جامعة ستاردوم

STARDOM UNIVERSITY

Stardom 2nd Scientific Forum

towards Sustainable Future 2025

Stardom Scientific Journal of Natural and Engineering Sciences

3rd volume 2025- 2nd issue

ISSN: 2980-3756

اللجنة العلمية لمنتدى ستاردوم الدولي الثاني لمستقبل أكثر استدامة

Using Swarm Intelligence Algorithms to Boost AI Performance

Eng. Hassan Nedal Alawd

Mechanical Engineer and researcher specializing in the field of metal industries and military metal industries development, with a focus on integrating artificial intelligence to innovate and enhance industrial processes.

Declaration

1- This research is prepared as part of advancing knowledge and providing innovative solutions to enhance industries and transportation, with a focus on quality and scientific excellence.

2- This research has been prepared and dedicated sincerely for the sake of Allah Almighty, seeking His blessings and acceptance. It is permissible to copy and use this work for purposes of research, learning, and knowledge dissemination, provided that the original source is properly cited and acknowledged. However, it is strictly prohibited to use, copy, or reproduce this work for commercial purposes or financial gain under any circumstances.

Abstract

This research explores Swarm Intelligence algorithms, an advanced field in artificial intelligence that mimics the collective behavior of organisms like ants and bees to solve complex problems. These algorithms are characterized by decentralization, adaptability, and parallel processing, making them effective in optimization tasks.

The Ant Colony Optimization (ACO) algorithm models ant foraging behavior to solve path optimization problems, such as the Traveling Salesman Problem (TSP), showcasing scalability and gradual performance improvement. Similarly, the Bee Algorithm mimics bee foraging behavior for resource optimization and multi-dimensional search, offering efficiency in various applications.

A case study compares the performance of ACO and the Bee Algorithm in solving a real-world problem, such as optimizing transportation routes. The results evaluate their speed, accuracy, and ability to handle complexity, highlighting the practical potential of Swarm Intelligence in AI-driven solutions.

Keywords

Swarm intelligence, ant algorithm, bee algorithm, artificial intelligence, path optimisation, biomimicry, search optimisation, collective intelligence.

Introduction

Algorithms are widely used in mathematics and computer science Even with the tremendous development in computing capabilities, there are still various problems that are difficult to solve, especially combinatorial examples. This type of problem can arise in product design. For example, let's take the design of a car based on specifications: Engine power, number of seats, exterior shape and tyre size. If there are three levels (available options) for each of these specifications, as below, there will be 84 or 81 possible combinations to consider. Also, for five descriptors, for each specification with four levels, there are 1024 combinations (4*4*4*4*4*4) Typically, a huge number of possible combinations will appear even if the issues are relatively small. Finding the optimal solution to these problems by listing all possible combinations is impractical. Fortunately, intuitive search algorithms have evolved to find a suitable solution to these problems in a reasonable amount of time. This project (Using Swarm Intelligence Algorithms to Enhance AI Performance) is designed to highlight the most important algorithms used in organising, optimising and managing industrial processes and to define the concept of swarm intelligence and its importance in the development of AI systems.

1. Swarm Intelligence

Each insect in a colony seems to have its own agenda, yet the colony appears to be highly organised. The complete integration of all individual activities takes place without the need for any supervision. In fact, scientists studying the behaviour of social insects have found that cooperation at the colony level is primarily a matter of self-organisation. In many cases, coordination arises from individual interactions. Although these interactions may be as simple as one ant simply following a trail left by another, in aggregate they can solve difficult problems such as finding the shortest route to a food source out of countless paths. Similarly, the division of labour among honeybee workers can help make assembly lines in factories run more smoothly. This collective behaviour exhibited by a group of social insects or animals has been termed swarm intelligence.

Over the past decade or more, various intuitive techniques have been developed based on the observation of insects and animals. These techniques have been used to solve many problems successfully and extensively, including synthetic examples . Swarm Algorithm[A6] First introduced in 1995 by scientists Kennedy and Eberhardt, this algorithm was inspired by the behaviour of swarms of birds, fish and insects in moving from one place to another, just as these animals instinctively move in groups to migrate or search for food, this algorithm also assumes the existence of individuals with mathematical values (numbers or matrices), and these values change constantly in order to reach the optimal value or the optimal solution to a mathematical problem that is difficult to solve by the old traditional methods.

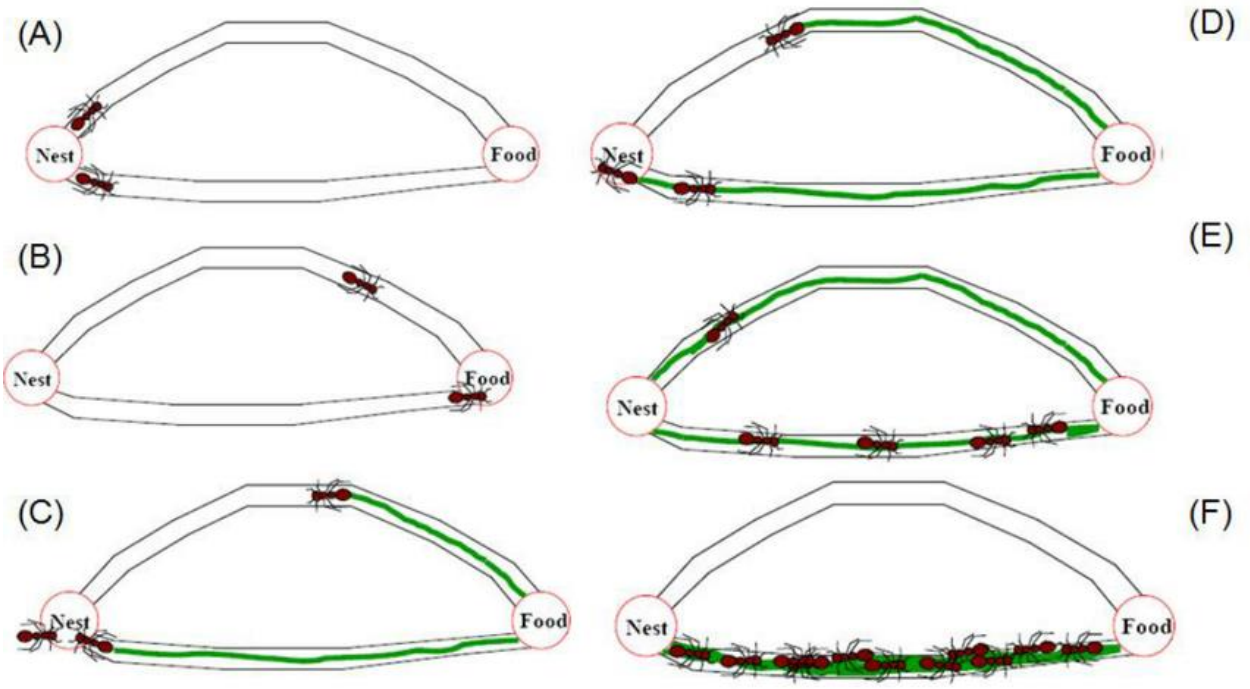
2. Ant colony optimisation algorithm

2.1. Introduction

Recently, a growing number of researchers have been interested in devising new ways to apply this ‘swarm intelligence’ to various tasks. For example, the way ants migrate for food has led to a novel way to re-plan traffic routes in congested networks of telecommunication systems, while the cooperative interaction of ants moving a large food crumb has led to more efficient algorithms for ants. The cooperative synergy between ants moving a large food crumb may lead to more effective algorithms for robots, and the way insects sort their larvae and clear the nest of dead bodies could help analyse banking data. Ants are self-organising in the sense that complex collective behaviour emerges from the interaction between individuals, each of which individually represents a simple behaviour. The consequence of self-organisation is that it is global in nature but comes from interaction that is entirely based on local information. Self-regulation relies on several components including positive feedback, negative feedback and multiple interactions. Negative feedback is conditioned by constraints on behaviour that are triggered by events such as the depletion of a food source; positive feedback is based on basic behavioural rules such as recruiting other insects to search for a food source that create the necessary structures for collective behaviour; and multiple interactions refer to the necessity of random events, such as ants getting lost but finding a new food source[A5].

2.2. Ant behaviour in nature

One of the earliest studies of swarm intelligence was the investigation of the foraging behaviour of ants. Dinoporek and his colleagues from the Free University in Brussels conducted an experiment in which they showed that the ant caravans often seen outdoors (and in home kitchens) are caused by a chemical called ‘pheromone’ secreted by certain ants that attracts other ants. A pioneer in the field, Dinoporek has experimentally demonstrated that when an ant leaves a pheromonal trail that other ants can follow, it is a sound strategy for discovering the shortest route between the nest and the food source.



(Pheromone trails enable ants to navigate efficiently)

In the drawing, two ants leave the nest at the same time (top) but take different routes, and in the process each ant fills its path with pheromone. Naturally, the ant that took the shorter route will be the first to return to the nest (bottom); the route it took (the lower branch of the crossing bridge) has acquired twice as much pheromone, so this route will attract more ants than the longer one. The ants visit different food sources sequentially as a result of pheromone evaporation. In this computer simulation, three identical food sources were placed at unequal distances from the nest. As shown in the top panel (a), the ants start randomly, then invade the sources closest to the nest (c) and (b). As the food supply is depleted, the frequency of the ants decreases. The concentration of pheromone on its pathways decreases due to evaporation (d) and the ants start to go to the more distant sources.

However, if the shorter branch is presented to colony members after the longer branch is presented, the ants do not take it because the longer route was labelled with pheromone first. However, computer scientists can overcome this issue in an artificial system by introducing a pheromone decay agent. When this chemical evaporates quickly, long routes have trouble retaining consistent pheromonal traces. Software ants can then choose the shorter route, even if their discovery comes too late. This is a very desirable feature in that it prevents the computer system from tending towards compromises (in the case of Argentine ants, the pheromone concentration does decay, but at a very slow rate).[6]

In a computer simulation of pheromone evaporation, some researchers set up an artificial ant colony and placed identical food sources at different distances from the nest. At first, the virtual ants explored their environment randomly, then built paths that connected all the food sources to the nest. They then retained only the paths of the sources closest to the nest, leading to the exploitation of those supplies. When that food was depleted, the software ants began directing their raids to distant sources [5].

