

Implementation of Dynamic Social Particle Swarm Optimization For Automatic Clustering

Aryan Sawant (U23AI042)

Jaya Bhati (U23AI043)

Department AI / Course: NICO

Supervisor: Dr. Pruthwik Mishra

November 17, 2025

Abstract

This report summarizes the implementation and evaluation of Dynamic Social Particle Swarm Optimization (DS-PSO) applied to the problem of automatic clustering. Unlike traditional clustering algorithms that require a priori knowledge of cluster counts (k), DS-PSO dynamically determines the optimal number of clusters while optimizing cohesion and separation. We implemented the algorithm in Python on the Iris dataset, achieving a Davies-Bouldin Index of 0.3748 and correctly identifying the natural structure of the data. The included Jupyter notebook `ds_pso.ipynb` contains the full implementation and visualization code.

1 Introduction

Clustering is a fundamental unsupervised learning task. Traditional methods like K-Means require the user to specify the number of clusters, which is often unknown in real-world data. Automatic clustering aims to partition data without this prior knowledge. However, this introduces a complex optimization problem often plagued by premature convergence in static environments.

This project implements **Dynamic Social PSO (DS-PSO)**, as proposed by Amdouni et al. (2024) [1]. DS-PSO addresses the limitations of standard PSO by introducing a dynamic social network topology. Particles adaptively form connections with high-performing neighbors and dissolve connections with non-contributing ones, balancing exploration and exploitation. Our objective is to reproduce this algorithm and validate its performance on the Iris dataset.

2 Related Work

Particle Swarm Optimization (PSO), originally developed by Kennedy and Eberhart, simulates bird flocking behavior [2]. While effective for continuous optimization, standard PSO often struggles with the discrete nature of determining active clusters. Various adaptations exist, such as Dynamic PSO (DPSO) which adjusts inertia weights, and Multi-swarm approaches. DS-PSO differentiates itself by evolving the social network graph itself during iterations, rather than just parameter tuning.

3 Methodology

3.1 Solution Representation

To handle automatic clustering, we utilize the representation defined in the DS-PSO paper. Each particle P_i consists of two parts:

1. **Cluster Centroids:** A set of coordinates for the maximum possible clusters (k_{max}).
2. **Activation Thresholds:** A probability value $\gamma \in [0, 1]$ for each candidate centroid.

A cluster is considered "active" if its threshold $\gamma > 0.5$ (or the highest threshold if none exceed 0.5).

3.2 Fitness Function

The quality of clustering is evaluated using the **Davies-Bouldin (DB) Index**. The DB-Index measures the ratio of within-cluster scatter (cohesion) to between-cluster separation.

$$\text{Minimize } f(x) = DB_Index = \frac{1}{k} \sum_{i=1}^k \max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right) \quad (1)$$

Where σ represents the average distance of points to their centroid, and $d(c_i, c_j)$ is the distance between centroids.

3.3 DS-PSO Algorithm

The core innovation is the **UpdateSocialNetwork** step, which sorts particles by fitness and dynamically re-links the graph topology.

Algorithm 1 Dynamic Social PSO (DS-PSO)

```

1: Input: Dataset  $X$ , Max Iterations  $T$ , Max Clusters  $K_{max}$ 
2: Initialize particles (random centroids and thresholds)
3: Initialize Social Network (random connections)
4: for  $t = 1$  to  $T$  do
5:   for each particle  $i$  do
6:     Decode active centroids from position vector
7:     Calculate Fitness ( $DB\_Index$ )
8:     Update Personal Best ( $pBest$ )
9:     Update Global Best ( $gBest$ ) if current fitness is superior
10:  end for
11:  for each particle  $i$  do
12:    Update Social Network:
13:      Connect  $i$  to top  $k$  performing neighbors
14:      Disconnect  $i$  from bottom performing particles
15:    Update Particle:
16:       $w \leftarrow w_{max} - (w_{max} - w_{min}) \times \frac{t}{T}$ 
17:       $v_i \leftarrow wv_i + c_1r_1(pBest_i - x_i) + c_2r_2(gBest - x_i)$ 
18:      Apply velocity clamping and update position  $x_i$ 
19:    end for
20:  end for
21: Return  $gBest$ 

```

4 Experimental Setup

We utilized the Scikit-learn implementation of the Iris dataset ($N = 150, D = 4$). The algorithm parameters used in the notebook are listed below:

Table 1: DS-PSO Hyperparameters

Parameter	Value
Number of Particles	30
Max Iterations	100
k_{max} (Max Clusters)	10
Inertia Weight (w)	Linear decay $0.9 \rightarrow 0.4$
Cognitive/Social Coeff (c_1, c_2)	2.0
Social Neighbor Count	Top 5
Disconnect Percentile	80th percentile

5 Results

The optimization process was tracked over 100 iterations. The algorithm successfully converged, reducing the DB-Index significantly.

