SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

A Systematic Survey on Foundational Swarm Intelligence: Comparative Mechanics, Applications, and Limitations of ACO and BCO

Ajay Kumar[1], Partibha Yadav[1], Palak[2], Mamta Yadav[3], Parshant Bansal[1], Jogender Singh Yadav[1], Sandeep[4], Jai Bhagwan[5*]

[1]Department of Computer Science & Engineering, Indira Gandhi University, Meerpur, Haryana, India.
[2]Department of Computer Science, Govt. College, Narnaul, Haryana, India.
[3]Department of Computer Science and Information Technology, Central University of Haryana, Mahendergarh, India.
[4]Department of Artificial Intelligence & Data Science, Guru Jambheshwar University of Science & Technology, Hisar, India.
[5]Department of Computer Science & Engineering, Guru Jambheshwar University of Science & Technology, Hisar, India.
*Corresponding Author: drjaicse@gmail.com

Abstract
Nowadays, swarm intelligence (SI) has become a significant concept in artificial intelligence. It refers to the idea that the collective behavior of self-directed systems can be used to solve challenging NP-hard optimization problems. A variety of research has been conducted recently to develop more efficient SI algorithms, and new metaheuristics have been introduced to handle distinct optimization tasks. However, most of the research effort is concentrated on ant colony optimization (ACO) and bee colony optimization (BCO). This article aims to provide an overview of these two approaches to make a comprehensive comparison between them. We focus on ACO's pheromone-based stigmergy and BCO's division of labor and recruitment mechanisms. Furthermore, we provide a detailed theoretical analysis of both algorithms. We compare their convergence, performance, and complexity and evaluate their applications and performance in practice (e.g., ACO-tagger demonstrated 96.87 % accuracy in Part-of-Speech tagging and ABC-RNN achieved 98.6% performance in financial fraud detection). Finally, we discuss their limitations and directions for future research to advance swarm intelligence research, including hybrid approaches that combine classic SI algorithms with state-of-the-art methods in machine learning.

Keywords: Ant Colony Optimization, Bee Colony Optimization, Swarm Intelligence, Combinatorial Optimization, Stigmergy, Dynamic Path Planning, Parameter Sensitivity.

1. Introduction

The term Swarm Intelligence was introduced by Gerardo Beni and Jing Wang in 1989 while working on cellular robotic systems [1]. Since then, it has grown to be one of the most important areas of study in the field of computational intelligence. It captures the behaviors of decentralized, self-organized systems of individuals who work together by communicating locally. Everyone in the system interacts with its environment in a way that leads to the emergence of higher-level patterns of intelligence and problem solving despite the lack of any central oversight.

There are two biological principles underlying the study of such systems: self-organization (or self-regulation) and stigmergy.

- **Self-organization** involves four elements: positive feedback (amplification), negative feedback (saturation or damping), randomness (exploration), and multiple interactions.
- **Stigmergy** is indirect communication via modifying the environment, where an action by an agent leaves a trace that encourages or induces another agent to perform a subsequent action [5].

ACO and BCO / ABC belong to the most prominent algorithms of the swarm intelligence paradigm which is one of the alternative approaches to distributed problem solving. The research on SI methods has produced a number of heuristics, the most successful among which are Ant Colony Optimization introduced by Marco Dorigo in 1991 and Bee Colony Optimization / Artificial Bee Colony developed by Dervis Karaboga et al.. While most of the other SI algorithms concentrate on real-number optimization (such as aerodynamic shape design, oceanic turbulence prediction, or animal school foraging path prediction), ACO and BCO act primarily in the discrete space of graphs and combinatorial optimization [2]. ACO was inspired by ants' pheromone trail laying/trailing behavior, and it is especially effective for finding shortest paths in graphs, while BCO simulates the collective behavior of hives and effectively explores the search space using scouts, employed bees, and onlookers [3] [4].

However, despite their popularity in the scientific community and successful applications in various industries, there are some drawbacks and unresolved issues. ACO's drawbacks include massive

computational complexity (due to maintaining pheromone trails) and its ability to stagnate on local optima for large graphs. BCO has fewer control parameters, but it may be too slow in late stages of local search [6]. Moreover, for both algorithms, setting the initial parameters (for instance, alpha, beta, and rho for ACO, and the limit for abandoned solutions for BCO) is crucial and may be highly non-trivial depending on the problem statement.

