEC 2015 Competition on Single Objective Multi-Niche Optimization," (Computational Intelligence Laboratory, Zhengzhou University, 2014), Tech. Rep.

Appendix A

Overview of the Reviewed Sources

Ref. No.	Title	Author(s), year	Method for best result in MMOPs
[6]	A clearing procedure as a niching method forgenetic algorithms	Pétrowski, 1996	The clearing technique utilizes the radius clearing distance.
[7]	Genetic algorithms with sharing for multimodal function optimization	Goldberg et al., 1987	Using the sharing radius, which divides the population into subgroups.
[8]	A sequential niche technique for multimodal function optimization	Beasley et al., 1993	It highly dependent on a parameter known as the niche radius and is determined based on the distance between two optima.
[9]	The crowding approach to niching in genetic algorithms	Mengshoel et al., 2008	The output population is evaluated using the optimal radius mechanism.
[10]	Probabilistic crowding: Deterministic crowding with probabilistic replacement	Mengshoel et al., 1999	To determine the optimal number in these approaches, the output population should be evaluated using the optimal radius-dependent mechanism.
[11]	A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization	Yin et al., 1993	With clustering strives to be independent of estimating the optimal radius.
[12]	A species conserving genetic algorithm for multimodal function optimization.	Li et al., 2003	Species conservation introduces a parameter, species distance, which is estimated based on knowing the number of optima in the optimization function.
[13]	Random search with species conservation for multimodal functions	Li et al., 2009	Species conservation introduces a parameter, species distance, which is estimated based on knowing the number of optima in the optimization function.
[14]	A new subpopulation model for evolutionary multimodal optimization	Lung et al., 2005	The roaming technique, combined with external memory for storing solutions, has proven effective in multimodal optimization.
[15]	Adaptive elitist-population based genetic algorithm for multimodal function optimization	Leung et al., 2003	Adjustments to genetic algorithm operators and a distance measure are employed to determine the proximity of two solutions to each other.
[16]	A gravitational search algorithm for multimodal optimization	Yazdani et al., 2014	The idea of subdividing the main population into smaller subgroups and preserving them is presented along with three strategies: K-nearest neighbors, elitism, and modification of the active gravitational mass formula.
[17]	Multi-modal forest optimization algorithm	Orujpour et al., 2020	Clustering techniques based on niching methods are integrated to transform the single-peaked forest optimization algorithm
[18]	A multi-modal bacterial foraging optimization algorithm	Farshi et al., 2021	The complexity of this algorithm is less than that of its single-peaked counterpart, as the dedispersion phase, like any other phase, is omitted.
[19]	A memetic animal migration optimizer for multimodal optimization	Farshi et al., 2022	It does not require prior knowledge of the problem space, as it avoids the need to predefine niche parameters, instead requiring specific adaptations to enable multimodal optimization.
[20]	A comparison of parallel and sequential niching methods	Mahfoud et al., 1995	The author categorizes the methods for solving multimodal optimization problems using evolutionary algorithms (EAs) into two main groups: iterative and parallel approaches.
[21]	Local binary pattern-based adaptive differential evolution for multimodal optimization problems	Zhao et al., 2019	The LBP utilizes the information from neighboring pixels to extract relevant patterns and identify multiple regions of interest, like finding multiple peaks in MMOP.
[22]	A penalty-based differential evolution for multimodal optimization	Wei et al., 2021	In PMODE, a dynamic penalty strategy is used to solve MMOPs, along with an elite selection mechanism to identify and select elite solutions.
[23]	A cluster-based differential evolution with self-adaptive strategy for multimodal optimization	Gao et al., 2013	This paper explores cluster-based differential evolution (DE) for multimodal optimization, where the entire population is divided into subpopulations to find different optima.
[24]	Niching particle swarm optimization techniques for multimodal buckling maximization of composite laminates	Huang et al., 2017	The advanced multimodal PSO algorithms adopted include the species-based PSO (SPSO), the fitness Euclidean-distance ratio-based PSO (FER-PSO), the ring topology-based PSO, and the Euclidean distance-based locally informed particle swarm (LIPS) optimizer.
[25]	Comprehensive learning particle swarm optimization algorithm with local search for multimodal functions	Cao et al., 2018	Taking advantage of CLPSO's strong global search capability and LS's fast convergence ability, it pursues higher optimization performance.