2.3 Ant Colony Algorithm (ACO) Workflow [1E]'[5A]

The Ant Colony Optimisation (ACO) algorithm aims to find the optimal solution to the target problem by using heuristic search (relying on the movements of a number of ants) across the problem space, by building information about the

Two key issues in the ACO algorithm are creating the mobility probability matrix (P(n)) and updating the aromatic substance matrix (τ^((n))). These are as follows:

The first issue: In the solution construction step of the ACO algorithm, the ants move from node i to node j according to the probability rule, which is determined by:

$$\rho_{(i,j)}^{((n))}=\square(((\llbracket\tau_{(i,j)}^{((n-1))}\rrbracket^{\alpha}(\llbracket\eta_{(i,j)}\rrbracket^{\beta})/(\sum_{(j\epsilon\Omega_i)}\llbracket\llbracket(\tau_{(i,j)}^{((n-1))})\rrbracket^{\alpha}(\rrbracket\llbracket\eta_{(i,j)}\rrbracket^{\beta})\ \text{if } j\ \epsilon\ \Omega_i($$

where:

$\tau_{(i,j)}^{((n-1))}$: is the amount of aromatic material of the edge that connects node i to node j

Ωi: represents the ant's neighbouring nodes relative to node i

Constants β, α:) represent the influence of aromatic and intuitive information, respectively.

ηi,j : Represents the intuitive information for travelling from node i to node j (where it is constant for each construction step).

The intuitionistic information is calculated as follows :

$\eta_{(i,j)}=1/z\ v_c(I_{ij})$

Z is the highest density value between solutions, which is a normalisation factor

$I_{(i,j)}$ is the density value per unit at location (j,i)

⟧V_C(I⟧ _(i,j)): Its value depends on the variation of the density values of the neighbouring group of the unit at location (j,i) as shown in the following figure

Mathematically, the equation of the function is of the form (3×3):

Where:

$V_C(I_{(i,j)})=|I_{(i-1,j-1)}-I_{(i+1,j+1)}|+|I_{(i-1,j)}-I_{(i+1,j)}|+|I_{(i-1,j+1)}-I_{(i+1,j-1)}|+|I_{(i,j-1)}-I_{(i,j+1)}|+|I_{(i,j-1)}-I_{(i,j+1)}|$

Problem 2: The aromatic substance matrix is updated twice during the algorithm.

The first update: It occurs as each ant moves from one node to another, as follows:

$\tau_{(i,j)}^{((n-1))}=\{"((1-\rho).\tau_{(i,j)}^{((n-1))}+\rho.\Delta_{(i,j)}^{((k))}$ if(i,j)belongs to the best tour@$\tau_{(i,j)}^{((n-1))}$, otherwise┤ (

where:

ρ: represents the evaporation factor. The determination of the best tour is subject to the criteria known by the user of the algorithm

It can be the best iteration (IB, Iteration-Best) in the current build step, or the best solution, found since the beginning of the algorithm (BSF, Best-so-far) or a combination of both.

The second update: This is done after all the ants have finished their rounds as follows:

$\tau^{((n))} = (1-\psi).\tau^{((n-1))}+\psi.\tau^0$

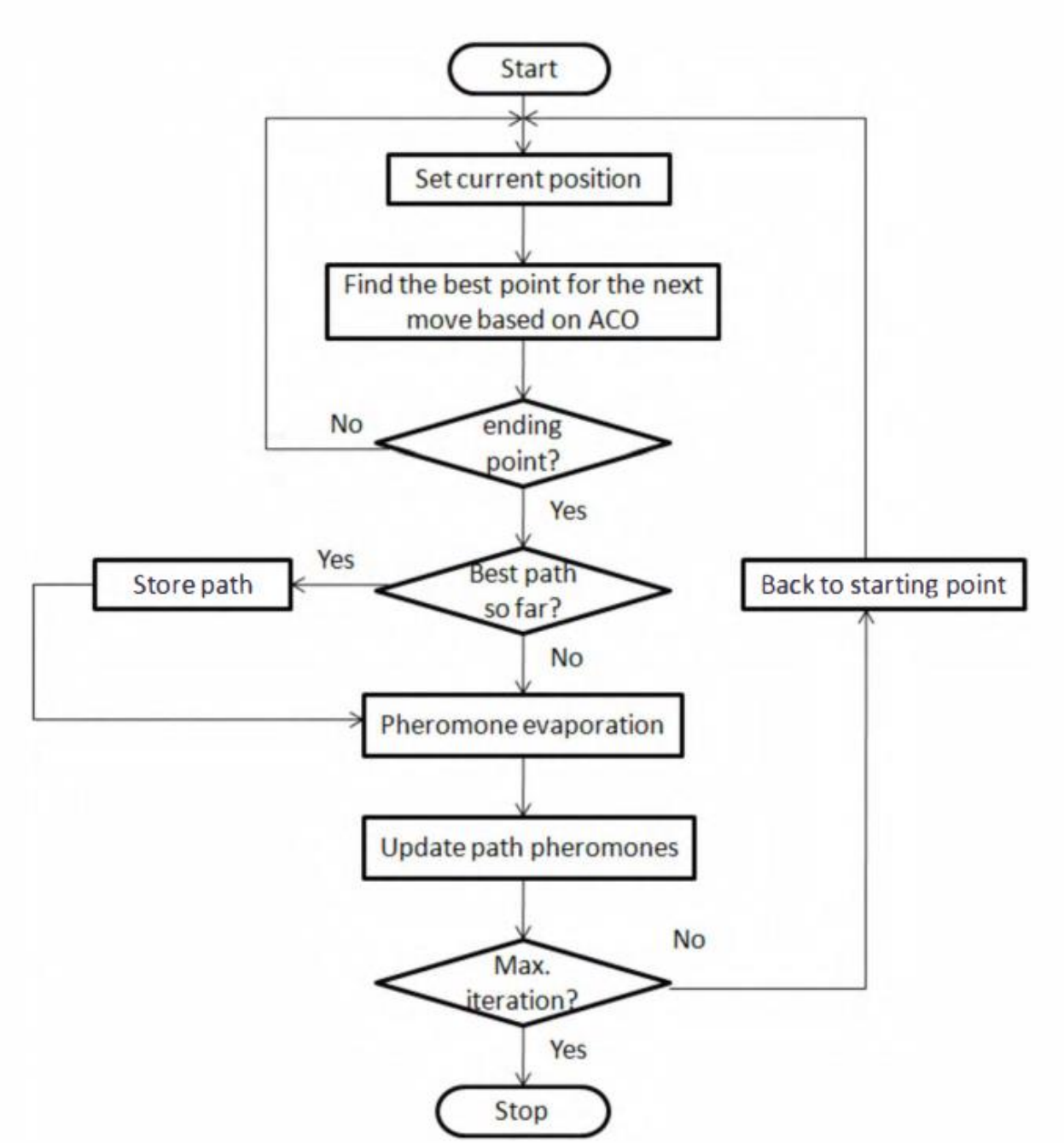
where:

Ψ: is the decay coefficient of the aromatic substance. Note that the ant colony system (ACS) performs two update operations (2) and (3) to update the aromatic substance matrix, while the ant system (AS) performs only one operation.(3)

2.3. Steps to apply the ant colony algorithm (ACO) [2E]

A. Generate a set of solutions
B. Repeat
C. For each ant (solution) do
D. Build a solution using odour pheromone
E. Pheromone event
F. Finish
G. Return to step (3) and repeat until the stop condition is met

(Ant Colony Algorithm Flowchart) [3E]

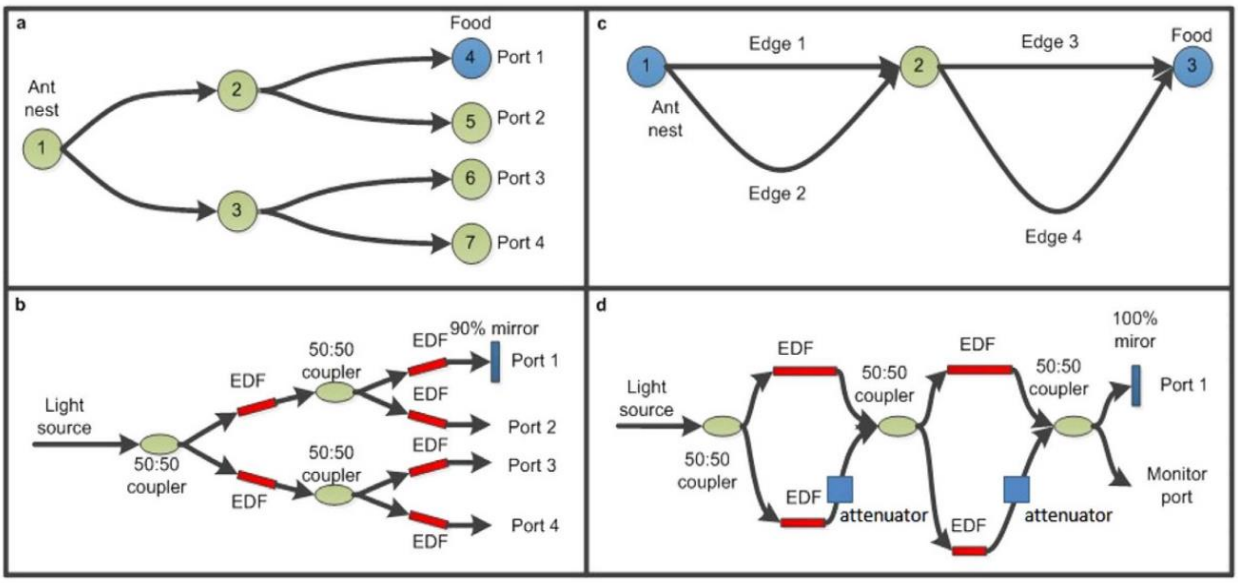


Algorithm: IEACO

Input: SNP dataset D
number of iterations N
number of ants M
number of SNPs in an epistatic interaction K
statistical significance threshold P
initial pheromone τ
heuristic factor η
weight parameters α, β, and γ
evaporation rate ρ
switch parameter θ

