Adaptive and Robust DBSCAN with Multi-agent Reinforcement Learning

Hao Peng, Xiang Huang, Shuo Sun, Ruitong Zhang, Xizhao Wang, *Fellow, IEEE* Philip S. Yu, *Fellow, IEEE*

Abstract—Density-Based Spatial Clustering of Applications with Noise (DBSCAN), a well-known density-based clustering algorithm, has gained widespread popularity and usage due to its effectiveness in identifying clusters of arbitrary shapes and handling noisy data. However, it encounters challenges in producing satisfactory cluster results when confronted with datasets of varying density scales, a common scenario in real-world applications. In this paper, we propose a novel **A**daptive and **R**obust **DBSCAN** with Multi-agent Reinforcement Learning cluster framework, namely **AR-DBSCAN**. First, we model the initial dataset as a two-level encoding tree and categorize the data vertices into distinct density partitions according to the *information uncertainty* determined in the encoding tree. Each partition is then assigned to an agent to find the best clustering parameters without manual assistance. The allocation is density-adaptive, enabling AR-DBSCAN to effectively handle diverse density distributions within the dataset by utilizing distinct agents for different partitions. Second, a multi-agent deep reinforcement learning guided automatic parameter searching process is designed. The process of adjusting the parameter search direction by perceiving the clustering environment is modeled as a Markov decision process. Using a weakly-supervised reward training policy network, each agent adaptively learns the optimal clustering parameters by interacting with the clusters. Third, a recursive search mechanism adaptable to the data's scale is presented, enabling efficient and controlled exploration of large parameter spaces. Extensive experiments are conducted on nine artificial datasets and a real-world dataset. The results of offline and online tasks show that AR-DBSCAN not only improves clustering accuracy by up to 144.1% and 175.3% in the Normalized Mutual Information (NMI) and the Adjusted Rand Index (ARI) metrics, respectively, but also is capable of robustly finding dominant parameters.

Index Terms—Density-based clustering, structural entropy, hyperparameter search, reinforcement learning, recursive mechanism

1 Introduction

Clustering serves as a fundamental technology for advancing machine intelligence. It reveals the intrinsic structure of data and aids machine intelligence systems in organizing and comprehending information. Clustering algorithms aim to divide data into multiple clusters based on specific rules. By grouping similar samples (defined by similar distance measurement) into the same cluster and allocating dissimilar samples into different ones, these algorithms reveal the differences between samples and the underlying patterns connecting them. During the last few decades, different types of clustering methods have been proposed [1], such as centroid-based clustering [2], density-based clustering [3], multi-view clustering [4], etc. As a typical density-based clustering method, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [5] defines clusters as the maximal set of density-connected points. It partitions regions of sufficient density into different clusters and identifies clusters with arbitrary shapes in spatial datasets with noise. Due to its simplicity and practicality, DBSCAN has gained widespread recognition and is used in various scientific and engineering fields, such as image segmentation [6], commercial research [7], biological analysis [8], neuroscience [9], target recognition [10], and more [11].

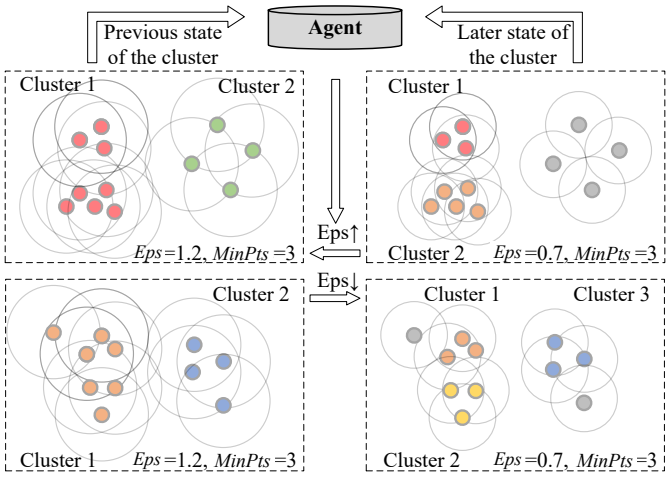


Fig. 1: The challenge of parameter searching through a single agent. Finding the optimal parameter combination of Eps and $MinPts$ in complex situations characterized by disparate density in the data is difficult. The gray vertices depict unclustered noise points.

However, the two predefined parameters of DBSCAN, i.e., the distance of the cluster formation Eps and the minimum data objects required inside the cluster $MinPts$, which typically need to be determined manually, pose four challenges to the clustering process. **First, density variety challenge.** When the data density distribution is not

- *Hao Peng, Xiang Huang, Shuo Sun, and Ruitong Zhang are with Beihang University, Beijing 1000191, China. E-mail: {penghao, huang.xiang, sun.shuo, rtzhang}@buaa.edu.cn;*
- *Xizhao Wang is with Shenzhen University, Shenzhen 518060, China. E-mail: xizhaowang@ieee.org;*
- *Philip S. Yu is with University of Illinois at Chicago, Chicago, IL 60607, USA. E-mail:psyu@uic.edu.*

Manuscript received May 2023, minor revised October 2025. (Corresponding author: Hao Peng)

uniform, and the intra-cluster distance varies significantly, DBSCAN with only one parameter combination can hardly perform well. In the scenario depicted in Figure 1, certain intra-cluster distances are small while others are large, eg., *Cluster 1* and *Cluster 2* in the upper left subfigure of Figure 1. When determining the value of Eps based on a smaller cluster scale, points with slightly larger distances may be classified as noise points incorrectly (the upper half of Figure 1). Conversely, if Eps is set too high, different clusters may be erroneously merged into a single cluster (the bottom half of Figure 1). Some researchers have attempted to address this issue [12], [13] by defining parameters for different groups based on varying densities, but they require introducing specific prior information, such as the total number of clusters or the value of $MinPts$. Therefore, the first challenge lies in devising a method to achieve accurate clustering when clusters have varying densities. **Second, parameters free challenge.** The clustering results are greatly subject to the two parameters Eps and $MinPts$, but they must be determined beforehand based on practical experience. Existing methods [14], [15] based on the k-distance estimate the possible values of the Eps through significant changes in the curve of the k-dist plot, but it still requires manual determination of the $MinPts$ parameter beforehand. Although some improved DBSCAN methods avoid the simultaneous adjustment of Eps and $MinPts$ to varying degrees, they also need to pre-determine the cutoff distance parameter [16], grid parameter [17], Gaussian distribution parameter [18], or fixed $MinPts$ parameter [6], [19]. Therefore, the second challenge is to automatically find optimal cluster parameters for DBSCAN without the need for manual tuning based on expert knowledge. **Third, adaptive policy challenge.** Traditional parameter search methods for DBSCAN, which rely on fixed patterns [14], [15], often face bottlenecks when dealing with unconventional data problems. This is due to the varying data distributions and cluster characteristics in clustering tasks. Even some hyperparameter optimization methods [20]–[22] that utilize an external clustering evaluation index based on label information as the objective function are ineffective when data label information is absent. On the other hand, approaches [23], [24] that only use the internal clustering evaluation index as the objective function are limited by accuracy issues, even though they do not require label information. Therefore, considering the absence of label information, the third challenge is effectively and adaptively adjusting the parameter search policy for the data in each process. **Fourth, search efficiency challenge.** The parameter space expands as the data scale increases. However, an excessively large search space hinders search efficiency [25] and introduces noise that affects clustering accuracy [26]. Thus, the fourth challenge lies in enhancing search efficiency while upholding high clustering accuracy.

In recent years, the theory of structural entropy [27] has demonstrated significant advantages in graph-structured data analysis [28]–[31]. An abstracting partitioning tree, also named *encoding tree*, is generated from a graph by minimizing structural entropy, where each node in the encoding tree has its corresponding node structural entropy. The structural entropy of the whole graph is calculated by summing up all non-root nodes' structural entropy in the encoding tree[1]. Meanwhile, by aggregating the structural entropy of the leaf nodes to the intermediate nodes within the encoding tree, it becomes feasible to quantify the density distribution, as partitions with similar densities exhibit similar structural entropy. On the other hand, the field of Dynamic Algorithm Configuration (DAC) [32]–[34] has emerged as a promising research direction, which often employs reinforcement learning to dynamically control the parameters of an algorithm during its execution. By framing parameter control as a sequential decision-making process, a DAC agent learns an optimal policy for adjusting parameters by continuously interacting with the environment.

To tackle the challenges mentioned above, we propose a novel **A**daptive and **R**obust **DBSCAN** with Multi-agent Reinforcement Learning cluster framework, namely **AR-DBSCAN**. The framework contains two methods: *agent allocation* and *automatic parameter search*, as well as an accelerated *recursive search mechanism*. First, we design a new structural entropy based agent allocation method, which partitions the initial dataset based on the *information uncertainty distribution* of intermediate nodes on the encoding tree and allocates an agent for each partition. A new considering data scale normalization based structured graph construction algorithm is given to convert initial discrete data into a graph, capturing neighborhood information for each data vertice. Unlike the previous one-dimensional structural entropy maximization based k-NN structuralization approach [35], the new algorithm considers mitigating the impact of data scale and the variation in k values of the k-NN graph. Besides, a new information uncertainty metric is devised to measure the density distribution of data partitions associated with the intermediary nodes. Second, we present a new multi-agent deep reinforcement learning based automatic parameter search method to find the optimal parameter combination for each partition. The parameter search process for each agent is transferred into a Markov Decision Process where the cluster changes observed after each clustering step serve as the state, and the parameter adjustment acts as the action. It is important to note that the agents operate in a non-competitive manner, where each is trained independently as a separate single-agent RL task. Through weak supervision, we construct rewards based on a limited number of external clustering indices. Additionally, we employ a new attention mechanism by fusing global and local states of multiple clusters, prompting the agents to attend to the global search environment and allocate varying levels of attention to distinct clusters. Third, a new personalized recursive search mechanism is devised for each agent to gradually reduce the search range and improve the precision of the search. The data scale determines the maximum search parameter boundaries, and the variations in search range and search precision are controllable.

Comprehensive experiments are carried out to evaluate the clustering performance of AR-DBSCAN on nine commonly used clustering datasets and one online task dataset. The results indicate that compared with the state-of-the-art parameter search approaches, AR-DBSCAN exhibits substantial improvements in accuracy, with enhancements of up to 144.1% and 175.3% in NMI and ARI metrics, respectively,

1. Vertices are defined in the graph, and nodes are in the tree.

demonstrating its effectiveness without requiring manual assistance. Moreover, AR-DBSCAN significantly enhances the robustness of clustering, with a reduction in the variance of up to 0.442 and 0.463 in the NMI and ARI metrics, respectively. All codes and datasets of this work are publicly available at GitHub[2].

Our preliminary work appeared at the proceedings of the ACM International Conference on Information and Knowledge Management 2022 [36] and won the Honorable Mention for Best Paper[3]. The journal version in this paper has extended the original parameter search framework into an adaptive and robust parameter search framework with multi-agent reinforcement learning. This full version involves several improvements in upgrading the methodology and the framework structure of the proposed architecture. In terms of model upgrades, different from learning parameters for the whole dataset by one agent, we introduce a multi-agent parameter search framework to learn adaptive parameters for different density partitions to improve the accuracy of clustering in complex datasets with various cluster densities. We integrate structural entropy theory into our framework to provide a view of the density distribution of the data, which guides the agent allocation. A multi-agent recursive mechanism is presented to personalize the adjustment of parameter search space and precision for various density partitions. More state-of-the-art performances are presented and analyzed in the experiments. Comprehensive experiments on ten datasets show the effectiveness and robustness of AR-DBSCAN.

In summary, the contributions of this paper are summarized as follows:

- A new parameter search framework guided by multi-agent deep reinforcement learning, named AR-DBSCAN, is presented. AR-DBSCAN achieves adaptive and robust clustering on datasets with varying densities, eliminating the requirement for pre-determining DBSCAN parameters.
- A novel agent assignment method, based on structural entropy, is designed to assign agents by considering the information uncertainty distribution of intermediate nodes in the encoding tree. The newly proposed information uncertainty metric provides a new perspective for quantifying the density distribution of data partitions.
- A new non-competitive multi-agent parameter search framework is devised to adaptively explore the optimal combination of parameters for partitions. This framework utilizes weak supervision and attention mechanism capturing both global information and local information within clusters.
- A new personalized recursive parameter search mechanism is introduced, enabling efficient and controlled search for all agents.
- Comprehensive experiments and analyses are conducted on ten datasets, demonstrating that AR-DBSCAN outperforms the existing state-of-the-art approaches in terms of accuracy, efficiency, and robustness.

This paper is organized as follows: Sec. 2 outlines the problem formulation and notation descriptions, Sec. 3 describes the detailed designs of AR-DBSCAN, Sec. 4 showcases the datasets and experiment details, Sec. 5 gives the results and analyses, and Sec. 6 introduces the related work before our conclusion in Sec. 7.

