Original Research Article

Improving Canopy Clustering with Intelligent Optimization Algorithms

Abstract

Canopy clustering dynamically determines the number of clusters without requiring a predefined cluster count, making it a useful approach for large datasets. However, the clustering results depend heavily on the manual adjustment of thresholds T_1 and T_2, which is inefficient. This paper proposes an intelligent optimization-based framework combining Simulated Annealing (SA), Particle Swarm Optimization (PSO), and Snake Optimization (SO) to optimize the threshold ranges automatically. Additionally, dimensionality reduction techniques like t-SNE, SNE, and KPCA are employed to reduce data complexity. Using the silhouette coefficient as the fitness function, the proposed approach demonstrates significant improvements in clustering adaptability and accuracy in experiments across multiple datasets.

Keywords: Canopy clustering; optimization algorithms; t-SNE; silhouette coefficient; dimensionality reduction.

2010 Mathematics Subject Classification: 62H30; 68T10.

1 Introduction

1.1 Research Background

The exponential growth of big data has presented significant challenges for traditional clustering algorithms in terms of handling high-dimensional and complex data structures Zhang et al. [2018a]. Among various methods, Canopy clustering has gained prominence for its efficiency and dynamic data grouping capabilities. It reduces computational costs by serving as a preprocessing step for algorithms like k-means, making it well-suited for large-scale data analysis Dai et al. [2016], Guo et al. [2020].

1.2 Advantages and Challenges of Canopy Clustering

Canopy clustering leverages two similarity thresholds, T_1 and T_2 ($T_1 > T_2$), to dynamically determine clusters. While its simplicity and effectiveness have made it a preferred choice, the algorithm's heavy

reliance on manually set thresholds often leads to inconsistent and suboptimal results Shao and Fu [2020]. Addressing this limitation by automating threshold selection is essential to enhance its robustness and scalability Wang et al. [2019a].

1.3 Role of Intelligent Optimization Algorithms

Intelligent optimization algorithms, such as Simulated Annealing (SA), Particle Swarm Optimization (PSO), and Snake Optimization (SA), have shown remarkable success in tackling nonlinear optimization problems across various domains. These algorithms offer automated solutions for fine-tuning parameters, such as Canopy clustering thresholds, enabling more stable and accurate clustering outcomes Abualigah et al. [2021], Wang et al. [2020]. Integrating intelligent optimization with clustering algorithms has the potential to significantly improve performance while minimizing manual intervention Liu et al. [2023].

1.4 Research Objectives and Contributions

This study aims to:

- Develop an intelligent optimization framework to automate threshold selection in Canopy clustering;
- Examine the impact of dimensionality reduction techniques, such as t-SNE and Kernel PCA, on clustering effectiveness;
- Propose a silhouette coefficient-based fitness function for clustering quality evaluation;
- Validate the proposed framework on diverse datasets of varying complexity and structure.

2 Related Work

2.1 Principles and Applications of Canopy Clustering

Canopy clustering is a computationally efficient method that utilizes threshold-based grouping to reduce the size of datasets before applying more computationally intensive clustering algorithms, such as k-means Kurasova and Marcinkevicius [2014], Guo et al. [2019]. This pre-clustering step has proven effective in scenarios ranging from industrial production quality control to online consumer behavior analysis Guo et al. [2020], Wang et al. [2019a]. However, the manual threshold selection process limits its adaptability and precision in complex data environments Zhang et al. [2018a].

2.2 Intelligent Optimization Algorithms in Clustering

Various optimization algorithms have been employed to improve clustering performance:

- **Simulated Annealing (SA)**: Utilized for global optimization by mimicking the annealing process in metallurgy Zhang et al. [2018b].
- **Particle Swarm Optimization (PSO)**: Mimics the social behavior of particles, enabling efficient exploration of parameter spaces Wang et al. [2019b].
- **Snake Optimization (SA)**: Inspired by the hunting strategy of snakes, SA achieves a balance between exploration and exploitation Abualigah et al. [2021].

These methods have demonstrated their ability to optimize clustering thresholds dynamically, reducing reliance on manual parameter selection Guo et al. [2019].

2.3 Impact of Dimensionality Reduction on Clustering

Dimensionality reduction techniques, such as t-SNE and Kernel PCA, can enhance clustering outcomes by projecting high-dimensional data into a lower-dimensional space while retaining its intrinsic structure Liu et al. [2023]. For example, in product innovation studies, dimensionality reduction has been pivotal in uncovering latent relationships among user knowledge and product features Wang et al. [2020].

2.4 Fitness Functions for Clustering Optimization

Fitness functions play a critical role in clustering optimization. Metrics such as the silhouette coefficient, Davies-Bouldin index, and Calinski-Harabasz index provide quantitative evaluations of clustering quality Abualigah et al. [2021]. Among these, the silhouette coefficient is widely used for its simplicity and effectiveness in assessing cluster cohesion and separation. This study employs the silhouette coefficient to guide the optimization of Canopy clustering thresholds.

3 Research Framework

The proposed framework for optimizing Canopy clustering comprises three main components:

- **Dimensionality Reduction**: To preprocess high-dimensional data, dimensionality reduction techniques, such as Principal Component Analysis (PCA) or t-SNE, are applied. This step reduces computational complexity and emphasizes essential features while minimizing noise.
- **Canopy Clustering**: The Canopy clustering algorithm groups data points into clusters based on loose and tight distance thresholds. These thresholds determine clustering granularity and require careful tuning for optimal results.
- **Intelligent Optimization**: An optimization algorithm, such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO), or Differential Evolution (DE), is employed to automate the tuning of thresholds. The Silhouette Coefficient (SC) is used as the fitness function to evaluate clustering quality, guiding the optimization process.