Objectives & Contributions of this Paper

In this paper, the authors perform a systematic review of the two algorithms in question that highlights their main differences, challenges, and application areas.

- Present mathematical formalisms and algorithmic implementations for both the standard ACO algorithm and BCO / ABC variants with sufficient detail to understand their mathematical principles and operational flows
- Compare the differences in graph traversal mechanisms and parameter sensitivity between the options,
- determining which is superior in terms of computational complexity (roughly within O(N)).
- Provide real-world application examples (Dynamic TSP, Permutation Flow Shop, Fraud Detection, AVR), analyzing actual performance differences.
- Discuss the limitations of the algorithms in optimizing discrete problems and suggest possible combinations of different metaheuristics to overcome them.

2. Ant Colony Optimization (ACO)

2.1 Biological Inspiration

Ant Colony Optimization is based on the behavior of a real-world ant colony. When searching for food, ants leave behind a trail of pheromone, a scent, on the ground, in an unconscious manner. Other ants that are searching for food encounter these scent trails with higher concentration and follow them [6]. Since it takes less time to travel through shorter paths, more ants are likely to traverse and reinforce these paths with pheromone, increasing their concentrations. Additionally, pheromone evaporates as time goes. Thus, paths that are not frequently travelled lose their scent trail due to evaporation. The combination of these two effects leads to ants quickly converging to the shortest path.

2.2 Mathematical Model

ACO models optimization problems as graph traversal tasks where ants navigate through a set of nodes V connected by edges E.

1. Probabilistic Transition Rule [2, 7]

When an ant k is located at node i, the probability $p_{ij}^k(t)$ of selecting node j as its next destination at iteration t is defined as:

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta}, & \text{if } j \in N_i^k \\ 0, & \text{otherwise} \end{cases}$$

Where:

- $\tau_{ij}(t)$ is the pheromone concentration on edge (i, j) at iteration t.
- $\eta_{ij} = \frac{1}{d_{ij}}$ is the heuristic information (visibility), where d_{ij} is the distance/cost between nodes i and j.
- $\alpha \geq 0$ is a parameter that controls the relative influence of the pheromone trail.
- $\beta \geq 1$ is a parameter that controls the relative influence of the heuristic information (distance).
- N_i^k is the feasible neighborhood set for ant k (unvisited nodes directly accessible from i).

2. Pheromone Evaporation

To prevent early stagnation and allow the colony to explore alternative routes, pheromone levels evaporate on all edges at a constant rate ρ:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t)$$

Where:

- $\rho \in (0,1]$ is the pheromone evaporation rate.
- $\Delta\tau_{ij}(t)$ is the total amount of new pheromone deposited on edge (i, j) across all m ants in the colony.

3. Pheromone Deposition

The total pheromone added to edge (i, j) at iteration t is calculated as:

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^k(t)$$

Where $\Delta\tau_{ij}^k(t)$ represents the amount of pheromone deposited by ant k:

$$\Delta\tau_{ij}^k(t) = \begin{cases} \frac{Q}{L_k(t)}, & \text{if ant } k \text{ traversed edge } (i,j) \text{ in its path} \\ 0, & \text{otherwise} \end{cases}$$

Where:
- Q is a constant related to the total pheromone quantity.
- $L_k(t)$ is the total length or cost of the tour constructed by ant k in iteration t.

3. Bee Colony Optimization (BCO / ABC)

3.1 Biological Inspiration

Bee Colony Optimization (and its continuous variant called Artificial Bee Colony - ABC) is based on the foraging behavior of honeybees. It is a swarm intelligence optimization technique, as a swarm of bees can distribute their foragers among food sources, according to their quality (nectar amount and concentration) and distance from the hive [4].

Communication happens inside the hive, where employed bees can meet and exchange information through a waggle dance performed in a special dance area. This information-sharing process allows employed bees to recruit other individuals to rich food sources (the longer and harder the waggle dance is, the more rewarding the food patch is).