2. https://github.com/RingBDStack/AR-DBSCAN
3. https://www.cikm2022.org/best-paper

TABLE 1: Forms and interpretations of notations.

Notation	Description
C; c_n	The cluster set; The n-th cluster
$\mathcal{P}$; $\mathcal{P}_i$	The data partition; The i-th data partiton
Eps	The distance of the cluster
$MinPts$	The minimum data required inside the cluster
P	The parameter combination for DBSCAN
G;V;E	The graph; The vertices set; The edges set
v;d_v	The data vertice;The degree of v
$e_{i,j}$	The edge between v_i and v_j
W;$w_{i,j}$	The edge weight set; The weight of $e_{i,j}$
$vol(G)$	The volume of graph G, i.e., degree sum in G
T	The encoding tree
λ;α	The root node of T ;The node of T
α^-;$\alpha^{\langle j \rangle}$	The parent node of α; The j-th child of α
N_α	The number of immediate successor nodes of α
T_α	The vertices set corresponding to node α
$H^1(G)$	The one-dimensional structural entropy of G
$H^1_{norm.}(G)$	The normalized one-dim. structural entropy
$H^K(G)$	The K-dimensional structural entropy of G
$H^T(G)$	The structural entropy of G under T
$H^T(G;\alpha)$	The structural entropy of node α
$\mathcal{H}^T(G;\alpha)$	The normalized structural entropy of node α
D_{ij}	The distance of vertice v_i and v_j
$\|\cdot\|$	The L2 norm function.
$M(T;\alpha,\gamma)$;$C(T;\alpha,\gamma)$	The merging operator; The combining operator
$\mathcal{S}$; s	The state set; The state for one step
$\mathcal{A}$; a	The action space; the action for one step
$\mathcal{R}$; r	The reward function; the reward for one step
$\mathbf{P}$	The policy optimization algorithm
$\mathcal{V}$; $\mathcal{V}_i$	The data block; The i-th data block
$\|\cdot\|$; $\cdot'$	The size of $\cdot$;The partial set of $\cdot$
e	The episode of parameter search process
I	The maximum step limit in an episode
$\cdot^{(e)(i)}$	The value of $\cdot$ in the i-th step of episode e
s_{global};$s_{local,n}$	The global state; The local state of cluster c_n
$\mathcal{D}_b$	The quaternary distances of P
R_{cn}	The ratio of $\|\mathcal{C}\|$ and $\mathcal{V}$
$\mathcal{X}$; $\mathcal{Y}$	The feature set; The true label set
$\cdot_{center,n}$	The value of $\cdot$ for the center object of c_n
F_G;F_L	The FCN for s_{global}; The FCN for $s_{local,n}$
F_S	The scoring FCN
$\alpha_{att,n}$	The attention weight for cluster c_n
$\sigma(\cdot)$	The `ReLU` activation function
$\|$	The splicing operation
β; δ	The reward impact factors
$\mathcal{T}$	The core element of one step
M	The sample number for $\mathcal{T}$
$\mathcal{L}_c$; $\mathcal{L}_a$	The loss for $Critic$; The loss for $Actor$
d	The dimension of feature
P_o, Eps_o, $MinPts_o$	The optimal parameter
$\cdot^{(l)}$	The value of $\cdot$ in layer l
$B_{p,1}$	The minimum boundaries for parameter p
$B_{p,2}$	The maximum boundaries for parameter p
π_p	The number of valid parameter
θ_p	The parameter search step
$\lfloor\ \rfloor$	The rounding down operation

2 Problem Formulation and Notation

In this section, we give the formula definitions of DBSCAN clustering, parameter search of DBSCAN, and corresponding concepts in the structural entropy theory. The forms and description of all necessary notations throughout our work are stated in Table 1.

Defination 2.1. (DBSCAN clustering). Given data vertices $V = \{v_1, ..., v_j, v_{j+1}, ...\}$, DBSCAN aims to find clusters

$C = \{c_1, ..., c_n, c_{n+1}, ...\}$ based on the parameter combination $P = \{Eps, MinPts\}$. Here, Eps means the maximum distance that two objects can be formed in a cluster, and $MinPts$ measures the minimum number of adjacent objects where a core object can reach within Eps distance. Concretely, DBSCAN first selects a random point having at least $MinPts$ points within Eps distance. Then, each point within the neighborhood of the core point is evaluated to see if it has $MinPts$ points within Eps distance (including the point itself). If the point satisfies the $MinPts$ criterion, it becomes another core point, and the cluster expands. If a point does not satisfy the $MinPts$ criterion, it becomes a boundary point. And a point is classified as noise if it does not meet the criteria to be considered a core point or a boundary point [5].

Defination 2.2. (Parameter search of DBSCAN). Given the partition set $\{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_t, \ldots\}$, for the t-th partition $\mathcal{P}_t = \{v_1^t, \ldots, v_i^t, v_j^t, \ldots\}$, the parameter search of DBSCAN clustering is the process of finding the optimal parameter combination $P_t = \{Eps_t, MinPts_t\}$ in corresponding parameter spaces.

Defination 2.3. (Parameter search of DBSCAN in the online task). Given the data blocks $\{\mathcal{V}_1, \ldots, \mathcal{V}_i, \mathcal{V}_{i+1}, \ldots\}$ as the online data stream, the parameter search in online clustering is the process of finding the optimal parameter combinations for each data block $\mathcal{V}_i$.

Defination 2.4. (Structured Graph). Let $G = \{V, E, W\}$ denote as a weighted undirected graph, where V is the vertices set, E is the edge set and $W \in \mathbb{R}^+$ is the edge weight set. The degree of vertice v, denoted as d_v, is defined as the sum of associated edge weights.

Defination 2.5. (Encoding Tree). Given a graph $G = \{V, E, W\}$, the encoding tree T of G is defined with the following properties: **(1)** For the root node denoted λ, we define the vertices set it associated $T_\lambda = V$. **(2)** For every node $\alpha \in T$, the immediate successors of α are $\alpha^{\langle j \rangle}$ ordered from left to right as j increases. It is required that every internal node has at least two immediate successors. **(3)** For each non-root node α, its parent node in T is denoted as α^-. **(4)** For node $\alpha \in T$, the vertex subsets $T_\alpha = \bigcup_{i=1}^{N_\alpha} T_{\alpha^{\langle j \rangle}}$, N_α is number of successors of α. We call an encoding tree the K-level encoding tree if its height is K.

Defination 2.6. (One-dimensional Structural Entropy). In a single-level encoding tree T, the one-dimensional structural entropy is defined as follows:

$$H^1(G) = -\sum_{v \in V} \frac{d_v}{vol(G)} \cdot \log_2 \frac{d_v}{vol(G)}, \tag{1}$$

where d_v is the degree of vertex v, and $vol(G)$ is the sum of the degrees of all vertices in G. As noted in previous research [27], the one-dimensional structure represents the maximum level of information embedding in graph G.

Defination 2.7. (High-dimensional Structural Entropy). For the encoding tree T, the high-dimensional structural entropy of G is defined as follows:

$$H^K(G) = \min_{\forall T: height(T) \leq K} H^T(G), \tag{2}$$

$$H^T(G) = \sum_{\alpha \in T, \alpha \neq \lambda} H^T(G; \alpha). \tag{3}$$

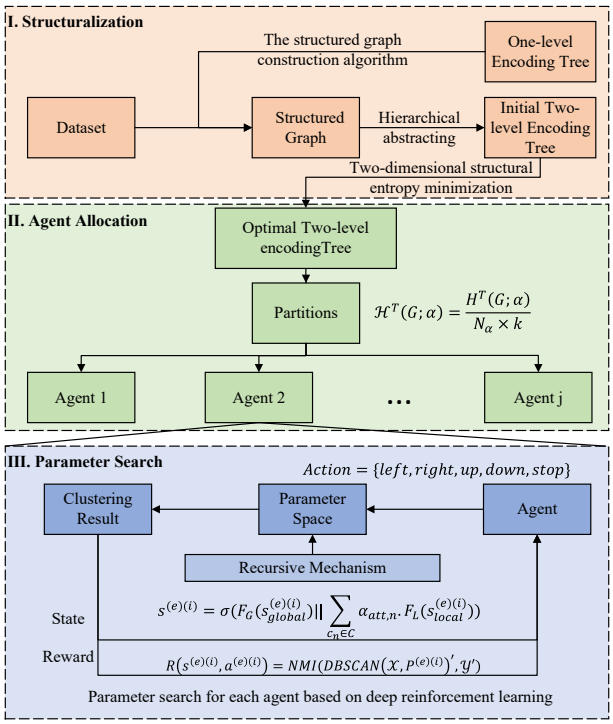


Fig. 2: Overall framework of AR-DBSCAN.

Here $H^T(G;\alpha)$ is the structural entropy of node α, and $H^T(G)$ is the structural entropy of encoding tree T. $H^K(G)$ is the K-dimensional structural entropy, with the optimal encoding tree of K-level.

Defination 2.8. (Structual Entropy of tree node α). Given an encoding tree T, and non-root node α in T, the structural entropy of node α is defined as follows:

$$H^T(G;\alpha) = -\frac{g_\alpha}{vol(G)} \log_2 \frac{V_\alpha}{V_{\alpha^-}}. \tag{4}$$

Here g_α represents the total weight of the cut edges linking the vertices in T_α with those in the complement set T_λ / T_α, where the latter comprises all vertices that are outside of T_α. V_α is the sum of degrees of all vertices in T_α.

3 AR-DBSCAN FRAMEWORK

As illustrated in Fig. 2, the details of AR-DBSCAN consist of *Structuralization*, *Agent allocation*, and *Parameter search*. (1) *Structuralization*. To begin with, the initial dataset is converted into a structural k-NN graph G, which captures the neighborhood structure information (Sec. 3.1.1). (2) *Agent allocation*. Following the structuralization step, the optimal two-level encoding tree is generated from the graph G based on two-dimensional structural entropy minimization (Sec. 3.1.2). The agent assignment method is then employed to partition the initial dataset based on information uncertainty distribution on the encoding tree. Each partition is assigned a corresponding agent (Sec. 3.1.3). (3) *Parameter search*. For each agent and corresponding data partitions, the separated parameter search process based on deep reinforcement learning is applied to find the optimal parameter combination respectively (Sec. 3.2.1). The recursive mechanism is introduced to improve search efficiency in Sec. 3.2.2.

Additionally, we design four working modes for online tasks in Sec. 3.3. The time complexity of AR-DBSCAN is given in Sec. 3.4.

3.1 Structural entropy based agent allocation

When dealing with clusters of varying densities, existing parameter search policies, such as those proposed in references [14], [36], [37], which rely on a single parameter combination, may fail to accurately recognize them (as discussed in Sec. 5.1.1 and Sec. 5.2). To tackle the collision of various density clusters in parameter search, a new structural entropy based agent allocation method is presented in Figure 3.

3.1.1 Structured graph construction

We transfer the initial dataset to a structured graph $G = \{V, E, W\}$, where V is the set of data points and E is determined through a k-nearest neighbors (k-NN) approach employing Euclidean distance. The Euclidean distance is defined as follows:

$$D_{ij} = \|r_i - r_j\|_2, \tag{5}$$

where r_i and r_j are the coordinates or feature vectors of two vertices v_i and v_j, and $\|\cdot\|_2$ is the L2 norm. The edge weight W, also referring to the similarity, is based on the Euclidean distance with normalization on the number of edges and the sum of distances. Specifically, the edge weight between vertices v_i and v_j is calculated as follows:

$$w_{i,j} = \exp(-D_{ij} \cdot \frac{|E|}{\sum_{e_{m,n} \in E} D_{m,n}}). \tag{6}$$

Here $e_{m,n}$ is the edge in the edge set E, $D_{m,n}$ is the Euclidean distance between vertices v_m and v_n, and $|E|$ is the size of the edge set. The edge weight (similarity) is inversely proportional to the Euclidean distance. Intuitively, the similarity increases as the distance between vertices decreases, leading to a larger edge weight.

Selecting the core parameter k in k-NN is a critical challenge. A larger k may create excessive edges and make G over-noisy. Conversely, a smaller k value may fail to capture the clusters' actual neighborhood structure. Different from the previous k-select algorithm based on one-dimensional structural entropy maximization [27], we consider the impact of data scale and normalize one-dimensional structural entropy as follows:

$$H^1_{norm.}(G) = \frac{H^1(G)}{k \times |V|}, \tag{7}$$

where $|V|$ is the size of V. $H^1_{norm.}(G)$ mitigates the impact of data scale and k, ensuring a stable search for the optimal value of k, as demonstrated in Sec 5.3.1. Then, we iterate over k from 1 to $|V|$, and construct corresponding k-NN graph G_k as well as calculating $H_{norm^1}(G_k)$. The optimal k is determined by minimum $H^1_{norm.}(G)$ in the $stable\_points$ that satisfying $H^1_{norm.}(G_k) < H^1_{norm.}(G_{k-1})$ and $H^1_{norm.}(G_k) < H^1_{norm.}(G_{k+1})$. The detailed process is depicted in Algorithm 1. Finally, we obtain the optimal k-NN graph G_k as the structured graph G.