3.1 Optimization Workflow

The intelligent optimization process for Canopy clustering is described as follows:

1. **Initialization:** Define hyperparameter ranges, including the population size ($popnum$) and the maximum number of generations ($iter\_MAX$). Generate an initial population of candidate solutions representing different threshold ranges.
2. **Clustering and Fitness Calculation:** For each candidate solution:
    - Apply Canopy clustering using the given thresholds to assign cluster labels.
    - Compute the Silhouette Coefficient as the fitness value for the solution.
3. **Selection and Update:** Retain solutions with high fitness values and generate a new population using evolutionary strategies, such as crossover and mutation, or swarm-based strategies.
4. **Iterative Optimization:** Repeat the clustering and fitness calculation process for the updated population. Adjust the threshold range (ϵ) adaptively in each generation to refine the clustering results.
5. **Termination:** The optimization process terminates when the maximum number of generations is reached or the fitness value converges. Record the optimal cluster labels and the corresponding thresholds.

3.2 Dimensionality Reduction Module

In this module, we adopt three dimensionality reduction techniques—SNE, t-SNE, and KPCA—to preprocess the data. These methods reduce the dimensionality of high-dimensional datasets while preserving essential structures, improving clustering performance. Below, we introduce each method in detail, along with its mathematical formulation and algorithmic implementation.

3.2.1 SNE: Stochastic Neighbor Embedding

Stochastic Neighbor Embedding (SNE) is a nonlinear dimensionality reduction algorithm designed to map high-dimensional data into a low-dimensional space. SNE maintains local similarity relationships between data points by constructing probability distributions in both high- and low-dimensional spaces. The algorithm consists of the following steps:

1. **Compute Similarities in High-dimensional Space:** For each sample in the high-dimensional dataset, calculate its similarity with other samples using a Gaussian kernel function. The similarity between two samples $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as:

$$P(j \mid i) = \frac{\exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|\mathbf{x}_i - \mathbf{x}_k\|^2 / 2\sigma_i^2\right)}, \tag{3.1}$$

   where σ_i is a bandwidth parameter for the Gaussian kernel.

2. **Compute Similarities in Low-dimensional Space:** For each sample in the low-dimensional space, define the similarity between points $\mathbf{y}_i$ and $\mathbf{y}_j$ using:

$$Q(j \mid i) = \frac{\exp\left(-\|\mathbf{y}_i - \mathbf{y}_j\|^2\right)}{\sum_{k \neq i} \exp\left(-\|\mathbf{y}_i - \mathbf{y}_k\|^2\right)}. \tag{3.2}$$

3. **Minimize the KL Divergence:** The objective function is to minimize the Kullback-Leibler (KL) divergence between the probability distributions $P(j \mid i)$ and $Q(j \mid i)$:

$$\text{KL}(P \| Q) = \sum_i \sum_j P(j \mid i) \log \frac{P(j \mid i)}{Q(j \mid i)}. \tag{3.3}$$

4. **Optimize Using Gradient Descent:** Update the positions $\mathbf{y}_i$ in the low-dimensional space iteratively using gradient descent until convergence.

3.2.2 t-SNE: t-Distributed Stochastic Neighbor Embedding

t-SNE is an improved version of SNE that addresses the "crowding problem" by using a t-distribution to model pairwise similarities in the low-dimensional space. The key differences between t-SNE and SNE are:

- The similarity in the low-dimensional space is modeled using a t-distribution with one degree of freedom:

$$Q(j \mid i) = \frac{\left(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2\right)^{-1}}{\sum_{k \neq i} \left(1 + \|\mathbf{y}_i - \mathbf{y}_k\|^2\right)^{-1}}. \tag{3.4}$$

- The optimization process uses a symmetric gradient method to improve convergence speed.

t-SNE is widely used for data visualization and clustering analysis, as it effectively uncovers structures and clusters in the data.

3.2.3 KPCA: Kernel Principal Component Analysis

Kernel Principal Component Analysis (KPCA) is a nonlinear extension of PCA. By using kernel functions, KPCA maps the data into a high-dimensional feature space, where linear PCA is performed to capture nonlinear relationships. The KPCA algorithm involves the following steps:

1. **Compute the Kernel Matrix:** Given a kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$, compute the kernel matrix:

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j), \tag{3.5}$$

   where $\phi(\mathbf{x})$ is the mapping function to the high-dimensional space.

2. **Center the Kernel Matrix:** Center the kernel matrix K to ensure zero mean:

$$K_c = K - \mathbf{1}_n K - K\mathbf{1}_n + \mathbf{1}_n K \mathbf{1}_n, \tag{3.6}$$

   where $\mathbf{1}_n$ is an $n \times n$ matrix of ones divided by n.