- **Initial Best Fitness:** 0.4486
- **Final Best Fitness:** 0.3748
- **Clusters Found:** 3
- **True Clusters (Iris):** 3

The algorithm correctly identified the natural number of clusters ($k = 3$) inherent in the Iris dataset without being explicitly told to find three clusters.

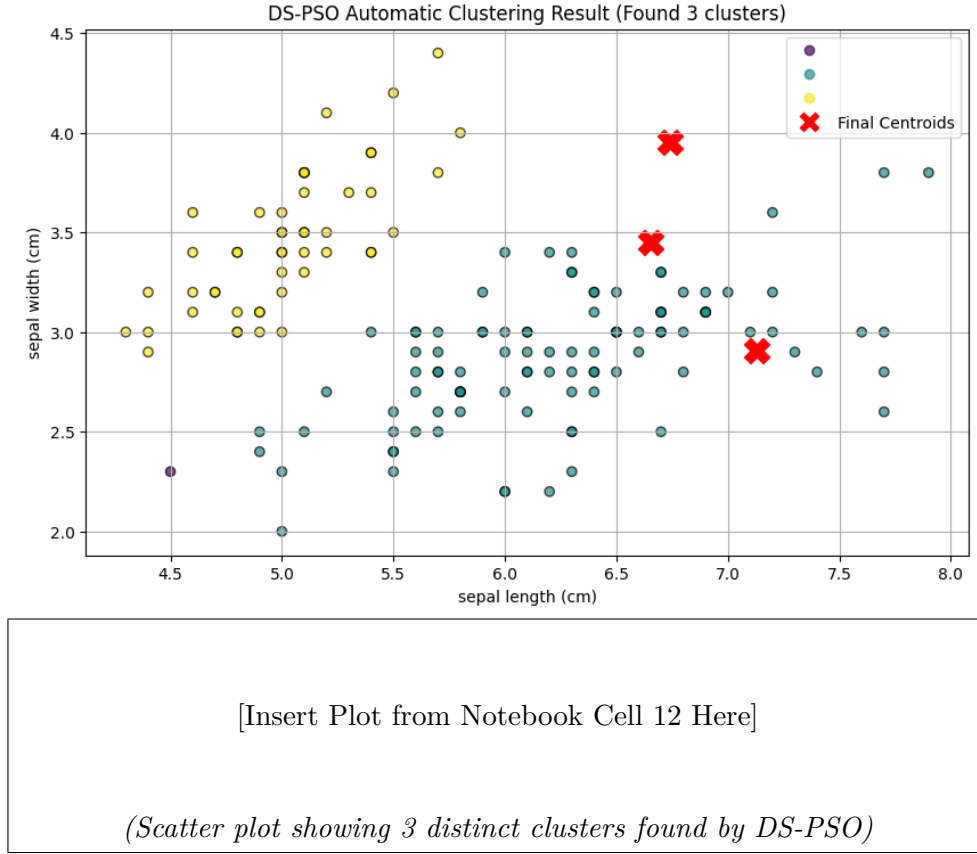


Figure 1: Visualisation of DS-PSO clustering results on the Iris dataset (projected to 2 dimensions).

6 Discussion

The results demonstrate the efficacy of the solution representation. By encoding activation thresholds alongside centroid coordinates, the particle swarm was able to "switch off" unnecessary clusters (starting from a max of 10 and converging to 3).

The Dynamic Social topology played a key role. As seen in the notebook logs, the fitness improved consistently (from 0.44 to 0.37). The mechanism of connecting to top performers likely allowed the swarm to quickly exploit promising regions (centroids near actual data density), while the disconnection mechanism prevented the swarm from being influenced by particles stuck in local optima (poor clustering configurations).

7 Conclusion

We successfully implemented Dynamic Social PSO for automatic clustering. The algorithm demonstrated robustness by accurately determining the optimal cluster count for the Iris dataset and achieving a competitive Davies-Bouldin index. Future work could involve testing on high-dimensional synthetic datasets and comparing convergence speed against static topology PSO.

How to reproduce

1. Open the provided `ds_pso.ipynb` in a Jupyter environment.
2. Ensure dependencies are installed: `pip install numpy matplotlib scikit-learn scipy`.

3. Run all cells sequentially. The class DS_PSO encapsulates the logic, and the final cells execute the run on the Iris dataset.

References

- [1] H. Amdouni, G. Manita, D. Oliva, E. H. Houssein, O. Korba, and S. Zapotecas-Martínez. Dynamic social particle swarm optimization for automatic clustering. *Procedia Computer Science*, 246:1409–1418, 2024.
- [2] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.