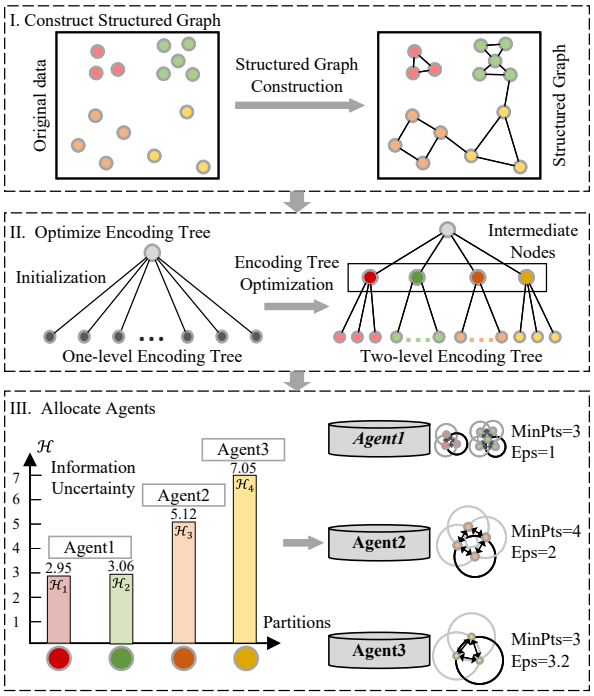


Fig. 3: Structural entropy based agent allocation.

Algorithm 1: The structured graph construction algorithm

Input: Data vertices set V
Output: Structured graph G

$stable\_point \leftarrow$ empty array list [];
for $k = 1, 2, 3, \cdots, |V|$ **do**
  $G_k \leftarrow$ construct initial k-NN graph via Eq. 6;
  $H^1(G_k) \leftarrow$ calculate the one-dimensional structural entropy via Eq. 1;
  $H^1_{norm.}(G)[k] \leftarrow$ correct $H^1(G_k)$ via Eq. 7;
  if $H^1_{norm.}(G)[k-1] < H^1_{norm.}(G)[k-2]$ *and* $H^1_{norm.}(G)[k-1] < H^1_{norm.}(G)[k]$ **then**
    $stable\_point$ append $k-1$;
for $k \in stable\_points$ **do**
  $k^* \leftarrow$ select k corresponding the minimum $H^1_{norm.}(G)[k]$ in stable point;
$G \leftarrow$ the k^*-NN graph G_{k*};
Return G;

3.1.2 Two-level encoding tree optimization

To attain an optimal hierarchical abstracting of the dataset and uncover its underlying density distribution, we optimize the two-level encoding tree of the graph G by minimizing the two-dimensional structural entropy within the structural graph G. The optimal two-level encoding tree is represented as follows:

$$T = \underset{\forall T: height(T) \leq 2}{argmin} H^T(G), \tag{8}$$

where $H^T(G)$ is the structural entropy defined in Eq. 3. We recursively employ two optimization operators introduced from deDoc [28], merging and combining, to optimize the

encoding tree based on the greedy policy. The merging operator $M(T;\alpha,\gamma)$ combines two subtrees under the same parent node into one subtree, while the combining operator $C(T;\alpha,\gamma)$ constructs a common new parent node for two subtrees. If the structural entropy $H^T(G)$ of the encoding tree T decreases after executing $M(T;\alpha,\gamma)$ or $C(T;\alpha,\gamma)$, we consider the operator to have executed successfully and denote as $M(T;\alpha,\gamma)\downarrow$ or $C(T;\alpha,\gamma)\downarrow$. The optimizing process is as follows.

1) Given encoding tree T, set λ as the root node, α and γ as the non-root nodes.
2) If there exist α and $\gamma \in T$ that satisfy $M(T;\alpha,\gamma)\downarrow$, execute $M(T;\alpha,\gamma)$ and obtain the updated encoding tree $T_M(\alpha,\gamma)$, then return to (2).
3) If there exist α and $\gamma \in T$ that satisfy $C(T;\alpha,\gamma)\downarrow$, execute $C(T;\alpha,\gamma)$ and obtain the updated encoding tree $T_C(\alpha,\gamma)$, then return to (2).
4) If there are no α and $\gamma \in T$ that satisfy $M(T;\alpha,\gamma)\downarrow$ or $C(T;\alpha,\gamma)\downarrow$, output the optimal encoding tree T.

Finally, the optimal two-level encoding tree T is generated. Notably, T also can be directly utilized for clustering purposes, and its effectiveness is investigated in Sec. 5.3.2.

3.1.3 Agent assignment

To address the challenge of varying densities within the dataset, the main solution is to partition the initial data based on its density distribution and perform a separate agent parameter search for each partition. Utilizing the optimal encoding tree T of the structural graph G, we evaluate the structural entropy of intermediate nodes using Eq. 4, which provides new insights into the density differences inherent in the data. Since the structural entropy of an intermediate node α is directly proportional to the number of its child nodes, we normalize it and give the formulation of information uncertainty as follows:

$$\mathcal{H}^T(G;\alpha) = \frac{H^T(G;\alpha)}{N_\alpha \times k}, \tag{9}$$

where N_α is the number of immediate successor nodes of α and k is selected in Sec. 3.1.1. The normalized information uncertainty $\mathcal{H}^T$ displays a similar scale of values across various datasets, which is also demonstrated in Sec. 5.3.3. Hence, we simply employ the DBSCAN algorithm to cluster the intermediate nodes with similar information uncertainty into the same partition. Specifically, as the scenario depicted in Figure 3 *Allocate agents*, the intermediate nodes with information uncertainty $\mathcal{H}_1$ and $\mathcal{H}_2$ are partitioned together as their information uncertainty difference is small, and the other two intermediate nodes are partitioned separately. This way, the leaf nodes in the encoding tree (i.e., the initial data points) are separated into different partitions. For each partition, we allocate a dedicated agent and conduct a parameter search specific to that agent.

3.2 Parameter search for each agent

Up to this point, we have successfully determined the number of agents and partitioned the corresponding data for them. In this subsection, we present the fundamental Markov decision process for parameter exploration and the recursive mechanism for navigating the parameter space, as illustrated in Figure 4.

3.2.1 Parameter search with DRL

The fixed DBSCAN parameter search policy is no longer suitable for various cluster tasks. As a solution, we proposed DRL-DBSCAN [36], a novel framework for automatic parameter search based on deep reinforcement learning (DRL). The core model of DRL-DBSCAN is formulated as a Markov Decision Process (MDP) with state set, action space, reward function, and policy optimization algorithm $(\mathcal{S},\mathcal{A},\mathcal{R},\mathbf{P})$ [38].

This process converts the DBSCAN parameter search into a maze game problem [39], [40] in the parameter space, with the objective of training an agent to iteratively navigate from the start point parameters to the endpoint parameters by interacting with the environment. The endpoint parameters obtained are considered the final search result of the game. The agent perceives the parameter space and DBSCAN clustering algorithm as the environment, with the search position and clustering result representing the state and the parameter adjustment direction serving as the action. To encourage exceptional behavior, a small number of samples are used to provide weakly supervised rewards to the agent. We optimize the agent's policy using the Actor-Critic [41] architecture. Specifically, we formulate the search process for episode $e(e = 1, 2, \dots)$ on the corresponding partition $\mathcal{P}$ of the agent as follows:

• ***State:*** To achieve an accurate and comprehensive state representation of the search environment, we construct it from global and local perspectives.

Firstly, we describe each agent's overall search and clustering situation by utilizing a 7-tuple to depict the global state of the i-th step $(i = 1, 2, \dots)$:

$$s_{global}^{(e)(i)} = P^{(e)(i)} \cup \mathcal{D}_b^{(e)(i)} \cup \{R_{cn}^{(e)(i)}\}. \tag{10}$$

Here, $P^{(e)(i)} = \{Eps^{(e)(i)}, MinPts^{(e)(i)}\}$ refers to the current DBSCAN parameter. $\mathcal{D}_b^{(e)(i)}$ represents a collection of quaternary distances, comprising the distance between $Eps^{(e)(i)}$ and its corresponding space boundaries $B_{Eps,1}$ and $B_{Eps,2}$, as well as the distance between $MinPts^{(e)(i)}$ and its corresponding boundaries $B_{MinPts,1}$ and $B_{MinPts,2}$ (the specific parameter boundaries are defined in Sec. 3.2.2). The ratio $R_{cn}^{(e)(i)} = \frac{|\mathcal{C}^{(e)(i)}|}{|\mathcal{P}|}$ corresponds to the number of clusters $|\mathcal{C}^{(e)(i)}|$ relative to the overall number of objects in the data partition $|\mathcal{P}|$. Secondly, to depict the state of each cluster, we define the $\{d+2\}$-tuple of the i-th local state of cluster $c_n \in \mathcal{C}$ as follows:

$$s_{local,n}^{(e)(i)} = \mathcal{X}_{center,n}^{(e)(i)} \cup \{D_{center,n}^{(e)(i)}, |c_n^{(e)(i)}|\}, \tag{11}$$

where $\mathcal{X}_{center,n}^{(e)(i)}$ represents the feature of the central object of c_n and d is its feature dimension, $D_{center,n}^{(e)(i)}$ denotes the Euclidean distance between the cluster center object and the central object of the entire data partition, and $|c_n^{(e)(i)}|$ means the number of objects contained in cluster $c_n^{(e)(i)}$. Thirdly, taking into account the variation in the number of clusters during different steps of the parameter search process, we use the Attention Mechanism [42] to consolidate the global state and multiple local states into a state representation of fixed length:

$$s^{(e)(i)} = \sigma\Big(F_G(s_{global}^{(e)(i)}) \parallel \sum_{c_n \in \mathcal{C}} \alpha_{att,n} \cdot F_L(s_{local,n}^{(e)(i)})\Big), \tag{12}$$

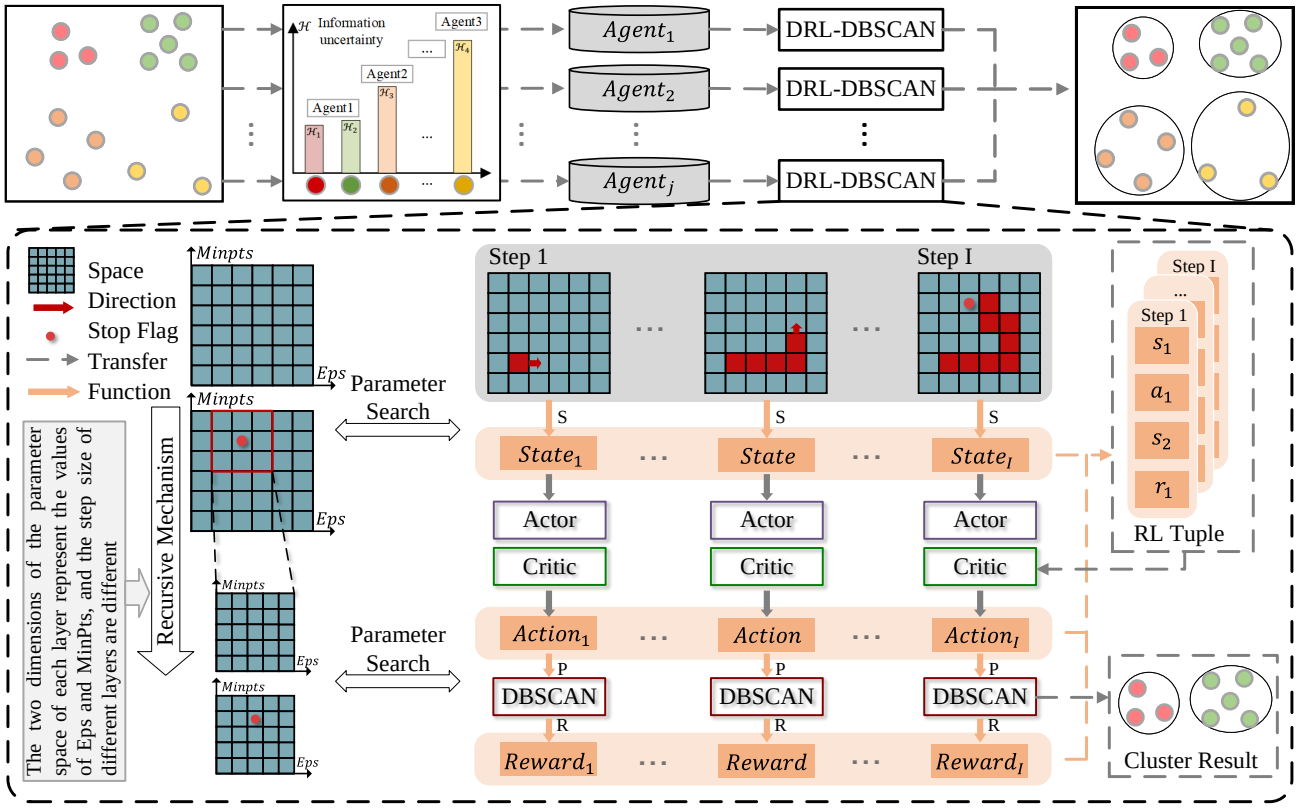


Fig. 4: Multi-agent deep reinforcement learning based automatic parameter search.

where F_G and F_L refer to the Fully-Connected Network (FCN) for the global state and local state, respectively. σ is the ReLU activation function, while $\|$ indicates the splicing operation. $\alpha_{att,n}$ denotes the attention weight assigned to cluster c_n, which is formulated as follows:

$$\alpha_{att,n} = \frac{\sigma\Big(F_S\big(F_G(s^{(e)(i)}_{global}) \parallel F_L(s^{(e)(i)}_{local,n})\big)\Big)}{\sum_{c_n \in \mathcal{C}} \sigma\Big(F_S\big(F_G(s^{(e)(i)}_{global}) \parallel F_L(s^{(e)(i)}_{local,n})\big)\Big)}. \quad (13)$$

We merge the global state with the local state of each cluster separately, feed it into a fully connected network F_S for scoring, and utilize the normalized score of each cluster as its attention weight. This method establishes attention toward the global search environment while representing local clusters. Additionally, it assigns distinct weights to different types of cluster information during the final state representation, thus augmenting the influence of significant clusters on the state.