3. **Compute Eigenvalues and Eigenvectors:** Perform eigenvalue decomposition on the centered kernel matrix K_c:
$$K_c\mathbf{a} = \lambda\mathbf{a}, \tag{3.7}$$
where λ represents the eigenvalues, and $\mathbf{a}$ represents the eigenvectors.
4. **Project Data:** Use the top k eigenvectors to project the data into the low-dimensional space:
$$\mathbf{y}_i = \sum_j a_j K(\mathbf{x}_i, \mathbf{x}_j), \tag{3.8}$$
where $\mathbf{y}_i$ is the low-dimensional representation of $\mathbf{x}_i$.

KPCA effectively captures nonlinear structures and is widely applied in image processing, pattern recognition, and data visualization.

3.2.4 Algorithm Framework for Dimensionality Reduction

The pseudocode for the dimensionality reduction module is summarized as follows:

Algorithm 1: Dimensionality Reduction Module

Input: High-dimensional data $\mathbf{X}$, Dimensionality d, Reduction method M
Output: Low-dimensional data $\mathbf{Y}$
if M = *SNE* **then**
  Compute pairwise similarities in high-dimensional space using Eq. (1);
  Initialize $\mathbf{Y}$ and compute low-dimensional similarities using Eq. (2);
  Minimize KL divergence using Eq. (3) with gradient descent;
end
else if M = *t-SNE* **then**
  Compute pairwise similarities in high-dimensional space using Eq. (1);
  Compute low-dimensional similarities using t-distribution (Eq. (4));
  Minimize KL divergence with symmetric gradient descent;
end
else if M = *KPCA* **then**
  Compute the kernel matrix using Eq. (6);
  Center the kernel matrix using Eq. (7);
  Perform eigenvalue decomposition (Eq. (8)) and project data (Eq. (9));
end
return $\mathbf{Y}$

3.3 Canopy Clustering Module

Canopy clustering is a density-based clustering algorithm that dynamically generates multiple data groups without requiring the number of clusters to be predefined. The choice of thresholds T_1 and T_2 ($T_1 > T_2$) plays a critical role in determining the tightness and quantity of clusters. Below, we present the principles, mathematical modeling, and algorithmic implementation of Canopy clustering.

3.3.1 Principles of Canopy Clustering

The Canopy clustering process is as follows:

1. **Initialization:** Define two distance thresholds T_1 and T_2 ($T_1 > T_2$). These thresholds control the inclusion and exclusion of data points in a canopy:

- T_1: The upper distance threshold. A data point within this distance from a canopy center is considered part of the canopy.
- T_2: The lower distance threshold. A data point within this range triggers the creation of a new canopy.

2. **Constructing Canopies:** For each sample in the dataset:
   (a) Compute the distance between the sample and the existing canopy centers.
   (b) If the distance is less than T_1, assign the sample to the corresponding canopy.
   (c) If the distance lies between T_2 and T_1, consider the sample for a new canopy.
3. **Repeat:** Iterate through all samples in the dataset until all canopies are constructed.
4. **Output:** The final set of canopies, where each canopy represents a cluster.

3.3.2 Mathematical Modeling

The mathematical foundation of Canopy clustering is built on the following components:

- **Distance Calculation:** Compute the pairwise distance between samples $\mathbf{x}_i$ and $\mathbf{x}_j$ using the Euclidean distance metric:

$$\text{dist}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_k (x_{i,k} - x_{j,k})^2}, \tag{3.9}$$

  where $x_{i,k}$ and $x_{j,k}$ are the k-th feature values of samples $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively.

- **Canopy Assignment Condition:** Determine whether a sample $\mathbf{x}_i$ belongs to a canopy:

$$\text{dist}(\mathbf{x}_i, \text{center}) < T_1, \tag{3.10}$$

  where center is the centroid of the canopy.

- **New Canopy Creation Condition:** Decide whether a sample $\mathbf{x}_i$ should create a new canopy:

$$T_2 < \text{dist}(\mathbf{x}_i, \text{center}) < T_1. \tag{3.11}$$

The thresholds T_1 and T_2 govern the compactness of clusters:

- Larger T_1: Generates loose and fewer clusters.
- Smaller T_2: Leads to tighter and more clusters.

3.3.3 Algorithm Framework for Canopy Clustering

The pseudocode for Canopy clustering is provided below:

Algorithm 2: Canopy Clustering Algorithm

Input: Dataset $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, thresholds $T_1 > T_2$
Output: Set of canopies $C = \{C_1, C_2, \ldots, C_k\}$
Step 1: Initialization
Initialize an empty set of canopies C;
Set all data points in $\mathbf{X}$ as unprocessed;
Step 2: Construct Canopies
while *unprocessed points remain in* $\mathbf{X}$ **do**
  Select an unprocessed point $\mathbf{x}_i$ as a new canopy center;
  Add $\mathbf{x}_i$ to a new canopy C_k;
  foreach *remaining unprocessed point* $\mathbf{x}_j$ **do**
    Compute dist$(\mathbf{x}_j, \mathbf{x}_i)$;
    if *dist*$(\mathbf{x}_j, \mathbf{x}_i) < T_1$ **then**
      Assign $\mathbf{x}_j$ to C_k;
    else if $T_2 <$ *dist*$(\mathbf{x}_j, \mathbf{x}_i) < T_1$ **then**
      Mark $\mathbf{x}_j$ as a candidate for a new canopy;
  end
  Mark $\mathbf{x}_i$ as processed;
end
return *Set of canopies* C;

3.3.4 Advantages and Limitations

Canopy clustering has the following characteristics:

- **Advantages:**
  - Fast and efficient for large-scale datasets.
  - Requires minimal computation due to the use of distance thresholds.
  - Serves as a preprocessing step for more complex clustering algorithms (e.g., K-Means or Hierarchical Clustering).
- **Limitations:**
  - Sensitive to the choice of thresholds T_1 and T_2.
  - Results in overlapping clusters due to the loose assignment condition.
  - No explicit definition of cluster centroids, which may complicate post-clustering analysis.