Ref. No.	Title	Author(s), year	Method for best result in MMOPs
[26]	A probabilistic multimodal optimization algorithm based on Buffon principle and Nyquist sampling theorem for noisy environment	Wang et al., 2021	The author proposes two new strategies to make the algorithm capable of probability prediction and multiple extreme points optimization.
[27]	A tree-structured random walking swarm optimizer for multimodal optimization	Zhang et al., 2019	The author has developed a niching strategy with a tree structure to help the algorithm explore multiple optima simultaneously.
[28]	A development on multimodal optimization technique and its application in structural damage detection	Chen et al., 2020	To improve the detection accuracy of the proposed algorithm, the Depth First Search (DFS) is adopted, and a new particle update scheme is proposed to maintain the diversity of particle populations.
[29]	Adaptive memetic differential evolution with niching competition and supporting archive strategies for multimodal optimization	Sheng et al., 2021	The authors have designed a niching competition strategy to competitively utilize niches according to their potential by encouraging niches with high potential for exploitation while niches with low potential for exploration, thus effectively Suitability searches the space to identify multiple optima.
[30]	Niching grey wolf optimizer for multimodal optimization problems	Ahmed et al., 2021	The authors propose a niching GWO (NGWO) that incorporates the personal best features of PSO and a local search technique to address these issues.
[31]	An optima-identified framework with brain storm optimization for multimodal optimization problems	Dai et al., 2021	(OIF) combined with brainstorm optimization (BSO) algorithm can identify global optimal solutions found during the search process and maintain these optima until the end of the run.
[32]	Solving multimodal optimization problems by a knowledge-driven brain storm optimization algorithm	Cheng et al., 2024	A knowledge-driven BSO in objective space (KBSOOS) algorithm is proposed to enhance the search performance and to maintain the diversity of the solutions for solving MMOPs.
[33]	A surrogate-assisted evolutionary algorithm with knowledge transfer for expensive multimodal optimization problems	Du et al., 2024	This study proposes a surrogate-assisted multimodal evolutionary algorithm with knowledge transfer (SAKT-MMEA), where a modality prediction method based on global surrogate-assisted sampling (GSSMP) and a joint surrogate-assisted local search method (JSLS) is designed for efficient modality exploration and exploitation, respectively.
[34]	A particle swarm optimization algorithm based on modified crowding distance for multimodal multi-objective problems	Feng et al., 2024	This study proposes a particle swarm optimization algorithm based on modified crowding distance (MOPSO_MCD). In MOPSO_MCD, a modified method for calculating the crowding distance (MCD) is devised, which allows for a more comprehensive assessment of the crowding relationship between individuals in the decision space and the objective space.
[35]	Computation of equilibria in finite games	McKelvey et al., 1996	This chapter provides an overview of the latest state of the art of methods for numerical computation of Nash equilibria and refinements of Nash equilibria—for general finite n-person games.
[36]	Glowworm swarm optimization for multimodal search spaces	Krishnanand et al., 2011	This chapter presents glowworm swarm optimization (GSO), a novel swarm intelligence algorithm that was recently proposed for simultaneous capture of multiple optima of multimodal functions.
[37]	Lotus effect optimization algorithm (LEA): a lotus nature-inspired algorithm for engineering design optimization	Dalirinia et al., 2023	The authors introduce a new evolutionary algorithm called the lotus effect algorithm. The article also highlights the practical application of LEA in reducing energy consumption in IoT nodes through clustering, resulting in increased packet delivery ratio and network lifetime.