- ***Action:*** The action $a^{(e)(i)}$ for the episode e and i-th step corresponds to the parameter search direction. To be precise, the action space $\mathcal{A}$ is defined as $\{left, right, down, up, stop\}$, where $left$ and $right$ refer to reducing or increasing the Eps parameter, $down$ and up correspond to reducing or increasing the $MinPts$ parameter, and $stop$ indicates the termination of the search process. To determine the action $a^{(e)(i)}$ based on the current state $s^{(e)(i)}$, we develop an Actor network as the policy network, which is described as follows:

$$a^{(e)(i)} = Actor(s^{(e)(i)}), \quad (14)$$

where the $Actor$ is a three-layer Multi-Layer Perceptron (MLP). The conversion process of an action from the i-th step to the $i+1$-th step is defined as follows:

$$P^{(e)(i)} \xrightarrow{a^{(e)(i)},\theta} P^{(e)(i+1)}. \quad (15)$$

$P^{(e)(i)}$ and $P^{(e)(i+1)}$ represent the parameter combinations of $Eps^{(e)(i)}, MinPts^{(e)(i)}$ and $Eps^{(e)(i+1)}, MinPts^{(e)(i+1)}$ at the i-th step and the $i+1$-th step, respectively. θ denotes the parameter adjustment step size, which will be discussed in detail in Sec. 3.2.2. It should be noted that if an action causes a parameter to go beyond its boundary, the parameter is set to the boundary value, and the corresponding boundary distance is set to -1 in the next step.

- ***Reward:*** Rewards play a crucial role in incentivizing the agent to improve its parameter search policy. Thus, we use a small number of external metric samples as the basis for providing rewards to the agent. The immediate reward function for the i-th step is defined as follows:

$$\mathcal{R}(s^{(e)(i)}, a^{(e)(i)}) = \mathrm{NMI}\big(\mathrm{DBSCAN}(\mathcal{X}, P^{(e)(i+1)})', \mathcal{Y}'\big), \quad (16)$$

where $\mathrm{NMI}(,)$ represents the external metric function, Normalized Mutual Information (NMI) [43] of DBSCAN clustering. $\mathcal{X}$ denotes the feature set, and $\mathrm{DBSCAN}(\mathcal{X}, P^{(e)(i+1)})'$ denotes the predicted labels for the partially labeled data. $\mathcal{Y}'$ refers to a set of partial true labels for the data partition. It is important to note that the labels are used solely for training purposes and during testing, the trained agent is utilized to find optimal parameters on the remaining unlabeled data.

Considering the optimal parameter search action sequence for one episode involves adjusting the parameters towards the optimal parameters and stopping the search at the optimal parameters, we propose using both the maximum immediate reward for subsequent steps and the endpoint immediate reward as the reward for the i-th step:

$$\begin{aligned} r^{(e)(i)} =& \beta \cdot \max\left\{\mathcal{R}(s^{(e)(m)}, a^{(e)(m)})\right\}|_{m=i}^{I} \\ &+ \delta \cdot \mathcal{R}(s^{(e)(I)}, a^{(e)(I)}), \end{aligned} \tag{17}$$

where $\mathcal{R}(s^{(e)(I)}, a^{(e)(I)})$ is the immediate reward for the I-th step end point of episode e. $\max$ computes the future maximum immediate reward prior to stopping the search in the current episode e. β and δ denote the reward impact factors, where $\beta = 1 - \delta$.

• ***Termination:*** To enhance the efficiency of the search process, we design three termination conditions for a complete episode search process, including parameter search going beyond the boundaries, the search step exceeding the maximum step limit in an episode, and the action turning to *stop*. The formulation for these conditions is as follows:

$$\begin{cases} \min(\mathcal{D}_b^{(e)(i)}) < 0, & \text{Out of bounds stop,} \\ i >= I_{max}, & \text{Timeout stop,} \\ a^{(e)(i)} = stop, where\ i \geq 2, & \text{Action stop,} \end{cases} \tag{18}$$

where I_{max} is the maximum step limit in an episode.

• ***Optimization:*** To enhance the performance of the policy network *Actor*, it is necessary to extract specific components from the network and enable the $Critc$ network to evaluate and adjust the *Actor* network accordingly. The parameter search process for episode e is described as follows: 1) Observe the current state $s^{(e)(i)}$ of DBSCAN clustering. 2) Predict the action of the parameter adjustment direction, $a^{(e)(i)}$, based on $s^{(e)(i)}$ through the *Actor* network. 3) Obtain the new state, $s^{(e)(i+1)}$. 4) Repeat the above steps until the end of the episode, and get a reward $r^{(e)(i)}$ for each step. Therefore, the core element of the i-th step is extracted as a *RL Tuple*:

$$\mathcal{T}^{(e)(i)} = (s^{(e)(i)}, a^{(e)(i)}, s^{(e)(i+1)}, r^{(e)(i)}). \tag{19}$$

We store $\mathcal{T}$ of each step in the memory buffer and sample M core elements to optimize the policy network *Actor*. Specifically, we define a three-layer MLP as the $Critic$ to learn the action value of the state [41] and use the $Critic$ network to optimize the *Actor*. The loss function for *Actor* and $Critic$ is defined as follows:

$$\begin{aligned} \mathcal{L}_c =& \sum_{\mathcal{T} \in buffer}^{M} \big(r^{(e)(i)} + \gamma \cdot Critic(s^{(e)(i+1)}, a^{(e)(i+1)}) \\ &- Critic(s^{(e)(i)}, a^{(e)(i)})\big)^2, \end{aligned} \tag{20}$$

$$\mathcal{L}_a = -\frac{\sum_{\mathcal{T} \in buffer}^{M} Critic\big(s^{(e)(i)}, Actor(s^{(e)(i)})\big)}{M}. \tag{21}$$

Here, γ denotes the reward decay factor. It should be noted that we adopt the Twin Delayed Deep Deterministic Policy Gradient algorithm (TD3) [44] for policy optimization in our framework, which is substituted with other Deep Reinforcement Learning policy optimization algorithms [41], [45].

3.2.2 Parameter Space and Recursion Mechanism

Here, we define the parameter space of the agent proposed in Sec. 3.2.1 and the recursive search mechanism that starts with a broad search space with coarse-grained precision and gradually narrows it down to a smaller search space with fine-grained precision. Firstly, to address the fluctuation of the parameter range caused by different data distributions, we normalize the data features by transforming the maximum search range of the Eps parameter into the $(0, \sqrt{d}]$ range, where d is the dimension of the data features. As $MinPts$ must be an integer greater than 0, we propose to define a coarse-grained maximum search range for $MinPts$ based on the size of the dataset. Secondly, in order to balance search efficiency and search precision, we propose a recursive mechanism for progressive search that narrows down the search range and increases the search precision layer by layer. Once the agent in the previous layer reaches termination, we replace it with a new agent in the next layer and search for the optimal parameter combination $P_o^{(l)} = \{Eps_o^{(l)}, MinPts_o^{(l)}\}$ according to the search precision and range requirements of the corresponding layer.

The minimum and maximum search boundaries $B_{p,1}^{(l)}$ and $B_{p,2}^{(l)}$ for parameter $p \in \{Eps, MinPts\}$ in layer l $(l = 0, 1, 2, \cdots)$ are defined as follows:

$$\begin{aligned} &B_{p,1}^{(0)} : B_{Eps,1}^{0} = 0, B_{MinPts,1}^{0} = 1, \\ &B_{p,2}^{(0)} : B_{Eps,2} = \sqrt{d}, B_{MinPts,2} = |\mathcal{P}|, \\ &B_{p,1}^{(l)} : \max\left\{B_{p,1}^{(0)},\ p_o^{(l-1)} - \frac{\pi_p}{2} \cdot \theta_p^{(l)}\right\}, \\ &B_{p,2}^{(l)} : \min\left\{p_o^{(l-1)} + \frac{\pi_p}{2} \cdot \theta_p^{(l)}, B_{p,2}^{(0)}\right\}, \end{aligned} \tag{22}$$

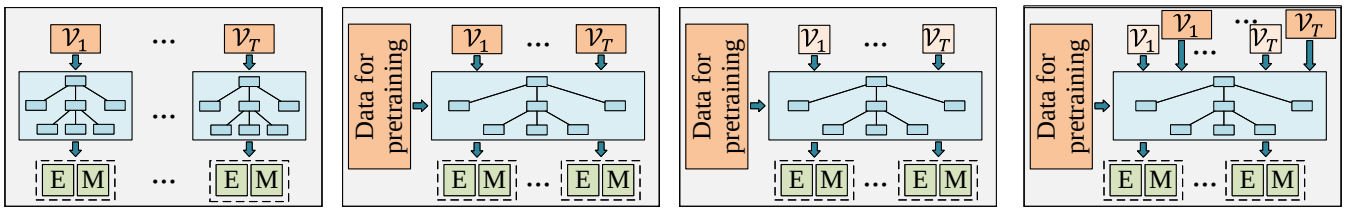
where d represents the dimensionality of the data features, $|\mathcal{P}|$ is the size of the data partition allocated to the agent in Sec. 3.1, $B_{p,1}^{(0)}$ and $B_{p,2}^{(0)}$ are the parameter boundaries of the 0-th layer that determine the maximum range for parameter search. π_p represents the number of parameters that can be searched in the parameter space of parameter p in each layer. $p_o^{(l-1)} \in P_o^{(l-1)}$ denotes the optimal parameter searched by the previous layer. For the 0-th layer, we define $p_o^{(0)} \in P_o^{(0)}$ as the midpoint between $B_{p,1}^{(0)}$ and $B_{p,2}^{(0)}$. Moreover, $\theta_p^{(l)}$ is the search step size that controls the search precision of parameter p in the l-th layer and is defined as follows:

$$\theta_p^{(l)} = \begin{cases} \frac{\theta_p^{(l-1)}}{\pi_p}, & if\ p\ is\ Eps; \\ \max\left\{\lfloor \frac{\theta_p^{(l-1)}}{\pi_p} + \frac{1}{2} \rfloor,\ 1\right\}, & otherwise. \end{cases} \tag{23}$$

Here, θ_p^{l-1} represents the search step size of the previous layer, while θ_p^0 is a manually defined value. The symbol $\lfloor\ \rfloor$ denotes rounding down to the nearest integer.

3.3 Working modes for online tasks

In order to improve the versatility of AR-DBSCAN in different online scenarios, we define four working modes as shown in Figure 5. Their corresponding definitions are as follows: **(1)Retraining Mode**. The optimal parameters are searched based on the training process for each agent, and agents are reinitialized when the dataset changes. **(2)Continuous Training Mode**. The agents are pre-trained in advance, and when the dataset changes, the parameter search



(a) Retraining Mode (b) Continuous Training Mode (c) Pretraining Testing Mode (d) Maintenance Testing Mode

Fig. 5: Four working modes for the online task. Dark orange square means partially labeled data block, and light orange square means unlabeled data block.

process is continued based on the training process using the already trained agents. **(3)Pretraining Testing Mode**. The agents are pre-trained in advance. When the dataset changes, the parameter search process for each agent is executed directly based on the testing process without labels. **(4)Maintenance Testing Mode**. Similarly to the pretraining testing mode, the agents are pre-trained in advance, and the parameter search process is executed directly based on the testing process without labels. The main difference with the pretraining testing mode is that following the pre-training phase, regular maintenance training is conducted using labeled historical data. Due to the limitation of structural entropy theory dynamic composition, the single-agent-based parameters search process can fit all four working modes, while multi-agent-based only fit the retraining mode. This is because the agent configuration is dynamically derived from the current data distribution and is not transferable to subsequent datasets.