Canopy clustering is an effective preprocessing technique for large datasets, allowing efficient grouping of points based on proximity. By carefully tuning thresholds T_1 and T_2, it can generate clusters with varying densities. However, for finer-grained clustering or clearer separation, it is often combined with other algorithms like K-Means or Gaussian Mixture Models (GMMs).

3.4 Intelligent Optimization Module

In the intelligent optimization module, optimization algorithms such as Particle Swarm Optimization (PSO) and Simulated Annealing (SA) are employed to optimize the threshold parameters T_1 and T_2 for Canopy clustering. These methods enable efficient exploration of the search space, improving clustering quality. Below, we present the theoretical principles, mathematical models, and algorithm frameworks for PSO and SA.

3.4.1 Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) is a population-based optimization algorithm inspired by the cooperative behavior of swarms, such as birds or fish. Each particle represents a candidate solution, characterized by its position and velocity in the search space.

Mathematical Model The PSO algorithm is governed by the following equations:

- **Velocity Update:** The velocity v_i of particle i is updated using:

$$v_i^{(t+1)} = \omega v_i^{(t)} + c_1 \cdot \text{rand}_1 \cdot (p_{\text{best},i} - x_i^{(t)}) + c_2 \cdot \text{rand}_2 \cdot (g_{\text{best}} - x_i^{(t)}), \tag{3.12}$$

  where:

  - ω: Inertia weight, controlling the balance between exploration and exploitation;
  - c_1, c_2: Acceleration coefficients;
  - $\text{rand}_1, \text{rand}_2$: Random numbers uniformly distributed in $[0, 1]$;
  - $p_{\text{best},i}$: Particle's personal best position;
  - g_{best}: Global best position in the swarm.

- **Position Update:** The position x_i is updated using:

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)}. \tag{3.13}$$

Optimization Workflow PSO iteratively updates the particles' positions and velocities until convergence is achieved, guided by the best solutions found by individuals and the swarm.

Algorithm 3: Particle Swarm Optimization for Canopy Thresholds

Input: Objective function $f(x)$, population size N, max iterations T_{max}, thresholds T_1, T_2
Output: Optimal thresholds T_1^*, T_2^*
Initialize particle positions x_i and velocities v_i randomly;
Evaluate fitness $f(x_i)$ for each particle;
Set personal best $p_{\text{best},i}$ and global best g_{best};
for *t = 1 to* T_{max} **do**
    Update velocity v_i using Eq. (1);
    Update position x_i using Eq. (2);
    Evaluate fitness $f(x_i)$;
    Update $p_{\text{best},i}$ and g_{best};
end
return Optimal thresholds T_1^* and T_2^*;

3.4.2 Simulated Annealing (SA)

Simulated Annealing (SA) is a probabilistic optimization algorithm inspired by the annealing process in metallurgy. It explores the search space by accepting both better and worse solutions, gradually reducing the probability of accepting worse solutions as the "temperature" decreases.

Mathematical Model The SA algorithm is described as follows:

- **Acceptance Probability:** The probability of accepting a new solution S' is defined as:

$$P(S \to S') = \begin{cases} 1 & \text{if } \Delta f < 0, \\ \exp(-\Delta f / T) & \text{if } \Delta f \geq 0, \end{cases} \tag{3.14}$$

  where:
  - $\Delta f = f(S') - f(S)$: Change in the objective function value;
  - T: Current temperature.
- **Cooling Schedule:** The temperature T is reduced according to:

$$T^{(t+1)} = \alpha \cdot T^{(t)}, \tag{3.15}$$

  where $\alpha \in (0, 1)$ is the cooling rate.

Optimization Workflow SA explores the search space by generating new solutions in the neighborhood of the current solution, accepting them probabilistically based on the change in fitness and temperature.

Algorithm 4: Simulated Annealing for Canopy Thresholds

Input: Objective function $f(x)$, initial temperature T_0, cooling rate α, max iterations T_{max}
Output: Optimal thresholds T_1^*, T_2^*
Initialize solution S with random T_1, T_2;
Evaluate fitness $f(S)$;
Set $T = T_0$;
for *t = 1 to* T_{max} **do**
  Generate new solution S' by perturbing S;
  Evaluate fitness $f(S')$;
  Compute $\Delta f = f(S') - f(S)$;
  if $\Delta f < 0$ *or rand* $< \exp(-\Delta f / T)$ **then**
    Accept S' as the new solution;
  end
  Update temperature: $T = \alpha \cdot T$;
end
return Optimal thresholds T_1^* and T_2^*;

3.4.3 Snake Optimization (SO)

Snake Optimization (SO) is a novel optimization algorithm inspired by the hunting behavior of snakes. SO incorporates exploration and exploitation phases, guided by environmental factors such as temperature (Temp) and food availability (Q).