Appendix B

Symbols

Symbol	Description
V_i^{t+1}	Speed of drop i in the next evolution iteration
V_i^t	Speed of drop i in current evolution iteration t
X_i^t	Possible solution i in current evolution iteration t
X_i^{t+1}	Possible solution i in the next evolution iteration
$X_{deep\ pit}^t$	Deepest drop pit
$Rand.r_1.r_2$	Random numbers between 0 and 1
q	Decremental coefficient of drop movement
$Select_i^t$	Pit selection
c_i^t	Pit capacity
t	Evolution iteration counter
L	Maximum number of evolution iteration
f_i^t	Value of the fitness function for possible solution i
f_{Max}	Maximum value of fitness function
f_{Min}	Minimum value of fitness function
$\boldsymbol{\sigma}$,$\boldsymbol{\beta}$. const	Constants
ΔX_i^{t+1}	Step length in the next evolution iteration's movement
w	Movement step coefficient
ΔX_i^t	Step length in current evolution iteration's movement
R	Decremental coefficient of flowers' growth area
s	Separation movement coefficient
S_i^t	Separation movement variable for search agent i in evolution round t
a	Alignment movement coefficient
A_i^t	Alignment movement variable for search agent i in evolution iteration t
c	Cohesion movement coefficient
C_i^t	Cohesion movement variable for search agent i in evolution iteration t
f	Food attraction coefficient
F_i^t	Food attraction coefficient for search agent i in evolution iteration t
e	Enemy distraction movement coefficient
E_i^t	Movement variable of enemy distraction for search agent i in evolution iteration t
X_-^t	The worst-found solution (enemy)
X_+^t	The best-found solution (food)
$GM(subpopulation_i)$	The population growth rate
x_i^*	The best answer in subpopulation$_i$
$Card\ B$	The main elements of set B
α	Archive parameter
β	Subpopulation stability level
x_1	Search factors
x_2	Search factors
x^*	The best search factor before moving
i_q	Two arbitrary points in the search space
i_p	Two arbitrary points in the search space
$HV(i_q, i_p)$	Hill-valley function
$samples$	Array is used to calculate the interior points in the hill-valley function
interior	Points inside the hill-valley function
σ^*	Nash equilibrium point
σ_i	The situation where the i player plays the desired strategy $\boldsymbol{\sigma_i}$
$v(\sigma)$	The Nash Lyapunov function
λ_i	The relaxation parameter
T_i	Deviation characteristic
y	The positions of robot