3.2 Role Division & Dynamics

The colony consists of three classes of bees

1. Bees that are working (employed). They are linked to food sources where they determine their quality and collect their resources by waggle dance.
2. Bees that are waiting in the hive to receive the information about successful food sources by watching the waggle dances of employed bees. That is why they are called onlooker bees, and they choose the place of food according to the received information.
3. Scout bees are those who have failed to find a food source again and again (more than the preset limit or abandonment threshold). Such bees change their status to scouts and randomly search for food sources.

3.3 Mathematical Formulation

In BCO, a food source location $X_i = (x_{i,1}, x_{i,2}, \dots \dots \dots D)$ represents a candidate solution to an optimization problem, where D is the dimensionality of the decision space. The nectar amount corresponds to the objective function value (fitness) of that solution [16, 18, 19, 20, 23].

1. Population Initialization

Initial food sources are generated uniformly at random within bounded upper (UB) and lower (LB) limits:

$$x_{i,j} = LB_j + \text{rand}(0,1) \cdot (UB_j - LB_j)$$

Where $i \in \{1,2, \dots, SN\}$ (SN is the number of food sources) and $j \in \{1,2, \dots, D\}$.

2. Employed Bee Phase (Neighborhood Local Search)

An employed bee modifies its current food position X_i to discover a neighbor solution V_i:

$$v_{i,j} = x_{i,j} + \phi_{i,j} \cdot (x_{i,j} - x_{k,j})$$

Where:
- $k \in \{1,2, \dots, SN\}$ is a randomly chosen food source index $k \neq i$.
- $j \in \{1,2, \dots, D\}$ is a randomly selected dimension index.
- $\emptyset_{i,j}$ is a random number generated uniformly in the range [-1, 1].

A **greedy selection** is then performed between X_i and V_i: if the fitness of V_i is higher, V_i replaces X_i; otherwise, X_i is retained, and its trial counter increases by 1.

3. Onlooker Bee Phase (Probabilistic Recruitment)

Onlooker bees evaluate the shared information from all employed bees and select a food source X_i with a probability p_i proportional to its fitness:

$$p_i = \frac{\mathbf{fit}_i}{\sum_{n=1}^{SN} \mathbf{fit}_n}$$

Where the fitness value fit_i is derived from the objective cost $f(X_i)$:

$$\text{fit}_i = \begin{cases} \dfrac{1}{1 + f(X_i)}, & \text{if } f(X_i) \geq 0 \\ 1 + |f(X_i)|, & \text{if } f(X_i) < 0 \end{cases}$$

Once an onlooker bee selects a food source X_i, it performs local search using the same neighbor update formula to find a new candidate solution V_i and applies greedy selection [25].

4. Scout Bee Phase (Abandonment & Re-initialization)

If a food source X_i cannot be improved after a predetermined parameter limit of consecutive trials, it is considered exhausted. The assigned employed bee abandons the source, transforms into a scout bee, and replaces X_i with a newly generated random solution:

$$x_{i,j} = LB_j + \text{rand}(0,1) \cdot (UB_j - LB_j)$$

4. Comparative Framework & Algorithmic Mechanics

While both Ant Colony Optimization (ACO) and Bee Colony Optimization (BCO / Artificial Bee Colony) are population-based and inspired by nature, they have fundamental differences in search space

representation, control structures, exploitation versus exploration trade-offs, and computational efficiency. The purpose of this section is to review and compare their basic mechanisms.

4.1 Search Space Topology: Graph Traversal vs. Solution Vector Manipulation

The main difference in design philosophy between ACO and BCO algorithms lies in the way the candidate solutions are formed in the search space [2], [11]:

1. **ACO (Graph-Based Discrete Traversal):** ACO is designed to solve TSPs in discrete space represented by a complete or sparse weighted graph G(V,E). The search space is a graph where agents move from node to node in a discrete state space by building solutions incrementally step by step along the edges until a complete path is formed. At each step, the agent makes a probabilistic choice using the learned pheromone trail and heuristic information. ACO algorithms cannot evaluate the solution before completing the entire traversal.
2. **BCO/ABC algorithms (Vector-Based Discrete/Continuous Search):** Search space is represented by a set of candidate solutions vectors $X_i \in R^D$ (discretized coordinates), with each vector corresponding to a food source. There is no adjacency or topology in the solution representation, and movement in search space is performed by applying the coordinate-wise neighborhood equations [3, 23] to modify the coordinates of the current solution vectors.