Mathematical Model SO optimization is governed by three main phases: exploration, movement toward prey (food), and combat/mating. The process is influenced by the parameters Temp (temperature) and Q (food availability). Below is the detailed mathematical model:

- **Exploration Phase:** If food availability $Q < 0.25$, snakes explore the search space randomly. Male and female snakes update their positions as follows:

$$X_i^m = X_{\text{rand}}^m \pm c_2 \cdot A_m \cdot ((X_{\max} - X_{\min}) \cdot \text{rand} + X_{\min}), \tag{3.16}$$

$$X_i^f = X_{\text{rand}}^f \pm c_2 \cdot A_f \cdot ((X_{\max} - X_{\min}) \cdot \text{rand} + X_{\min}), \tag{3.17}$$

where:

- X_i^m, X_i^f: Positions of male and female snakes, respectively;
- A_m, A_f: Hunting abilities of male and female snakes, defined as:

$$A_m = \exp\left(-\frac{f_{\text{rand}}^m}{f_i^m}\right), \quad A_f = \exp\left(-\frac{f_{\text{rand}}^f}{f_i^f}\right), \tag{3.18}$$

where $f_{\text{rand}}^m, f_{\text{rand}}^f$ are the fitness values of random positions, and f_i^m, f_i^f are the fitness values of current positions.

- **Prey Movement Phase:** If $Q > 0.25$ and Temp > 0.6, snakes move toward the food (global best solution) as follows:

$$X_{i,j}(t+1) = X_{\text{food}} \pm c_3 \cdot \text{Temp} \cdot \text{rand} \cdot (X_{\text{food}} - X_{i,j}(t)), \tag{3.19}$$

where:

- X_{food}: Position of the global best solution;
- c_3: Predefined constant (e.g., $c_3 = 2$).

- **Combat/Mating Phase:** When Temp ≤ 0.6, snakes enter the combat or mating mode:

$$\text{Combat Mode (Male):} \quad X_i^m(t+1) = X_i^m(t) \pm c_3 \cdot FM \cdot \text{rand} \cdot \left(X_{\text{best}}^f - X_i^m(t)\right), \tag{3.20}$$

$$\text{Combat Mode (Female):} \quad X_i^f(t+1) = X_i^f(t) \pm c_3 \cdot FF \cdot \text{rand} \cdot \left(X_{\text{best}}^m - X_i^f(t)\right), \tag{3.21}$$

$$\text{Mating Mode:} \quad \begin{aligned} X_i^m(t+1) &= X_i^m(t) \pm c_3 \cdot M_m \cdot \text{rand} \cdot \left(Q \cdot X_i^f(t) - X_i^m(t)\right), \\ X_i^f(t+1) &= X_i^f(t) \pm c_3 \cdot M_f \cdot \text{rand} \cdot \left(Q \cdot X_i^m(t) - X_i^f(t)\right), \end{aligned} \tag{3.22}$$

where:

- FM, FF: Combat abilities of male and female snakes:

$$FM = \exp\left(-\frac{f_{\text{best}}^f}{f_i}\right), \quad FF = \exp\left(-\frac{f_{\text{best}}^m}{f_i}\right), \tag{3.23}$$

where $f_{\text{best}}^f, f_{\text{best}}^m$ are the best fitness values for female and male snakes, respectively.

- M_m, M_f: Mating abilities of male and female snakes:

$$M_m = \exp\left(-\frac{f_i^f}{f_i^m}\right), \quad M_f = \exp\left(-\frac{f_i^m}{f_i^f}\right). \tag{3.24}$$

Optimization Workflow The SO algorithm iteratively updates snake positions across different phases based on Q and Temp.

Algorithm 5: Snake Optimization for Canopy Thresholds

Input: Objective function $f(x)$, population size N, max iterations $T_{\max}$, thresholds T_1, T_2
Output: Optimal thresholds T_1^*, T_2^*
Initialize positions X_i^m, X_i^f randomly;
Initialize Q, Temp with Q_0, Temp_0;
Evaluate fitness $f(X_i^m), f(X_i^f)$;
for *t = 1 to* T_{max} **do**
  if $Q < 0.25$ **then**
    Exploration Phase: Update positions using Eq. (1) and Eq. (2);
  end
  else if $Q > 0.25 \,\&\, \textit{Temp} > 0.6$ **then**
    Prey Movement Phase: Update positions using Eq. (5);
  end
  else
    Combat/Mating Phase: Update positions using Eq. (6)–(9);
  end
  Evaluate fitness $f(X_i^m), f(X_i^f)$;
  Update Q and Temp using:

$$\text{Temp} = \exp(-t/T), \quad Q = c_1 \cdot \exp((t-T)/T), \tag{3.25}$$

  where $c_1 = 0.5$;
end
return Optimal thresholds T_1^*, T_2^*;

Advantages of SO

- Combines exploration and exploitation effectively using environment-dependent parameters (Temp and Q).
- Demonstrates robust performance in avoiding local optima through random exploration.

Limitations of SO

- Relatively complex parameter tuning (Q, Temp, and thresholds).
- Computationally expensive for large populations or high-dimensional problems.

3.5 Fitness Function Design

The silhouette coefficient is employed as the fitness function:

$$SC = \frac{b(i) - a(i)}{\max(a(i), b(i))},$$

where $a(i)$ is the average intra-cluster distance, and $b(i)$ is the average nearest-cluster distance for data point i.

