

**National University of Computer and Emerging Sciences**

**FAST University Islamabad Campus**

BSCS-22-Semester Course Section: **D**

Subject: Design And Analysis of Algorithms

Semester End Project

Adaptive Multimodal Continuous

Ant Colony Optimization

Submitted by: **Salman Khan 22I-1285**

**Hassan Ismail 22I-1011**

Submitted to: **Mrs. Hira Mastoor**

December 12th, 2024

**Introduction**

This paper explores the challenges of solving optimization problems with multiple equally optimal solutions, highlighting the limitations of traditional algorithms like Genetic Algorithms and Ant Colony Optimization (ACO), which are designed to find a single best solution. The ability to identify multiple optimal solutions is crucial for providing decision-makers with diverse choices. However, as the complexity of such problems grows, the number of optimal solutions can increase exponentially, complicating their identification.

Key challenges include:

* Ensuring diversity within the solution population to avoid overlooking potential optimal paths.
* Addressing the high computational costs and memory usage of existing methods like niching, which divides populations into subgroups to locate multiple solutions.

**Proposed Method: Adaptive Multimodal Continuous ACO (AM-ACO)**

The paper introduces **AM-ACO**, a novel approach that enhances traditional ACO for multimodal optimization by incorporating:

* **Niching Strategy**: Divides the population into subgroups, each tasked with exploring specific solutions.
* **Adaptive Parameter System**: Removes the need for manual parameter tuning, enabling more efficient and thorough exploration of solutions.

**Results and Benefits**

* Tested on 10 fitness functions, AM-ACO outperformed existing methods.
* Identified more optimal solutions than competitors.
* Maintained efficiency even for high-dimensional, complex problems.

The study highlights AM-ACO as a promising tool for tackling multimodal optimization problems with improved solution diversity and computational efficiency problems

**Algorithm 1 Implementation:**

This algorithm organizes a population into smaller groups, referred to as **crowds**, based on proximity between individuals. The process involves iteratively grouping individuals until the entire population is divided.

**Key Steps:**

* A **group size** is chosen, and a **random reference point** is generated.
* The **distance** from the reference point to each individual in the population is calculated.
* The individual **closest to the reference point** is selected, and a group is formed by adding the next closest individuals.
* These grouped individuals are **removed** from the population.
* The process repeats until no individuals remain.

The result is a population divided into smaller groups, with each group consisting of closely located individuals.

**Algorithm 2 Implementation:**

This algorithm organizes a population into smaller groups called **species** based on fitness and proximity, ensuring each group is centered around the most fit individuals.

**Key Steps:**

* **Sort Population by Fitness**: Individuals are ranked with the most fit at the top.
* **Seed Selection**: The best individual is chosen as the **seed** for a new species.
* **Species Formation**: The seed is combined with the next closest individuals to form a species.
* **Removal**: These grouped individuals are removed from the population.
* **Repeat**: The process continues until no individuals remain.

The population is divided into species, each centered around one of the best individuals, ensuring groups are based on both fitness and closeness.

**Algorithm 4 Implementation:**

This algorithm creates new solutions using a **niching method**, focusing on fitness-based selection and small adjustments to generate diversity.

**Key Steps:**

* For each niche:
  + Identify the **highest and lowest fitness values**.
  + Calculate a niche-specific value, **σᵢ**, to guide selection probabilities.
* Use a **roulette wheel selection** method to choose solutions from the niche based on fitness.
* For each selected solution:
  + Generate a new starting point, **μ**.
  + Calculate a small adjustment, **δ**, and apply it to **μ** to create a new solution.
* Repeat the process across all niches until the required number of new solutions is generated.

**Outcome:**

* Tracks the fitness values of all newly created solutions to ensure diversity and quality.
* Encourages solution exploration within niches while maintaining efficiency.

**Algorithm 5 Implementation:**

This algorithm improves the fitness of a set of seeds through an iterative refinement process, ensuring that seeds with better fitness are prioritized and updated.

**Key Steps:**

* **Fitness Adjustment**:
  + Calculate the **minimum (Fmin)** and **maximum (Fmax)** fitness values.
  + If any seed has fitness ≤ 0, adjust all fitness values by adding **|Fmin| + η** (a small positive value) to make them positive.
* **Probability Calculation**:
  + Compute a **selection probability** for each seed by normalizing its fitness (dividing by Fmax).
* **Seed Replacement**:
  + For each seed, compare a random number with its probability:
    - If the random number is smaller, evaluate potential replacements.
    - Replace the seed if a better individual is found.
* **Repeat for All Seeds**: Continue until all seeds have been evaluated and updated.