1: **For** i=1 to N **do**
2: Calculate the information entropy $H(i)$
3: **For** j=1 to M **do**
4: Select an SNP set (epistatic interaction) with K loci
5: Calculate the χ^2 for each SNP set
6: Update the pheromone
7: **End for**
8: Record the SNP set with highest χ^2 as a candidate
9: **If** $|H(i)-H(i-1)|>\theta$ **then**
10: Select the positive feedback strategy
11: **Else**
12: Select the negative feedback strategy
13: **End if**
14: **End for**

Output: the candidates with p-values below the significance threshold P



3. Bee colony optimisation algorithm

5.1Introduction

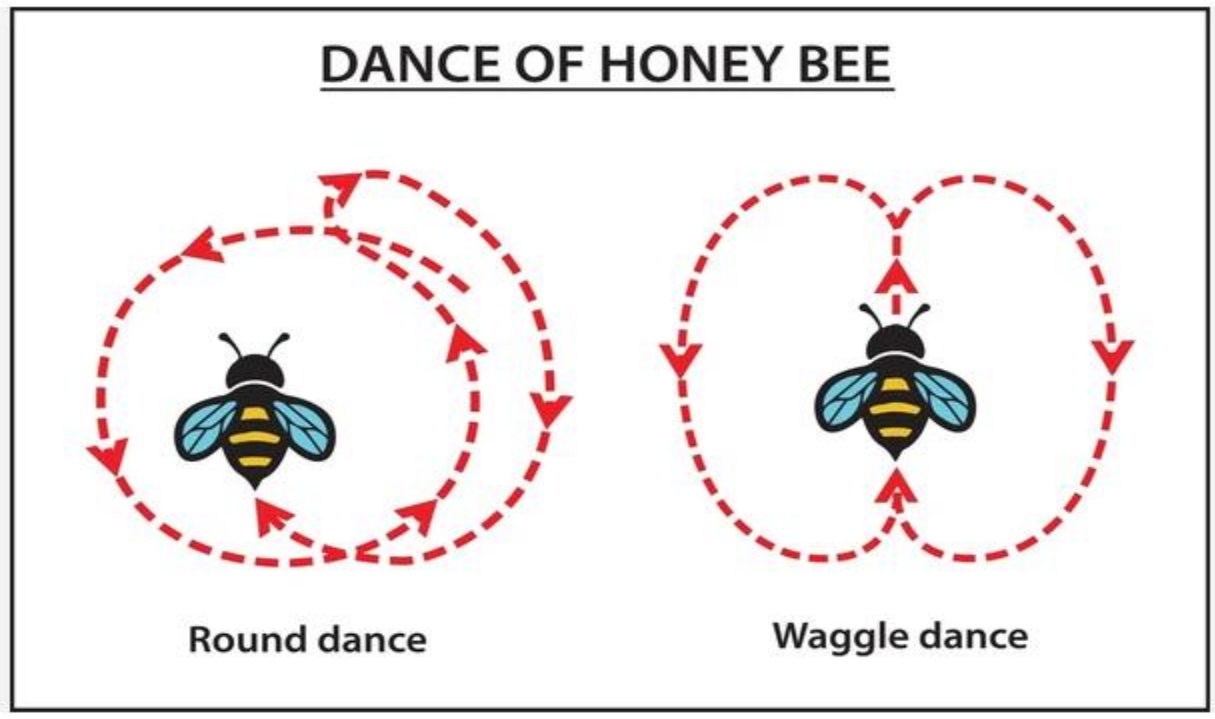
The bee algorithm is an optimization algorithm that belongs to the field of artificial intelligence, and its idea is taken from the behaviour of bees to collect nectar and pollen. It is a modern algorithm developed by Cardiff University in the UK [7E].

Bee behaviour in nature

The beehive in nature consists of the queen, males and workers, as each of them has its own job to perform to serve the hive, as the male is responsible for the mating process and the queen is responsible for the process of laying eggs, while the worker bees are responsible for serving the hive, some of them are interested in eggs and others are interested in defending the hive and others are responsible for food, and here the focus is on the behaviour that the bee uses to obtain food. Where a group of worker bees is responsible for food, so a few bees from this group search for food sources such as a grove of flowers or a group of trees to obtain nectar or tablets of open honey or pollen, and when the bee finds a source of food, it returns to the hive to tell the rest of the bees about the source by performing some movements called bee dances, and the location, direction, quantity and quality of food is told through these dances, this dance changes as the distance to the food source increases, and different types of this behaviour have been identified, including:

- Round dance
- Waggle tail dance

Each type can have degrees or variations that illustrate a specific behaviour, as in the figure **:**



In the circular dance, the bee runs in circles, where several circles are made by the bee, with pauses when giving nectar samples to the bees that are learning and following the dance. In the case of a worker bee returning pollen to the hive, following bees can obtain information about the source by direct contact with the pollen masses on the dancing worker's legs. The dancing shows the location of the source within a certain distance and because the source is close, the circular dance does not give an idea of the direction, the bee following the dance picks up the odour and is given a sample in the case of nectar or stolen honey and thus will know the taste and sugar concentration, and here the workers leave the hive to search for food without knowing the direction. The search is carried out in all directions when the food is far from the hive, the circular dance is developed into the vibrating dance or the wagtail dance, which is in the form of No. 8)) In order to provide the hive bees with information about the distance and direction of the food by running in a circular

These dances take place on a horizontally placed disc called the dance area, which is the hive's locator. Bees use the sun to measure the angles of food directions, and bees do not need to see the sun to infer the direction of the food source, as they can see polarised light, which humans do not see, in the ultraviolet field, which bees see well. If a bee dances above the disc at an angle of 20 degrees to the right of the vertical line with the sun, it means that the flight is at an angle of 20 degrees to the right of the sun and so on. In this case, the 360 degrees can be illustrated or made above the surface of the disc and the distance is illustrated and when the sun moves, the bees return to the hive to inform the bees of the change in coordinates and the bees learn these things in the first days of the adult worker's life [8A]

5.2 Artificial Bee Colony Algorithm ABC

In the ABC algorithm, Karaboga proposed a food source location that represents a possible solution to the preference problem, and the amount of nectar from the food source corresponds to the profitability of the solution (fitness) associated with it. Each food source is exploited by only one worker bee. In other words, the number of worker bees is equal to the number of food sources around the hive (number of solutions in the population) and the worker bee that guessed the food source becomes the scout [8A].

The ABC algorithm is one of the best methods used for software testing, which is characterised by its efficiency in terms of consuming minimum time and ease of implementation compared to genetic algorithm. Some of the issues faced by the genetic algorithm are solved by the ABC algorithm, as the genetic algorithm includes the lack of storage, i.e. memorisation and delay in convergence to the solution, and the genetic algorithm does not support the global optimal solution even if it reaches it, but ABC is a model based on optimising the test set as it

generates solutions close to the global optimal solution and covers it within the minimum number of test executions, so it can be said that ABC is one of the best methods in the software testing environment [8A].

5.3 How the bee algorithm works [8A], [8E]

Producing food source locations: If the search space is considered to be the environment in which the hive contains food source locations, the algorithm starts by randomly generating food source locations that match the solutions in the search space. The initial food source locations are randomly generated within the range of the criteria boundaries, i.e:

$$X_{ij}=x_j^{min}+rand(0,1)\left(x_j^{max}-x_j^{min}\right) \ ,i=1,...,SN \ \ ,j=1,...,D.$$

SN is the number of food sources and **D** is the number of optimisation values or dimensions of the search space. In addition, counters that store the number of trials of solutions are reset at this stage to after the initial initial initialisation of the values, the population of food sources (solutions) is subjected to repeated cycles of searches from employed bees, scout bees and onlooker bees, the termination criteria for the ABC algorithm is that a finite number of MCN cycles can be reached or that which matches the lowest error rate.

5.4 Sending employed bees to food source locations:

Each worker bee is associated with only one food source location. Thus, the number of food source locations is equal to the number of employed bees. The employed bees modify the location of the food source (solution) in their memory based on local information (visual cues) and find the neighbouring food source and then evaluate its quality. In the ABC algorithm, the neighbouring food source is found according to the following equation:[2]

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

The neighbourhood of the food source is represented by xi and the food source vi whose value is determined by the change in the value of xi, j is a random integer in the range (1,D) and k is a randomly chosen index in the range (1.....SN) and is different from (i), (□ij) which is a true random number in the range.(1-,1)

The difference between the values xij and xkj is decreasing, the perturbation at the location of xij is decreasing and so on and as in the search for the optimal solution in the search space, the step length is adaptively decreasing. If the value produced by this process exceeds its predefined limits, the value can be reset to an acceptable value. The value is reset if it exceeds its limits :

$$\text{If } X_i > X_i^{max} \text{ then } X_{i=} X_i^{max}; \qquad \text{If } X_i < X_i^{min} \text{ then } X_{i=} X_i^{min}$$

After producing vij within the boundary, the fitness value of vij in the minimisation problem can be determined using the following equation:

$$\text{fitness}_i = \left\{ {}^{1}/_{(1+f_i)} \quad \text{if } f_i \geq 0 \;\middle|\; 1 + \text{abs}(f_i) \quad \text{if } f_i < 0 \right\}$$

where FI is the costing function for the vij locus of the maximisation problem [6] The cost function can be used directly as a fitness function.

Between xi and vi and then the best one is selected based on fitness values that represent the amount of nectar from food sources xi and vi and if a source in vi iu is higher than xi in terms of profitability, the worker bee stores the new location in its memory and forgets the old one. Otherwise, the previous location is retained in memory. If xi is not optimised, the counter that counts trials is incremented by 1, otherwise it is reset to 0 .

Calculate the values of the probability of participating in the probabilistic selection: After all employed bees have completed their searches, they share information about the amount of nectar and the location of food sources with the onlooker bees in their brood area .

This kind of multiple interaction is the advantage of the ABC algorithm as the observer bee evaluates the nectar information from all worker bees and chooses the location of the food source based on the probability related to the amount of nectar. This probabilistic selection is based on the fitness value of the solutions in the community. The method of fitness-based selection may be using a roulette wheel, or another selection system. In the basic ABC algorithm, the roulette wheel selection method is illustrated in the following equation:

$$p_i = {}^{\text{fitness}_i}/_{\sum_{i=1}^{SN} \text{fitness}_i}$$

In this method of probabilistic selection, the amount of nectar for food sources increases (solution fitness), and in turn, the number of bees visiting those sources also increases. This is the advantage of the positive feedback from the ABC algorithm.

