Exploration-Exploitation Tradeoffs in Metaheuristics: A Review

Bayadir Abbas Al-Himyari [1], Hiba Al-khafaji [2] and Noor Fadel Hussain [3]

[1] Cyber Security, College of Information Technology, University of Babylon, Iraq
Email: sci.bayadir.abbas@uobabylon.edu.iq

[2] Software, College of Information Technology, University of Babylon, Iraq.
Email: hibamj.alkhafaji@uobabylon.edu.iq

[3] Cyber Security, College of Information Technology, University of Babylon, Iraq.
Email: noor.fadel@uobabylon.edu.iq

ABSTRACT— ***A metaheuristic algorithm is an algorithmic framework at a high level, independent of problems, which offers a set of recommendations or strategies to develop heuristic optimization algorithms. Evolutionary computation and other metaheuristics have long been focused on the trade-off between exploration and exploitation. This subject is present in a wide range of domains, including modeling and prediction, search and optimization, machine learning and cognition, and many more situations where uncertainty is present. The trade-off between exploration and exploitation is crucial for all optimization techniques. Efficient optimization and lower computing costs can be achieved by striking a fair balance between both.***
The paper introduces a study that discusses metaheuristic algorithms and previous studies in three areas: 1) What elements of metaheuristic facilitate exploration and exploitation; 2) When and how exploration control is applied. 3) how to find a balance between the two.

Keywords— Metaheuristics, Exploration, Exploitation, Trade-off

1. INTRODUCTION

It has been shown that the most useful methods for solving challenging hard optimization problems are the metaheuristics (MHs) family. The MH algorithm is an algorithmic framework that is high-level and independent of problems. It offers a collection of principles or tactics for creating heuristic optimization algorithms. Over the past few decades, a wide variety of MHs have been presented. The majority of MHs were devoted to applications and experimental research. Exploration and exploitation are the two fundamental search behaviors employed in MH algorithms in general and swarm intelligence in particular. While exploitation refers to searching the area around a promising region, exploration refers to finding an unexplored area of the viable region [1, 2].

Although MHs are capable of both exploration and exploitation, it has frequently been discovered that they are stuck in a local rather than global optima. The primary cause is the challenge of appropriately balancing the two abilities, exploitation and exploration.

The terms "exploration" and "exploitation" have been defined differently in the literature on MHs. Exploration and exploitation are interpreted as global and local search, respectively, from an inherent viewpoint. This explanation unquestionably captures a facet of their characters, however based just on search scope, it is unreliable to differentiate between exploration and exploitation because there is no universally accepted cutoff point for what constitutes "local." Strictly limiting exploration to stay inside the entire search space is a clear-cut but pointless approach, and any search inside a smaller space will be seen as exploitation.

Exploration and exploitation in machine learning algorithms represent the gathering and application of knowledge, respectively. Some academics believe that the degree of randomness can distinguish between exploitation and exploration. In order to obtain knowledge about unknown problems, Chen et al. classified two types of behavior: exploration and exploitation [3]. The information acquisition process was viewed as a sampling process that progressively transforms an optimization problem from a "black-box model" into a "whitebox." Chen states that a sampling behavior is considered exploratory if and only if its sampling point is created without reference to the data gathered by previous sampling points. In the same way, sampling behavior involves exploitation of the corresponding

generation. In general, exploitation refers to the ability to find high quality solutions inside the search regions, while exploration refers to the capacity to visit numerous and diverse regions of the search space.

A search algorithm ought to achieve a tactical balance between these two potentially conflicting objectives. The majority of conventional MHs consist of multiple components for both exploration and exploitation. Evolutionary computation and other machine learning fields have long been interested in the trade-off between exploration and exploitation. This subject is present in a wide range of domains, including modeling and prediction, search and optimization, machine learning and cognition, and many more situations where uncertainty is present. The trade-off between exploration and exploitation is crucial for all optimization techniques. Efficient optimization and lower computing costs can be achieved by striking a fair balance between both [4, 5].

The paper introduces a study that discusses metaheuristic algorithms and previous studies in three areas: 1) What elements of metaheuristic facilitate exploration and exploitation; 2) When and how exploration control is applied. 3) how to find a balance between the two.

2. METAHEURISTIC ALGORITHMS CHARACTERISTICS

Throughout the past 20 years, metaheuristic techniques have grown in popularity and been used in a variety of scientific and industrial fields. The following factors account for this popularity [6]:

- Simplicity: The majority of them were motivated by straightforward ideas about natural occurrences, animal behavior, or evolutionary theory. Because of its simplicity, scientists and researchers are able to replicate a wide range of natural notions, enhance existing meta-heuristics, combine two or more of them, or suggest new meta-heuristics.
- Flexibility: The meta-heuristics can be readily applied to a wide range of problems without requiring any special structural modifications. They also view problems as black boxes. This indicates that the input and output of a system are its most crucial components for a meta-heuristic.
- Gradient-Free (Derivation-Free) Mechanism: Metaheuristics optimize problems stochastically, meaning that random solutions are first found and the derivative of search spaces does not need to be calculated in order to discover the optimum.
- Avoiding Local Optima: Meta-heuristics are more effective in avoiding local optima. Because metaheuristics are stochastic, stagnation in local solutions is avoided and the entire search space is searched. Therefore, it appears that using meta-heuristics to optimize problems with a large number of local optima is an effective approach.

Meta-heuristic algorithms are able to provide reasonable results in a suitable amount of time. The solution obtained may not always be appropriate, just as meta-heuristic algorithms may not always guarantee the best answers. Put another way, depending on the application to which an evolved algorithm is employed, its overall performance may vary. Although an algorithm can be highly effective in solving a particular problem, it might not be the best option for another one. As previously stated, the meta-heuristic algorithm is run on random inputs and obtained outputs, and it is not reliant on the problem. Having an effective algorithm that can produce reasonable and quality solutions is the goal.

3. TYPES OF MHs

Based on the manipulation of the solutions, three basic classes of MHs may be identified. Iteratively, local search MHs tweak a single solution slightly. Constructive MHs build solutions out of their component pieces. Population-based MHs create novel solutions by combining old ones repeatedly. These classifications don't have to be mutually exclusive, though, as many MH algorithms use concepts from several classes. We refer to these techniques as hybrid MHs [5, 2].

3.1 Local Lookup MHs

Iterative improvement, also known as local search (LS), identifies effective solutions by repeatedly modifying a single answer, known as the current solution. This class of MHs gets its name from these changes, which are called movements and are usually "small" (such that neighboring solutions are relatively close to each other according to a natural metric). The neighborhood of a given solution is the set of solutions that can be acquired by making a single move on that solution. Different move types can be established depending on how the solution is represented. Every kind of relocation creates a different neighborhood structure. The current answer is swapped out for one from its neighborhood in each cycle. The move strategy, also known as the search strategy, is the rule that is applied to choose the new current solution. The steepest ascent or descent approach, which chooses the best move from the neighborhood, is a popular search technique. MHs who employ this tactic are frequently referred to as hill-climbers. The option that little enhances the present solution is chosen using the mildest descent/ascent approach. Other move strategies include the first improving approach, which, naturally, depends on the sequence in which the moves are checked, chooses the first step that makes the existing solution better.

Instances of this group include, in order to explore the solution space, Tabu search (TS) keeps track of a short-term memory (tabu list), Variable Neighborhood Search (VNS) algorithms systematically explore different neighborhoods to escape local optima, and Simulated Annealing (SA) mimics the annealing process in metallurgy.

Rather of enhancing whole solutions, constructive MHs, as their name implies, build solutions from their constituent parts. This is accomplished by gradually adding elements to a partial solution. Frequently, greedy algorithms that add the best member at each iteration are adapted into constructive maximum hierarchy algorithms. Most constructive MHs include a local search phase following the construction phase in order to enhance the quality of the final solutions.
This category includes models that mimic the collective behavior of ant colonies, such as the greedy randomized adaptive search process (GRASP), pilot method, Large neighborhood search (LNS), and Ant colony optimization (ACO).

3.2 MHs Depending on Populations

Population-based MHs choose and combine existing solutions from a set, typically referred to as the population, iteratively until they identify good solutions. Evolutionary algorithms (EAs) are the most significant members of this class because they closely resemble the rules of natural evolution. We refer to the broad class of evolutionarily based MHs as evolutionary algorithms. This includes, among many other things, particle swarm optimization (PSO), which mimics the social behavior of particles in a swarm, genetic programming (GP), which evolves computer programs or solutions represented as trees, evolutionary computation (EC), evolution strategies (ES), and genetic algorithms (GA), which evolve a population of solutions using genetic operators like crossover, mutation, and selection. "Pure" evolutionary algorithms are uncommon when used to solve combinatorial optimization problems; instead, most of them incorporate some sort of improvement operator, most frequently in the form of local search.

Evolutionary algorithms work with a set, or population, of solutions and employ two methods to find good solutions: first, they select from the population the solutions that are primarily of high quality, and then they recombine those solutions into new ones by using specialized operators that combine the properties of two or more solutions. Following recombination, new solutions are reintroduced into the population to replace older, typically lower-quality solutions. These new solutions may need to meet requirements like feasibility or minimum quality needs. Almost often, the operators (selection, recombination, and replacement) in evolutionary algorithms heavily rely on randomness. It is also common to use a mutation operator, which randomly modifies a solution, if slightly, after it has been recombined. The majority of evolutionary algorithms reflect the population's optimal solution after repeatedly iterating through the stages of selection, recombination, mutation, and replacement. For evolutionary algorithms to guarantee that the best solutions endure over multiple iterations while maintaining population diversity, some sort of "population management" is usually necessary.

3.4 Hybrid MHs

There has been a trend in recent years to see MH frameworks as general ideas or components that can be applied to the development of optimization techniques [7]. Since most modern MH algorithms integrate concepts from several classes, the term "hybrid MH" has lost most of its ability to discriminate. Specialized heuristics are used by many recent MHs to effectively tackle subproblems generated by the MH. In a similar vein, a significant amount of local search MHs begin the neighborhood search by using a construction phase to identify an initial solution, or a group of initial solutions. Actually, the greedy randomized construction phase should be followed by a local search phase, according to the original GRASP MH description [8]. Recombination operators from the family of evolutionary algorithms are combined with local search MHs in algorithms that fall within the memetic algorithms class [9].

4. FACTORS AFFECT THE TRADE-OFF OF EXPLORATION VS EXPLOITATION IN MH SEARCH

In addition to the MH's strategy, a number of problem- and implementation-related factors also influence the trade-off between exploration and exploitation. The following are significant variables in MH algorithms that impact the exploration-exploitation trade-off [10, 1]:

- Problem complexity: Exploration becomes more difficult in high-dimensional spaces due to the large size of the solution space. To successfully navigate complicated landscapes, algorithms may need to strike a balance between exploring multiple regions and exploitation of interesting spots.
- Search Space Topology: The trade-off may be impacted by the existence of several local optima. While broad exploration may be necessary to avoid local optima, algorithms should also take use of effective solutions to increase the likelihood of convergence towards a global optimum.