4.2 Exploitation vs. Exploration Dynamics

The key to maintaining a proper balance between exploration (global search) and exploitation (local search) is essential to avoid premature convergence to local optima and ensure convergence speed [7, 18].

1. Mechanism of Exploitation

- **ACO:** Exploitation is realized by positive feedback (stigmergy): ants reinforce edges they traverse by increasing the amount of pheromone on them. Therefore, the probability of choosing edges with a high number of pheromones increases; parameter α determines the level of exploitation.
- **BCO:** Exploitation is realized in the Employed Bee Phase and Onlooker Bee Phase. In the Employed Bee Phase, the bees perform a local search near their respective food source by using a greedy search method. In the Onlooker Bee Phase, the bees perform exploitation by intensifying the search around the best food sources found by the employed bees using the roulette wheel selection method.

2. Exploration Mechanics

- **ACO:** Exploration relies on two elements:

1. The heuristic parameter β (encouraging unexplored short-distance edges).
2. **Pheromone Evaporation ρ**, which acts as negative feedback by steadily decaying unused trails and preventing early lock-in to suboptimal paths.

- **BCO:** Exploration is governed mainly by the Scout Bee Phase. If a food source ceases to yield improvement after a certain number of iterations (limit), an entirely new solution is generated uniformly at random from the search space to replace it. Exploration is therefore explicitly controlled through an inertia-type mechanism that prevents premature convergence to local optima.

4.3 Control Parameters & Sensitivity Analysis

The performance of both algorithms depends heavily on initial parameter configuration. However, their sensitivity profiles and tuning complexity differ significantly [1, 4, 5]:

Parameter Category	Ant Colony Optimization (ACO)	Bee Colony Optimization (BCO / ABC)
Primary Parameters	α (Pheromone influence) β (Heuristic visibility influence) ρ (Evaporation rate) Q (Pheromone deposit scale) **τ** (Initial pheromone level) m (Number of ants)	SN (Number of food sources / Employed bees) Limit (Abandonment threshold) maxIter (Max iterations)
Parameter Tuning Overhead	**High:** ACO requires tuning interdependent continuous variables (α, β, ρ). Poor choices lead directly to premature convergence (α >> β) or random search behavior (ρ → 1).	**Low:** BCO uses fewer key parameters. Setting Limit = SN * D works well across diverse benchmarks.
Adaptability	Sensitive to static parameter setups; often requires adaptive schemes (e.g., MAX-MIN Ant System) for stability.	Inherently robust due to self-balancing scout bee re-initialization.

4.4 Time and Space Computational Complexity

To evaluate computational efficiency, let N be the number of nodes/variables, m the number of agents (ants or total bees 2 * SN), D the problem dimension, and I the maximum iteration count.

1. Time Complexity

- **ACO:** In each iteration, m ants construct a tour of length N. Choosing the next node requires calculating probabilities across up to N unvisited neighbors, taking $O(N^2)$ operations per ant. Adding the pheromone update step yields an overall time complexity per iteration of:

$$\mathcal{O}\big(I \cdot (m \cdot N^2 + |E|)\big)$$

For dense graphs where $|E| \approx N^2$, this scale becomes computationally expensive for large N.

- **BCO:** In each iteration, SN employed bees and SN onlooker bees evaluate a single dimension modification per food source (O(D)). The scout phase checks food source limits in O(SN) time. The overall time complexity per iteration is:

$$\mathcal{O}(I \cdot SN \cdot D)$$

Since SN and D scale linearly, BCO is generally less computationally demanding per iteration than ACO on large-scale problems.

2. Space Complexity

- **ACO:** ACO must maintain a pheromone matrix and heuristic distance matrix for all edge pairs in memory, resulting in a space complexity of:

$$\mathcal{O}(|V|^2 + m \cdot N)$$

- **BCO:** BCO only needs to store the coordinate positions, fitness values, and trial counters for SN food sources, leading to a much lower space complexity of:

$$\mathcal{O}(SN \cdot D)$$

5. Application Domains & Benchmark Performance Analysis

Both Ant Colony Optimization (ACO) and Bee Colony Optimization (BCO/ABC) have been successfully transitioned from proof-of-concept metaheuristics to practical industrial applications. However, the fundamental nature of their algorithmic structure favors one optimization category over the other; ACO being more effective for graph-discrete routing/probability tasks, while BCO is used for continuous numerical optimization, assignment, and rule-based prediction modeling.