3.4 Time complexity of AR-DBSCAN

The time complexity of agent allocation is $O(n^2 + n\log^2 n)$, where n is the number of nodes. Separately, the construction of the structured graph has time complexity $O(n^2)$, while optimizing the two-level encoding tree has time complexity $O(n\log^2 n)$. CoDeSEG [46] proposes a game-theoretic framework that reduces the latter to near-linear time complexity $O(n)$ in large-scale networks. Regarding the time complexity of the parameter search, according to the recursive mechanism, the time complexity of the parameter search for each agent is $O(\log N)$, where N denotes the size of the parameter space $\theta_p^{(0)}/\theta_p^L = (\pi_p)^L$. Consequently, the total time complexity of AR-DBSCAN is $O(n^2 + n\log^2 n) + O(\log N)$.

4 Experimental Setup

4.1 Hardware and Software

All experiments are performed on a Linux server consisting of an 18-core Intel i9-10980XE CPU, 256GB of RAM, and an NVIDIA RTX A6000 GPU. We implement all models using Pytorch 2.0 and Python 3.10. As for baselines, we utilize open-source implementations from the library Hyperopt[4] and Scikit-opt[5], or codes provided by the authors.

4. http://github.com/hyperopt/hyperopt
5. https://github.com/guofei9987/scikit-opt

4.2 Baselines

We compare AR-DBSCAN to traditional hyperparameter search schemes, meta-heuristic optimization algorithms, and existing DBSCAN parameter search methods. The baselines are listed below.

Rand [37]. The Rand algorithm finds better models by effectively searching a larger, less promising configuration space. It searches over the same domain to find models that are as good or better within a small fraction of the computation time.

BO-TPE [47]. The BO-TPE algorithm uses random search and two new greedy sequential methods based on the expected improvement criterion to optimize hyperparameters.

Anneal [48]. The Anneal introduces the idea of annealing into the field of combinatorial optimization. It is a stochastic optimization algorithm based on the Monte-Carlo iterative solution strategy, which is based on the similarity between the annealing process of solid matter in physics and general combinatorial optimization problems.

PSO [49]. The PSO algorithm is a swarm optimization algorithm in which each member continuously changes its search pattern by learning from its own experience and the experience of other members.

GA [22]. The GA algorithm combines the genetic algorithm and support vector machine method. Support vector machines are used to solve classification tasks, while genetic algorithms are optimization heuristics that combine direct and random searches within a solution space.

DE [50]. The Differential Evolution (DE) algorithm is based on evolutionary ideas such as genetic algorithms. It is essentially a multi-objective (continuous variable) optimization algorithm for solving the overall optimal solution in multidimensional space.

KDist(VDBSCAN) [14]. The VDBSCAN algorithm selects several values of parameter Eps for different densities according to a k-dist plot before adopting the traditional DBSCAN algorithm. And clusters with varied densities simultaneity are found with different values of Eps.

BDE [21]. The BDE-DBSCAN adopts the binary coding scheme and represents each individual as a bit string. It leverages the Binary Differential Evolution and Tournament Selection method to simultaneously quickly and automatically specify appropriate parameter values in DBSCAN.

4.3 Datesets

To provide a comprehensive evaluation of the presented AR-DBSCAN framework, we utilized nine artificial cluster-

TABLE 2: Statistics of benchmark datasets.

Type	Dataset	Classes	Size	Dim.	Time
Offline	Aggregation	7	788	2	✗
	Compound	6	399	2	✗
	Pathbased	3	300	2	✗
	D31	31	3100	2	✗
	Unbalance2	8	6500	2	✗
	Asymmetric	5	1000	2	✗
	Skewed	6	1000	2	✗
	Cure-t1-2000n-2D	6	2000	2	✗
	Cure-t2-4k-2D	7	4000	2	✗
Online	Powersupply	24	29928	2	✓

ing benchmark datasets and one public real-world streaming dataset for our experiments. Table 2 gives the statistics of these datasets, and further information is provided below.

Aggregation [51]. Features clusters of uneven sizes connected by narrow bridges, presenting a challenge for standard clustering methods.

Compound [52]. Combines three distinct cluster types: two clearly separated, two connected by a small neck, and a dense cluster surrounded by a sparse one.

Pathbased [53]. Contains an open circular cluster with two internal Gaussian clusters, each with 100 data points.

D31 [54]. Consists of 31 randomly placed, 2D Gaussian clusters, each with 100 data points.

Unbalance2 [55]. Contains three large, dense clusters (2000 points each) and five small, sparse clusters (100 points each) positioned nearby.

Asymmetric [55]. Features five clusters with uneven sizes and densities, where two are grouped together and three are separate.

Skewed [55]. Comprises six oblong clusters elongated along a 45-degree axis, testing the algorithm's ability to handle non-spherical shapes.

Cure-t1-2000n-2D [56]. A challenging dataset featuring a mix of high-density circular, large sparse, and two elliptical clusters connected by narrow bridges.

Cure-t2-4k-2D [56]. Extends the Cure-t1-2000n-2D dataset by adding 2000 noise points, significantly increasing clustering difficulty.

Powersupply [57]. A real-world streaming dataset from an Italian electricity company (30,000 data points from 1995-1998), with the task of predicting the hour of power supply usage (24 clusters).

4.4 Evaluation metrics

We evaluate experiments based on two metrics: accuracy and efficiency. Specifically, we use the normalized mutual information (NMI) and the adjusted rand index [58] (ARI) to measure accuracy. For efficiency, we employ the DBSCAN clustering round to measure the efficiency of various models. Since our approach, AR-DBSCAN, involves multiple agents searching for optimal parameters on separate subsets of the dataset, we align all agents' clustering results and aggregate their results for each round to obtain the final result of AR-DBSCAN for that round. In cases where one agent has fewer total rounds than others due to early stopping, we repeat the last round's result for alignment purposes.

4.5 Implementation Details

All experiments comprise two parts: the offline evaluation and the online evaluation. The continuous training mode and the two testing work modes only support a single agent, and AR-DBSCAN degenerates into DRL-DBSCAN in this scenario. Therefore, during the online task, we only compare AR-DBSCAN in the retraining model with the baselines. Due to the randomness of most algorithms, we perform ten runs on all datasets with different seeds and report the mean and variance as results, except for KDist, because it is a heuristic and does not involve random problems. For the allocation of agents in AR-DBSCAN, we set Eps to 0.3 and $MinPts$ to 1 for all datasets. In the parameter search part, we use a unified label training proportion of 0.2, an Eps parameter space size π_{Eps} of 5, a $MinPts$ parameter space size π_{MinPts} of 4, a maximum number of search steps I_{max} of 30, and a reward factor δ of 0.2. The FCN and MLP dimensions are uniformly set to 32 and 256, respectively, the reward decay factor γ of Critic is set to 0.1, and the number of samples M from the buffer is set to 16. We uniformly set the maximum number of episodes to 15. In the offline tasks, the maximum number of recursive layers L_{max} is 3, and the maximum search boundary in the 0-th layer of $MinPts$, $B^{(0)}_{MinPts,2}$, is 0.25, while in the online task, they are set to 6 and 0.0025 times the size of data block $\mathcal{V}$, respectively. All baselines use the same objective function (Eq. 16), parameter search space, and parameter minimum step size as our models if they support the settings.

5 Evaluation

In this section, we conduct several experiments to evaluate the performance of AR-DBSCAN. Firstly, we experiment with models on the offline task, using nine artificial datasets. Following this, we evaluate models through the design of an online task. Additionally, we conduct the ablation study to explore the impact of each component in AR-DBSCAN. Finally, we present a detailed case study to demonstrate the practicality and effectiveness of AR-DBSCAN. Further experiments on computational cost and the impact of label proportion are reported in Appendix A and Appendix C.

5.1 Offline Evaluation

The offline evaluation is conducted on nine artificial benchmark datasets. Since there is no pre-training data available in offline scenarios, each parameter search process is initiated from the beginning. To obtain a more accurate estimate for all models, we perform all experiments using ten different random seeds. The Cure-t1 and Cure-t2 datasets correspond to the Cure-t1-2000n-2D and Cure-t2-4k-2D datasets introduced in Sec. 4.3.

5.1.1 Accuracy and Stability Analysis

We perform a parameter search process on AR-DBSCAN, DRL-DBSCAN, and eight other baselines in 30 clustering rounds. We report the best NMI and ARI metrics obtained

TABLE 3: Offline evaluation of NMI performance. The best results are bolded, and the second-best results are underlined. The Avg. column shows the mean performance of the model across all datasets while ignoring the variances.

	Model	Aggregation	Compound	Pathbased	D31	Unbalance2	Asymmetric	Skewed	Cure-t1	Cure-t2	Avg.
Traditional models	**Rand**	.759±.114	.748±.054	.659±.232	.308±.329	.942±.053	.744±.059	.604±.176	.794±.156	.564±.266	.684
	BO-TPE	.716±.141	.702±.241	.784±.069	.229±.243	.954±.005	.777±.034	.472±.253	.808±.173	.708±.115	.683
Meta-heuristic models	**Anneal**	.749±.271	.520±.361	.649±.237	.170±.193	.744±.395	.721±.190	.487±.296	.724±.356	.509±.323	.586
	PSO	.593±.354	.455±.335	.600±.278	.359±.334	.893±.221	.756±.052	.525±.260	.840±.200	.578±.245	.622
	GA	.752±.146	.704±.254	.684±.188	.229±.202	.896±.193	.731±.070	.509±.268	.703±.269	.480±.285	.632
	DE	.277±.372	.326±.354	.222±.277	.236±.259	.592±.446	.590±.228	.214±.277	.534±.372	.342±.315	.365
Dedicated models	**KDist**	.601	.392	.404	.074	.979	.468	.516	.449	.569	.495
	BDE	.634±.281	.723±.251	.512±.332	.414±.361	.945±.047	.770±.050	.649±.152	.851±.146	.644±.181	.682
Ours	**DRL-DBSCAN**	.956±.021	.784±.039	.817±.028	.667±.024	.952±.002	.789±.065	.738±.079	.929±.109	.745±.113	.820
	AR-DBSCAN	**.978±.005**	**.951±.005**	**.825±.061**	**.773±.062**	**.995±.004**	**.909±.028**	**.838±.032**	**.987±.013**	**.764±.104**	**.891**

TABLE 4: Offline evaluation of ARI performance. The best results are bolded, and the second-best results are underlined. The Avg. column shows the mean performance of the model across all datasets while ignoring the variances.

	Model	Aggregation	Compound	Pathbased	D31	Unbalance2	Asymmetric	Skewed	Cure-t1	Cure-t2	Avg.
Traditional models	**Rand**	.679±.163	.726±.040	.634±.211	.138±.262	.971±.085	.550±.129	.419±.187	.719±.236	.482±.270	.591
	BO-TPE	.627±.185	.679±.238	.789±.104	.044±.046	.989±.003	.621±.106	.312±.186	.771±.222	.642±.175	.608
Meta-heuristic models	**Anneal**	.704±.272	.513±.353	.657±.250	.034±.042	.770±.409	.567±.192	.347±.236	.708±.360	.473±.306	.530
	PSO	.512±.367	.420±.361	.552±.379	.088±.219	.926±.232	.584±.117	.374±.227	.805±.240	.523±.251	.532
	GA	.679±.192	.684±.237	.674±.260	.040±.041	.916±.216	.538±.128	.350±.220	.639±.295	.396±.284	.546
	DE	.253±.354	.314±.343	.184±.284	.061±.087	.606±.464	.398±.170	.136±.191	.467±.377	.283±.286	.300
Dedicated models	**KDist**	.524	.391	.381	.004	.998	.122	.154	.127	.235	.326
	BDE	.542±.281	.704±.251	.482±.397	.207±.280	.975±.066	.605±.130	.475±.174	.800±.234	.577±.206	.596
Ours	**DRL-DBSCAN**	.959±.034	.760±.032	**.849±.038**	.262±.017	.988±.001	.684±.119	.600±.134	.931±.125	**.705±.110**	.749
	AR-DBSCAN	**.987±.003**	**.945±.005**	.808±.094	**.400±.104**	**.999±.001**	**.899±.067**	**.772±.084**	**.994±.016**	.634±.225	**.826**

using the optimal DBSCAN parameters identified during the search process. The results are reported in Table 3 and Table 4 in the format of $mean \pm variance$. Specifically, for the NMI metric, AR-DBSCAN shows mean improvements of 30.3%, 30.5%, 52.0%, 43.2%, 41.0%, 144.1%, 80.0%, and 30.6% across all datasets compared to the other eight baselines, respectively. For the ARI metric, AR-DBSCAN shows mean improvements of 39.8%, 35.9%, 55.8%, 55.3%, 51.3%, 175.3%, 153.4%, and 38.6% compared to the other eight baselines. These results demonstrate the superiority of our models. Furthermore, the decrease in variances across various datasets demonstrates the robustness of our models. For AR-DBSCAN, variances decrease by at most 0.367 & 0.364, 0.356 & 0.356, 0.271 & 0.303, 0.299 & 0.176, 0.442 & 0.463, 0.2 & 0.125, 0.264 & 0.152, 0.359 & 0.361, 0.219 & 0.081 on NMI and ARI for the nine datasets compared to the other eight baselines. The evident benefits regarding accuracy and robustness demonstrate that, in comparison to other hyperparameter optimization baselines with the same objective function, AR-DBSCAN can consistently find exceptional parameter combinations over several rounds of parameter search. Moreover, when compared to the submodel DRL-DBSCAN, AR-DBSCAN demonstrates consistent improvements in accuracy across all nine datasets, except for the Pathbased and Cure-t2 datasets, where AR-DBSCAN underperforms DRL-DBSCAN with 0.041 and 0.071 on the ARI metric, respectively. In addition, AR-DBSCAN exhibits greater stability on six of the nine datasets compared to DRL-DBSCAN. This indicates that allocating different density partitions to different agents can enhance the accuracy and robustness of clustering.