- Algorithmic factors: including crossover probability and mutation rates. can have a big impact on the trade-off between exploration and exploitation, just like GAs. While high crossover probability may result in increased exploitation, high mutation rates promote exploration.
- Maintaining diversity methods in the population: Some algorithms include methods to keep the population diverse

while encouraging exploration. Diversity guarantees a more complete exploration of the solution space and helps avoid early convergence.

- Adaptation mechanisms: During the search, the exploration-exploitation balance can be efficiently adjusted by algorithms that dynamically modify parameters in response to the performance and features of the optimization issue.
- Dynamics of the problem: Algorithms must modify their approaches in dynamic optimization problems where the objective function or constraints vary over time. As the issue develops, a balance between exploration and exploitation must be kept.

- Quality of initial solutions: The subsequent balance between exploration and exploitation can be influenced by the quality of the initial solutions discovered during the early phases of the algorithm. Inadequate initial solutions could necessitate a greater focus on exploration.
- Computational resources: The algorithm's capacity to explore and exploit can be impacted by the availability of computational resources, such as memory and time, as well as computational budget. A more concentrated strategy to either exploration or exploitation may be necessary because to limited resources.

- Specific characteristics of the problem: Certain problems may contain exploitable structures that can be used to their advantage for effective exploitation. Designing exploration and exploitation tactics can be influenced by an understanding of specific characteristics of the problem at hand

- Performance metrics: As convergence criteria, the success or convergence criteria have an impact on the trade-off between exploration and exploitation. Algorithms can have to strike a compromise between guaranteeing a comprehensive investigation of the solution space and obtaining a solution rapidly (exploitation).

A MH algorithm's efficacy frequently rests on how well these parameters are adjusted to the specifics of the given optimization task. Experiments and sensitive analyses are frequently carried out by researchers and practitioners to gain insight into how modifications to these parameters affect the exploration-exploitation trade-off and overall algorithm performance.

4.1 Evaluation of the Balance

In order to identify suitable solutions for an optimization problem, MH algorithms employ a set of candidate solutions to explore the search space. The search procedure is typically directed toward search agents who have the best solutions [11]. This attraction results in a decrease in the distance between search agents and an increase in the effect of exploitation. However, the impact of the exploration process becomes more noticeable as search agents get farther apart. A diversity measurement called the dimension-wise diversity measurement [10] is taken into consideration in order to compute the increase and decrease in distance among search agents. The population diversity is defined as follows with this method:

$$Div_j = \frac{1}{n}\sum_{i=1}^{n} \left| median(x^j) - x_i^j \right| \quad (1)$$

$$Div = \frac{1}{m}\sum_{j=1}^{m} Div_j \quad (2)$$

where the $median(x_j)$ denotes the population-wide median of dimension j. The search agent's dimension is denoted by X_{ij}, The number of search agents in the population is represented by n, and the number of design variables in the optimization problem is represented by m.

The gap between each search agent's dimension (j) and the averaged median of that dimension is known as the diversity in each dimension (Div_j). Then, by averaging each Div_j in each dimension, the diversity of the overall population Div is determined. Every iteration computes both values. The entire balance response can be defined as the portion of an MH scheme's exploration and exploitation. In every iteration, these values are calculated using the following models:

$$XPL\% = \left(\frac{Div}{Div_{max}}\right) \times 100 \quad (3)$$

$$XPT\% = \left(\frac{\left|Div - Div_{max}\right|}{Div_{max}}\right) \times 100 \quad (4)$$

where Div_{max} is the highest diversity value discovered during the whole optimization procedure. The link between the diversity attained at the maximum and the diversity in each iteration is represented by the exploration percentage, or

XPL%. The level of exploitation is reflected in the XPT% percentage of exploitation. Since the concentration of search agents produces the difference between the maximal diversity and the current diversity of an iteration, it is calculated as the complemental percentage to XPL%. It is evident that elements XPL% and XPT% are complementary to one another and in conflict with one another.

By employing a reference element, the median value prevents inconsistencies in the evaluation of the balance response. Another intriguing characteristic of the balance response is that the maximum diversity (Div_{max}) discovered throughout the optimization procedure also affects the *XPL%* and *XPT%* values. However, the filtering effect that the average combination of all dimensions produces is one drawback of the dimension-wise diversity index.

5. LITERATURE REVIEW

Here, various metaheuristic algorithms will be discussed including how they achieve the balance between exploration and exploitation.

5.1 Genetic Algorithms

Population-based search algorithms known as genetic algorithms (GAs) are modeled after the process of natural evolution. They iteratively develop a population of candidate solutions toward better solutions by utilizing genetic operators, such as selection, crossover, and mutation [12, 13].

i. Exploration and Exploitation in GA

By exploring, the GA aims to discover novel solutions, uncover hidden gems, and avoid getting stuck in local optima. By exploiting known points, the GA hones in on promising areas, aiming for convergence toward the global optimum. Here are some GA operators are their relation of the trade-off:

- Selection Operator: The selection operator is a critical factor in this trade-off. It determines which individuals (genetic solutions) survive and reproduce.
- Essence of Exploitation: Fitness-proportional selection emphasizes exploiting well-performing solutions.
- Influence of Exploration: Linear rank selection leans toward exploration, encouraging diversity.
- Novel Selection Scheme: Researchers have proposed a fresh approach—a selection scheme that strikes an optimal balance between exploration and exploitation. This scheme adjusts selection pressure dynamically, avoiding premature convergence and enhancing overall performance.

Balancing exploration and exploitation in a genetic algorithm (GA) is crucial for achieving optimal results. Here are some strategies to implement this trade-off:

1. Population Diversity:
   - Exploration: Maintain diversity within the population by using techniques like tournament selection or roulette wheel selection. These methods allow less fit individuals to survive, promoting exploration.
   - Exploitation: Use elitism to preserve the best solutions from one generation to the next. Elitism ensures that high-performing individuals are not lost during evolution.

   Strategies implemented:

   - Tournament Selection: Allows less fit individuals to survive, promoting exploration.

   - Elitism: Preserves the best solutions from one generation to the next.

   - Dynamic Parameters: Gradually adjust population size and mutation rate during evolution.

   - Niching Techniques: Penalize solutions in crowded regions to encourage diversity.

2. Crossover and Mutation Rates:
   - Exploration: Increase the mutation rate. Mutations introduce randomness, allowing the GA to explore new regions of the search space.
   - Exploitation: Adjust the crossover rate. High crossover rates emphasize exploitation by combining existing solutions.
3. Dynamic Parameters:
   - Exploration: Start with a larger population size and higher mutation rate. Gradually decrease these parameters as the GA progresses.
   - Exploitation: Decrease the population size and mutation rate over time. This encourages convergence toward promising solutions.
4. Niching Techniques:

- Exploration: Implement crowding or fitness sharing. These methods encourage diversity by penalizing solutions in crowded regions of the fitness landscape.
- Exploitation: Use speciation to maintain separate niches. Speciation prevents premature convergence by promoting diversity.

5. Multi-Objective Optimization:
   - Exploration: Optimize multiple objectives simultaneously. Pareto-based approaches explore trade-offs between conflicting objectives.
   - Exploitation: Use dominance-based ranking to select solutions that balance multiple objectives effectively.

The ideal balance between exploration and exploitation depends on the problem domain, the specific GA, and the characteristics of the fitness landscape. Hussain and Muhammad [14] presented a study aims to find a balance between premature convergence and population diversity in GAs. The researchers introduce a new selection scheme called split-based selection (SBS)**.** This operator strikes a fine balance between two extremes: exploration and exploitation. It eliminates fitness scaling issues and dynamically adjusts selection pressure. The study evaluates the SBS operator using TSPLIB instances, demonstrating significantly improved results in terms of average and standard deviation values. Exploration **i**nvolves venturing into new areas of the search space. Exploitation **c**apitalizes on existing knowledge to refine solutions.

Alba and Dorronsoro [15] presented a paper which explores the trade-off between exploration and exploitation in dynamic cellular GAs. The study investigates both static and dynamic decentralized versions of the cellular genetic algorithm (cGA). In cGA, individuals are situated in a specific topology and interact only with their neighbors. It examines how feedback from search experiences stored in an archive impacts the algorithm's behavior. The study evaluates two different replacement strategies within cGA. These strategies influence how individuals are replaced during evolution. The proposed algorithms are tested against a benchmark of problems. the best-performing algorithm is compared with two state-of-the-art genetic algorithms for multi-objective optimization. Dynamic cGAs exhibit the most desirable behavior in terms of efficiency and accuracy among all evaluated versions.

Vafaee et al. [16] presented a study that focuses on maintaining a delicate balance between exploration and exploitation in GAs. The researchers propose a GA that emphasizes diversity. The GA uses a novel mutation operator with site-specific mutation rates. Mutation rates are adjusted based on the underlying pattern of highly-fit solutions. The proposed approach is evaluated using benchmark problems, demonstrating its effectiveness in achieving the desired trade-off.

Zhang et al. [17] presented a paper that introduces a survival analysis method to tackle the trade-off issue. Results from the analysis guide the selection of appropriate solution creation operators that favor either exploration or exploitation. Specifically, a differential evolution recombination operator is used for exploration, while a novel clustering-based operator is proposed for exploitation. The developed algorithm is compared with four well-known multi-objective evolutionary algorithms.

5.2 Simulated Annealing

A stochastic search technique called "simulated annealing" (SA) mimics the annealing procedure used in metallurgy. Early in the search phase, it accepts worse solutions using a probabilistic acceptance criterion and then progressively concentrates on enhancing the objective function. [18, 19].

5.2.1 Schedule of Annealing and Neighborhood Structure

In SA, a temperature parameter is set and an initial solution is generated. By altering the existing solution and admitting new ones based on a probability function that is dependent on the objective function difference and the current temperature, the algorithm iteratively searches the solution space. The set of possible moves from the existing solution is defined by the neighborhood structure.

5.2.2 Acceptance Criteria

Based on the current temperature and the objective function value of the new solution, the acceptance criterion in SA decides whether to accept it or not. Exploration is aided by the algorithm's initial higher probability of accepting worse solutions. The algorithm progressively becomes more selective and concentrates on enhancing the objective function as the temperature drops.

5.2.3 Exploration and Exploitation in Simulated Annealing:

Exploration involves searching for new solutions in unexplored regions. Allows occasional uphill moves to explore diverse areas. While **e**xploitation refines existing solutions to improve their fitness. Balances exploration by accepting better solutions. Simulated annealing dynamically balances exploration and exploitation, adapting its behavior to the problem domain.

Amine [20] presented a study which focuses on extending SA to multiobjective optimization problems introducing Multiobjective Simulated Annealing (MOSA). MOSA constructs an estimated Pareto front by gathering nondominated solutions during exploration. Challenges include balancing exploration and exploitation due to the number of objective functions. Researchers propose various MOSA variants to achieve this balance. These variants adapt mutation rates, use clustering-based operators, and manage diversity.

Chen [21] presented a research which introduces a strategy called "threshold convergence" to balance exploration and exploitation in stochastic search techniques. It enhances performance in multi-modal search spaces. This research contributes to achieving a delicate balance in SA by incorporating threshold-based control. Early-stage local search steps are "held back" by a threshold function. Preventing interference with concurrent global search mechanisms. Threshold convergence leads to significant performance improvements.