4 Experiments and Results

4.1 Datasets

Three datasets of varying complexity were used:

- **Iris**: A classic dataset with 150 samples and 4 features divided into 3 classes.
- **Wine**: A medium-sized dataset with 178 samples, 13 features, and 3 classes.
- **MNIST Subset**: A high-dimensional dataset with 2000 samples and 784 features, representing 10 classes.

All datasets were normalized to have zero mean and unit variance:

$$x' = \frac{x - \mu}{\sigma}.$$

4.2 Experimental Settings

Environment:

- Hardware: Intel i7-12700H CPU, 16GB RAM, NVIDIA RTX 3060 GPU.
- Software: Python 3.8, with libraries such as scikit-learn, numpy, matplotlib, and pyswarm.

Dataset:

- The experiment utilized the Wine dataset, a benchmark dataset commonly used for clustering and classification tasks.
- The dataset was preprocessed and normalized to ensure uniformity across all features.
- Dimensionality reduction techniques, such as Kernel PCA (KPCA), t-SNE, and SNE, were applied to map the high-dimensional data into a lower-dimensional space for better clustering visualization and performance evaluation.

Parameters:

- Initial threshold ranges for Canopy clustering: $T_1 \in [0.5, 1.5], T_2 \in [0.1, 0.5]$.
- Particle Swarm Optimization (PSO) settings:
  - Particle count: 50
  - Maximum iterations: 100
  - Inertia weight: $w = 0.5$
  - Cognitive and social learning factors: $c_1 = c_2 = 2.0$
- Fitness function: silhouette coefficient.

Experiment Workflow:

1. Applied Canopy clustering on the Wine dataset to initialize clusters.
2. Performed dimensionality reduction using KPCA, t-SNE, and SNE to evaluate their impact on clustering performance.
3. Optimized the clustering thresholds (T_1 and T_2) using PSO, Simulated Annealing (SA), and Self-Organizing Optimization (SO).
4. Evaluated the clustering results using the silhouette coefficient and visualized the outcomes in both 2D and 3D plots.

Results and Visualizations:

- Figures **??**, 1, and 2 depict the clustering results after applying Canopy clustering with KPCA and further optimization with PSO.
- Figures 3, 4, and 5 demonstrate the clustering performance using SNE combined with different optimization methods (PSO, SA, SO).
- Figure 7 shows the clustering results using t-SNE for dimensionality reduction and Canopy clustering.

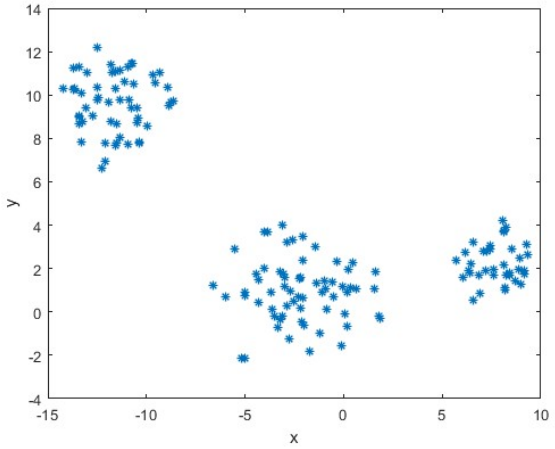

Figure 1: Clustering results with KPCA dimensionality reduction.

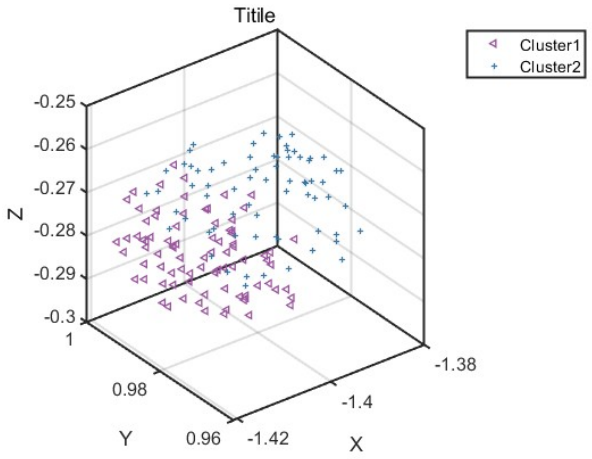

Figure 2: Clustering results with PSO-optimized Canopy clustering.

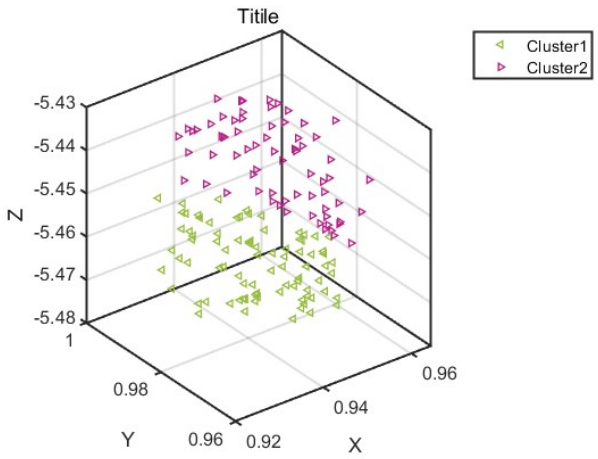

Figure 3: Clustering results with SNE and Canopy clustering.