5.1 Routing, Logistics, and Path Planning

Routing Problems are a natural extension of ACO's original theoretical basis of graph traversal through pheromone trail laying.

1. Dynamic Traveling Salesman Problem / Vehicle Routing

The original Traveling Salesman Problem (TSP) is a purely static case of finding a Hamiltonian cycle of shortest possible length. However, in dynamic graph variants, the distance between graph edge nodes takes on different values over time, such as in cases of traffic jams or additional delivery requests. Conventional ACO methods fail to account for these sudden changes to the graph distance matrix, as the search is biased towards shorter paths with higher pheromone concentrations. A hybrid approach of ACO + Simulated Annealing (SA) allows a reset of the local search space whenever a sudden change to the graph occurs. The hybrid ACO-SA method was found to have a significantly shorter convergence time and better path cost results than conventional Genetic Algorithms and pure ACO methods [2].

2. Mobile Robot Path Planning

In the application of path planning for mobile robots, the BCO algorithm has been adapted for use in static and dynamic environments alike by conceptualizing the path itself as a Bezier curve waypoint array or control point vector. Employed bees adjust the path coordinates to avoid collisions, while onlooker bees search the solution space for high-fitness paths with minimal curvature and distance [23, 25].

5.2. Industrial and Operations Scheduling

Scheduling operations usually involve assigning jobs to processing agents while satisfying a set of constraints.

1. Permutation Flow Shop Scheduling

In the case of Permutation Flow Shop (PFSP) scheduling, jobs are assigned to processing machines in a specific order, resulting in the makespan (total production time) being dependent on the sequence of jobs. ACO approaches to PFSP involve the sequence itself being encoded as the path, with ants laying pheromone trails on the jobs that they visited previously. MAX-MIN Ant System approaches were found to be the most effective at avoiding premature convergence and finding near-optimal solutions for industry-standard Taillard test cases [18].

2. Group Shop Scheduling / Medical Shift Allocation

A particularly challenging scheduling case is medical workers' shift allocation, which usually has many hard and soft constraints, including worker availability, skill level, staff rotation rules, and shift limits per worker. ACO's ability to find solutions in such a constrained solution space makes it a good fit for the medical scheduling problem. In cases where the constraints are too difficult to encode as separate rules, they can be embedded into the heuristic guidance η_{ij} for the ants' path selection probability function.

5.3. NLP: ACO-based POS-Tagging

In Natural Language Processing, Part-of-Speech (POS) tagging involves 372abelling the part of speech for each word in each text, either as a noun, adjective, verb, etc. Using ACO for POS-tagging involves encoding the possible parts of speech as the solution graph's state space. An ACO-based POS tagger (ACO-tagger) was

tested on standard linguistic datasets and found to have a tag accuracy of 96.867%. The method was particularly effective at ambiguous cases and out-of-dictionary word tagging, where Hidden Markov Models usually underperformed.

5.4. Financial Security / Predictive Analytics: ABC-RNN Fraud Detection

The task of fraud detection involves distinguishing between genuine and fraudulent transactions, usually in the case of credit card or insurance fraud. Fraudulent transactions usually have significantly fewer observed features than genuine ones, resulting in the classes being highly imbalanced and non-linearly separated. As such, using a deep Recurrent Neural Network (RNN) to classify transactions is challenging, as the optimization function can get stuck in local minima or have exploding/vanishing gradients. The ABC-RNN approach uses the Artificial Bee Colony (ABC) algorithm to find the optimal weight configuration for an RNN.

1. Fraud Detection with ABC-RNN

The weight and bias vector of the RNN is encoded as the food source vector X. The employed bees tweak the weights of the neural network to find better minima in the loss function, while the onlooker bees reinforce the best-performing configurations. If the search process gets stuck in a local minimum, the scout bees reset the configuration vector X and restart the search process. The ABC-RNN approach was tested on three standard credit insurance fraud datasets from the UC Irvine repository, with the following results: Insurance Fraud – 98.60%, Mortgage Fraud – 97.92%, Credit Card Fraud – 96.03% accuracy.