5.3 Differential Evolution (DE)

In 2005, Storn and Price introduced differential evolution as a reliable and easily parallelizable substitute for global optimization techniques. DE takes its concept of self-organization from the well-liked heuristic search technique, Nelder-Mead's simplex search algorithm [22]. DE begins with a population of randomly initialized solution vectors, just like other population-based metaheuristics. However, DE modifies an existing solution in the population using the difference vector of two randomly selected members rather than variation operators with predefined probability distributions [23].

5.3.1 Exploration and Exploitation in DE

Balancing exploration and exploitation in DE is crucial for achieving effective optimization. Some strategies to achieve this delicate balance are:

1. Population Diversity:
   - Exploration Aspect: Maintain diversity within the population by using techniques like tournament selection or crowding. These methods allow less fit individuals to survive, promoting exploration.
   - Exploitation Aspect: Use elitism to preserve the best solutions from one generation to the next. Elitism ensures that high-performing individuals are not lost during evolution.
2. Mutation Strategies:
   - Exploration: Experiment with different mutation strategies. For instance, consider using large-scale mutations to explore distant regions of the search space.
   - Exploitation: Fine-tune mutation parameters. Smaller-scale mutations can help refine existing solutions.
3. Crossover Rate:
   - Exploration: Adjust the crossover rate. Higher crossover rates encourage exploration by combining genetic material from different individuals.
   - Exploitation: Lower crossover rates emphasize exploitation by preserving existing solutions.
4. Adaptive Parameters:
   - Exploration: Start with a larger population size and higher mutation rate. Gradually decrease these parameters as the DE progresses.
   - Exploitation: Decrease the population size and mutation rate over time. This encourages convergence toward promising solutions.
5. Niching Techniques:
   - Exploration: Implement techniques like fitness sharing or speciation. These methods prevent premature convergence by promoting diversity.
   - Exploitation: Use niching to maintain separate niches of solutions.
6. Hybrid Approaches:
   - Combine DE with other optimization methods (e.g., local search, particle swarm optimization, or simulated annealing).
   - Hybridization can enhance both exploration and exploitation capabilities.

The optimal balance depends on the problem domain, the specific DE variant, and the characteristics of the fitness landscape. Epitropakis et al. [24] propose a hybrid approach that combines DE mutation operators to achieve a balance between their exploration and exploitation capabilities. Combines different DE mutation operators which aims to strike an optimal balance between exploring new regions and refining existing solutions. Extensive experiments demonstrate that the proposed hybrid approach effectively enhances DE's ability to accurately locate solutions in the search space.

A method known as DEIE (Differential Evolution with Information Entropy-Based Mutation Strategy) is put forth by Wang et al. [25]. This work provides insights into how information entropy-based mutation techniques can be used to achieve a delicate equilibrium in DE. Using the crowding technique, DEIE calculates the total number of Markov

states. It uses historical evolutionary data to infer the transition matrix between states. The evolutionary process is divided into stages of exploration and exploitation dynamically by DEIE. The Markov state model's information entropy serves as the basis for this divide. DEIE adaptively uses stage-specific mutation operations. The usefulness of DEIE is shown through experiments conducted on classical functions and benchmark sets.

A paper examining the trade-off between exploration and exploitation in DE was provided by Sá et al. [26]. DE may have trouble avoiding local optima and preserving diversity. To improve its behavior, the study proposes changing typical DE techniques. In order to improve the behavior of the DE algorithm, this study suggests modifying the usual mechanisms by altering the exploration vs. exploitation balance. To create population diversity, a probabilistic selection process for the new member of the population is one suggested option. Their cost function is used to define the selection probability of the population members.

Zhang et al. [27] presented a paper which conduct a comprehensive survey on strategies to achieve a delicate balance between exploration and exploitation in DE across different scales like: Parameter adaptation techniques, strategies to enhance population diversity, and selection mechanisms with varying pressure. Achieving the right balance is crucial for DE's performance on diverse optimization problems.

A novel search framework built on an explicit strategy of control is put forth by Cai et al. [28]. Striking a balance between exploration and exploitation in (EAs). Three different forms of transference are included in an explicit control technique that is introduced to balance exploration and exploitation. Operators for exploration and exploitation are defined formally. A novel differential evolution technique based on triple transference is put forth.

5.4 Ant Colony Optimization

Ant Colony Optimization (ACO), is a metaheuristic algorithm that draws inspiration from ants' foraging habits. It builds high-quality solutions iteratively through the use of stigmergy and pheromone-based communication [29].

5.4.1 Stigmergy and Pheromone-Based Communication

Artificial ants build solutions in ACO by repeatedly going from one place to another and leaving pheromone trails in their path. An edge's desirability is indicated by the pheromone concentration on that edge. Ants use pheromone levels and heuristic information from nearby areas to make a probabilistic decision on where to visit next. The dynamic updating of pheromone trails is ensured by pheromone evaporation and reinforcement mechanisms. Reinforcement builds the pheromone trail of high-quality solutions the ants have discovered, but evaporation gradually lowers pheromone levels to prevent convergence to suboptimal solutions.

5.4.2 Route Construction and Pheromone Updating

By repeatedly choosing places to visit based on pheromone and heuristic information, each ant in ACO builds an entire set of routes. In order to update the pheromone, the current pheromone is evaporated, and then new pheromone is applied to the edges that the ants have traveled, depending on how well a solution was discovered.

5.4.3 Exploration and Exploitation in ACO

ACO achieves a delicate balance between exploration and exploitation [2]**.** During exploration, ants explore new paths by depositing pheromones on edges. This encourages diversity and helps discover novel solutions. While during exploitation, ants follow paths with higher pheromone levels. This exploits known good solutions. There are some challenges, for example: striking the right balance is crucial for ACO's effectiveness, too much exploration can lead to slow convergence, and too much exploitation can cause premature convergence. There are a number of methods to achieve balance:

1. Dynamic Pheromone Update:
   - Adjust pheromone levels dynamically based on solution quality.
   - Encourage exploration when solutions are suboptimal.
2. Heterogeneous Approach:
   - Combine different ACO variants with varying exploration-exploitation trade-offs.
   - Improve performance and reduce parameter tuning efforts.
3. Information Entropy-Based Strategies:
   - Use entropy measures to guide exploration and exploitation.
   - Enhance the algorithm's ability to explore diverse regions.

Jabbar [30] presented a study which improves the results of the Traveling Salesman Problem (TSP) produced by ACO. Trying to balance the exploration and exploitation components within ACO by overcoming the drawbacks of the exploration problem. The study produced global optimal results in high-dimensional space. Experiments on six variants of ACO show that the proposed work produces high-quality results in terms of the shortest route.

The authors Liu et al. [31] proposed a novel approach to enhance the performance of ACO by addressing the exploration-exploitation trade-off. The approach combines two techniques: epsilon greedy and Levy flight. An exploration method widely used in reinforcement learning. Adapted to ACO as a pseudo-stochastic mechanism and based on Levy distribution, lead to balancing search space exploration and speed for global optimization. The Greedy-Levy ACO incorporates both approaches to solve complex combinatorial optimization problems. The study specifically applied to the TSP. Greedy-Levy ACO outperforms existing ACO variants and other TSP solvers.

The authors Kumazawa et al. [32] proposed two exploration strategies to enhance the performance of model checking based on ACO and strike the balance between exploration and exploitation. Model checking is a formal verification technique for software systems. Traditional exhaustive search techniques often fail for large systems due to resource demands. The study introduces different kinds of randomized selection mechanisms and diversify solutions found by many agents. Strategies help the search algorithm extend reachable regions effectively. Computation time and memory requirements are reduced compared to existing ACO methods.

5.5 Tabu Search

The metaheuristic algorithm Tabu Search (TS) explores the solution space using a memory-based search technique. It keeps track of recent visited solutions in a short-term memory known as the tabu list, which prevents them from being revisited. Iteratively exploring within neighborhood of the existing solution, TS permits moves that satisfy particular tabu conditions or enhance the objective function [33].

5.5.1 Exploration and Exploitation in TS:

TS can achieve balance in different contexts:

- Exploration: TS explores the solution space by allowing moves that might not lead to immediate improvement. It uses a tabu list to prevent revisiting recently explored solutions.
- Exploitation: TS intensifies the search by focusing on promising regions. It prioritizes moves that lead to better solutions.

The tabu tenure (how long a move remains forbidden) determines the balance between exploration and exploitation. A shorter tabu tenure encourages exploration, while a longer one emphasizes exploitation. The authors Chandran et al. [34] introduced a hybrid approach that combines GA and TS to address the challenge of efficient resource allocation in cloud data centers. Traditional exhaustive search techniques are resource-intensive and inefficient for large-scale systems. GA provides exploration capabilities, while TS emphasizes exploitation. The goal is to strike a balance between exploring a wide solution space and refining existing solutions. A novel approach is introduced, called Tabu Job Master. It combines the strengths of GA and TS. It improved convergence speed, effective utilization of variables, and enhanced energy consumption results.

The authors Hanafi et al. [35] proposed a novel approach called the Alternating Ascent (AA) algorithm. The paper focuses on binary combinatorial optimization problems, specifically, it addresses the challenge of escaping local optima during search. The AA Algorithm combines the strengths of GA and TS. It alternates between two phases: Ascent Phase (exploration) moves toward a local optimum and Post-Ascent Phase (exploitation) moves away from the local optimum and other previous local optima. This step encourages exploitation by exploring new directions and escaping local optima. The AA Algorithm dynamically adjusts its behavior based on thresholds and adaptive memory. The approach improved convergence speed, efficient utilization of variables, and enhanced energy consumption results.

5.6 Particle Swarm Optimization

The population-based optimization technique known as Particle Swarm Optimization (PSO) was motivated by the social behavior of fish schools and flocks of birds. PSO involves the movement of candidate solutions, or particles, through the solution space according to both the swarm's and their own best-known positions. PSO updates the particle locations and velocities iteratively in an attempt to converge to optimal solutions [36].

5.6.1 Exploration and Exploitation in PSO

In PSO algorithms, achieving a delicate balance between exploration (global search) and exploitation (local search) is crucial for effective optimization. Here are some strategies:

1. Diversification Strategies: Various diversification techniques exist in PSO, such as:
   - Particle diversity maintenance: Encouraging particles to explore different regions.
   - Dynamic neighborhood structures: Adapting the neighborhood topology during optimization.
   - Randomization: Introducing randomness to escape local optima.
2. Adaptive Memory Approaches: Some PSO variants use adaptive memory to balance exploration and exploitation. Examples include:

- Tabu search: Maintaining a list of recently visited solutions.

- Memory-based PSO: Incorporating historical information to guide particle movement.

3. Threshold-Based Techniques: These methods dynamically adjust parameters based on specific conditions. For example, adjusting inertia weight or neighborhood size based on convergence progress.