It is notable that DE underperforms other baselines in terms of accuracy due to its bias towards continuous parameters during parameter search, which requires more iterations to achieve optimal results. BO-TPE, on the other hand, uses a probabilistic surrogate model to learn from previously searched parameter combinations, effectively balancing exploration and exploitation. This approach provides advantages over other baselines when applied to multiple datasets. However, all these baselines are found to be stuck in high variances due to limited cluster rounds. KDist demonstrates strong performance on the Unbalance2 dataset, as the clusters within Unbalance2 exhibit significant density disparities and well-defined cluster boundaries. However, it underperforms in comparison to other methods when applied to the D31 dataset. This is due to the uniform density of clusters within the D31 dataset, which violates the underlying assumption of density variation made by the KDist algorithm. In contrast, AR-DBSCAN and DRL-DBSCAN use a recursive structure that progressively narrows the search space of parameters for each layer while also learning from historical experience. This approach is more suitable for searching DBSCAN clustering parameter combinations. Additionally, AR-DBSCAN introduces the pre-partition of different densities based on structural entropy, which further improves the cluster accuracy and robustness by easing the optimal parameter combination collision of different density clusters.

5.1.2 Efficiency Analysis

To showcase the parameter searching efficiency of our models, we report the average historical maximum normalized mutual information (NMI) results for AR-DBSCAN, DRL-DBSCAN, and the other baseline algorithms on eight datasets, with respect to the number of clustering rounds. The results are depicted in Figure 6, with shaded areas indicating the range of fluctuations (variance) of NMI across multiple runs. For comparison, the shaded area representing the variance of NMI for the BO-TPE algorithm is also

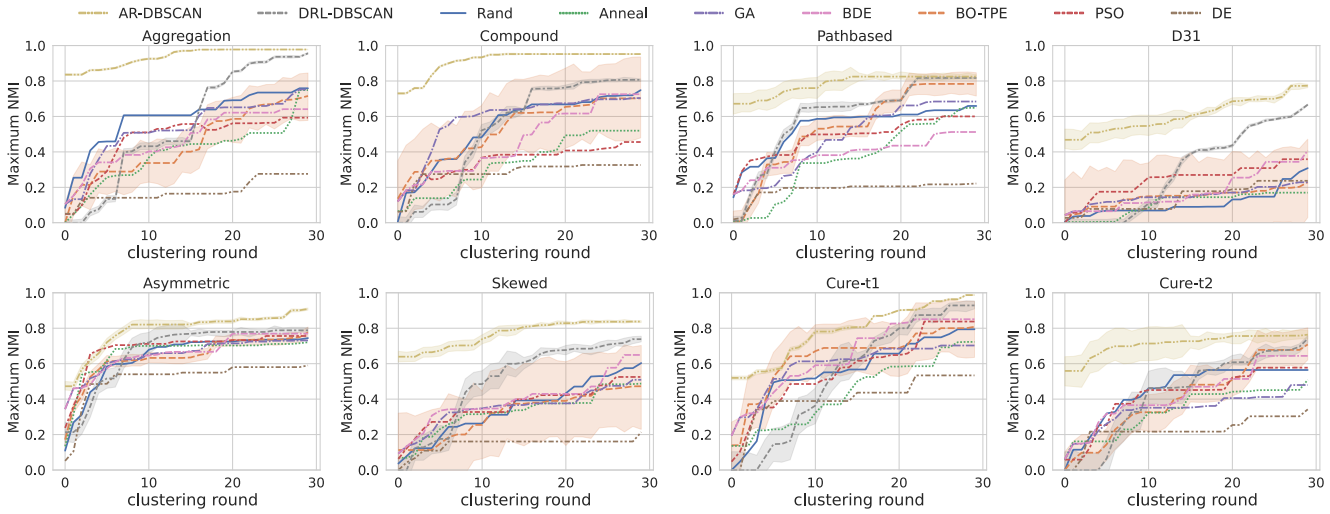


Fig. 6: Offline clustering efficiency comparison.

TABLE 5: Online evaluation of NMI performance. The best results are bolded, and the second-best results are underlined.

Blocks	Rand	BO-TPE	Anneal	PSO	GA	DE	KDist	BDE	DRL$_{re}$	DRL$_{con}$	AR-DBSCAN
$\mathcal{V}_1$	.082±.072	.152±.039	.104±.085	.096±.081	.130±.069	.024±.046	.170	.067±.073	.176±.004	.176±.005	**.179±.007**
$\mathcal{V}_2$	.081±.048	.096±.043	.048±.058	.066±.051	.114±.036	.072±.050	**.178**	.072±.050	.135±.005	.139±.003	.138±.012
$\mathcal{V}_3$	.159±.085	.148±.098	.071±.090	.193±.069	.160±.102	.131±.099	.000	.155±.100	**.238±.009**	.236±.007	**.238±.009**
$\mathcal{V}_4$	.087±.044	.110±.012	.048±.057	.087±.045	.077±.053	.023±.034	**.191**	.105±.036	.117±.011	.115±.014	.122±.003
$\mathcal{V}_5$	.107±.095	.181±.061	.064±.098	.172±.086	.148±.097	.078±.097	.000	.118±.100	.218±.014	.219±.005	**.223±.004**
$\mathcal{V}_6$	.167±.057	.199±.018	.130±.087	.128±.084	.170±.059	.054±.083	.104	.156±.071	**.204±.003**	.203±.004	.196±.019
$\mathcal{V}_7$	.086±.036	.130±.010	.104±.053	.105±.040	.075±.056	.039±.046	.000	.058±.056	**.133±.005**	.131±.006	**.133±.005**
$\mathcal{V}_8$	.129±.072	.173±.058	.097±.097	.120±.086	.156±.073	.024±.054	.000	.064±.078	**.204±.009**	.198±.003	**.204±.009**

TABLE 6: Online evaluation of ARI performance. The best results are bolded, and the second-best results are underlined.

Blocks	Rand	BO-TPE	Anneal	PSO	GA	DE	KDist	BDE	DRL$_{re}$	DRL$_{con}$	AR-DBSCAN
$\mathcal{V}_1$	.021±.022	.038±.014	.027±.022	.024±.021	.032±.021	.005±.014	.002	.014±.020	.047±.005	.046±.003	**.049±.005**
$\mathcal{V}_2$	.020±.014	.024±.012	.012±.014	.016±.014	.027±.010	.017±.013	.007	.016±.013	.033±.003	.034±.002	**.035±.002**
$\mathcal{V}_3$	.041±.029	.041±.028	.018±.024	.055±.023	.047±.032	.036±.029	.000	.044±.029	**.074±.007**	.071±.004	**.074±.007**
$\mathcal{V}_4$	.020±.010	.024±.002	.010±.012	.020±.010	.016±.011	.005±.009	.009	.021±.008	.025±.002	**.025±.001**	.024±.002
$\mathcal{V}_5$	.026±.027	.044±.018	.017±.027	.050±.026	.040±.011	.019±.025	.000	.033±.032	.058±.009	.066±.004	**.066±.003**
$\mathcal{V}_6$	.046±.019	.051±.017	.036±.027	.034±.024	.047±.019	.015±.023	.004	.043±.024	.059±.003	**.062±.002**	.055±.012
$\mathcal{V}_7$	.018±.011	.033±.003	.026±.013	.024±.013	.017±.016	.008±.012	.000	.013±.016	**.036±.001**	.034±.002	**.036±.001**
$\mathcal{V}_8$	.030±.023	.049±.018	.028±.028	.032±.026	.042±.023	.005±.013	.000	.015±.023	.057±.004	**.058±.003**	.057±.004

displayed as a representative of the baseline algorithms. It is worth noting that efficiency in this context refers to the efficiency of parameter searching.

Compared to the baseline algorithms, DRL-DBSCAN outperforms them across all eight datasets before the 23rd clustering round, which demonstrates the effectiveness of the deep reinforcement learning and recursion mechanisms. In the D31 dataset, DRL-DBSCAN achieves a speedup of 2.09 times compared to the BO-TPE baseline when both algorithms reach an NMI of 0.2. On the other hand, AR-DBSCAN maintains an advantage over all datasets at any clustering round and converges to higher accuracy. We analyze this from two aspects: 1) The pre-partitioning based on a two-level encoding tree demonstrates its ability to cluster and provides a sufficiently good pre-clustering result. 2) The allocation of agents for different density partitions helps to improve the performance of the DRL-DBSCAN submodel as it increases the convergence upper bound of DRL-DBSCAN. We also observe that the shaded area of the curves for AR-DBSCAN and DRL-DBSCAN gradually decreases over time, while the shaded area of BO-TPE tends to remain constant. This improvement in stability is attributed to the use of DRL, which allows the action decider ($Actor$) to learn and improve gradually over time.

5.2 Online Evaluation

To comprehensively explore the AR-DBSCAN performance, we design an online evaluation and compare the performance of AR-DBSCAN on online tasks with DRL-DBSCAN and other baselines. Since the remaining three working modes for the online task of AR-DBSCAN involve a single agent and result in the degeneration of our model into DRL-DBSCAN, the comparison is limited to AR-DBSCAN in retraining mode against eight baselines and the training-based modes of DRL-DBSCAN (retraining mode DRLre and continuous training mode DRLcon) on a streaming dataset named Powersupply. The Powersupply dataset is divided into eight blocks of equal size. All baselines are initialized before the start of each block, like DRL_{re}.

We report the results of NMI and ARI as the performance metric on Table 5 and Table 6, respectively. Since the $stop$ action may lead to an automatic termination mechanism for the search process in AR-DBSCAN, we set a maximum

TABLE 7: Impact of normalization. The best results are bolded, and the second-best results are underlined. "w/o norm. W" and "w/o norm. $H^1(G)$" denote the removal of edge weight and one-dimensional structural entropy normalization, respectively.

Dataset	w/o norm. W	w/o norm. $H^1(G)$	AR-DBSCAN
Aggregation	**.992±.005**	.977±.005	.978±.005
Compound	.944±.016	.946±.008	**.951±.005**
Pathbased	.737±.228	.783±.160	**.825±.061**
D31	.729±.022	.692±.266	**.773±.062**
Unbalance2	.995±.006	.990±.004	**.995±.004**
Asymmetric	.907±.022	**.910±.021**	.909±.028
Skewed	.773±.042	.776±.044	**.838±.032**
Cure-t1	.985±.012	.960±.074	**.987±.013**
Cure-t2	**.938±.022**	.757±.102	.764±.104

TABLE 8: Comparition the clustering results of AR-DBSCAN, optimal two-level encode tree, and DRL-DBSCAN. The best results are bolded, and the second-best results are underlined. The SE column is the clustering results of the optimal two-level encode tree.

