Monkey Search Algorithm

Analysis of Algorithm

BSCS 4-1

Semester Project

Supervised By

Mr. Usman Sharif

Faculty of Computing

Riphah International University

Contents

Monkey Search Algorithm: Theoretical Analysis

Introduction

Monkey Search (MS) is a relatively novel metaheuristic optimization algorithm inspired by the climbing behavior of monkeys in search of food. Introduced by Mucherino and Seref in 2007, it is a stochastic, population-based search technique that blends exploration and exploitation in a unique tree-based framework. The algorithm simulates a monkey climbing a tree, backtracking, and memorizing fruitful paths to eventually reach the best fruit (solution).

Biological Inspiration

The natural foraging behavior of monkeys involves dynamically exploring trees, backtracking on non-promising paths, and progressively memorizing successful routes. This behavior maps effectively to optimization problems where the search space can be represented as a tree, and each path can be evaluated for its potential.

Core Concepts and Mechanisms

1. **Tree Structure**

   The solution space is modeled as a tree, where each node represents a partial or complete solution. The monkey climbs from the root node to the leaf nodes, exploring various paths.

2. **Climbing Process**

   A climbing phase involves the monkey selecting branches (i.e., solution components) to explore. Each move is stochastic, with a bias towards better options.

3. **Backtracking**

   When no further progress can be made or when a poor region is encountered, the monkey backtracks and explores a different path.

4. **Memory**

   The algorithm maintains a memory of good climbing sequences, helping to intensify the search in promising areas while preserving diversity.

5. **Rest and Watch**

   After each climb, a monkey rests and watches the landscape (solution space), adjusting its strategy based on observed improvements or stagnation.

Algorithm Phases

1. **Exploration Phase**

   Randomized climbing with minimal bias. High diversity ensures broader coverage of the solution space.

2. **Climbing and Evaluation Phase**

   Guided climbs using accumulated memory and fitness evaluation of paths.

3. **Backtracking and Intensification Phase**

   Reallocation of search efforts toward promising areas using previously successful routes.

4. **Resting and Learning Phase**

   Incorporation of successful climbing patterns into memory to refine the search strategy.

Convergence Properties

The Monkey Search Algorithm belongs to the class of stochastic metaheuristic algorithms, which typically lack formal mathematical proofs of convergence to the global optimum. However, we can analyze its convergence behavior under specific conditions:

1. **Asymptotic Convergence**: As the number of iterations approaches infinity, the probability of finding the global optimum approaches 1, provided:
   - The somersault process maintains sufficient exploration
   - The climbing process allows fine-tuning of solutions
   - The population size is large enough to cover the search space adequately
2. **Exploration-Exploitation Balance**:
   - The climbing process provides exploitation (local search)
   - The watch-jump process offers mid-range exploration
   - The somersault process ensures global exploration
3. **Parameter Impact on Convergence**:
   - Larger climbing steps accelerate convergence but may reduce solution quality
   - Larger somersault ranges increase exploration but may slow convergence

- Population size trades off computational cost against search coverage

Complexity Analysis

A. Time Complexity

Let's denote:

- n = number of **dimensions**
- p = **population size** (number of monkeys)
- k = **max iterations**

Now break it down per step:

1. Initialization

- Each monkey is initialized with a position of size n: $O(n)$
- This is done p times: $O(p * n)$
- Fitness is evaluated for each monkey: assuming evaluation takes $O(n)$ (like the Sphere function): total = $O(p * n)$

Hence, **Initialization Time: $O(p * n)$**

2. Main Loop

It runs k times. For each iteration, we perform 3 main operations:

i. Climb Process:

- For each monkey: iterate n dimensions and evaluate twice (up and down): $O(n)$ per monkey
- Done for p monkeys: $O(p * n)$

ii. Watch-Jump Process:

- For each monkey, randomly pick one and compare/jump in n dimensions $\rightarrow O(n)$ per monkey
- Total: $O(p * n)$

iii. Somersault Process:

- Compute center of gravity: $O(p * n)$
- Then update all monkey positions again: $O(p * n)$

iv. Update Best Monkey:

- Sort the population: `O(p log p)`

Therefore, Per iteration total: `O(p * n + p log p)`. However, `n` is usually much larger than `log p`. So, we can say that each iteration takes `O(p * n)`.

Overall time complexity: `O(k * p * n)`

B. Space Complexity

1. **Monkeys**
   - Each monkey has a position array of size `n`: `O(n)`
   - You have `p` monkeys: `O(p * n)`
2. **Best Monkey**
   - A clone of a monkey: `O(n)`
3. **Temporary arrays in operations**
   - Climb, watch-jump, and somersault each create `newPosition[]`: up to `O(n)` at a time (not per monkey, reused)

Total space complexity: `O(p * n)`

Search Space Characteristics

The algorithm's performance is influenced by the characteristics of the search space:

1. **Modality**
    - Performs well on unimodal functions due to the climbing process
    - Can handle multimodal functions through the somersault process
    - Highly multimodal landscapes may require larger population sizes
2. **Separability**
    - The dimension-by-dimension climbing approach is particularly effective for separable functions
    - Non-separable functions (where variables interact) present greater challenges
3. **Ruggedness**
    - Smooth functions allow effective climbing
    - Rugged functions with many local optima benefit from the somersault process
4. **Deceptiveness**
    - Functions where local optima give misleading information about the global optimum
    - MSA can overcome deception through the somersault process

Extensions and Variants

Several extensions to the basic Monkey Search Algorithm can improve performance:

1. **Adaptive Parameter Control**
    - Dynamically adjust climbing step and somersault range during the search
    - Example: Decrease somersault range over time to focus on exploitation
2. **Hybrid Approaches**
    - Combine MSA with local search methods for faster convergence
    - Integrate with gradient-based methods for smooth functions
3. **Constraint Handling**
    - Penalty functions for soft constraints
    - Repair mechanisms for solutions violating hard constraints
4. **Parallel Implementation**
    - Island model with multiple populations
    - Shared memory for cooperative search
5. **Multi-objective Extensions**
    - Maintain a Pareto front of non-dominated solutions
    - Adapt the somersault process to maintain diversity in objective space

Conclusion

The Monkey Search Algorithm (MSA) presents a biologically inspired, metaheuristic approach to solving complex optimization problems through a tree-based exploration framework. By mimicking the intelligent foraging behavior of monkeys—such as climbing, backtracking, memorizing successful paths, and adjusting strategies through rest and observation—MSA effectively balances exploration and exploitation across the search space. Its stochastic nature, combined with memory-driven guidance and adaptive movements, allows it to perform competitively on both unimodal and multimodal functions.

While the algorithm lacks formal convergence guarantees like many metaheuristics, its probabilistic convergence behavior and flexibility in handling diverse optimization landscapes make it a strong candidate for a variety of problem domains. The theoretical time and space complexity analyses indicate that MSA is scalable with respect to population size, dimensionality, and iteration count, though its efficiency is closely tied to the characteristics of the search space.

Furthermore, MSA's adaptability through parameter tuning, hybridization, and parallelization offers numerous opportunities for enhancement. With extensions for constraint handling and multi-objective optimization, the Monkey Search Algorithm continues to be a promising and evolving tool in the field of computational intelligence.

Submitted By
Syed Jaffar Raza Kazmi
57375