The authors Binkley & Hagiwara [37] propose a study using VBR to enhance the performance of PSO algorithms. Standard PSO often converges quickly to local minima, missing better opportunities in multimodal functions. While VBR monitors particle velocities during evolution. When the median velocity of swarm particles drops below a threshold, the entire swarm is reinitialized. VBR alleviates premature convergence, allowing PSO to focus on one minimum at a time. Experimental results show improved performance on multimodal benchmark functions.

The authors Nakisa et al. [38] proposed a combinatorial optimization method based on Particle Swarm Optimization (PSO) and local search algorithms for multi-robot search systems. The method aims to strike a balance between exploration and exploitation by reinitializing the swarm when the distance between the target and a robot becomes small. This approach encourages local exploration beyond reaching the target, resulting in efficient search times.

The author Ghalia [39] proposed enhancements to the standard Particle Swarm Optimization (PSO) algorithm. The new approach, called PSO with Active Velocity Penalty (PSO-AVP)**,** addresses the disadvantage of allowing particles to move outside the search space. PSO-AVP actively penalizes particle velocities to ensure confinement within the search space. By preventing particles from moving outside the feasible region, PSO-AVP ensures better exploration and consistent behavior.

Zhang [40] proposed a particle swarm optimization algorithm with an empirical balance strategy (EBPSO), which selects a better search strategy from two equations using an adaptive adjustment mechanism. The algorithm dynamically adjusts the influence weight of the search equations and introduces a dynamic random search mechanism.

Liu et al. [41] introduced random PSO (RPSO) which aims to improve PSO's search ability by maintaining diversity and preventing premature convergence. The RPSO introduces Gaussian white noise (GWN) with adjustable intensity that perturbs acceleration coefficients during velocity updates. This strategy enhances exploration and helps escape local optima traps. The proposed RPSO outperforms existing PSO variants on widely used optimization benchmark functions.

5.7 Iterated Local Search

A metaheuristic algorithm called Iterated Local Search (ILS) integrates local search with diversification and perturbation [42, 43]. It begins with a solution and refines it using a local search algorithm. ILS carries on searching after obtaining a local optimum by introducing perturbations to escape the local optima. To raise the quality of the solution, the local search and perturbation processes are repeated iteratively.

5.7.1 Exploration and Exploitation in ILS

The key idea in ILS is to strike a balance between exploration and exploitation by creating new starting solutions from perturbations of previously found solutions.

- Exploration: Creating new starting solutions by perturbing existing ones.
- Exploitation: Using local search to improve the current solution.

It balances exploration (via perturbations) and exploitation (via local search). An acceptance criterion determines whether to keep the new solution or maintain the previous one. ILS has been successfully applied to tackle hard combinatorial optimization problems. ILS combines the principles of local search (hill climbing) with iterative improvement. The basic idea is to iteratively apply a move operator to the current solution and then restart local search from the perturbed solution.

HyperILS is an extension of ILS that incorporates ideas from hyper-heuristics and reinforcement learning. It aims to automate the design of heuristic methods for solving computational search problems. It uses reinforcement learning to dynamically choose the best operator or heuristic at each iteration. Unlike traditional ILS, HyperILS modifies the design of the perturbation and improvement stages. By adaptively selecting operators, HyperILS maintains this balance while searching for optimal solutions.

A study was introduced by Al-Behadili et al. [44], where the ILS-AntMiner algorithm is used for rules-based classification. ILS plays a crucial role in enhancing exploitation. In optimization algorithms, exploitation refers to refining the current solution by focusing on promising regions of the search space. Exploration, on the other hand, involves searching for new solutions in unexplored areas. The local search explores the neighborhood of the current solution, aiming for improvements. Perturbation introduces randomness, allowing exploration of different regions. Acceptance criteria determine whether to accept a new solution based on its quality. In the context of ILS-AntMiner, Local search explores rule space, refining existing rules. Perturbation introduces variations (e.g., rule modifications). Acceptance criteria guide the selection of improved rules. By combining Ant Colony Optimization (ACO) with ILS, the

algorithm achieves effective exploitation: It intensifies the search around promising solutions, and it escapes local optima by exploring diverse rule combinations. This balance leads to accurate classification models.

A study was introduced by Zhao et al. [45], a HybridILS algorithm combines success-history based parameter adaptation for differential evolution (SHADE) and limited-memory Broyden–Fletcher–Goldfarb–Shanno (LBFGS) as its perturbation and local search strategies. By integrating the exploration capability of SHADE and the local search performance of LBFGS, HybridILS aims to solve numerical optimization problems effectively. The algorithm employs a simulated annealing acceptance criterion to balance exploration and exploitation within the iterated local search framework.

5.8 Artificial Bee Colony Optimization (ABC)

Karaboga introduced the ABC metaheuristic algorithm in 2005. It is among the most frequently referenced new generation metaheuristics [46, 47]. Since ABC is a population-based method, it has been used to solve a number of optimization issues. Since solutions are represented as the food resources themselves, and candidate solutions are represented as bees exploring/exploiting food resources, ABC is a natural aspiration. A solution denotes a food supply, and the quantity of nectar in each resource denotes the fitness or quality of each solution.

Within the hive, there are three different kinds of bees: "employed," "onlooker," and "scout" bees. In nature, bee employees search for food, return to their colony, and dance to share information. After gathering all of the nectar, an employed bee becomes a scout, searching for fresh sources of food. While scout bees explore for food sources, onlooker bees observe how employed bees dance and select food sources.
When compared to other metaheuristic algorithms, the ABC has a number of advantages, including minimal control parameters, ease of implementation, and exceptional exploration capabilities. Because it searches for the optimal solution both locally and globally for each iteration, the probability of finding the optimal solution is significantly increased.

5.8.1 Exploration and Exploitation in ABC

Exploration and exploitation are achieved as the following:

- Exploration (Scout Bees): ABC emphasizes exploration by employing scout bees to search for new solutions. However, some researchers have reported that ABC may focus too much on exploration, especially as problem dimensions increase. High-dimensional problems pose challenges for the scout bee operator.
- Exploitation (Employed Bees): Employed bees exploit existing solutions by improving them. The balance between exploration and exploitation is crucial for finding global optima. If ABC leans too heavily toward exploration, it risks missing true global optima.

Achieving an optimal trade-off between exploration and exploitation is essential for ABC's performance. Recent studies suggest that ABC's scout bee operator may become redundant in high-dimensional problems. Contrary to popular belief, ABC may not excel in exploration ability for such scenarios. Researchers have observed these limitations and questioned the algorithm's behavior. How a bee colony is composited affects how well an ABC is explored and exploited. The performance of ABC is compared against GA and PSO by Karaboga et al. [46]. When it comes to multi-variable function optimization, the ABC method is seen to perform better than other algorithms. A better fitness equation (influenced by DE) based on the bees' search behavior around the best nectar in previous iterations is proposed by Gao and Liu [48], who note that ABC's solution search equation is inadequate during the exploitation phase. In comparison to traditional ABC algorithms, the experimental findings show that the modified ABC algorithm performs well in addressing complex numerical problems.

By utilizing their understanding of the global best solution throughout the exploitation, Zhu and Kwong [49] proposed a gbest-guided ABC algorithm (GABC). According on the experimental results, the suggested GABC algorithm works better than the traditional ABC method. The authors' goal is to address ABC's deficiency with reference to its solution search equation.
For numerical optimization, TSai et al. [50] provide a novel ABC optimization approach. The method selects a more appropriate exploration/exploitation ratio in order to enhance the quality of the solutions by including the universal gravitation notion into the affection consideration between the onlooker and employed bees.

W. Xiao et al [51], presented a study where chaotic and neighborhood search-based artificial bee colony algorithm (CNSABC) is used. The CNSABC variant introduces three improved mechanisms: (1**)** Bernoulli Chaotic Mapping with Mutual Exclusion Mechanism: Enhances diversity and exploration ability. (2) Neighborhood Search Mechanism with Compression Factor: Improves convergence efficiency and exploitation capability. (3) Sustained Bees: Aids in maintaining a balance between exploration and exploitation. CNSABC demonstrates better convergence efficiency and search ability compared to traditional ABC.

5.9 Bacterial Foraging Optimization (BFO)

Passino in 2002 suggests the BFO metahuristic. The BFO algorithm processes parallel non-gradient optimization by modeling the foraging behavior of bacteria over a landscape. A set of tensile agella provides the movement (locomotion) that enables an E. Coli bacteria to swim or tumble when it is foraging. As the agella rotates in a clockwise manner, each agellum tightens its cell. As a result, the agella behaves independently and the bacteria tumbles less frequently. It falls sharply to create a nutrient gradient in a harmful area. To make the bacterium swim more quickly, turn the agella counterclockwise. The bacteria can face chemotaxis while moving away from a toxic environment and toward a nutritional gradient.

In an environment that is friendly to them, bacteria can move longer distances. The bacteria have the ability to replicate themselves when they are given enough food. This event serves as inspiration for Passino as he develops the BFO algorithm. Abrupt alterations in the surroundings might prevent chemotactic progress. It is possible for a group of bacteria to travel to new locations or for other groups of bacteria to settle where the original group of bacteria was. This process—known as elimination-dispersal—involves either the termination of bacteria from a specific region or the dispersal of a group of bacteria into a new environmental location.

5.9.1 **Exploration** and **Exploitation in BFO**

Exploration and exploitation are achieved as the following:

- Exploration (Tumbling): Bacteria explore the search space by tumbling based on random directions. This introduces diversity and helps discover new regions.
- Exploitation (Swimming): Bacteria exploit existing solutions by swimming with certain step sizes. This aims to improve the current solution's quality.

Researchers have proposed several enhancements to improve BFO's exploration-exploitation tradeoff:

- Conjugated Novel Step-size BFO (CNS-BFO) [54]**:** This algorithm modifies the step-size strategy and introduces a learning mechanism. By adjusting the step size dynamically during evolution, CNS-BFO strikes a good balance between exploration and exploitation, significantly mitigating premature convergence.
- Self-Adaptive BFO (SA-BFO) [55]: A-BFO adjusts the run-length unit parameter dynamically during evolution to balance exploration and exploitation. It shows improved performance over the original BFO.
- Improved Chemotaxis Strategy [56]: Some variants incorporate novel chemotaxis strategies to enhance exploration and exploitation.

5.10 Bat Algorithm (BA)

Yang presents the BA metaheuristic for the first time [57]. Bats utilize echolocation, a form of sonar, to find their nests in the dark, avoid obstacles, and locate prey. A bat makes a sound and tracks the reflections it receives from objects in the environment. Using echolocation, bats can also distinguish between barriers and food/prey. The idea behind BA is that bats' ability to echolocate may be formalized as a way to solve an objective function's optimal solution. Even in a single run, the nature of exploration and exploitation can be altered and controlled with many parameters [58].

5.10.1 Exploration and Exploitation in BA

The main focus of MHAs like BA is to maintain this delicate balance.

- Exploration involves exploring the entire search space, seeking out new regions that might contain optimal solutions.
- Exploitation, on the other hand, focuses on exploiting promising areas that are already known to be fruitful.

A chaotic BA is suggested by Gandomi and Yang [59] in order to boost the global search capability of BA for robust optimization. The authors examine many chaotic maps on benchmark problems. The outcomes confirm that chaotic BA can perform better than traditional versions of BA.