Both Ant Colony Optimization (ACO) and Bee Colony Optimization (BCO / ABC) have transitioned from theoretical metaheuristics to robust industrial problem solvers. However, their structural differences dictate their suitability across distinct real-world domains. ACO dominates graph-based discrete routing and permutation scheduling problems, while BCO excels in continuous numerical optimization, complex feature assignment, and rule-based predictive classification models.

5.5 Quantitative Performance Matrix

Table 1 summarizes the empirical performance metrics, benchmark domains, and key structural advantages of ACO and BCO/ABC across the reviewed literature [2, 13, 20, 24].

Table 1: Application Domain & Benchmark Performance Summary for ACO and BCO

Algorithm	Domain / Benchmark Application	Underlying Dataset / Benchmark Setup	Key Quantitative Performance Metric	Core Advantage Demonstrated
ACO	Part-of-Speech (POS) Tagging	Standard Text Corpus (ACO-tagger)	**96.867% Accuracy**	Effective disambiguation in discrete contextual path selection.
ACO	Dynamic Traveling Salesman (DTSP)	Dynamic Graph Benchmarks (Hybrid ACO-SA)	Outperformed classical GA and basic ACO in path cost and convergence.	Temperature-driven pheromone adaptation prevents local stagnation.
ACO	Frequency Assignment & Scheduling	Permutation Flow Shop & Group-Shop Benchmarks	High global optimal hit rate in reasonable processing time.	Heuristic visibility integration allows easy embedding of soft/hard constraints.
BCO / ABC	Insurance Fraud Detection	UCI Financial Insurance Benchmark (ABC-RNN)	**98.60% Classification Accuracy**	Prevents neural network local minima traps via scout bee global re-initialization.
BCO / ABC	Mortgage Fraud Detection	UCI Mortgage Benchmark (ABC-RNN)	**97.92% Classification Accuracy**	High feature space exploration rate with minimal parameter tuning.
BCO / ABC	Credit Card Fraud Detection	UCI Credit Card Benchmark (ABC-RNN)	**96.03% Classification Accuracy**	Robust performance across dynamic, non-convex error surfaces.
BCO / ABC	Automatic Voltage Regulator (AVR) Systems	Optimal Power Flow / Dispatch Systems	Fast response times and low steady-state error	Strong parameter robustness in continuous control space tuning.

6. Critical Limitations & Open Research Challenges

Despite their versatility, ACO and BCO face structural bottlenecks when applied to ultra-large-scale or dynamic real-world problems:

- **Premature Convergence & Local Optima:**
  - **ACO (Pheromone Stagnation):** Early high-reward paths can trigger a positive feedback loop that over-concentrates pheromones on suboptimal edges, causing the colony to lock into local optima [6, 12].
  - **BCO (Late-Stage Slowdown):** Random coordinate perturbations in neighbor generation (X) lack gradient guidance, leading to slow local refinement near true global optima [14].
- **Scalability Bottlenecks on Large Topologies:**
  - **Memory Density:** Maintaining an N * N pheromone matrix in ACO requires $O(N^2)$ memory, creating compute bottlenecks on dense, large-scale graphs.
  - **Selection Overhead:** Calculating probabilistic transition rules across dense candidate sets slows execution without artificial neighborhood restrictions.
- **Hyperparameter Dependencies:**
  - **ACO Sensitivity:** The balance between history (α) and heuristic foresight (β) is fragile; poor ratios cause instant stagnation or blind greedy search.
  - **BCO Thresholding:** A small limit abandons promising search spaces prematurely, while a large limit wastes compute cycles on stagnant regions.

7. Future Perspectives & Hybrid Strategies

To address these challenges, recent work has proposed several avenues of research that combine swarm principles with modern machine learning techniques [10, 16, 19]:

- **Hybrid metaheuristics**: combine global search heuristics with local optimization procedures (e.g. Simulated Annealing, Tabu Search) to refine candidate solutions.
- **Machine learning guided adaptation**: use reinforcement learning to learn adaptation policies that modify parameters (e.g. α, β, ρ) and limits based on the dynamics of the search process.
- **Neural stigmergy:** replace pheromone tables with graph neural networks, which learn to encode predictions about the desirability of graph edges – including edges that have not been traversed during the search.
- **Hardware acceleration:** represent swarm operations as tensor contractions and utilize GPU acceleration to reduce computation time on large graphs.