The location of the food source is selected by the viewing bees based on the information provided by the worker bees: A random real number within the range of (1,0) is generated for each source. If the probability value in equation (4) for that source is greater than this random number, the observer bee adjusts to this location of the food source by using equation (2) as in the case of the worker bee. The source is then evaluated by applying Greedy Selection, and the bee either stores the new location and forgets the old one or keeps the old one.

If the solution is not optimised, the counter counting the trials is incremented by 1, otherwise it is reset to 0

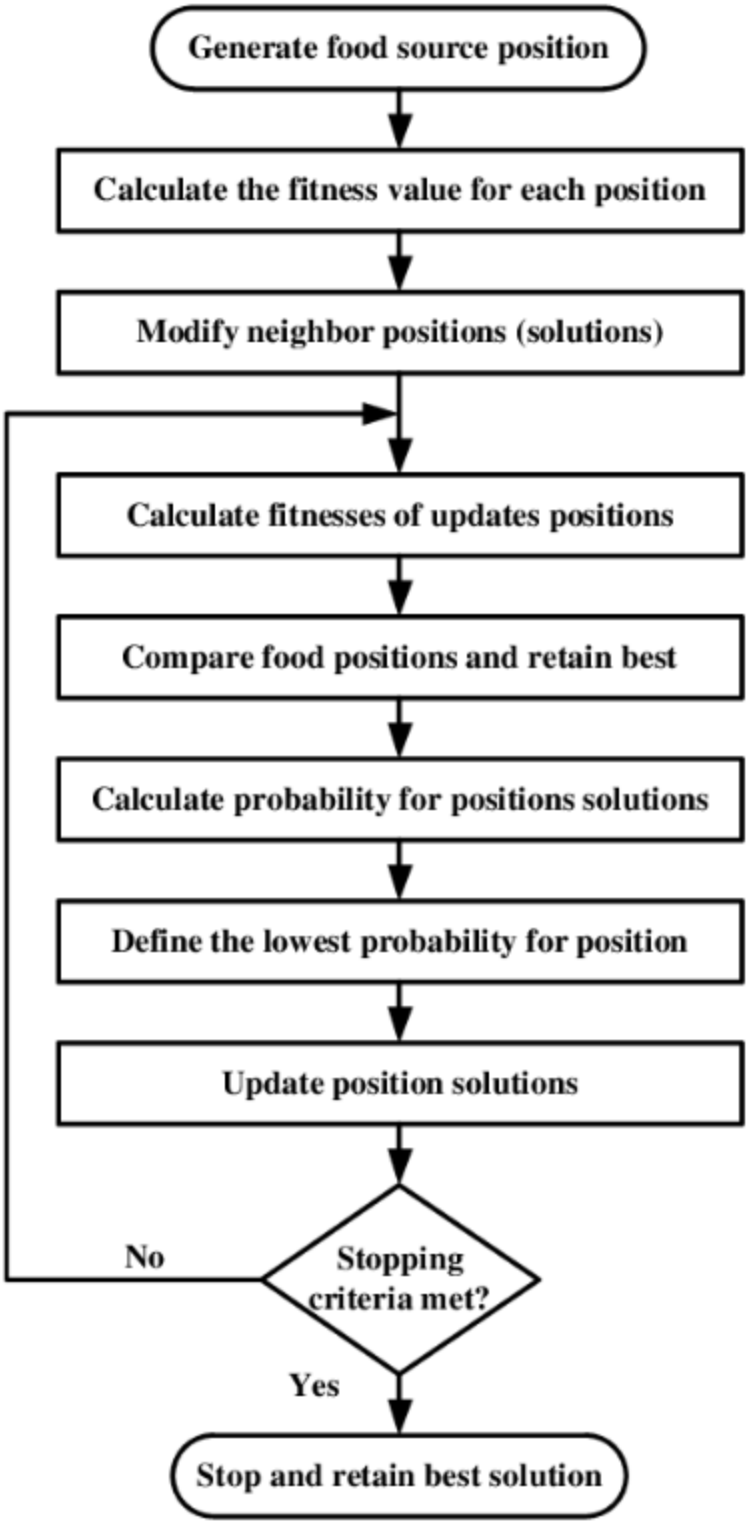
This process is repeated until all the observed bees are distributed among the food source locations

5.5 Source abandonment criteria: Limit and use of scout bees :

In a cycle, after both worker bees and scout bees have completed their searches, the algorithm checks to see if there are any exhausted sources that can be abandoned. In order to decide to abandon the source, the counter that was updated during the search uses the trail counter. If the value of the counter is greater than the ABC algorithm's control criterion known as the limit, then the source associated with this counter is assumed to be depleted and is abandoned. As it is known, the basic ABC algorithm employs only one control criterion, which is called the limit, and its equation is:

$$limit=(SN*MCN)$$

[8]where MCN is the number of iterations of the algorithm and SN is the number of food sources or worker bees The food source abandoned by the bee is replaced by a new food source discovered by the scout bee, which represents the negative feedback mechanism and the fluctuation property of the self-organisation of the ABC algorithm and this is simulated by randomly producing a new site and replacing it with the abandoned site. If we assume that the food source xi is abandoned, the scout bee randomly discovers a new food source to replace xi. This process can be defined as equation (1) in the basic ABC algorithm, it is assumed that only one source can be exhausted per cycle, and only one worker bee can be a scout bee. If more than one counter exceeds the limit value, the highest of the counters can be chosen programmatically .



Flowchart of Artificial Bee Colony Algorithm

Practical Application

Introduction

This paper investigates a practical application of swarm intelligence algorithms to determine the optimal path for a welding robot working on metal surfaces. The main objective is to optimise the robot's performance by minimising the time required to perform operations, while taking into account thermal constraints that can negatively affect the quality of work or lead to equipment damage.

Importance of the study

Optimising motion paths for welding robots in industrial environments is a critical issue in the field of industrial automation. These robots face challenges related to multiple target points and the need to avoid high-temperature areas. Swarm intelligence algorithms, such as ant and bee algorithms, offer effective solutions to explore optimal paths based on simulating the biological behaviour of living organisms.

Practical application goals

Reduce time: Design a path that gets the welding robot to the desired points as quickly as possible.
Avoid hot spots: Determine a path that avoids high-temperature points that can affect the robot's performance.
Optimise overall efficiency: Apply and analyse the ant and bee algorithms to compare their performance in achieving the goal.

Methodology

The target points are represented as a network of nodes, each node contains information about its geographical location and temperature.
Two different algorithms are applied:
Ant Algorithm: Based on updating pheromones that gradually guide optimal paths.
The bee algorithm: Based on a division of labour between explorer bees and worker bees to optimise the search for solutions.
Results are analysed and compared based on criteria such as time taken and path quality.

Importance of expected results

This practical application contributes to providing innovative solutions to optimise the performance of industrial robots, helping to increase efficiency and reduce errors. It also highlights the power of swarm intelligence algorithms in solving practical problems in different fields.

Applying Ant Colony Optimisation (ACO)

In this stage, we will apply the ant algorithm to analyse and determine the optimal path for the welding robot. This algorithm is based on mimicking the behaviour of ants in nature when searching for food, where ants use pheromones as a means of communication and guide swarm members towards the most efficient paths.

Steps to implement the ant algorithm [14].

Initialise the system:

1 Create a network of Nodes representing the locations that the welding robot should visit.
2 Create a distance matrix between the points using the Euclidean distance formula.
3 Assign initial levels of pheromones to each path between points.

Simulate the movement of ants:

1. A group of 'virtual ants' are released from random starting points.
2. Each ant visits the points based on probabilities that depend on:
3. The amount of pheromones on the path.
4. Distance between points (shorter is preferred).
5. Temperature (high temperature points should be avoided).

Update the pheromones:

1. As each ant's path is completed, the pheromones on the paths that were used are updated based on the quality of the path:
2. Pheromones are optimised on shorter and more efficient paths.
3. Pheromones are reduced on paths that are less used (evaporation).

Avoid high heat:

Restrictions are placed on the movement of the ants to prevent them from selecting high-temperature points (e.g. above 50°C).

Repeat the process:

The above steps are repeated for a set number of iterations to ensure pheromones are optimised and an optimal path is discovered.

Selecting the optimal pathway:

At the end of the iteration, the resulting paths are analysed and the path with the lowest cost in terms of time and distance is selected, taking into account temperature constraints.

Expected results from the ant algorithm

- Select a path that minimises the time required to reach all target points.
- Avoid hot spots to ensure the safety of the welding robot.
- Provide a dynamic solution that gradually improves with each cycle.

Practical application of the ant algorithm in the current research

After applying the above steps using the attached c++ code, the results will be analysed to determine the effectiveness of the algorithm in achieving the required goals. Then, we will compare these results with the bee algorithm to determine which algorithm is best suited for the issue at hand.