BA falls into the category of population-based MHAs. Unlike single-solution MHAs, which modify a single candidate solution iteratively, population-based MHAs generate a set of solutions. These algorithms emphasize global exploration by maintaining a diverse population of solutions. By doing so, they avoid premature convergence and enhance the chances of discovering optimal solutions. The Bat Algorithm dynamically adjusts its exploration and exploitation strategies, much like bats fine-tuning their echolocation to locate prey.

5.11 Cuckoo Search Algorithm (CSA)

CSA is suggested by Yang and Deb [60]. The CSA uses fruit flies' and birds' Lévy flight patterns to mimic the brood parasite behavior of cuckoo species. Birds of the Cuckoo species breed aggressively. To improve the likelihood that their own eggs will hatch, they put their eggs in other birds' nests and remove the other eggs. The CSA metaheuristic employs

three basic guidelines: 1) Cuckoos lay one egg at a time, and they leave their eggs in random nests; 2) Nests with higher-quality eggs can survive; 3) There is a constant number of host nests, and the host bird has a probability of $p_a \in [0, 1]$ of detecting the egg. Either the bird leaves the nest and builds a new one, or the egg is thrown outside of the nest. One advantage of CSA is its ease of application to a larger range of optimization problems due to its less number of parameters that need to be modified compared to most other metaheuristic algorithms. However, a significant portion of the new solutions should be generated by randomness during a run to guarantee that the optimization process does not become stuck in a local optima. The CSA randomization is more effective when the step length is chosen by employing a heavy-tailed distribution [61, 62].

5.11.1 **Exploration** and **Exploitation in** CSA

Achieving a balance between exploration (searching widely) and exploitation (focusing on promising areas) is crucial for optimization algorithms. CS achieves this balance through its population-based approach and the use of Levy flights.

- Exploration: The random walk Levy flight behavior enables CS to explore the entire search space thoroughly.
- Exploitation: A fixed number of better fitness cuckoos survive in the environment, ensuring that promising solutions are retained.

Researchers have explored various modifications and hybrid versions of CS. These adaptations aim to enhance its performance, convergence speed, and applicability across different fields, including engineering, machine learning, and deep learning. By combining CS with other techniques, researchers continue to refine its exploration-exploitation balance. Some studies focus on addressing premature convergence in metaheuristic algorithms. For instance, the Automatic Cuckoo Search (AuCS) algorithm dynamically updates step sizes in each generation to strike a better balance between exploration and exploitation, avoiding premature convergence [63].

Salgotra et al. [64] suggest CS enhancements. Even though CS is a useful algorithm, it can still perform better by incorporating exploration and exploitation in the search process. Three modified forms of CS are put forth in this study to enhance the features of exploration and exploitation. Rather of using Lévy flights to effectively search the search space, all of these versions use the Cauchy operator to produce the step size. In order to balance exploration and exploitation, two more notions are also introduced in CS: the division of populations and the division of generations. The effects of probability switch with various population and dimension sizes of the proposed versions of CS have been examined, using 24 standard benchmark problems.

Kanagaraj et al. [65] aims to address reliability and redundancy allocation problems using a hybrid approach that combines the CS algorithm with the well-known GA. Reliability–Redundancy Allocation Problems (RRAP) involve optimizing system reliability while allocating redundant components efficiently. CS is a metaheuristic inspired by cuckoo bird behavior, emphasizing random exploration. GA is a popular evolutionary algorithm for optimization. The authors introduce a novel hybrid algorithm called CS-GA. CS and GA are combined to exploit their complementary strengths. CS handles exploration, while GA focuses on exploitation. CS enhances diversity, preventing premature convergence. GA contributes local search capabilities. CS-GA aims to find optimal redundancy allocation strategies that maximize system reliability while minimizing costs. By balancing exploration and exploitation, it achieves better convergence rates and accuracy.

Hussain et al. [66] propose improvements to CS. The enhanced variant is called Personal Best Cuckoo Search (pBestCS). pBestCS incorporates personal best information during solution generation, improving local search capabilities. Instead of a constant value, pBestCS dynamically updates the switching parameter, enhancing global search. Experimental results across various test problems demonstrate the efficiency of pBestCS compared to standard CS, as well as other optimization algorithms like Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC). The proposed modifications in pBestCS enhance both local and global search capabilities, making it a promising choice for solving optimization problems.

Huang et al. [67] aims to address the Combined Heat and Power Economic Dispatch (CHPED) problem using a novel optimization algorithm called Heterogeneous Evolving Cuckoo Search (HECS). HECS combines the features of CS with a novel constraint-handling mechanism. HECS maintains a diverse population of solutions. The algorithm evolves over time, adapting to the problem landscape. A specialized mechanism ensures feasible solutions throughout the iteration. HECS is evaluated on large-scale CHPED problems. Results demonstrate its effectiveness in finding optimal solutions while considering both economic and emission constraints.

5.12 Firefly Algorithm (FA)

Yang [68] proposed the FA proposal. The brief, rhythmic flashing patterns of fireflies serve as an inspiration for FA. Such ashes have two primary purposes: they may attract mates or serve as a warning to predators. Sexes are brought together by the rhythmic ash and the velocity of flashing. For combinatorial algorithms, the flashing can be expressed as

a function that needs to be optimized. The following principles idealize these flashing properties. 1) Gender is irrelevant when it comes to fireflies' ability to attract one another. 2) What makes a firefly attractive is its brightness. As a result, the less bright firefly travels in the direction of the brighter ones. As the distance between fireflies increases, they become less attractive. When there isn't a brighter one, it moves at random. 3) A firefly's brightness depends on the search space of the goal function [69].

5.12.1 Exploration and Exploitation in FA

Like other optimization algorithms, FA faces the classic trade-off between exploration and exploitation:

- Exploration: Fireflies explore new regions of the solution space, seeking potentially better solutions.
- Exploitation: Fireflies exploit known promising areas, aiming to converge towards optimal solutions.

FA's exploration process can sometimes lead to premature convergence, limiting its ability to escape local optima. Striking the right balance between exploration (diversity) and exploitation (intensity) is crucial. Many scientific studies and adaptations of FA are available. In their paper, Yang and He [69] provide a thorough review of the fundamentals of FA. The authors focus on the significance of striking a balance between exploitation and exploration.

Chaos is added to FA by Gandomi et al. [70] in order to enhance its global search for robust optimization. To set the firefly's attractive motion, chaotic maps are used. Additionally, Brajevic and Stanimirovic [71] suggested a newly designed chaotic FA that used a Gauss map to address the global optimization problems raised by the original FA. They also evaluated the performance of the ICFA (improved chaotic firefly algorithm) using 19 benchmark functions.

The enhanced firefly algorithm (UFA), which was proposed by Brajevic and Ignjatovic [72], aims to prevent local optimum trapping and enhance the original firefly algorithm's problem-solving capabilities. This novel firefly technique was developed using a chaotic map. They used twenty-four benchmark functions to assess how well the suggested chaotic map-based UFA technique performed.

Researchers in [73] have proposed modifications to enhance FA's performance. (1) Adaptive Randomness and Absorption Coefficients: Adjusting these coefficients over time/iterations to balance exploration and exploitation. (2) Gray Relational Analysis: Allocating information from appealing fireflies effectively. (3) Global Best-guided Strategies: Incorporating global best solutions to guide fireflies during movement. These enhancements aim to maintain a delicate balance between exploring new regions and exploiting known promising areas. By adapting randomness and absorption, FA can achieve better convergence rates and overall performance.

Inspired by the scout bee behavior in the ABC algorithm, the authors in [74] introduce the Scouting FA. Fireflies stuck in local optima take additional random walks to escape towards the optimal solution region. Scouting FA aims to improve convergence accuracy by preventing premature convergence. Scouting FA introduces a novel search mechanism to address FA's limitations, leading to improved optimization performance by maintaining a delicate balance between exploration and exploitation. Empirical experiments on standard benchmark functions validate the effectiveness of Scouting FA. Results show that Scouting FA outperforms the original FA, demonstrating its superiority.

The study [75] aims to enhance the global searching ability of the FA by embedding the Cross-Entropy (CE) method into FA. CE is known for its ergodicity, adaptability, and robustness. The goal is to achieve an effective balance between exploration and exploitation, avoiding local optima and improving convergence rates. The authors introduce a novel hybrid meta-heuristic algorithm by combining FA with CE known for its ergodicity, adaptability, and robustness. The proposed hybrid algorithm combines the strengths of FA and CE, achieving a delicate equilibrium between exploring new regions and exploiting known solutions for effective global optimization. Numerical experiments demonstrate the effectiveness of the hybrid algorithm: Improved global search capacity, enhanced optimization accuracy, and better performance across various test functions.

5.13 Grey Wolf Algorithm (GWO)

In 2014, Mirjalili et al. [76] proposed the GWO metaheuristic. The GWO is a pack animal that belongs to the family of predatory animals. There is a social hierarchy in every wolf pack. There are various kinds of wolves in a normal wolf hierarchy, including "alpha dogs," "beta dogs," "omega dogs," and "subordinates." The dog in the pack with the greatest responsibility is the alpha. It commands the group and is dominant. The dog ranked second in the hierarchy is called a beta. In the event that alpha dog becomes dysfunctional, he or she is the most likely candidate to be the alpha dog. The lowest ranking dogs are called omegas. If a dog is not one of the aforementioned breeds, it is referred to as subordinate (or delta). The most fascinating swarm activity exhibited by these wolves is group hunting. The social hierarchy, tracking, encircling, and attacking prey are the main components of the GWO algorithm as a mathematical model. In this model, alpha dog is the best solution. Delta dogs and beta dogs are the second and third best solutions,

respectively. Omega dogs are the last members of the swarm. Grey wolves circle their prey during the encirclement phase [77].

5.13.1 **Exploration** and **Exploitation in GWO**

Like other optimization algorithms, GWO faces the classic trade-off between exploration and exploitation:

- Exploration: Wolves explore new areas of the solution space, seeking potentially better solutions.
- Exploitation: Wolves focus on leveraging known promising areas to converge towards optimal solutions.

GWO maintains a delicate balance by:

- Pack Dynamics: The hierarchical structure within the wolf pack ensures a mix of exploration (by lower-ranking wolves) and exploitation (by alpha wolves).
- Search Intensity: The algorithm dynamically adjusts the search intensity based on the fitness landscape.
- Solution Movement: Wolves move towards better solutions while occasionally exploring new regions.

Mittal et al. [78] propose a modified GWO to balance the exploration and exploitation efforts of GWO that improves the performance of the algorithm. Kohli and Arora [79] introduce the chaotic GWO algorithm to accelerate its global convergence speed. Experiments are studied to carry out to solve standard constrained benchmark problems.

The study [80] proposes a novel approach called Hybrid Evolutionary GWO (HE-GWO) to enhance the effectiveness of the GWO for handling dynamic landscapes and constrained optimization problems. HE-GWO combines the strengths of the canonical GWO with Differential Evolution (DE) to increase diversity. HE-GWO is rigorously benchmarked against various meta-heuristics, including standard GWO, recent variants, and state-of-the-art algorithms. It consistently outperforms competitors in benchmarking tests, achieving the highest profitability in multi-unit production planning problems.