**Outcome:**

The algorithm outputs refined seeds with improved fitness values, ensuring progress toward better solutions while maintaining diversity.

**Algorithm 6 Implementation:**

This algorithm integrates **Algorithm 1, 4, and 5** to create a niching-based optimization approach, generating and refining solutions iteratively.

**Key Steps:**

* **Initialization**:
  + Randomly initialize a set of solutions and evaluate their fitness.
  + Determine the **maximum and minimum fitness values** in the solution archive.
* **Niching (Algorithm 1)**:
  + Randomly select a niching size from a predefined set.
  + Divide the archive into niches based on proximity and fitness.
* **New Solution Construction (Algorithm 4)**:
  + Create new solutions within niches.
  + Compare each new solution with its closest counterpart in the archive and replace the original if the new solution is better.
* **Local Search (Algorithm 5)**:
  + Refine the updated solutions through a localized fitness improvement process.
* **Iteration**:
  + Update fitness values and repeat the process until termination criteria are met.

**Outcome:**

The algorithm produces a refined archive of solutions that leverage niching, fitness-based selection, and local search for improved optimization.

**Algorithm 7 Implementation:**

This algorithm integrates **Algorithm 2, 4, and 5** to create a species-based optimization approach, similar to Algorithm 6 but using species grouping instead of niching for solution clustering.