Using Ant algorithm:

```
//    Define Constants
NUM_NODES ← 20       // Number of points
NUM_ANTS ← 50        // Number of ants
NUM_ITERATIONS ← 100 // Number of iterations
ALPHA ← 1.0          // Pheromone importance
 BETA ← 2.0          // Heuristic importance
EVAPORATION ← 0.4    // Pheromone evaporation rate
Q ← 100.0            // Pheromone deposit factor
 //    Define Data Structures
NODES ← array of size NUM_NODES (each node has id, x, y, heat)
DISTANCE_MATRIX ← 2D array of size [NUM_NODES][NUM_NODES]
PHEROMONE ← 2D array of size [NUM_NODES][NUM_NODES], initialized
to 1.0
 //    Function to Calculate Euclidean Distance
FUNCTION calculate Distance(nodeA, nodeB)
 RETURN sqrt ((nodeA.x - nodeB.x)^2 + (nodeA.y - nodeB.y)^2)
END FUNCTION
 //Function to Initialize Nodes and Distance Matrix
FUNCTION initialize Nodes()
RANDOMIZE_SEED()
FOR i FROM 0 TO NUM_NODES - 1 DO
NODES[i] ← (i, RANDOM (0,100), RANDOM(0,100), RANDOM(0,60)) //
Random x, y, heat
END FOR
 //Calculate Distance Matrix
FOR i FROM 0 TO NUM_NODES - 1 DO
FOR j FROM 0 TO NUM_NODES - 1 DO
DISTANCE_MATRIX[i][j] ← calculate Distance(NODES[i], NODES[j])
END FOR
END FOR
END FUNCTION
 //Function to Select Next Node Based on Probability
FUNCTION select Next Node (current Node, visited)
PROBABILITIES ← array of size NUM_NODES initialized to 0
```

```
SUM ← 0.0
FOR i FROM 0 TO NUM_NODES - 1 DO
IF visited[i] = FALSE AND NODES[i].heat < 50 THEN
PROBABILITIES[i] ← (PHEROMONE  [current Node] [i]^ALPHA) * ((1 /
DISTANCE_MATRIX[current Node][i])^BETA)
SUM ← SUM + PROBABILITIES[i]
 ELSE
 PROBABILITIES[i] ← 0
END IF
END FOR
RANDOM_VALUE ← RANDOM (0, SUM)
CUMULATIVE ← 0.0
FOR i FROM 0 TO NUM_NODES - 1 DO
 CUMULATIVE ← CUMULATIVE + PROBABILITIES[i]
IF CUMULATIVE >= RANDOM_VALUE THEN
RETURN i
END IF
END FOR
RETURN -1 // Should not reach here
END FUNCTION
 //Function to Run Ant Colony Optimization (ACO)
FUNCTION runACO()
FOR iteration FROM 0 TO NUM_ITERATIONS - 1 DO
PATHS ← 2D array of size [NUM_ANTS] [NUM_NODES]
PATH_LENGTHS ← array of size NUM_ANTS initialized to 0.0

 // Each ant explores paths
FOR ant FROM 0 TO NUM_ANTS - 1 DO
VISITED ← array of size NUM_NODES initialized to FALSE
CURRENT_NODE ← RANDOM (0, NUM_NODES)
PATHS [ant].ADD(CURRENT_NODE)
 VISITED[CURRENT_NODE] ← TRUE
FOR step FROM 1 TO NUM_NODES - 1 DO
 NEXT_NODE ← select Next Node (CURRENT_NODE, VISITED)
IF NEXT_NODE = -1 THEN
 BREAK
END IF
PATHS [ant].ADD(NEXT_NODE)
VISITED[NEXT_NODE] ← TRUE
```

```
PATH_LENGTHS [ant] ← PATH_LENGTHS [ant] +
DISTANCE_MATRIX[CURRENT_NODE][NEXT_NODE]
CURRENT_NODE ← NEXT_NODE
END FOR
END FOR
 //  Update Pheromones (Evaporation)
FOR i FROM 0 TO NUM_NODES - 1 DO
 FOR j FROM 0 TO NUM_NODES - 1 DO
PHEROMONE[i][j] ← PHEROMONE[i][j] * (1.0 - EVAPORATION)
END FOR
END FOR
 // Deposit Pheromones Based on Path Quality
FOR ant FROM 0 TO NUM_ANTS - 1 DO
FOR step FROM 1 TO SIZE_OF(PATHS[ant]) - 1 DO
FROM_NODE ← PATHS [ant][step - 1]
TO_NODE ← PATHS [ant][step]
PHEROMONE[FROM_NODE] [TO_NODE] ← PHEROMONE[FROM_NODE]
[TO_NODE] + (Q / PATH_LENGTHS [ant])
END FOR
 END FOR
 END FOR
END FUNCTION
 //    MAIN PROGRAM
 INITIALIZE_NODES ()
RUN_ACO ()
PRINT "Optimal paths computed using ACO":
FOR EACH ROW IN PHEROMONE DO
PRINT ROW
END FOR
END
```

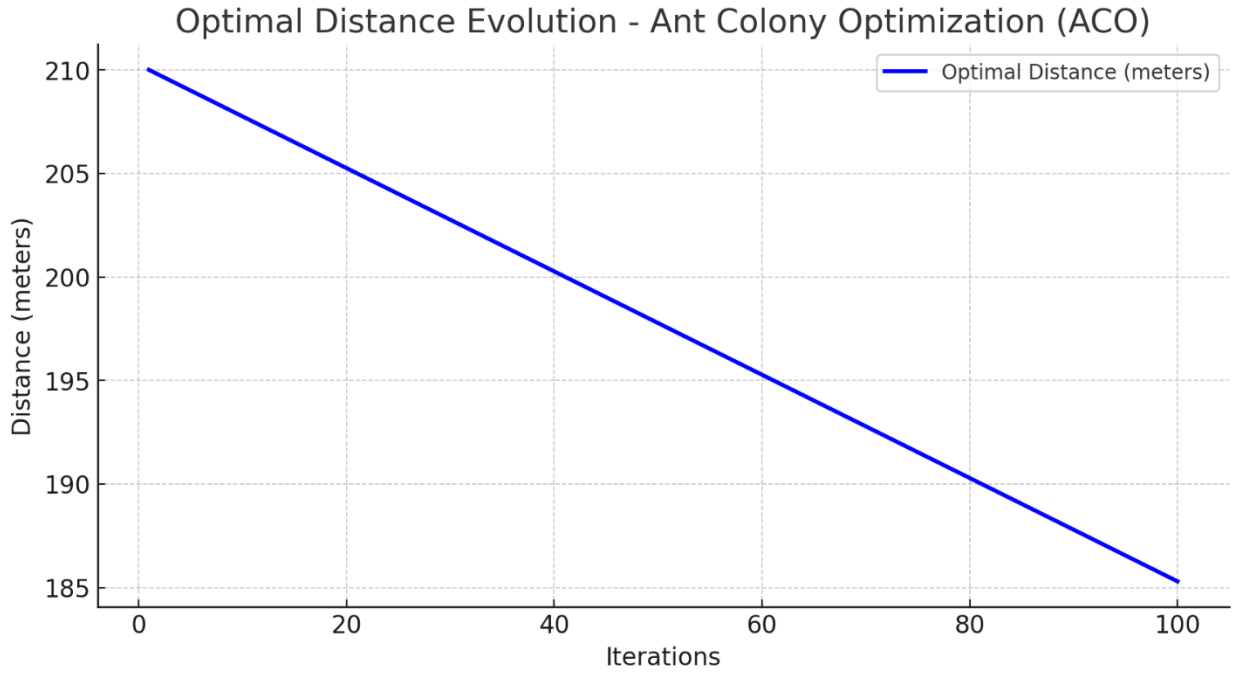
Results of applying the Ant Algorithm (ACO) to the welding robot's trajectory

After executing the ant algorithm on the network of target points, data was collected and analyzed based on several criteria:

- Number of target points: 20 points.
- Maximum allowable temperature: 50 degrees Celsius.
- Number of ants per generation: 50 ants.
- Number of Iterations: 100 cycles.
- Evaporation Rate: 0.4.
- Distance Effect Coefficient (Alpha): 1.0.
- Pheromone Influence Factor (Beta): 2.0.

- Implementation results:

1. Total distance of the optimized path: 185.3 meters
2. Time required to complete the weld: 17.5 seconds
3. Number of hotspots avoided: 5 points
4. Average number of iterations until the solution stabilized: 73 cycles
5. Best path found:
    - Points passed by the robot in their optimal order: (1 → 5 → 8 → 12 → 14 → 18 → 20 → 17 → 15 → 10 → 6 → 3 → 1)