Appendix C

Test Functions

Function name	Equations	Search range	Number of local/global
F1: Ackley function	$f(x) = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) -$ $\exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos\left(cx_i\right)\right) + a + \exp(1)$	$-5 \le x_i \le 5$	120/1
F2: Six-hump camel function	$f(x,y) = \left(4 - 2.1x^2 + \frac{x^4}{3}\right)x^2 + xy + \left(-4 + 4y^2\right)y^2$	$-1,9 \le x_i \le 1,9$ $-1,1 \le x_i \le 1,1$	4/2
F3: Decreasing maxima	$ex\left(-\frac{2lo(2)\ (x-0.08)}{0.854}\right)\cdot\left(\left(5\pi x^{3/4} - 0.05\right)\right)$	$0 \le x_i \le 1$	4/1
F4: Equal maxima	$f(x) = (5\pi x)$	$0 \le x_i \le 1$	0/5
F5: Himmelblau function	$f(x,y) = \left(x^2 + y - 11\right)^2 + \left(x + y^2 - 7\right)^2$	$-6 \le x_i \le 6$	0/4
F6: Modified Rastrigin	$\sum_{i=1}^{n}\left[10 + 9\cdot\cos\left(2\pi k_i x_i\right)\right]$	$0 \le x_i \le 1$	0/12
F7: Vincent function	$f(x) = \sum_{i=1}^{d}\sin\left(10\log\left(x_i\right)\right)$	$0.25 \le x_i \le 1$	0/36
F8: Shubert function	$f(x,y) =$ $\sum_{i=1}^{5} i\cos\left((i+1)x + i\right)\sum_{j=1}^{5} j\cos\left((j+1)y + j\right)$	$-5 \le x_1 \le 5$ $-5 \le x_2 \le 5$	742/18
F9: Composition Function 1	f_1, f_2 : Griewank′s function f_3, f_4 : Weierstrass function f_5, f_6 : Sphere function $f(x) = \sum_{i=1}^{6}\left[\omega_i \cdot f_i\left(\frac{x-o_i}{\lambda_i}\cdot M_i\right) + \text{bias}_i\right] + f_{\text{bias}}$	$-5 \le x \le 5$	119/6
F10: Composition Function 2	f_1, f_2 : Rastrigin function f_3, f_4 : Weierstrass function f_5, f_6 : Griewank function f_7, f_8 : Sphere function $f(x) = \sum_{i=1}^{8}\left[\omega_i \cdot f_i\left(\frac{x-o_i}{k_i}\cdot M_i\right) + \text{bias}_i\right] + f_{\text{bias}}$	$-5.5 \le x \le 5$	264/8
F11: Five-uneven-peak trap	$f(x) = 700 \cdot \sum_{i=1}^{D} t_i + 200D$ t_i is defined as : $-200 + x_i^2$ *if* $x_i < 0$ $-80 \times \left(2.5 - x_i\right)$ *if* $0 \le x_i < 2.5$ $-64 \times \left(x_i - 2.5\right)$ *if* $2.5 \le x_i < 5$ $-64 \times \left(7.5 - x_i\right)$ *if* $5 \le x_i < 7.5$ $-28 \times \left(x_i - 7.5\right)$ *if* $7.5 \le x_i < 12.5$ $-28 \times \left(17.5 - x_i\right)$ *if* $12.5 \le x_i < 17.5$ $-32 \times \left(x_i - 17.5\right)$ *if* $17.5 \le x_i < 22.5$ $-32 \times \left(27.5 - x_i\right)$ *if* $22.5 \le x_i < 27.5$ $-80 \times \left(x_i - 27.5\right)$ *if* $27.5 \le x_i \le 30$ $-200 + \left(x_i - 30\right)^2$ *if* $x_i > 30$	$-100 \le x_1 \le -45$ $30 \le x_2 \le 80$	21/4
F12: Expanded equal maxima	$f(x) = 307 - \sum_{i=1}^{D} t_i + D$ $t_i =$ y_i^2 *if* $y_i < 0$ or $y_i > 1$ $sin^6\left(5\pi \cdot y_i\right)$ *if* $0 \le y_i \le 1$	$10 \le y_1 \le 50$ $-40 \le y_2 \le 10$	0/25
F13: Expanded uneven maxima	$f(x) = 508 - \sum_{i=1}^{D} t_i + D$ $t_i =$ y_i^2 if $y_i < 0$ *or* $y_i > 1$ $\sin^6\left(5\pi\left(y_i^{(3/4)} - 0.05\right)\right)$ if $0 \le y_i \le 1$	$-40 \le y_1 \le 0$ $-70 \le y_2 \le -30$	0/25
F14: Modified Vincent function	$f(x) = 805 - \frac{1}{D}\sum_{i=1}^{D} t_i + 1.0$ $t_i =$ $\left(0.25 - y_i\right)^2 + \sin\left(10 \cdot log(2.5)\right)$ if $y_i < 0.25$ $\sin\left(10 \cdot log\left(y_i\right)\right)$ if $0.25 \le y_i \le 10$ $\left(y_i - 10\right)^2 + \sin\left(10 \cdot log(10)\right)$ if $y_i > 10$	$20 \le y_1 \le 80$ $40 \le y_2 \le 100$	0/36