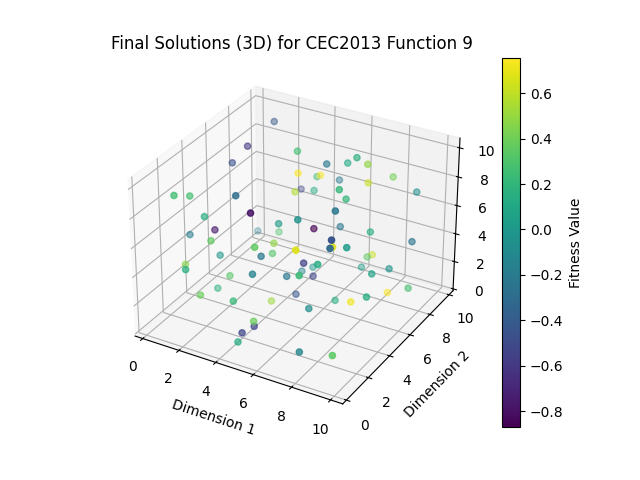
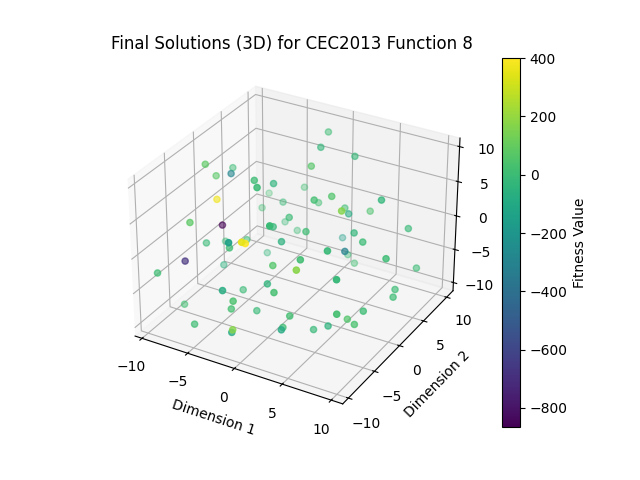
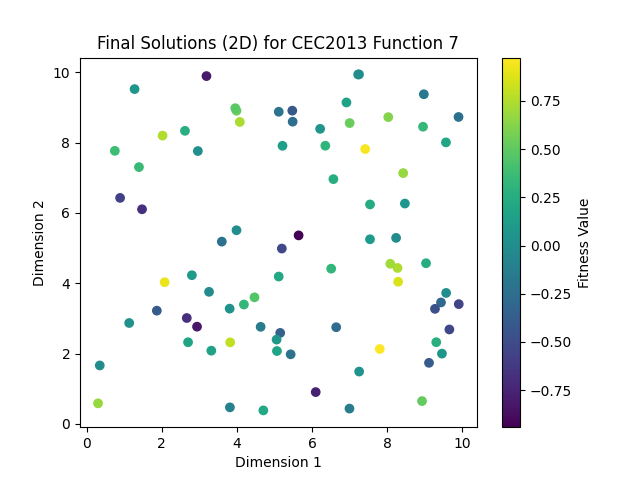
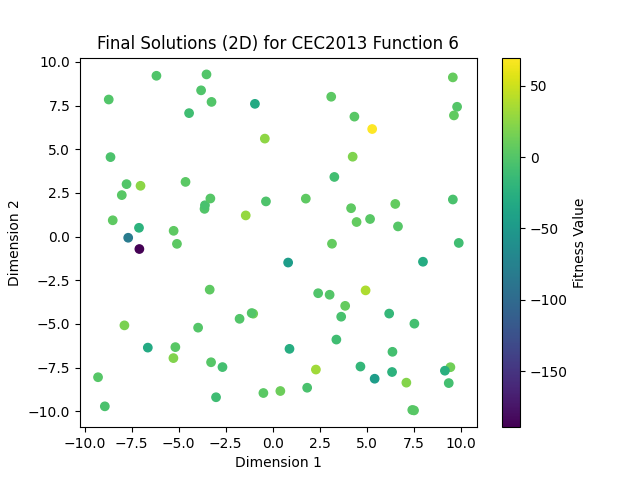
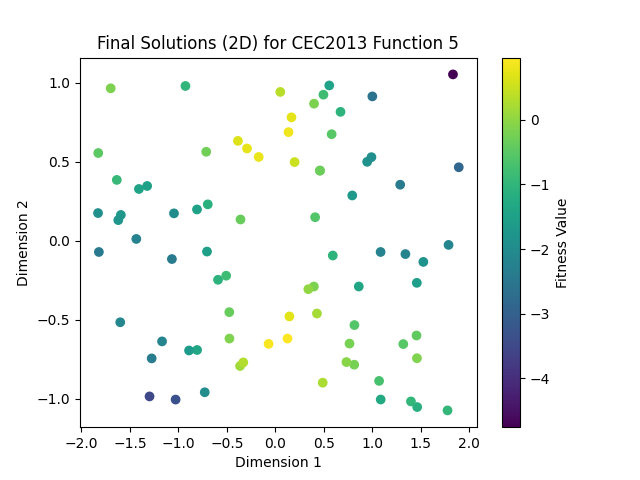
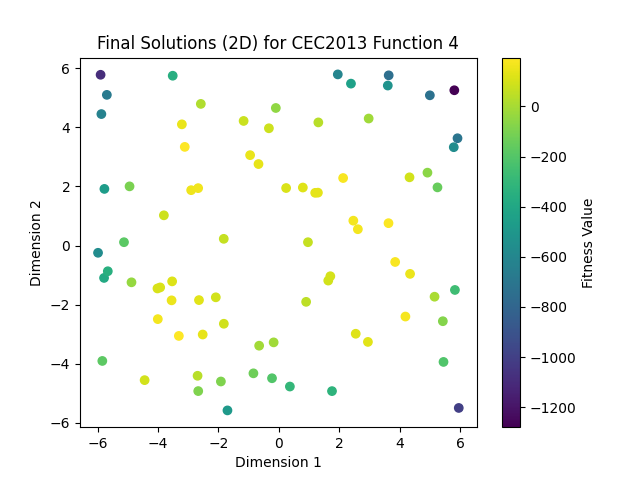
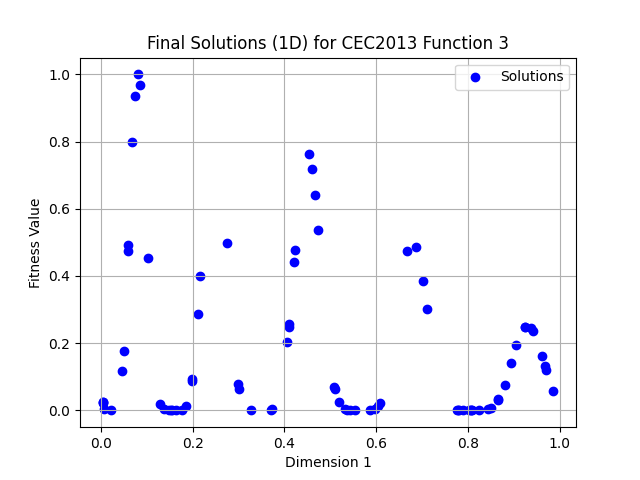
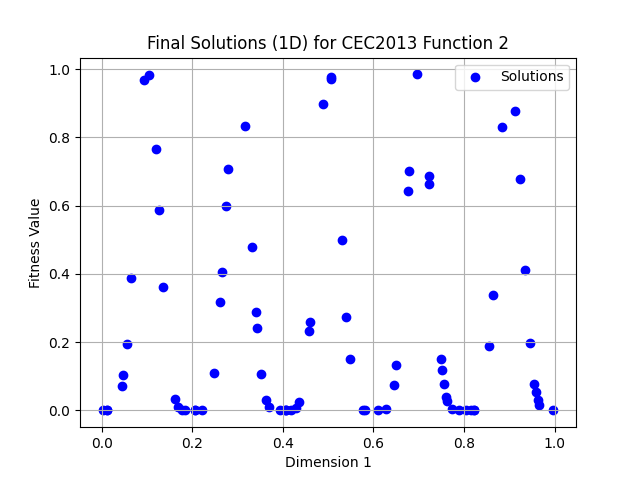
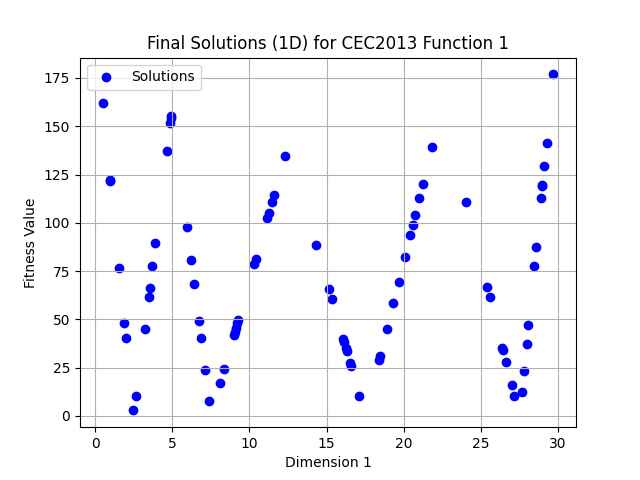
**Key Steps:**