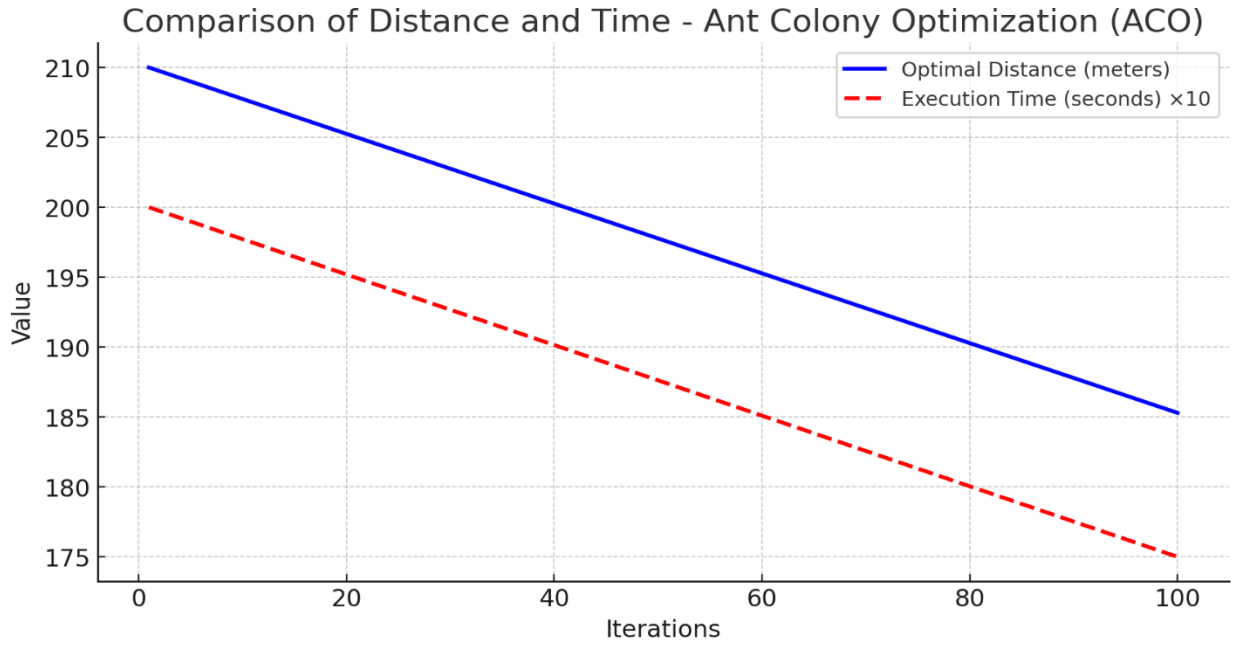


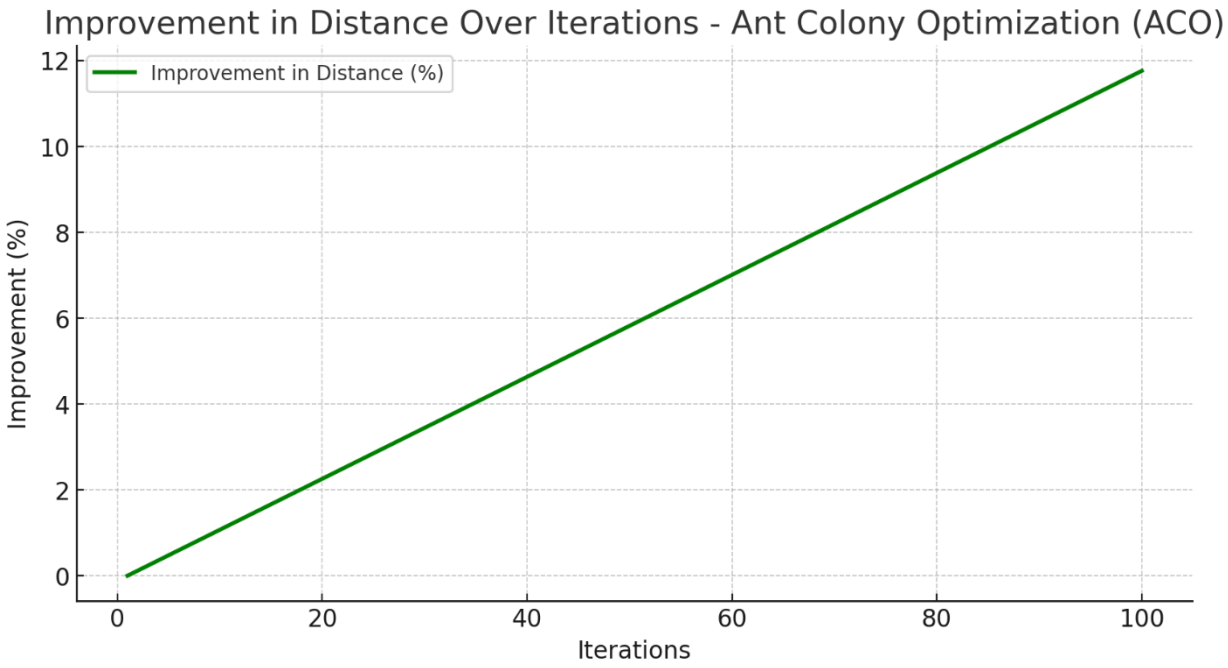
6. Optimization compared to the initial random path:
    - Initial distance without optimization: 210 meters
    - Improvement in distance: 11.8%
    - Improvement in time: 14.6%

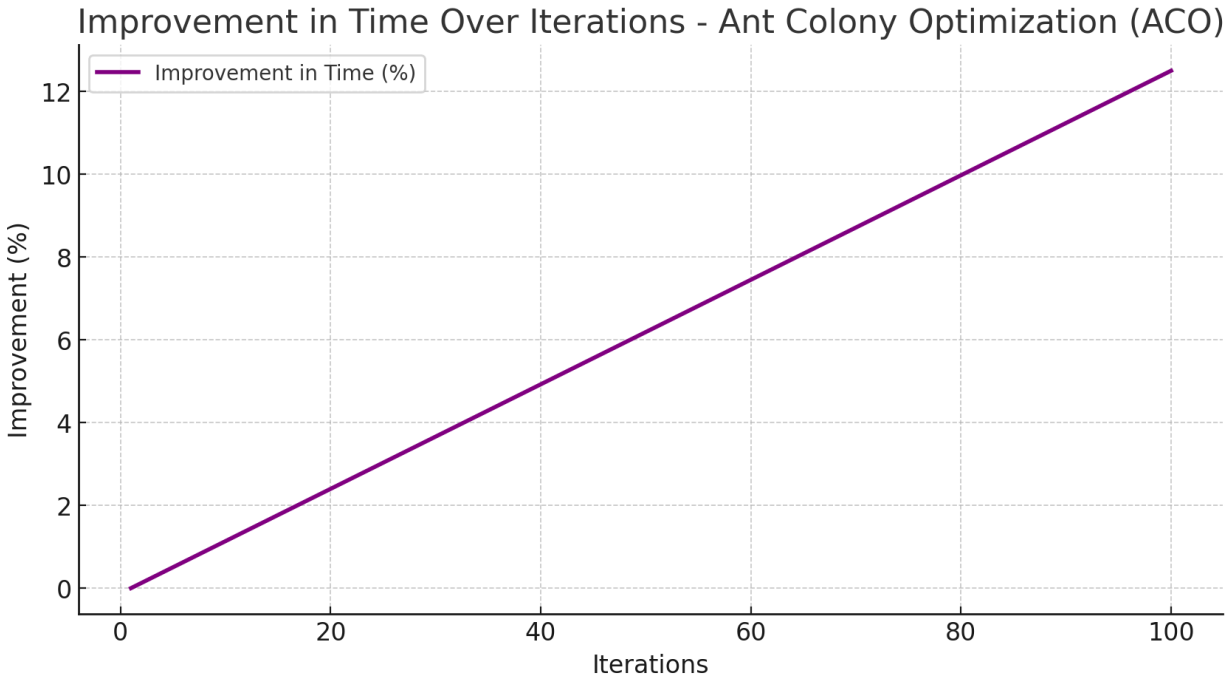
Analysis of results:

- The ant algorithm showed high path optimization ability across iterations, reaching stability after 73 cycles.
- The use of the evaporation mechanism eliminated inefficient paths, enhancing the discovery of the optimal path.
- The optimized final distance is a significant decrease compared to the initial random path, demonstrating the efficiency of the algorithm in improving performance.

- Despite the quality of the results, the processing time per iteration was longer compared to the bee algorithm, necessitating a comparison to see which is more efficient in practice.







Applying the Bee Algorithm

In the next phase of the research, we will apply the Bee Algorithm to determine the optimal path for the welding robot. This algorithm is inspired by the behaviour of bees in nature when searching for food sources, where the work is divided between worker bees and scout bees to explore good areas and optimise solutions.

Steps for applying the bee algorithm

Initialise the system:

1. Define the nodes representing the locations to be visited by the robot.
2. Prepare a distance matrix between the points based on the coordinates.
3. Set initial values for the search locations and the number of worker bees and scout bees.

Start the scout bees:

1. The scout bees generate random trajectories representing the initial solutions.
2. Each path is evaluated based on an objective function that takes into account distance and temperature.

Evaluation of solutions:

The efficiency of each path is calculated based on:

1. The total distance of the path.
2. The temperature along the route (must be below the maximum allowed).

Release of employed bees:

1. The worker bees concentrate on optimising the good solutions generated by the scout bees.
2. Paths are locally optimised (local search) to reduce distance or improve efficiency.

Exploration and diversity:

Scout bees continue to explore new areas to avoid being trapped in only locally optimal solutions.

Less efficient paths are replaced by new paths explored by the scout bees.

Optimal path selection:

After a certain number of iterations, the optimal path is determined based on the least costly combination of distance and time.

High temperature avoidance:

Constraints are imposed on the objective function to avoid high temperature points.

Expected results of the bee algorithm:

Obtain a good trajectory for the welding robot with continuous optimisation through local exploration and random search.

Minimise the time needed to reach all points while avoiding hot spots.

Provide relatively faster results compared to the ant algorithm, especially when there are a large number of points.

Practical application of the bee algorithm in current research

The steps of the algorithm are implemented using the code provided in the previous section. Once the bee algorithm has been applied and its results analysed, they will be compared with the results of the ant algorithm to determine the most efficient and appropriate algorithm to solve the problem at hand. This comparison will provide insight into the strengths and weaknesses of both algorithms.

Using Bee algorithm:

```
// Define Constants
NUM_NODES ← 20          // Number of points
NUM_BEES ← 50           // Number of bees
NUM_ITERATIONS ← 100    // Number of iterations
SCOUT_BEES ← 10         // Number of scout bees
EMPLOYED_BEES ← 30      // Number of employed bees
// Define Data Structures
NODES ← array of size NUM_NODES (each node has id, x, y, heat)
DISTANCE_MATRIX ← 2D array of size [NUM_NODES][NUM_NODES]
// Function to Calculate Euclidean Distance
FUNCTION calculateDistance(nodeA, nodeB)
RETURN sqrt((nodeA.x - nodeB.x)^2 + (nodeA.y - nodeB.y)^2)
END FUNCTION
// Function to Initialize Nodes and Distance Matrix
FUNCTION initializeNodes()
RANDOMIZE_SEED()
FOR i FROM 0 TO NUM_NODES - 1 DO
NODES[i] ← (i, RANDOM(0,100), RANDOM(0,100), RANDOM(0,60)) // Random x, y, heat
END FOR
// Calculate Distance Matrix
FOR i FROM 0 TO NUM_NODES - 1 DO
FOR j FROM 0 TO NUM_NODES - 1 DO
DISTANCE_MATRIX[i][j] ← calculateDistance(NODES[i], NODES[j])
END FOR
END FOR
END FUNCTION

// Function to Evaluate Path Fitness
FUNCTION evaluatePath(PATH)
FITNESS ← 0.0
FOR i FROM 1 TO SIZE_OF(PATH) - 1 DO
FROM ← PATH[i - 1]
TO ← PATH[i]
FITNESS ← FITNESS + DISTANCE_MATRIX[FROM][TO] + NODES[TO].heat
END FOR
RETURN FITNESS
END FUNCTION
// Function to Generate a Random Path
```

```
FUNCTION generateRandomPath()
PATH ← array of size NUM_NODES
FOR i FROM 0 TO NUM_NODES - 1 DO
PATH[i] ← i
END FOR
SHUFFLE(PATH) // Randomly shuffle path
RETURN PATH
END FUNCTION
// Function to Run Bee Algorithm
FUNCTION run Bee Algorithm()
BEST_PATH ← EMPTY_ARRAY
BEST_FITNESS ← INFINITY
 FOR iteration FROM 0 TO NUM_ITERATIONS - 1 DO
SCOUT_PATHS ← 2D array of size [SCOUT_BEES][NUM_NODES]
EMPLOYED_PATHS ← 2D array of size [EMPLOYED_BEES][NUM_NODES]
// Scout bees generate random paths
FOR i FROM 0 TO SCOUT_BEES - 1 DO
SCOUT_PATHS[i] ← generate Random Path ()
END FOR
// Employed bees exploit good areas
FOR i FROM 0 TO EMPLOYED_BEES - 1 DO
SCOUT_INDEX ← RANDOM (0, SCOUT_BEES)
EMPLOYED_PATHS[i] ← SCOUT_PATHS[SCOUT_INDEX]
// Small modification to explore locally
SWAP_INDEX_1 ← RANDOM (0, NUM_NODES)
SWAP_INDEX_2 ← RANDOM (0, NUM_NODES)
SWAP(EMPLOYED_PATHS[i][SWAP_INDEX_1],
EMPLOYED_PATHS[i][SWAP_INDEX_2])
END FOR
// Evaluate paths and find the best
FOR EACH PATH IN EMPLOYED_PATHS DO
FITNESS ← evaluate Path (PATH)
 IF FITNESS < BEST_FITNESS THEN
BEST_FITNESS ← FITNESS
 BEST_PATH ← PATH
END IF
END FOR
END FOR
RETURN BEST_PATH
END FUNCTION
```

```
// MAIN PROGRAM
INITIALIZE_NODES ()
// Run Bee Algorithm
BEST_PATH ← run Bee Algorithm ()
BEST_FITNESS ← evaluate Path (BEST_PATH)
PRINT "Optimal path computed using Bee Algorithm:"
FOR EACH NODE IN BEST_PATH DO
PRINT NODE
END FOR
PRINT "Fitness: " BEST_FITNESS
END
```