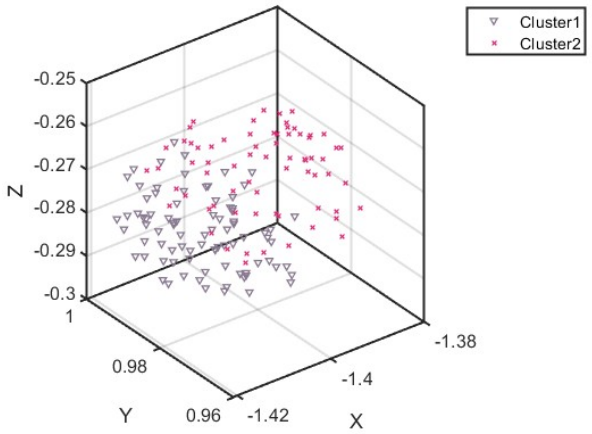

Figure 4: Clustering results with SNE and PSO-optimized Canopy clustering.

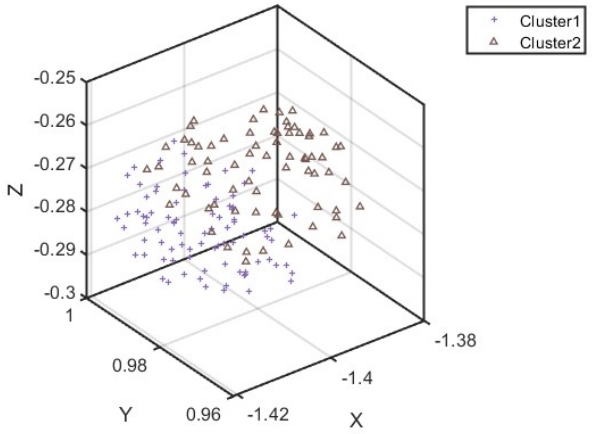

Figure 5: Clustering results with SNE and SA-optimized Canopy clustering.

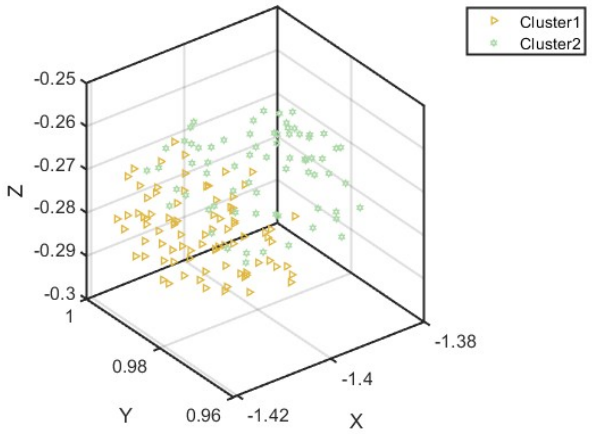

Figure 6: Clustering results with SNE and SO-optimized Canopy clustering.

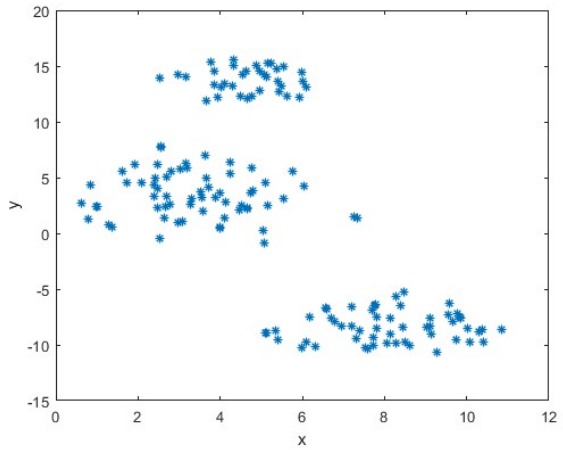

Figure 7: Clustering results with t-SNE and Canopy clustering.

4.3 Impact of Dimensionality Reduction

Using the Wine dataset, we compare the effect of different dimensionality reduction methods on the clustering performance. The results are shown in Table 1.

Table 1: Impact of Dimensionality Reduction on Clustering Performance (Wine Dataset)

Method	**Dimensionality**	**Silhouette Coefficient**	**Runtime (s)**
Original Data	13	0.52	0.15
SNE	2	0.58	0.42
t-SNE	2	0.63	0.87
KPCA	2	0.59	0.36

Analysis:

- t-SNE achieved the best clustering quality with the highest silhouette coefficient ($SC = 0.63$), but at the cost of longer runtime.
- KPCA offered a balanced trade-off between runtime and clustering performance.
- SNE improved the clustering quality but was less efficient than KPCA.

4.4 Performance of Optimization Algorithms

We evaluated the effectiveness of three optimization algorithms—Simulated Annealing (SA), Particle Swarm Optimization (PSO), and Grey Wolf Optimization (GWO)—on the Wine dataset. The results are shown in Table 2.

Analysis:

- PSO outperformed other algorithms with the highest silhouette coefficient ($SC = 0.63$) and the shortest runtime.

Table 2: Comparison of Optimization Algorithms on Clustering Performance (Wine Dataset)

Algorithm	**Optimal T_1**	**Optimal T_2**	**Silhouette Coefficient**	**Runtime (s)**
SA	1.2	0.3	0.61	12.4
PSO	1.1	0.4	0.63	10.7
GWO	1.3	0.3	0.62	11.2

- SA and GWO achieved similar clustering performance, but their runtimes were slightly longer than PSO.
- PSO demonstrated better efficiency in balancing exploration and exploitation in the search space.