8. Conclusion

This paper provides a systematic comparative overview of two solution approaches to optimization problems based on swarm intelligence. On the one hand, Ant Colony requires building an appropriate graph and following the trails of pheromones layer by layer before obtaining an acceptable solution. On the other hand, Bee Colony uses the principle of division of labour to find the best possible candidate solution vector, making use of both employed and unemployed bees and using their combined probabilities, to perturb the solutions in the entire search space.

Both algorithms were successfully applied to several tasks resulting in high accuracy rates, be it the ACO-tagger achieving 96.87% accuracy score in POS tagging or ABC-RNN reaching 98.60% in detecting financial fraud. Nevertheless, it was found out that each of the two swarming approaches also has its shortcomings, namely, ACO's ease of trapping in local optima and massive memory consumption (complexity of $O(N^2)$ for large graphs and BCO's occasional stagnation at the end of iterations.

Further research on the topic should be made about combining the two heuristics to offset their disadvantages, implementing the algorithms on GPUs for increased performance, and applying machine learning methods to determine optimal parameter configurations. Altogether, the current paper aimed to provide a comprehensive overview of the basic swarm-based optimization algorithms to facilitate researchers' choices in terms of implementation and parameter selection when tackling discrete and combinatorial problems.

References

[1] C. Wang, S. Zhang, T. MA, Y. Xiao, M. Z. Chen and L. Wang. Swarm intelligence: A survey of model classification and applications. *Chinese Journal of Aeronautics*, 38(3), 102982, 2025. doi.org/10.1016/j.cja.2024.03.019.

[2] P. Stodola, K. Michenka, J. Nohel, and M. Rybanský. Hybrid Algorithm Based on Ant Colony Optimization and Simulated Annealing Applied to the Dynamic Travelling Salesman Problem. *Entropy*, 22(8), 884, 2020. https://doi.org/10.3390/e22080884.

[3] N. Rahnema, and F. S. Gharehchopogh. An improved artificial bee colony algorithm based on whale optimization algorithm for data clustering. *Multimedia Tools and Applications*, 79(43), 32169-32194, 2020. https://doi.org/10.1007/s11042-020-09639-2.

[4] A. Sharma, S. Choudhary, R. K. Pachauri, A. Shrivastava, and D. Kumar. A review on artificial bee colony and it's engineering applications. *Journal of Critical Reviews*, 7(11), 4097-4107, 2020. DOI: 10.31838/jcr.07.11.558.

[5] J. Yang, L. Qu, Y. Shen, Y. Shi, S. Cheng, J. Zhao, and X. Shen. Swarm Intelligence in Data Science: Applications, Opportunities and Challenges. In *Proc. Advances in Swarm Intelligence: 11th International Conference, ICSI 2020*, Belgrade, Serbia, 12145: 3–14, 2020. https://doi.org/10.1007/978-3-030-53956-6_1.

[6] K. R. Kalantari, A. Ebrahimnejad, and H. Motameni. Efficient improved ant colony optimisation algorithm for dynamic software rejuvenation in web services. *The Institution of Engineering and Technology*, 14(4), 369-376, 2020. doi: 10.1049/iet-sen.2019.0018.

[7] Y. Liu, B. Cao, and H. Li. Improving ant colony optimization algorithm with epsilon greedy and Levy flight. *Complex Intell. Syst*, 7: 1711–1722, 2021. https://doi.org/10.1007/s40747-020-00138-3.

[8] A. M. Mayet, V. P. T. Ijyas, J. K. Bhutto, J. W. G. Guerrero, N. K. Shukla, E. Eftekhari-Zadeh, and H. H. -Alhashim. Using Ant Colony Optimization as a Method for Selecting Features to Improve the Accuracy of Measuring the Thickness of Scale in an Intelligent Control System. *Processes*, 11(6), 2023. https://doi.org/10.3390/pr11061621.

[9] A. Z. Ye, B. R. Li, C. W. Zhou, D. M. Wang, E. M. Mei, F. Z. Shu, and G. J. Shen. High-Dimensional Feature Selection Based on Improved Binary Ant Colony Optimization Combined with Hybrid Rice Optimization Algorithm. *International Journal of Intelligent Systems*, 2023. https://doi.org/10.1155/2023/1444938.