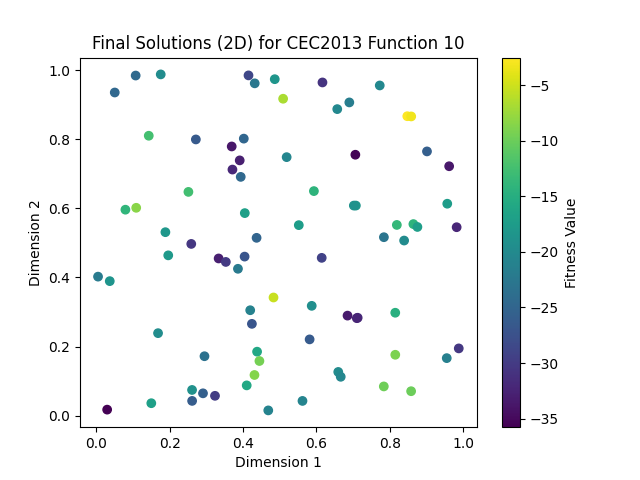
* **Species Formation (Algorithm 2)**:
  + Group the population into species centered around the most fit individuals based on proximity and fitness.
* **New Solution Construction (Algorithm 4)**:
  + Generate new solutions within each species.
  + Compare each new solution with its closest counterpart in the archive and replace the original if the new solution is better.
* **Local Search (Algorithm 5)**:
  + Perform localized fitness improvement on the updated solutions to refine them further.
* **Iteration**:
  + Update fitness values and repeat the process until termination criteria are met.

**Outcome:**

The algorithm produces a refined archive of solutions by combining species-based grouping, fitness-based selection, and local search, ensuring diversity and high-quality solutions.

**Graph Results Of Functions 1-10 For AMC-ACO (Crowding Technique):**





**Graph Results Of Functions 1-10 For AMS-ACO (Speciation Technique):**