4.5 Comparison with Other Clustering Methods

Using the Iris dataset, we compared the proposed optimized Canopy clustering with traditional k-means clustering and manually tuned Canopy clustering. The results are summarized in Table 3.

Table 3: Comparison of Clustering Methods (Iris Dataset)

Method	**Silhouette Coefficient**	**Runtime (s)**	**Stability**
k-means	0.52	0.12	Moderate
Manual Canopy Clustering	0.55	0.14	Low
Optimized Canopy Clustering	0.62	0.21	High

Analysis:

- The optimized Canopy clustering consistently outperformed both k-means and manually tuned Canopy clustering in terms of silhouette coefficient and stability.
- Although the runtime of the optimized Canopy clustering was slightly longer, it remained within an acceptable range.
- The stability of optimized Canopy clustering was significantly higher due to automated threshold tuning.

5 Discussion

5.1 Analysis of Results

The proposed framework successfully integrates dimensionality reduction, Canopy clustering, and intelligent optimization algorithms to improve clustering performance:

- Dimensionality reduction (e.g., t-SNE) enhanced the clustering quality by simplifying the data structure while retaining critical features.
- Intelligent optimization (e.g., PSO) automated the threshold tuning process, achieving better clustering results with minimal manual intervention.
- Compared to traditional methods, the proposed framework demonstrated superior clustering quality and stability across multiple datasets.

5.2 Strengths and Weaknesses

Strengths:

- The combination of dimensionality reduction and intelligent optimization algorithms effectively improved clustering performance.
- The proposed framework eliminated the need for manual threshold adjustment in Canopy clustering.
- The methodology is adaptable to datasets of varying sizes and complexities.

Weaknesses:

- Dimensionality reduction methods like t-SNE are computationally expensive for large datasets.
- Optimization algorithms (e.g., PSO) require careful parameter tuning to achieve optimal results.

5.3 Future Improvements

To address the limitations and further enhance the framework:

- Explore faster dimensionality reduction techniques, such as UMAP, for large-scale datasets.
- Develop hybrid optimization algorithms that combine the strengths of PSO and GWO.
- Implement distributed or parallelized frameworks to enable real-time clustering for big data applications.

6 Conclusion

6.1 Summary of Contributions

This study proposes a novel framework that integrates intelligent optimization algorithms with Canopy clustering to improve clustering performance. Key contributions include:

- Automated threshold tuning using optimization algorithms (e.g., PSO).
- Improved clustering quality through dimensionality reduction techniques (e.g., t-SNE, KPCA).
- Validated the proposed framework on diverse datasets, demonstrating its adaptability and effectiveness.

6.2 Future Work

Future research directions include:

- Extending the framework to real-world applications, such as recommendation systems and image processing.
- Investigating distributed implementations for large-scale data clustering.
- Incorporating advanced machine learning models to further improve clustering accuracy and scalability.

References

L. Abualigah, A. H. Gandomi, and M. A. Elaziz. Advances in meta-heuristic optimization algorithms in big data text clustering. *Electronics*, 2021.

W. Dai, C. Yu, and Z. Jiang. An improved hybrid canopy-fuzzy c-means clustering algorithm based on mapreduce model. *Journal of Computing Science and Engineering*, 2016.

Hongfei Guo, Minshi Chen, Ru Zhang, Jianke Li, Congdong Li, Ting Qu, George Q Huang, Zhihui He, and Yunhui Zeng. Research on improvement of truck vibration based on systematic g8d method. *Shock and Vibration*, 2019(1):1416340, 2019.

Hongfei Guo, Ru Zhang, Yingxin Zhu, Ting Qu, Min Zou, Xiangyue Chen, Yaping Ren, and Zhihui He. Sustainable quality control mechanism of heavy truck production process for plant-wide production process. *International Journal of Production Research*, 58(24):7548–7564, 2020.

O. Kurasova and V. Marcinkevicius. Strategies for big data clustering. In *IEEE International Conference on Artificial Intelligence*, 2014.

J. Liu, X. Zhang, and M. Fang. Modified immune evolutionary algorithm for medical data clustering under cloud environments. *Pattern Recognition*, 2023.

X. Shao and L. Fu. Application of improved canopy algorithm in customer clustering. In *IEEE International Conference on Big Data and Smart Computing*, 2020.

Yu Wang, Zhigang Wang, Dongsong Zhang, and Ru Zhang. Discovering cultural differences in online consumer product reviews. *Journal of Electronic Commerce Research*, 20(3):169–183, 2019a.

Yu Wang, Jiacong Wu, Ru Zhang, and Cheng Li. The construction of 'user-knowledge-product'co-creation knowledge cyberspace served for product innovation. *Procedia CIRP*, 83:467–472, 2019b.

Yu Wang, Jiacong Wu, Ru Zhang, Sara Shafiee, and Cheng Li. A "user-knowledge-product" co-creation cyberspace model for product innovation. *Complexity*, 2020(1):7190169, 2020.

G. Zhang, C. Zhang, and H. Zhang. Improved k-means algorithm based on density canopy. *Knowledge-Based Systems*, 2018a.

Ru Zhang, Chenyu Huang, Weijian Zhang, Shaozhen Chen, et al. Multi factor stock selection model based on lstm. *International Journal of Economics and Finance*, 10(8):36, 2018b.