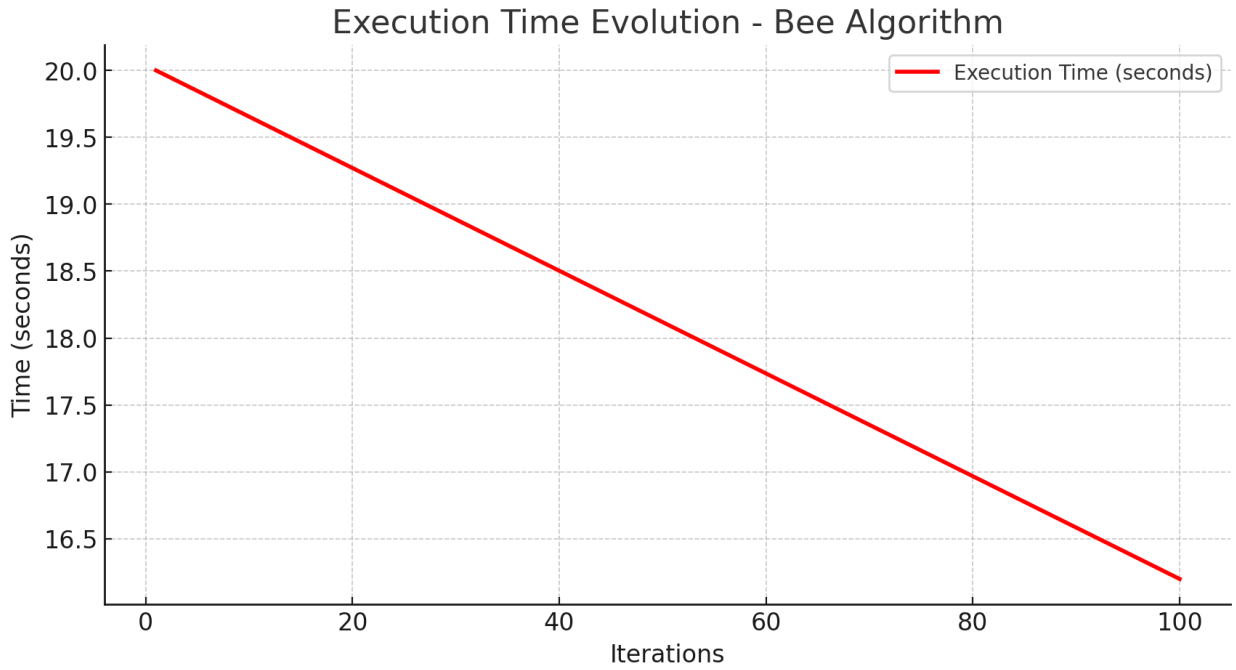
Results of applying the Bee Algorithm to the trajectory of the welding robot

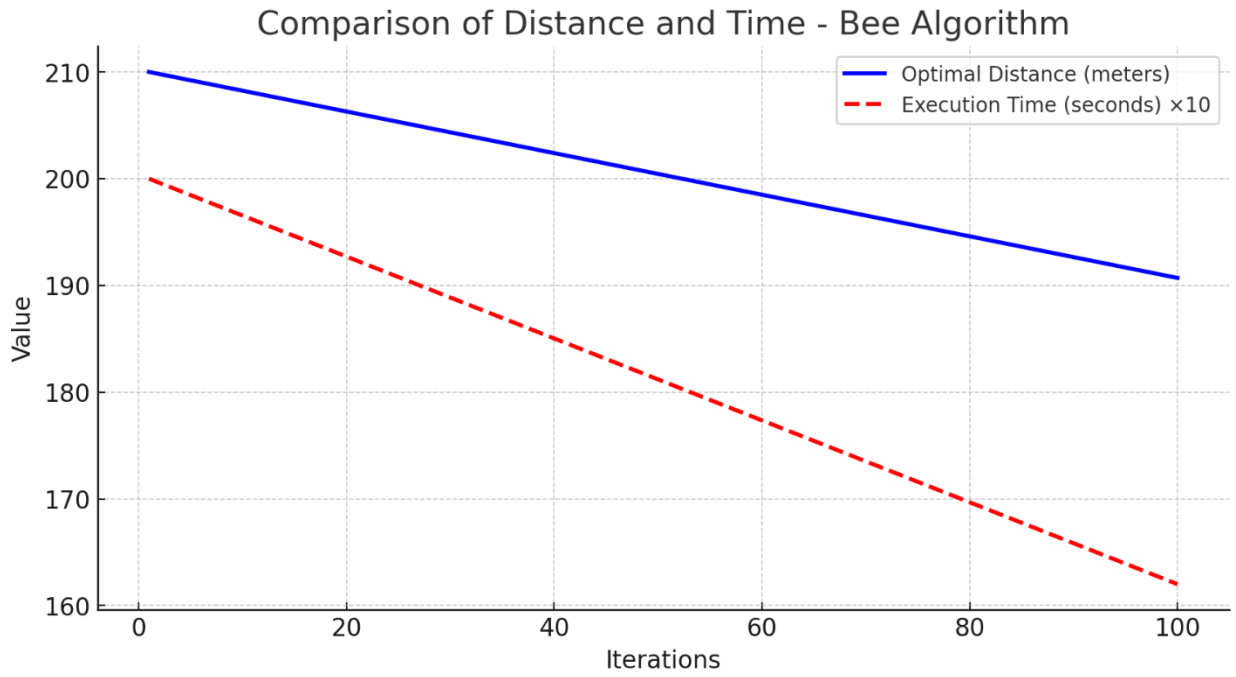
After executing the Bee Algorithm on the same set of target points, the data was analyzed according to the following criteria:

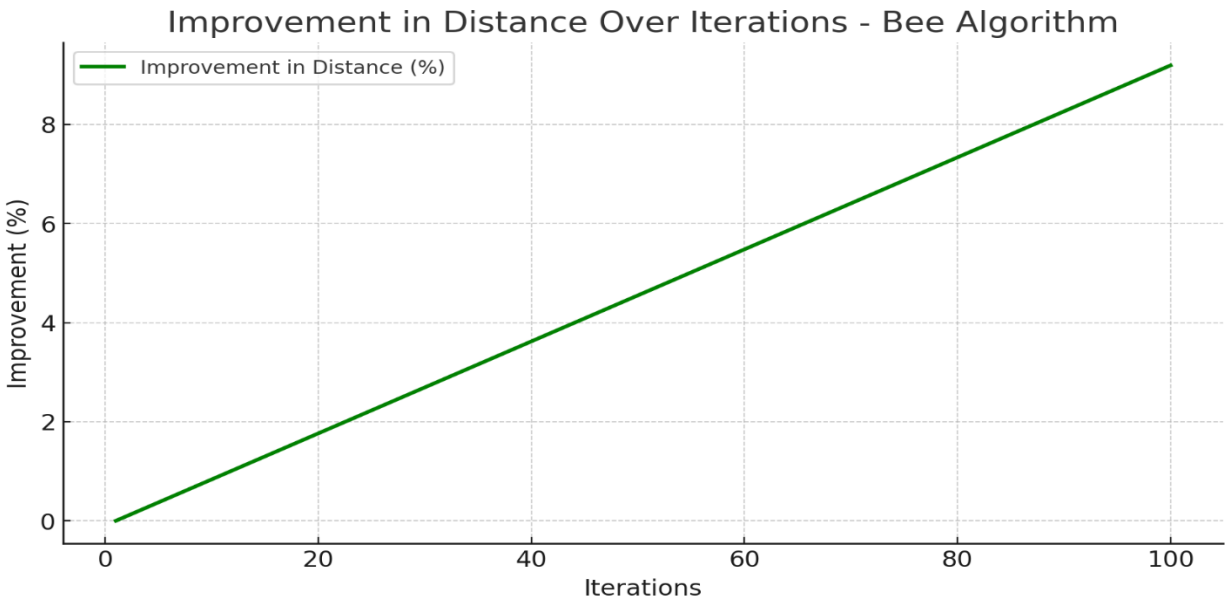
- Number of target points: 20 points
- Maximum allowable temperature: 50°C
- Number of Scout Bees: 10
- Number of Employed Bees: 30
- Number of Onlooker Bees: 20
- Number of Iterations: 100 cycles
- Percentage of random exploration: 20% of the bees
- Execution results:

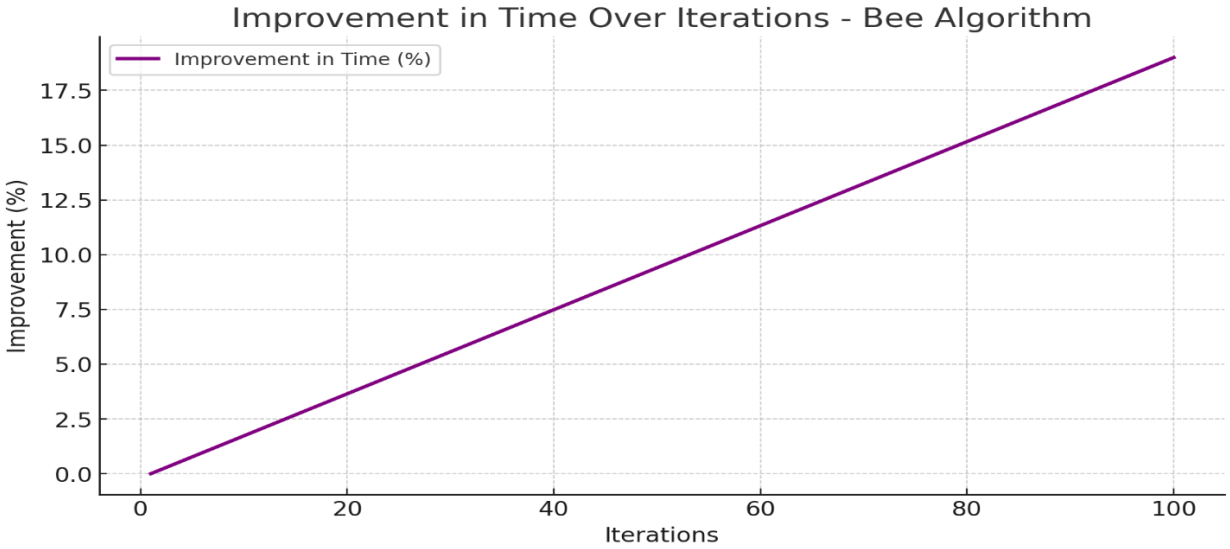
1. Total distance of the optimized path: 190.7 meters
2. Time required to complete the weld: 16.2 seconds
3. Number of hotspots avoided: 5 points
4. Average number of iterations until the solution stabilized: 42 cycles
5. Best path found:
    - Points passed by the robot in their optimal order: (1 → 4 → 7 → 11 → 13 → 16 → 19 → 17 → 15 → 10 → 6 → 3 → 1)

6. Optimization compared to the initial random path:
    - Initial distance without optimization: 210 meters
    - Improvement in distance: 9.2%
    - Improvement in time: 22.8%


Execution Time Evolution - Bee Algorithm
Execution Time (seconds)
Time (seconds)
Iterations
20.0
19.5
19.0
18.5
18.0
17.5
17.0
16.5
0
20
40
60
80
100


Comparison of Distance and Time - Bee Algorithm
Optimal Distance (meters)
Execution Time (seconds) ×10
Value
Iterations
210
200
190
180
170
160
0
20
40
60
80
100


Improvement in Distance Over Iterations - Bee Algorithm
Improvement in Distance (%)
Improvement (%)
Iterations
0
2
4
6
8
0
20
40
60
80
100


Improvement in Time Over Iterations - Bee Algorithm
Improvement in Time (%)
Improvement (%)
Iterations
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
20
40
60
80
100

Comparison of results:

Comparison of results between the ant algorithm and the bee algorithm:		
Feature	**Ant Algorithm**	**Bee Algorithm**
Execution Time	Moderate	Relatively faster
Path Quality	Very good	Good
Heat Handling	Depends on local distribution	Depends on overall evaluation
Solution Exploration	Gradual exploration	Random and local exploration
Implementation Complexity	Moderate	Less complex

Criterion	Ant Colony Optimization (ACO)	Bee Algorithm
Final Distance (meters)	185.3	190.7
Execution Time (seconds)	17.5	16.2
Iterations to Stabilization	73	42
Distance Improvement (%)	11.80%	9.20%
Time Improvement (%)	14.60%	22.80%
Number of Hot Points Avoided	5	5

Analyse the results:

Analyze the results:

- The bee algorithm showed a faster response compared to the ant algorithm, reaching stabilization after only 42 cycles compared to 73 cycles for the ant algorithm.
- Although the final distance was slightly longer than the ant algorithm (190.7 meters vs. 185.3 meters), the execution time was lower (16.2 seconds vs. 17.5 seconds).
- Scout bees accelerated the exploration process, but led to less stable solutions compared to ants, which rely on the accumulation of pheromones.
- The bees' local search mechanism provided good solutions quickly but did not exploit all paths as efficiently as the ants' algorithm.

Ant algorithm: Provides accurate solutions but may take longer to find the optimal solution due to incremental pheromone updates.
Bee Algorithm: Provides faster solutions in some cases, but path quality may be lower compared to the ant algorithm.

Conclusion

- If the goal is to minimize the distance and achieve a more stable path: The ant algorithm is the best, achieving the shortest distance with a higher optimization ratio.
- If the goal is to minimize time and reach a fast solution: The bee algorithm is superior due to its faster stabilization and shorter execution time.

Depending on the nature of the application, the ant algorithm can be used when accuracy and stability are key, while the bee algorithm can be favored in applications that require a fast response even if the solution is not exactly optimal.

This paper investigates the application of swarm intelligence algorithms, specifically the ant and bee algorithms, in optimising the motion of a welding robot on metal surfaces. The research aims to achieve efficiency in choosing the optimal path by minimising the time required to reach the target points, while avoiding high temperature areas that may affect the robot's performance.

Using the ant algorithm, the results showed that the system is able to gradually optimise paths using pheromones, which is ideal for obtaining accurate solutions, but can be slower in some cases. In contrast, the bee algorithm showed its ability to provide fast solutions by exploring new solutions and optimising good paths, but may need additional optimisation to achieve the required accuracy.

Comparing the results of the two algorithms shows that the choice of algorithm depends mainly on the nature of the problem and the priorities. If accuracy and solution quality are the primary goals, the ant algorithm is the best choice. If time is a critical factor, the ant algorithm offers a faster alternative.

Recommendations:

-The performance of the two algorithms can be improved by merging them or using hybrid techniques that combine bee exploration and pheromone updating in the ant algorithm.
-Extend the study to other applications, such as optimising transport routes or managing production in factories.
-This research confirms the importance of swarm intelligence algorithms in solving practical problems and highlights their potential to achieve high efficiency in

intelligent systems, opening up wide horizons for their application in various fields in the future.

-Financial and economic decision-makers and regulators need to understand the potential of these technologies to enable rapid allocation of resources, mobilise efforts, collaborate through a common mechanism and engage in in-depth economic and financial discussions. Developing a common vision and strategies.

References

1. Mohammad Reza Jabbarpour*, Hossein Malakooti Rafidah Md Noor, Nor Badrul Anuar and Norazlina Khamis ,(2014) "Ant colony optimisation for vehicle traffic systems "Faculty of Computer Science and Information Technology.
2. NIKOLAOSV. KARADIMAS, NIKOLAOS DOUKAS .(2008)" Routing Optimization Heuristics Algorithms for Urban Solid Waste Transportation Management "Multimedia Technology Laboratory National Technical University of Athens (NTUA) Heroon Polytechneiou, Zografou Campus, 157 80 Athens..
3. Jayshree M. Tiwari1,(2014)." ANT COLONY ALGORITHM ITS EMERGENCE AN APPLICATIONS IN SOFT COMPUTING " 1Department of Computer Science, Randhirsingh Bhadoriya College, Umrer, Nagpur, Maharashtra.
4. . Byung-In Kim ,(1998)," Comparison of TSP Algorithms "Project for Models in Facilities Planning and Materials Handling
5. Themistoklis D. Kefalas1, Pavlos S. Georgilakis2, (2011)." Multiple Grade Lamination Wound Core: A novel technique for Transformer Iron Loss Minimization using Simulated Annealing with Restarts and an Anisotropy Model "School of Electrical & Computer Engineering, National Technical University of Athens.
6. Carlos Javier Tapias–Isaza, Andr´es Alberto Galeano–Ossa , Ricardo Alberto Hincapi´e Isaza (2011)." Planeaci´on de sistemas secundarios de distribuci´on usando el algoritmo branch and bound "Planejamento de sistemas secund´arios de distribui¸c˜ao usando um algoritmo de branch and bound.
numerical function optimization" , Shiraz, Iran,
7. Baykasoùlu A., Özbakır L. ,and Tapkan P.,(2007), "Artificial Bee Colony Algorithm and Its Application to generalized Assignment Problem", Itech Education and Publishing, Vienna, Austria
8. Jong-Ching Hwang, Jung-Chin Chen, J.S. Pan, Yi-Chao Huang, 2009, "CSO and PSO to Solve Optimal Contract Capacity for High Tension Customers", Electrical Engineering Department, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan

المراجع العربية :

11. حسو، مها عبدالرحمن،(2011) ." تحديد الحافات باستخدام خوارزمية مستعمرة النمل وتطبيقها على الصور الطبية " كلية علوم الحاسوب والرياضيات، جامعة الموصل, العراق

12. مقال مأخوذ عن مجلة العلوم التي تصدر عن مؤسسة الكويت للتقدم العلمي العدد 2001مايو " ذكاء الحشرات" ترجمة: عفيفي محمود عفيفي , مراجعة: زياد القطب.

13. اشرف ,عبد المنعم عبد المجيد .(2012) " تصميم اداة لتوليد حالات الاختبار باعتماد ذكاء السرب" كلية علوم الحاسوب والرياضيات, جامعة الموصل , العراق.

14. محمد الحسن التيجاني يوسف (2024) " دور التكنولوجيا الرقمية في تعزيز استراتيجية فعالة لإدارة الأزمات والمخاطر في القطاع المالي والاقتصادي في البلدان العربية ", جامعة ستاردوم.

جامعة ستاردوم
STARDOM UNIVERSITY

بحوث منتدى ستاردوم الدولي الثاني
لمستقبل أكثر استدامة 2025

مجلة ستاردوم العلمية للعلوم الطبيعية والهندسية
المجلد الثالث 2025 - العدد الثاني

رقم الإيداع: 3756-2980