[10] J. M. Challab, and F. Mardukhi. Ant Colony Optimization–Rain Optimization Algorithm Based on Hybrid Deep Learning for Diagnosis of Lung Involvement in Coronavirus Patients. *Iran J Sci Technol Trans Electr Eng*, 47: 887–902, 2023. https://doi.org/10.1007/s40998-023-00611-y.

[11] A. Rezvanian, S. M. Vahidipour, and A. Sadollah. An Overview of Ant Colony Optimization Algorithms for Dynamic Optimization Problems. *IntechOpen*, 2023. doi: 10.5772/intechopen.111839.

[12] M. Faisal, and F. Albogamy. Ant Colony Optimization Algorithm Enhancement for Better Performance. *IEEE*, 2023, 0701-0710, 2023. doi: 10.1109/AIIoT58121.2023.10174442.

[13] A. Mohammadi, S. Hajiaghajani, and M. Bahrani. ACO-tagger: A Novel Method for Part-of-Speech Tagging using Ant Colony Optimization. *arXiv preprint arXiv*, 2303, 16760, 2023.

[14] T. Wen, H. Liu, L. Lin, B. Wang, J. Hou, C. Huang, T. Pan, and Y. Du. Multiswarm Artificial Bee Colony algorithm based on spark cloud computing platform for medical image registration. *Computer Methods and Programs in Biomedicine*, 192: 2020. doi: 10.1016/j.cmpb.2020.105432.

[15] A. K. Alazzawi, H. M. Rais, S. Basri, and Y. A. Alsariera. Pairwise Test Suite Generation Based on Hybrid Artificial Bee Colony Algorithm. *Lecture Notes in Electrical Engineering*, 619: 137-145, 2020.

[16] J. F. Farfán, and L. Cea. Coupling artificial neural networks with the artificial bee colony algorithm for global calibration of hydrological models. *Neural Computing and Applications*, 33(14), 8479-8494, 2021. https://doi.org/10.1007/s00521-020-05601-3.

[17] I. J. Jacob, and E. Darney. Artificial Bee Colony Optimization Algorithm for Enhancing Routing in Wireless Networks. *Journal of Artificial Intelligence and Capsule Networks*, 03(01), 62-71, 2021. DOI:https://doi.org/10.36548/jaicn.2021.1.006.

[18] X. Long, J. Zhang, K. Zhou, and T. Jin. Dynamic Self-Learning Artificial Bee Colony Optimization Algorithm for Flexible Job-Shop Scheduling Problem with Job Insertion. *Processes*, 10(3), 2022. https://doi.org/10.3390/pr10030571.

[19] E. Kaya. A New Neural Network Training Algorithm Based on Artificial Bee Colony Algorithm for Nonlinear System Identification. *Mathematics*, 10(19), 2022. https://doi.org/10.3390/math10193487.

[20] T. Karthikeyan, M. Govindarajan, and V. Vijayakumar. Intelligent Financial Fraud Detection Using Artificial Bee Colony Optimization Based Recurrent Neural Network. *Intelligent Automation & Soft Computing*, 37(2) 1483-1498, 2023. https://doi.org/10.32604/iasc.2023.037606.

[21] F. Iqbal, A. Raziq, Zil-E-Huma, C. Tirink, A. Fatih, and M. Yaqoob. Using the artificial bee colony technique to optimize machine learning algorithms in estimating the mature weight camels. *Trop Anim Health Prod*, 55(2), 2023. doi: 10.1007/s11250-023-03501-x.

[22] Y. Cui. Optimizing decision trees for English Teaching Quality Evaluation (ETQE) using Artificial Bee Colony (ABC) optimization. *Heliyon*, 9(8), 2023. doi: 10.1016/j.heliyon.2023.e19274.

[23] Y. Cui, W. Hu, and A. Rahmani. Fractional-order artificial bee colony algorithm with application in robot path planning. *European Journal of Operational Research*, 306(1), 47-64, 2023. https://doi.org/10.1016/j.ejor.2022.11.007.

[24] M. Sutar, and H.T. Jadhav. A modified artificial bee colony algorithm based on a non-dominated sorting genetic approach for combined economic-emission load dispatch problem. *Applied Soft Computing*, 144, 2023. https://doi.org/10.1016/j.asoc.2023.110433.