The study [81] introduces a new variant of the Gravitational Search Algorithm (GSA) by combining it with the GWO. The proposed algorithm balances exploration and exploitation by splitting the swarm into two groups: 1) One group focuses on better exploitation. 2) The other group is responsible for better exploration.The modified search process aims to achieve optimal solutions. The algorithm is tested on benchmark functions. Results demonstrate that this approach achieves a better balance between exploration and exploitation, leading to optimal solutions.

The study [82] proposes a hybrid optimization technique that combines the Mean Grey Wolf Optimizer (MGWO) with the Whale Optimizer Algorithm (WOA). The goal is to enhance the exploration and exploitation capabilities of both algorithms. The hybrid algorithm, called Hybrid Approach GWO (HAGWO), utilizes the spiral equation from WOA for two purposes: 1) Balancing exploration and exploitation in the Grey Wolf Optimizer. 2) Preventing premature convergence and local minima trapping by applying the spiral equation to the entire population. The hybrid algorithm is tested on standard benchmark functions. Results demonstrate improved stability, faster convergence rates, and computational accuracy compared to other nature-inspired metaheuristics."

The study [83] aims to enhance the performance of the GWO by addressing its limitations related to exploration and exploitation. Specifically, it focuses on preventing stagnation and improving convergence when solving complex and multimodal optimization problems. In the proposed algorithm: Reinforced Exploitation and Exploration GWO (REEGWO)**,** the top three wolves are given different weights based on their knowledge about the location of the prey (optimal solution). A random search based on tournament selection is used to enhance exploration. A well-designed mechanism balances exploration and exploitation. Experimental results demonstrate that REEGWO outperforms both the standard GWO and its four recently top variants.

In 2023, Shial et al. [84] proposed that the exponential decay equation helps in transforming the exploration to exploitation process from initial iterations to final iterations, with proportions of 70% and 30% respectively, enhancing the exploration capability significantly.

By Long et al. [85], a random opposition-based GWO was employed. This technique was employed to prevent the GWO from becoming trapped in local minima. Furthermore, the suggested approach seeks to achieve a more optimal balance between exploration and exploitation. 30 benchmarks from IEEE CEC 2014 and 23 benchmarks test functions were used to evaluate the proposed approach. The outcomes were comparable to those of other optimization methods.

5.14 Harmony Search Algorithm (HSA)

The HSA algorithm is a metaheuristic that draws inspiration from musical compositions and the compositional writing process [86]. In order to model optimization problems, HSA uses techniques used by artists to compose harmonic music. In HSA, a musician can improvise a song in one of three ways: (1) by naturally playing any well-known piece of music (pitches in harmony) from memory; (2) by mimicking an existing piece of music (pitch adjusted); or (3) by

composing random harmonic notes. Geem et al. [86] applies these possible ways through the problem's optimization process.

Using harmony memory is comparable to selecting the optimal chromosomes in genetic algorithms. The greatest harmonies are preserved for new harmony memories ensured by the harmony memory. Harmony memory acceptance rate, or $r_{accept} \in [0; 1]$, is the parameter allocated to it. The HS method converges more slowly when the rate is too low since only a small number of the best harmonies are chosen. A rate that is excessively high (a number near 1) can make it difficult to explore every harmonie. This may result in incorrect solutions. To avoid his issue, the parameter r_{accept} is chosen between [0.7, 0.95].

5.14.1 Exploration and Exploitation in HAS

HS has distinct exploration and exploitation strategies than traditional metaheuristic algorithms. Studies show that while HS is not a parametrically sensitive algorithm, its performance can be enhanced by tuning its parameters.

- Exploration: HSA explores new regions of the solution space, seeking potentially better solutions.
- Exploitation: HSA exploits known promising areas, aiming to converge towards optimal solutions.

HSA achieves this balance through: 1) Harmony Memory Consideration: Maintaining a memory of harmonious solutions. 2) Pitch Adjustment: Adjusting musical parameters (solution components) to strike a balance. 3) Improvisation and Adaptation: Creating new harmonies while preserving existing good ones. HSA is stated to be a robust optimization algorithm for solving NP-Hard engineering optimization problems.

The goal of the study [87] is to enhance the exploration and exploitation capabilities of the optimization process. he proposed method, called Mine Blast Harmony Search (MBHS), uses MBA for exploration and HS for exploitation. Numerical experiments validate that MBHS provides better exploitation ability, especially in the final iterations. It achieves mature convergence to the optimum solution across various optimization problems.

The study [88] aims to enhance the performance of three HS variants by addressing their slow convergence rates and improving overall optimization efficiency. The three HS variants are improved by integrating an enhanced version of OBL, called Improved Opposition-Based Learning (IOBL) which introduces randomness to increase solution diversity and enhance exploration. The hybrid algorithms are evaluated on benchmark functions to compare their performance with the original HS variants. The new hybrid algorithms demonstrate improved efficiency compared to the original HS variants.

The authors in [89] proposed a novel approach called the Dual-Memory Dynamic Search Harmony Search (DMDS-HS) algorithm. A dual-memory structure is introduced to rank and hierarchically organize the harmonies in the harmony memory. This creates an effective and selectable trust region, reducing blind searching and improving convergence. The trust region is dynamically adjusted during optimization. This balances convergence improvement while maintaining global search capability. To boost convergence speed, a phased dynamic convergence domain concept is introduced. It strategically devises a global random search strategy. An adaptive parameter adjustment strategy rationalizes exploration and exploitation abilities. This fine-tunes the algorithm’s search strategies. Results show that DMDS-HS outperforms other Harmony Search variants and state-of-the-art algorithms in terms of diversity, freedom from local optima, and solution accuracy. Additionally, DMDS-HS exhibits superior clustering performance in solving complex data clustering problems.

The goal of the study [90] is to enhance the searching effectiveness of HSA by addressing its limitations related to exploration and exploitation. The enhancements in EHS_CRP is summarized as**:** 1) Global and Local Dimension Selection: Designed to accelerate the search speed. 2) Selection Learning Operator: Based on global and local mean levels to improve the balance between exploration and exploitation. 3) Circular Region Perturbation: Prevents algorithm stagnation and expands exploration regions. EHS_CRP outperforms other state-of-the-art swarm intelligence approaches in terms of accuracy, convergence speed, stability, and robustness. It performs exceptionally well in engineering design optimization problems.

5.15 Whale Optimization Algorithm (WOA)

WOA was proposed by Mirjalili and Lewis [91]. The WOA is a novel metaheuristic that draws inspiration from humpback whale social behavior. Because they are sociable creatures, humpback whales search for fish in groups using a bubble-net strategy. Humpack whales have adapted their collective hunting and feeding behavior to their benefit because it is easier for them to safeguard their young. This kind of hunting involves whales diving beneath a large group of prey and creating bubbles that force fish into a bubble-net known as bubble-net feeding. Since humpback whales can only swallow small prey as whole due to their narrow throats and lack of teeth, they adopt this foraging strategy to hunt big groups of small fish or krill.

WOA solves NP-Hard optimization issues by using a mathematical model of humpback whales' spiral bubble-net feeding method. Three basic WOA functions include encircling prey, searching for prey, and spiral bubble-net feeding

techniques. When humpback whales locate their target prey, they begin to circle around them. Many search agents are employed by the WOA, and they all begin with a random solution. The other search agents update their locations in the direction of the global best solution, which is carried out by humpback whales, after determining which search agent has the best solution.
WOA is a new metaheuristic, but because it requires few parameters to be tuned during optimization, it appears to have a lot of potential to draw in researchers [92, 93].

5.15.1 Exploration and Exploitation in WOA and Trade-Off

Here are the key points related to the exploration-exploitation trade-off in WOA:

- Exploration involves searching the global solution space randomly, exploring new regions.
- Exploitation focuses on refining known promising areas, aiming to converge towards optimal solutions.

In WOA, the distance control parameter a plays a crucial role in balancing exploration and exploitation. It determines how far a whale moves during its search process. The standard WOA optimizes the distance control parameter $\boldsymbol{a}$ using linear control strategy (LCS). However, the predation behavior of whales is not simply linear. To address this limitation, a new approach called NCS-Arcsin is proposed. NCS-Arcsin accurately describes the process of whales' predation. It significantly improves the exploration and exploitation capabilities of WOA.

The authors in [94] proposed an enhanced WOA specifically designed for time-optimal trajectory planning in traditional manipulator systems. An inertia weight factor is introduced into the surrounding prey and bubble-net attack formulas of the WOA. Reinforcement learning techniques control this weight factor, enhancing the global search capability of the algorithm. Additionally, the Variable Neighborhood Search (VNS) algorithm is incorporated to improve local optimization. The proposed WOA is compared with several commonly used optimization algorithms. The improved WOA produces smooth and continuous manipulation trajectories. Also the authors in [95] propose an enhanced WOA for effective trajectory planning for quadruped robots. The proposed approach improves the WOA by combining it with SA. This hybrid algorithm, called IWOA-SA, prevents the WOA from falling into local optima and balances its exploration and exploitation abilities. Adaptive weights are introduced to enhance optimization performance. Markov chains from stochastic process theory are used to analyze the global convergence of the proposed algorithm. The IWOA-SA algorithm outperforms six representative optimization algorithms across multiple dimensions and test function suites.

The authors in [96] proposed a novel hybrid algorithm that combines the WOA and the FA. This hybrid approach, called Firefly-Whale Optimization Algorithm (FWOA), aims to address mobile robot path planning (MRPP) while achieving a balance between exploration and exploitation. The WOA imitates whale foraging behavior, while the FA mimics firefly behavior. The FWOA combines these two algorithms using multi-population dynamics and opposite-based learning. It aims to quickly find optimal paths in complex mobile robot working environments. The FWOA is tested on 23 benchmark functions and applied to optimize MRPP. Comparative experiments with ten other classical metaheuristic algorithms demonstrate the FWOA's remarkable performance. It excels in terms of convergence speed and exploration capability, outperforming other methods.

6 CONCLUSION

MH is a dynamic system made up of numerous "individuals" that communicate with one another. The components of the system (such as the problem essence, the structure of the algorithm, the algorithm's control parameters, and the procedure for evaluating candidate solutions) are intricately related to one another. Each of them affects the algorithm differently. The performance of MHs is the emergency of each individual's interaction and the overall behavior of the system. Metaheuristic algorithms perform better when exploration and exploitation are balanced. It's critical to strike a balance between these two factors because: 1) An excessive amount of exploration may result in inefficiency since the algorithm may spend too much time looking in useless areas. Overuse of exploitation can lead the algorithm to prematurely converge to a suboptimal solution, hence obstructing the possibility of finding better solutions elsewhere.
In order to efficiently locate high-quality solutions, metaheuristic algorithms combine the advantages of both exploration and exploitation by striking a balance while navigating the solution space.