Dataset	SE	DRL-DBSCAN	AR-DBSCAN
Aggregation	.836	.956±.021	**.978±.005**
Compound	**.952**	.784±.039	.951±.005
Pathbased	**.913**	.817±.028	.825±.061
D31	.632	.667±.024	**.773±.062**
Unbalance2	.980	.952±.002	**.995±.004**
Asymmetric	.737	.789±.065	**.909±.028**
Skewed	.639	.738±.079	**.838±.032**
Cure-t1	.903	.929±.109	**.987±.013**
Cure-t2	.561	.745±.113	**.764±.104**

number of search rounds to ensure that the experimental conditions for the baselines are synchronized. Specifically, we use the average number of clustering rounds consumed when AR-DBSCAN is automatically terminated as the maximum round (30 for Table 5 and Table 6). Compared to baselines other than DRL-DBSCAN, AR-DBSCAN consistently shows improvement across all eight data blocks. Although K-Dist can determine parameters without relying on labels and iterations and achieves the best performance in blocks $\mathcal{V}2$ and $\mathcal{V}3$ according to the NMI metric, it exhibits considerable instability and may even yield 0 in some data blocks. Additionally, AR-DBSCAN outperforms DRL-DBSCAN in five out of eight data blocks, as measured by both the NMI and ARI metrics. Although DRLcon can leverage historical environment information and achieve performance enhancement compared to DRLre in certain blocks, it only surpasses AR-DBSCAN in three blocks according to the ARI metric. These results demonstrate the capability of AR-DBSCAN to cluster streaming data and highlight the advantages of our agent allocation method and the learnable DBSCAN parameter search in terms of accuracy and stability. It is noteworthy that in block $\mathcal{V}_3$, $\mathcal{V}_7$ and $\mathcal{V}_8$, AR-DBSCAN yields the same results as DRLre. This occurrence arises from the fact that, in these specific data blocks, the inner cluster structure of the data remains unclear. Consequently, our agent allocation method assigns all vertices within the block to a single agent, resulting in the degeneration of AR-DBSCAN into DRLre.

5.3 Ablation Study

In the ablation study, we investigate the effects of normalization on edge weight W and one-dimensional structural entropy $H^1(G)$, the capabilities of the optimal two-level encoding tree, the sensitivity of hyperparameters in the agent allocation and parameter search process, as well as the role of the recursive mechanism.

5.3.1 Normalization

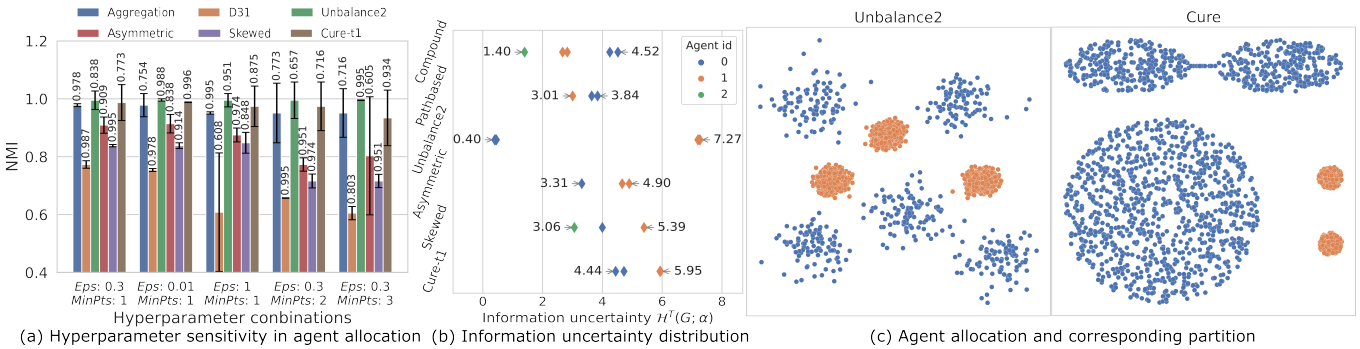
In Sec. 3.1.1, we propose to normalize the edge weight in Eq. 6 and the one-dimensional structural entropy in Eq. 7. Here, we experiment with the impact of normalization on AR-DBSCAN and report the NMI results in Table 7. The normalization of W and $H^1(G)$ enhances the model performance on six of nine datasets. On the Pathbased dataset, the normalization on edge weight W and one-dimensional structural entropy $H^(1)(G)$ leads to an 11.9% and 5.4% improvement in performance, respectively, while simultaneously decreasing the variance by 0.167 and 0.099. We also observe that not normalizing the edge weight is a preferable choice on the Cure-t2 dataset. This is attributed to the fact that half of the data points in the Cure-t2 dataset are noise. These overall improvements in performance and stability provide evidence of the positive effect of normalization in our model.

5.3.2 Cluster based on Encoding Tree

In our framework, there are three models that can be used to cluster datasets: AR-DBSCAN, optimal two-level encode tree, and the submodel DRL-DBSCAN. Here, We experiment with the cluster accuracy of these three models and report the NMI results in Table 8. The results of the optimal two-level encoding tree remain constant because the value of k selected by Algorithm 1 remains unchanged for the same dataset. The optimal two-level encode tree demonstrates strong performance in cluster applications as an unsupervised cluster model, even achieving the highest performance on the Compound and Pathbased datasets. As the overall model, AR-DBSCAN shows superiority over other models on seven of the nine datasets. This demonstrates the overall effectiveness of the design, highlighting the effectiveness of its design in enhancing cluster accuracy and improving the robustness of the submodel.

5.3.3 Hyperparameter on Agent allocation

To pre-partition the dataset based on density, we calculate the information uncertainty of the intermediate nodes in the optimal encode tree and simply employ a DBSCAN to cluster the intermediate nodes based on information uncertainty. Here, we investigate the influence of different combinations of DBSCAN parameters on agent allocation and report the results in Figure 7. Figure 7 (a) illustrates the impact of the parameter combinations on model performance. It shows the parameter Eps has a negligible effect on the model, except for the D31 dataset. This is due to the fact that the clusters in the D31 dataset exhibit similar densities, resulting in equivalent levels of information uncertainty for intermediate nodes. Consequently, any adjustment to Eps on the D31 dataset can lead to a significant alteration in the allocation of agents. On the other hand, the value of $Minpts$ is more significant since the optimal two-level encoding tree contains only a small number of intermediate nodes. Figure 7 (b) depicts that the information uncertainty proposed in Eq. 9 exhibits similar magnitudes across diverse



(a) Hyperparameter sensitivity in agent allocation (b) Information uncertainty distribution (c) Agent allocation and corresponding partition

Fig. 7: The impact of hyperparameter combination on agent allocation. (a) is the NMI results of AR-DBSCAN under different parameter combinations, (b) is the information uncertainty distribution of intermediate nodes in the optimal two-level encoding tree, (c) is the agent allocation results on Unbalance2, Cure-t1, and Compound dataset with the parameter $Eps = 0.3$ and $MinPts = 1$. Each color corresponds to an agent.

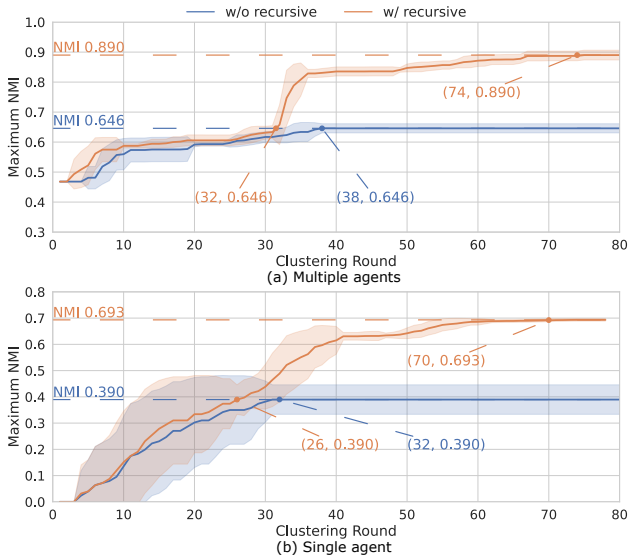


Fig. 8: The impact of recursive mechanism. The "w/o recursive" in the legend denotes a model without using the recursive mechanism, while "w/ recursive" indicates the utilization of the recursive mechanism. (a) is the results for multiple agents, (b) is the results for a single agent, where AR-DBSCAN degenerates into DRL-DBSCAN.

datasets. Figure 7 (c) shows the number of agents and the corresponding dataset partition for various datasets with the parameter combination of Eps=0.3 and $MinPts$=1. It is evident that for the Unbalance2 and Cure-t1 datasets, the clusters with high density are assigned to one agent, while the clusters with low density are assigned to the other agent. This illustrates the effectiveness of our agent assignment method in accurately differentiating cluster densities.

5.3.4 Recursive Mechanism

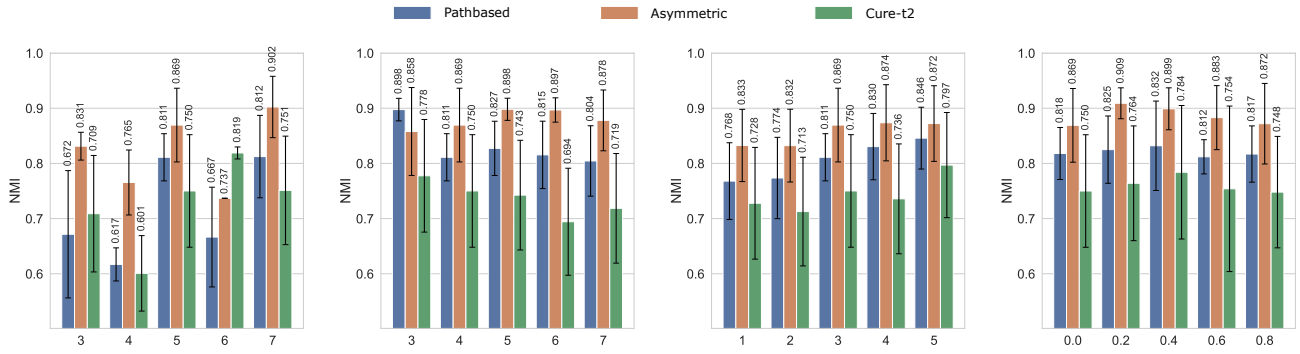
We evaluate the impact of the recursive mechanism on the D31 dataset and report the maximum NMI achieved across clustering rounds in Figure 8. To better compare, we turn off the early stop mechanism so that AR-DBSCAN can perform a longer search. In order to consider the influence of agent allocation on the overall search boundary for each agent, we conduct the experiment in scenarios involving both multiple agents and a single agent in Figure 8 (a) and (b), respectively. It can be observed that the utilization of the recursive mechanism can enhance the efficiency of parameter search by saving six rounds of search to reach the same peak as the variant without the recursive mechanism. Furthermore, we also observe that the recursive mechanism contributes to the improvement of the final convergence performance. This improvement can be attributed to the recursive mechanism narrowing down the search range and incrementally enhancing search precision layer by layer. Consequently, it mitigates the issue of becoming trapped in local optima, which can occur when employing a direct search with high precision.

5.3.5 Hyperparameter on Parameter Search Process

Figure 9 illustrates the results of AR-DBSCAN on the Pathbased, Asymmetric, and Cure-t2 datasets with various hyperparameters. Figures 9 (a) and (b) show the effects of varying the parameter space size π_{Eps} and π_{MinPts}. The results indicate that the space size of Eps is a critical factor for model performance, and the optimal value varies for different datasets. The optimal value of π_{Eps} for the Pathbased and Asymmetric datasets is 5, while 6 is more suitable for the Cure-t2 dataset. Contrastively, AR-DBSCAN is less sensitive to π_{MinPts} compared to π_{Eps}. Figure 9 (c) indicates that more layers generally lead to better model performance, as deeper recursion in the parameter search space results in a more precise search. Unlike DRL-DBSCAN, which encounters performance degradation in some datastes when L_{max} is too large [36], AR-DBSCAN is not affected by large L_{max}. This is attributed to our multi-agent design, which diminishes the parameter optimization conflict of the single agent in DRL-DBSCAN. Figure 9 (d) shows the impact of the reward factor δ, which influences the weights of the endpoint immediate reward and future maximum immediate reward on the final reward (Eq. 17). The results demonstrate that balancing the weight of the two immediate rewards generally benefits the model.