REFERENCES

[1] E. Cuevas, P. Diaz, and O. Camarena, "Experimental Analysis Between Exploration and Exploitation," in *Intelligent Systems Reference Library*, vol. 195, 2021, pp. 249–269. doi: 10.1007/978-3-030-58100-8_10.
[2] K. Hussain, M. N. M. Salleh, S. Cheng, and Y. Shi, "On the exploration and exploitation in popular swarm-based metaheuristic algorithms," *Neural Comput. Appl.*, 2019, doi: 10.1007/s00521-018-3592-0.
[3] J. Chen, B. Xin, Z. Peng, L. Dou, and J. Zhang, "Optimal contraction theorem for exploration-exploitation tradeoff in search and optimization," *IEEE Trans. Syst. Man, Cybern. Part ASystems Humans*, 2009, doi: 10.1109/TSMCA.2009.2012436.
[4] E. Bayona, J. E. Sierra-García, and M. Santos, "Comparative Analysis of Metaheuristic Optimization Methods

for Trajectory Generation of Automated Guided Vehicles," *Electron.*, 2024, doi: 10.3390/electronics13040728.

[5] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, "Metaheuristic algorithms: A comprehensive review," in *Computational Intelligence for Multimedia Big Data on the Cloud with Engineering Applications*, 2018. doi: 10.1016/B978-0-12-813314-9.00010-4.

[6] D. A. Marín-Idárraga, J. M. Hurtado González, and C. Cabello Medina, "Factors affecting the effect of exploitation and exploration on performance: A meta-analysis," *BRQ Bus. Res. Q.*, 2022, doi: 10.1177/2340944420972707.

[7] Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*. 2004. doi: 10.1007/978-3-662-07807-5.

[8] T. A. Feo, M. G. C. Resende, and S. H. Smith, "Greedy randomized adaptive search procedure for maximum independent set," *Oper. Res.*, 1994, doi: 10.1287/opre.42.5.860.

[9] P. Moscato and C. Cotta Porras, "An Introduction to Memetic Algorithms," *Intel. Artif.*, 2003, doi: 10.4114/ia.v7i19.721.

[10] J. Xu and J. Zhang, "Exploration-exploitation tradeoffs in metaheuristics: Survey and analysis," in *Proceedings of the 33rd Chinese Control Conference, CCC 2014*, 2014. doi: 10.1109/ChiCC.2014.6896450.

[11] B. Morales-Castañeda, D. Zaldívar, E. Cuevas, F. Fausto, and A. Rodríguez, "A better balance in metaheuristic algorithms: Does it exist?," *Swarm Evol. Comput.*, 2020, doi: 10.1016/j.swevo.2020.100671.

[12] O. Samuel Sowole, "A Comparative Analysis of Search Algorithms for Solving the Vehicle Routing Problem," in *Search Algorithms - Applications for Pure Science and Industry [Working Title]*, 2023. doi: 10.5772/intechopen.112067.

[13] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, and A. Cosar, "A survey on new generation metaheuristic algorithms," *Comput. Ind. Eng.*, 2019, doi: 10.1016/j.cie.2019.106040.

[14] A. Hussain and Y. S. Muhammad, "Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator," *Complex Intell. Syst.*, 2020, doi: 10.1007/s40747-019-0102-7.

[15] E. Alba and B. Dorronsoro, "The exploration/exploitation tradeoff in dynamic cellular genetic algorithms," *IEEE Trans. Evol. Comput.*, 2005, doi: 10.1109/TEVC.2005.843751.

[16] F. Vafaee, G. Turán, P. C. Nelson, and T. Y. Berger-Wolf, "Balancing the exploration and exploitation in an adaptive diversity guided genetic algorithm," in *Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014*, 2014. doi: 10.1109/CEC.2014.6900257.

[17] H. Zhang, J. Sun, T. Liu, K. Zhang, and Q. Zhang, "Balancing exploration and exploitation in multiobjective evolutionary optimization," *Inf. Sci. (Ny).*, 2019, doi: 10.1016/j.ins.2019.05.046.

[18] H. Yang, Z. Yang, Y. Yang, and L. Zhang, "An improved particle swarm optimization algorithm based on simulated annealing," in *2014 10th International Conference on Natural Computation, ICNC 2014*, 2014. doi: 10.1109/ICNC.2014.6975891.

[19] R. K. Kincaid and A. Ninh, "Simulated Annealing," in *Springer Optimization and Its Applications*, 2023. doi: 10.1007/978-3-031-38310-6_10.

[20] K. Amine, "Multiobjective Simulated Annealing: Principles and Algorithm Variants," 2019. doi: 10.1155/2019/8134674.

[21] S. Chen, C. Xudiera, and J. Montgomery, "Simulated annealing with thresheld convergence," in *2012 IEEE Congress on Evolutionary Computation, CEC 2012*, 2012. doi: 10.1109/CEC.2012.6256591.

[22] S. Singer and S. Singer, "Efficient Implementation of the Nelder–Mead Search Algorithm," *Appl. Numer. Anal. Comput. Math.*, 2004, doi: 10.1002/anac.200410015.

[23] Bilal, M. Pant, H. Zaheer, L. Garcia-Hernandez, and A. Abraham, "Differential Evolution: A review of more than two decades of research," *Eng. Appl. Artif. Intell.*, 2020, doi: 10.1016/j.engappai.2020.103479.

[24] M. G. Epitropakis, V. P. Plagianakos, and M. N. Vrahatis, "Balancing the exploration and exploitation capabilities of the Differential Evolution Algorithm," in *2008 IEEE Congress on Evolutionary Computation, CEC 2008*, 2008. doi: 10.1109/CEC.2008.4631159.

[25] L. Wang, X. Zhou, T. Xie, J. Liu, and G. Zhang, "Adaptive Differential Evolution with Information Entropy-Based Mutation Strategy," *IEEE Access*, 2021, doi: 10.1109/ACCESS.2021.3119616.

[26] Â. A. R. Sá, A. O. Andrade, A. B. Soares, and S. J. Nasuto, "Exploration vs. Exploitation in differential evolution," in *AISB 2008 Convention: Communication, Interaction and Social Intelligence - Proceedings of the AISB 2008 Symposium on Swarm Intelligence Algorithms and Applications*, 2008.

[27] Y. Zhang, G. Chen, L. Cheng, Q. Wang, and Q. Li, "Methods to balance the exploration and exploitation in Differential Evolution from different scales: A survey," *Neurocomputing*, 2023, doi: 10.1016/j.neucom.2023.126899.

[28] Z. Cai, X. Yang, M. C. Zhou, Z. H. Zhan, and S. Gao, "Toward explicit control between exploration and exploitation in evolutionary algorithms: A case study of differential evolution," *Inf. Sci. (Ny).*, 2023, doi: 10.1016/j.ins.2023.119656.

[29] M. N. M. Salleh *et al.*, "Exploration and exploitation measurement in swarm-based metaheuristic algorithms: An empirical analysis," in *Advances in Intelligent Systems and Computing*, 2018. doi: 10.1007/978-3-319-72550-5_3.

[30] A. M. Jabbar, "Controlling the Balance of Exploration and Exploitation in ACO Algorithm," *J. Univ. BABYLON Pure Appl. Sci.*, 2018, doi: 10.29196/jub.v26i4.678.
[31] Y. Liu, B. Cao, and H. Li, "Improving ant colony optimization algorithm with epsilon greedy and Levy flight," *Complex Intell. Syst.*, 2021, doi: 10.1007/s40747-020-00138-3.
[32] T. Kumazawa, M. Takimoto, and Y. Kambayashi, "Exploration Strategies for Model Checking with Ant Colony Optimization," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2021. doi: 10.1007/978-3-030-88081-1_20.
[33] M. Gendreau, "An Introduction to Tabu Search," in *Handbook of Metaheuristics*, 2006. doi: 10.1007/0-306-48056-5_2.
[34] R. Chandran, S. Rakesh Kumar, and N. Gayathri, "Genetic algorithm-based tabu search for optimal energy-aware allocation of data center resources," *Soft Comput.*, 2020, doi: 10.1007/s00500-020-05240-9.
[35] S. Hanafi, Y. Wang, F. Glover, W. Yang, and R. Hennig, "Tabu search exploiting local optimality in binary optimization," *Eur. J. Oper. Res.*, 2023, doi: 10.1016/j.ejor.2023.01.001.
[36] A. G. Gad, "Particle Swarm Optimization Algorithm and Its Applications: A Systematic Review," *Arch. Comput. Methods Eng.*, 2022, doi: 10.1007/s11831-021-09694-4.
[37] K. J. Binkley and M. Hagiwara, "Balancing exploitation and exploration in particle swarm optimization: Velocity-based reinitialization," *Trans. Japanese Soc. Artif. Intell.*, 2008, doi: 10.1527/tjsai.23.27.
[38] B. Nakisa, M. N. Rastgoo, and M. J. Norodin, "Balancing exploration and exploitation in particle swarm optimization on search tasking," *Res. J. Appl. Sci. Eng. Technol.*, 2014, doi: 10.19026/rjaset.8.1117.
[39] M. Ben Ghalia, "Particle swarm optimization with an improved exploration-exploitation balance," in *Midwest Symposium on Circuits and Systems*, 2008. doi: 10.1109/MWSCAS.2008.4616910.
[40] Y. Zhang and X. Kong, "A particle swarm optimization algorithm with empirical balance strategy," *Chaos, Solitons Fractals X*, 2023, doi: 10.1016/j.csfx.2022.100089.
[41] W. Liu, Z. Wang, N. Zeng, Y. Yuan, F. E. Alsaadi, and X. Liu, "A novel randomised particle swarm optimizer," *Int. J. Mach. Learn. Cybern.*, 2021, doi: 10.1007/s13042-020-01186-4.
[42] T. Stützle and R. Ruiz, "Iterated local search," in *Handbook of Heuristics*, 2018. doi: 10.1007/978-3-319-07124-4_8.
[43] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search: Framework and applications," in *International Series in Operations Research and Management Science*, 2019. doi: 10.1007/978-3-319-91086-4_5.
[44] H. N. K. Al-Behadili, K. R. Ku-Mahamud, and R. Sagban, "Hybrid ant colony optimization and iterated local search for rules-based classification," *J. Theor. Appl. Inf. Technol.*, 2020.
[45] F. Zhao, X. He, G. Yang, W. Ma, C. Zhang, and H. Song, "A hybrid iterated local search algorithm with adaptive perturbation mechanism by success-history based parameter adaptation for differential evolution (SHADE)," *Eng. Optim.*, 2020, doi: 10.1080/0305215X.2019.1595611.
[46] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Glob. Optim.*, 2007, doi: 10.1007/s10898-007-9149-x.
[47] E. Kaya, B. Gorkemli, B. Akay, and D. Karaboga, "A review on the studies employing artificial bee colony algorithm to solve combinatorial optimization problems," 2022. doi: 10.1016/j.engappai.2022.105311.
[48] W. F. Gao and S. Y. Liu, "A modified artificial bee colony algorithm," *Comput. Oper. Res.*, 2012, doi: 10.1016/j.cor.2011.06.007.
[49] G. Zhu and S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function optimization," *Appl. Math. Comput.*, 2010, doi: 10.1016/j.amc.2010.08.049.
[50] P. W. Tsai, J. S. Pan, B. Y. Liao, and S. C. Chu, "Enhanced artificial bee colony optimization," in *International Journal of Innovative Computing, Information and Control*, 2009.
[51] W. sheng Xiao, G. xin Li, C. Liu, and L. ping Tan, "A novel chaotic and neighborhood search-based artificial bee colony algorithm for solving optimization problems," *Sci. Rep.*, 2023, doi: 10.1038/s41598-023-44770-8.
[52] K. M. Passino, "Biomimicry of Bacterial Foraging for Distributed Optimization and Control," *IEEE Control Syst.*, 2002, doi: 10.1109/MCS.2002.1004010.
[53] C. Guo, H. Tang, B. Niu, and C. Boon Patrick Lee, "A survey of bacterial foraging optimization," *Neurocomputing*, 2021, doi: 10.1016/j.neucom.2020.06.142.
[54] M. Chen, Y. Ou, X. Qiu, and H. Wang, "An Effective Bacterial Foraging Optimization Based on Conjugation and Novel Step-Size Strategies," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2020. doi: 10.1007/978-3-030-57884-8_32.
[55] H. Chen, Y. Zhu, and K. Hu, "Self-adaptation in bacterial foraging optimization algorithm," in *Proceedings of 2008 3rd International Conference on Intelligent System and Knowledge Engineering, ISKE 2008*, 2008. doi: 10.1109/ISKE.2008.4731080.
[56] B. Pang, Y. Song, C. Zhang, H. Wang, and R. Yang, "Bacterial foraging optimization based on improved chemotaxis process and novel swarming strategy," *Appl. Intell.*, 2019, doi: 10.1007/s10489-018-1317-9.
[57] X. S. Yang, "A new metaheuristic Bat-inspired Algorithm," in *Studies in Computational Intelligence*, 2010. doi:

10.1007/978-3-642-12538-6_6.
[58] T. Agarwal and V. Kumar, "A Systematic Review on Bat Algorithm: Theoretical Foundation, Variants, and Applications," 2022. doi: 10.1007/s11831-021-09673-9.
[59] A. H. Gandomi and X. S. Yang, "Chaotic bat algorithm," *J. Comput. Sci.*, 2014, doi: 10.1016/j.jocs.2013.10.002.
[60] X. S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings*, 2009. doi: 10.1109/NABIC.2009.5393690.
[61] A. Sharma, A. Sharma, V. Chowdary, A. Srivastava, and P. Joshi, "Cuckoo search algorithm: A review of recent variants and engineering applications," in *Studies in Computational Intelligence*, 2021. doi: 10.1007/978-981-15-7571-6_8.
[62] H. Malik, A. Iqbal, P. Joshi, S. Agrawal, and I. B. Farhad, *Metaheuristic and Evolutionary Computation : Algorithms and Applications*. 2021.
[63] S. K. Nayak, B. R. Senapati, and D. Mishra, "Balancing exploration and exploitation: Unleashing the adaptive power of automatic cuckoo search for meta-heuristic optimization," *Intell. Decis. Technol.*, 2024, doi: 10.3233/IDT-idt230275.
[64] R. Salgotra, U. Singh, and S. Saha, "New cuckoo search algorithms with enhanced exploration and exploitation properties," *Expert Syst. Appl.*, 2018, doi: 10.1016/j.eswa.2017.11.044.
[65] G. Kanagaraj, S. G. Ponnambalam, and N. Jawahar, "A hybrid cuckoo search and genetic algorithm for reliability-redundancy allocation problems," *Comput. Ind. Eng.*, 2013, doi: 10.1016/j.cie.2013.08.003.
[66] K. Hussain, M. N. Mohd Salleh, Y. A. Prasetyo, and S. Cheng, "Personal best Cuckoo search algorithm for global optimization," *Int. J. Adv. Sci. Eng. Inf. Technol.*, 2018, doi: 10.18517/ijaseit.8.4.5009.
[67] Z. Huang, Z. Gao, L. Qi, and H. Duan, "A Heterogeneous Evolving Cuckoo Search Algorithm for Solving Large-Scale Combined Heat and Power Economic Dispatch Problems," *IEEE Access*, 2019, doi: 10.1109/ACCESS.2019.2933980.
[68] X. S. Yang, "Firefly algorithm, stochastic test functions and design optimization," *Int. J. Bio-Inspired Comput.*, 2010, doi: 10.1504/IJBIC.2010.032124.
[69] X. S. Yang and X. He, "Firefly algorithm: recent advances and applications," *Int. J. Swarm Intell.*, 2013, doi: 10.1504/ijsi.2013.055801.
[70] A. H. Gandomi, X. S. Yang, S. Talatahari, and A. H. Alavi, "Firefly algorithm with chaos," *Commun. Nonlinear Sci. Numer. Simul.*, 2013, doi: 10.1016/j.cnsns.2012.06.009.
[71] I. Brajević and P. Stanimirović, "An improved chaotic firefly algorithm for global numerical optimization," *Int. J. Comput. Intell. Syst.*, 2018, doi: 10.2991/ijcis.2018.25905187.
[72] I. Brajević and J. Ignjatović, "An upgraded firefly algorithm with feasibility-based rules for constrained engineering optimization problems," *J. Intell. Manuf.*, 2019, doi: 10.1007/s10845-018-1419-6.
[73] M. Sababha, M. Zohdy, and M. Kafafy, "The enhanced firefly algorithm based on modified exploitation and exploration mechanism," *Electron.*, 2018, doi: 10.3390/electronics7080132.
[74] J. A. Villaruz, B. D. Gerardo, A. O. Gamao, and R. P. Medina, "Scouting Firefly Algorithm and its Performance on Global Optimization Problems," *Int. J. Adv. Comput. Sci. Appl.*, 2023, doi: 10.14569/IJACSA.2023.0140350.
[75] G. Li, P. Liu, C. Le, and B. Zhou, "A novel hybrid meta-heuristic algorithm based on the cross-entropy method and firefly algorithm for global optimization," *Entropy*, 2019, doi: 10.3390/e21050494.
[76] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," *Adv. Eng. Softw.*, 2014, doi: 10.1016/j.advengsoft.2013.12.007.
[77] Y. Liu, A. As'arry, M. K. Hassan, A. A. Hairuddin, and H. Mohamad, "Review of the grey wolf optimization algorithm: variants and applications," *Neural Comput. Appl.*, 2024, doi: 10.1007/s00521-023-09202-8.
[78] N. Mittal, U. Singh, and B. S. Sohi, "Modified Grey Wolf Optimizer for Global Engineering Optimization," *Appl. Comput. Intell. Soft Comput.*, 2016, doi: 10.1155/2016/7950348.
[79] M. Kohli and S. Arora, "Chaotic grey wolf optimization algorithm for constrained optimization problems," *J. Comput. Des. Eng.*, 2018, doi: 10.1016/j.jcde.2017.02.005.
[80] V. K. R. Aala Kalananda and V. L. N. Komanapalli, "Hybrid evolutionary grey wolf optimizer for constrained engineering problems and multi-unit production planning," *Evol. Intell.*, 2024, doi: 10.1007/s12065-024-00909-8.
[81] S. Joshi and J. C. Bansal, "Grey Wolf gravitational search algorithm," in *IWCI 2016 - 2016 International Workshop on Computational Intelligence*, 2017. doi: 10.1109/IWCI.2016.7860371.
[82] N. Singh and H. Hachimi, "A New Hybrid Whale Optimizer Algorithm with Mean Strategy of Grey Wolf Optimizer for Global Optimization," *Math. Comput. Appl.*, 2018, doi: 10.3390/mca23010014.
[83] X. Yu, W. Y. Xu, X. Wu, and X. Wang, "Reinforced exploitation and exploration grey wolf optimizer for numerical and real-world optimization problems," *Appl. Intell.*, 2022, doi: 10.1007/s10489-021-02795-4.
[84] G. Shial, S. Sahoo, and S. Panigrahi, "An Enhanced GWO Algorithm with Improved Explorative Search Capability for Global Optimization and Data Clustering," *Appl. Artif. Intell.*, 2023, doi: 10.1080/08839514.2023.2166232.
[85] W. Long, J. Jiao, X. Liang, S. Cai, and M. Xu, "A Random Opposition-Based Learning Grey Wolf Optimizer,"

IEEE Access, 2019, doi: 10.1109/ACCESS.2019.2934994.

[86] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A New Heuristic Optimization Algorithm: Harmony Search," *Simulation*, 2001, doi: 10.1177/003754970107600201.

[87] A. Sadollah, H. Sayyaadi, D. G. Yoo, H. M. Lee, and J. H. Kim, "Mine blast harmony search: A new hybrid optimization method for improving exploration and exploitation capabilities," *Appl. Soft Comput. J.*, 2018, doi: 10.1016/j.asoc.2018.04.010.

[88] A. A. Alomoush, A. R. A. Alsewari, K. Z. Zamli, A. Alrosan, W. Alomoush, and K. Alissa, "Enhancing three variants of harmony search algorithm for continuous optimization problems," *Int. J. Electr. Comput. Eng.*, 2021, doi: 10.11591/ijece.v11i3.pp2343-2349.

[89] J. Wang, H. Ouyang, Z. Zhou, and S. Li, "Harmony Search Algorithm Based on Dual-Memory Dynamic Search and Its Application on Data Clustering," *Complex Syst. Model. Simul.*, 2023, doi: 10.23919/CSMS.2023.0019.

[90] W. Wu, H. Ouyang, A. W. Mohamed, C. Zhang, and S. Li, "Enhanced harmony search algorithm with circular region perturbation for global optimization problems," *Appl. Intell.*, 2020, doi: 10.1007/s10489-019-01558-6.

[91] S. Mirjalili and A. Lewis, "The Whale Optimization Algorithm," *Adv. Eng. Softw.*, 2016, doi: 10.1016/j.advengsoft.2016.01.008.

[92] X. Wu, S. Zhang, W. Xiao, and Y. Yin, "The Exploration/Exploitation Tradeoff in Whale Optimization Algorithm," *IEEE Access*, 2019, doi: 10.1109/ACCESS.2019.2938857.

[93] N. Rana, M. S. A. Latiff, S. M. Abdulhamid, and H. Chiroma, "Whale optimization algorithm: a systematic review of contemporary applications, modifications and developments," 2020. doi: 10.1007/s00521-020-04849-z.

[94] J. Du, J. Hou, H. Wang, and Z. Chen, "Application of an improved whale optimization algorithm in time-optimal trajectory planning for manipulators," *Math. Biosci. Eng.*, 2023, doi: 10.3934/mbe.2023728.

[95] R. Xu, C. Zhao, J. Li, J. Hu, and X. Hou, "A Hybrid Improved-Whale-Optimization–Simulated-Annealing Algorithm for Trajectory Planning of Quadruped Robots," *Electron.*, 2023, doi: 10.3390/electronics12071564.

[96] T. Tian, Z. Liang, Y. Wei, Q. Luo, and Y. Zhou, "Hybrid Whale Optimization with a Firefly Algorithm for Function Optimization and Mobile Robot Path Planning," *Biomimetics*, 2024, doi: 10.3390/biomimetics9010039.