5.4 Case Study

To demonstrate the practicality and effectiveness of AR-DBSCAN, we conduct a comprehensive case study and report the results in Figure 10. Specifically, we evaluate AR-DBSCAN, DRL-DBSCAN, Rand, BO-TPE, Anneal, PSO, GA, and DE across nine datasets and 30 clustering rounds. The figure illustrates the best clustering results obtained from each model. It can be found that DRL-DBSCAN effectively distinguishes the three clusters located in the bottom left



(a) Eps Size (π_{Eps}) (b) MinPts Size (π_{Minpts}) (c) Number of Layers (L_{max}) (d) Reward Factor (δ)

Fig. 9: Sensitivity analysis of the hyperparameters in the parameter search process.

corner of the Aggregation dataset. AR-DBSCAN showcases a similar capability, but it incorrectly assigns the two small clusters to the same cluster. On the other hand, the other models mistakenly group them as a single cluster. DE performs the least satisfactorily, as it separates the seven-class dataset into two clusters. On the Compound dataset, AR-DBSCAN is the only model that is able to accurately separate the dense cluster situated in the right half from the surrounding sparse cluster. This is attributed to the pre-partitioning approach we proposed based on information uncertainty, which allows for advanced separation of these two clusters. Other models struggle with clustering the dense cluster and part of the surrounding data points together, leading to inaccurate results. We also observe that all models fail to separate the two clusters located in the upper-left region, which are connected by a small neck. Most models perform well on the Pathbased dataset. However, on the D31 dataset, most models are only able to distinguish two clusters, while AR-DBSCAN is capable of identifying a larger number of clusters. Except for AR-DBSCAN, all other models are affected by the three high-density clusters in the Unbalance2 dataset during the parameter search process, resulting in the misidentification of the other five sparse clusters as noise. In contrast, AR-DBSCAN assigns an agent to the high-density clusters and the other agent to the low-density clusters. This allocation enables AR-DBSCAN to achieve the nearly perfect distinction of all eight clusters. In the case of the Asymmetric dataset, AR-DBSCAN is the only model capable of separating all five clusters, while others incorrectly cluster some of the clusters together. The Skewed dataset is a challenging dataset for all models, as they all struggle with incorrectly dividing the end of certain oblong clusters into other clusters. Still, AR-DBSCAN only misses one of the clusters completely, while other baselines miss at least 3 of the clusters. On the Cure-t1 dataset, all models fail to separate the narrow bridges between the two elliptical clusters. AR-DBSCAN successfully clusters all other clusters, while DRL-DBSCAN struggles to cluster two small-scale clusters into separate clusters. We attribute this to the Eps value being excessively large, primarily influenced by the presence of two elliptical clusters and a large, sparse circular cluster situated in the lower-left region. The Cure-t2 dataset is generated by adding the same number of noise points as the data size of the Cure-t1 dataset. We observe that all models fail to correctly separate the two elliptical clusters in this dataset. Nonetheless, AR-DBSCAN still recognizes the largest number of clusters correctly.

6 Related Work

Structural Entropy. Information entropy was proposed to meet the demand for measuring uncertainty in information transmitted through communication systems. Correspondingly, to measure the information uncertainty in graph-structured data, structural entropy also was proposed and used to evaluate the complexity of the hierarchical structure of a graph by defining the encoding tree and structural entropy [27]. The process of constructing and optimizing the encoding tree is also a natural vertices clustering method for graphs. Due to the theoretical completeness and interpretability of structural entropy theory, it has great potential for application in graph analyses such as graph hierarchical pooling [30], graph structure learning [31], and role discovery [59]. Moreover, the two-dimensional and three-dimensional structural entropy, which measure the complexity of hierarchical structures at two and three dimensions, respectively, have found applications in fields such as medicine [35], bioinformatics [28], and network security [60].

Automatic DBSCAN Parameter Determination. The clustering results of DBSCAN are heavily dependent on the settings of two sensitive hyperparameters, Eps and $MinPts$, which are usually determined previously by practical experience. Numerous methods were proposed to address the challenge above. OPTICS [61] obtains the Eps by establishing cluster sorting based on reachability, but the acquisition of Eps needs to interact with the user. V-DBSCAN [14] and KDDClus [15] plot a curve based on the sorted distances of each object to its k-th nearest neighbor and identify significant changes in the curve to generate a series of candidate values for the Eps parameter. However, these methods fail to determine an appropriate value for the $MinPts$ parameter automatically. Despite the advancements in clustering algorithms such as DSets-DBSCAN [6], outlier detection-based DBSCAN [19], and RNN-DBSCAN [62], these methods still rely on fixed $MinPts$ values or predetermined k-nearest neighbors [62]. In addition to the approaches above, there have been efforts to identify the density of raw data points based on the size and shape of each data region determined by pre-defined grid partition parameters [16], [63]. While these methods alleviate the challenge of parameter selection to some extent, they still rely on heuristic decisions from the user, limiting their adaptability to changing data.

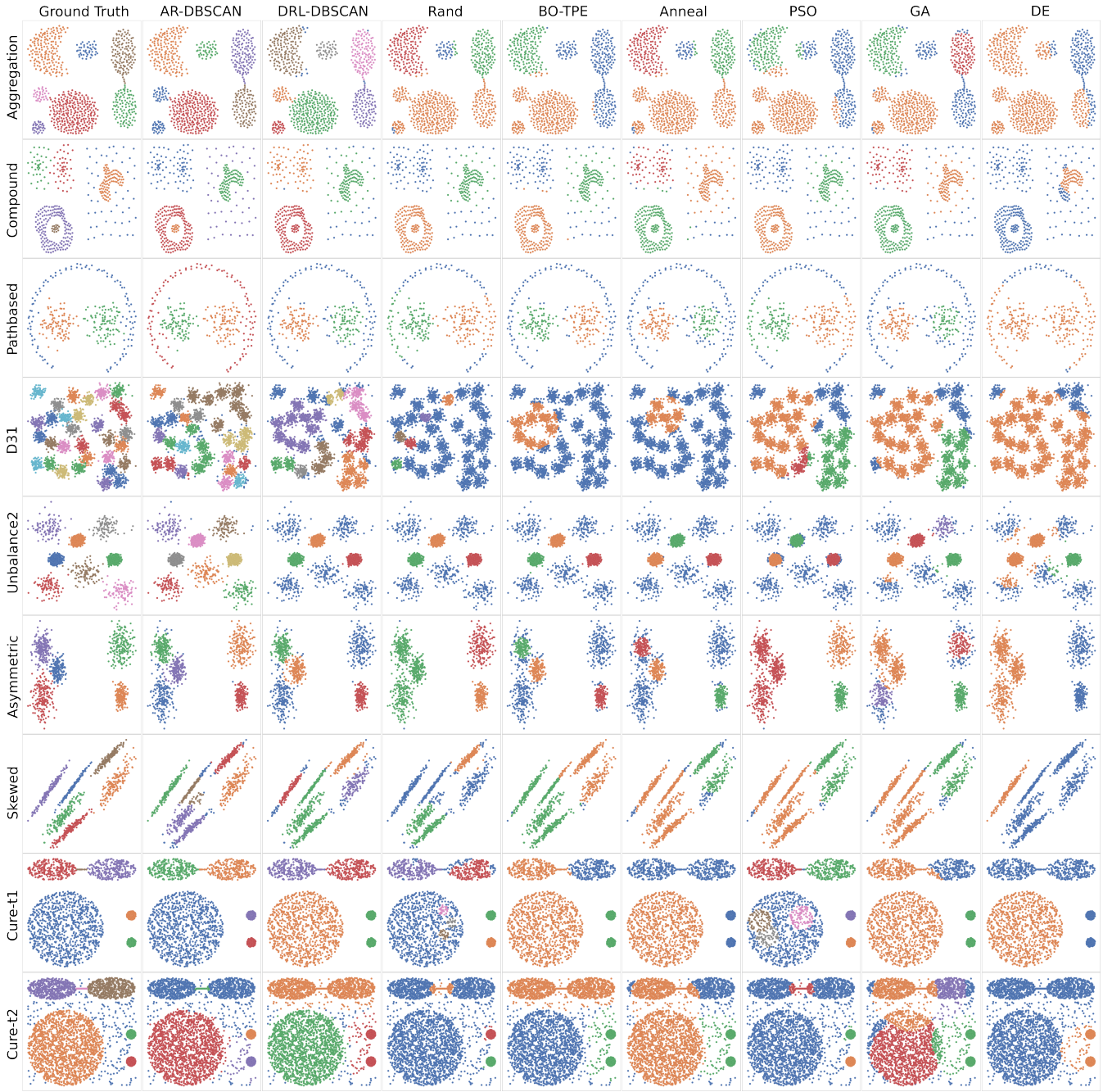


Fig. 10: The best results of multiple models across 30 clustering rounds. The datasets organized from top to bottom: Aggregation, Compound, Pathbased, D31, Unbalance2, Asymmetric, Skewed, Cure-t1, and Cure-t2.

Another viable method for parameter selection is based on the Hyperparameter Optimization (HO) algorithm. BDE-DBSCAN [21] aims to optimize an external purity index by employing a binary differential evolution algorithm to select the $MinPts$ parameters and a tournament selection algorithm to determine the Eps parameters. The MOGA-DBSCAN approach proposes the outlier index as an internal index [64]. This index is utilized as the objective function, and a multi-objective genetic algorithm is employed to optimize and select the parameters.

Reinforcement Learning for Algorithm Configuration and Clustering. Recently, the field of Dynamic Algorithm Configuration (DAC) has explored using reinforcement learning to dynamically control algorithm parameters during execution [32]. [33] establishes DAC as a new meta-algorithmic framework and presents DACBench, the first benchmark library for DAC. MA-DAC [34] proposes a multi-agent DAC to address the dynamic configuration of algorithms with multiple types of hyperparameters. The DAC framework is typically general-purpose, whereas AR-DBSCAN is a specialized solution for DBSCAN. Beyond these general configuration frameworks, RL has also been integrated more directly into clustering algorithms for specific tasks. AC-DRL [65] proposes an experience-driven approach based on an Actor-Critic based DRL framework to efficiently select the cluster head for managing the resources of the network, aiming to solve the challenges of the noisy nature of the Internet of Vehicles environment. In the particle physics

task, MCTS Clustering [66] utilizes Monte Carlo tree search to construct high-quality hierarchical clusters. In the health and medical domain, [67] proposes three distance metrics based on the state of the users and leverages two clustering algorithms and RL to cluster users who exhibit similar behaviors. The methods mentioned above are field-specific RL clustering methods, but they cannot implement a general clustering framework.

7 Conclusion

In this paper, we proposed an adaptive and robust DBSCAN with the multi-agent reinforcement learning cluster framework, AR-DBSCAN. The framework consists of two main components: a cluster density-based agent allocation method developed from the structure entropy theory and a deep reinforcement learning based parameter search method. The cluster density-based agent allocation method is designed to pre-partition the dataset based on the cluster density, which mitigates the problem of DBSCAN parameter search in varying density clusters. The deep reinforcement learning based parameter search method is applied to each agent for sensing the clustering environment and searching for the optimal parameter through weak supervision and an attention mechanism. A recursive search mechanism is devised to avoid the search performance decline caused by a large parameter space. The experimental results on various tasks and datasets demonstrate the high accuracy, efficiency, and stability of our model. In future work, we plan to explore additional concepts for the interaction among multi-agents to avoid the impact of mis-partitioning on the agent allocation method.

Acknowledgement

This work is supported by the NSFC through grants U25B2029, 62322202, 62441612, 62432006 and U24A20322, Beijing Natural Science Foundation through grant L253021, the Pioneer and Leading Goose R&D Program of Zhejiang through grant 2025C02044, Local Science and Technology Development Fund of Hebei Province Guided by the Central Government of China through grant 254Z9902G, National Key Laboratory under grant 241-HF-D07-01, Hebei Natural Science Foundation through grant F2024210008, Major Science Technology Special Projects of Yunnan Province through grants 202502AD080012 and 202502AD080006, and the Fundamental Research Funds for the Central Universities.

References

[1] A. E. Ezugwu, A. M. Ikotun, O. O. Oyelade, L. Abualigah, J. O. Agushaka, C. I. Eke, and A. A. Akinyelu, "A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects," *EAAI*, vol. 110, 2022.

[2] J. MacQueen, "Multivariate observations," in *Proceedings ofthe 5th Berkeley Symposium on Mathematical Statisticsand Probability*, vol. 1, 1967, pp. 281–297.

[3] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *TPAMI*, vol. 24, no. 5, pp. 603–619, 2002.

[4] Z. Huang, Y. Ren, X. Pu, and L. He, "Non-linear fusion for self-paced multi-view clustering," in *ACM MM*, 2021, pp. 3211–3219.

[5] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *ACM SIGKDD*, vol. 96, no. 34, 1996, pp. 226–231.

[6] J. Hou, H. Gao, and X. Li, "Dsets-DBSCAN: A parameter-free clustering algorithm," *TIP*, vol. 25, no. 7, pp. 3182–3193, 2016.

[7] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "DBSCAN revisited, revisited: why and how you should (still) use DBSCAN," *TODS*, vol. 42, no. 3, pp. 1–21, 2017.

[8] Y. Wang, Y. Gu, and J. Shun, "Theoretically-efficient and practical parallel DBSCAN," in *SIGMOD*, 2020, pp. 2555–2571.

[9] S. T. Mai, S. Goebl, and C. Plant, "A similarity model and segmentation algorithm for white matter fiber tracts," in *ICDM*. IEEE, 2012, pp. 1014–1019.

[10] Z. Zhu, G. Huang, J. Deng, Y. Ye, J. Huang, X. Chen, J. Zhu, T. Yang, D. Du, J. Lu *et al.*, "Webface260M: A benchmark for million-scale deep face recognition," *TPAMI*, 2022.

[11] S. T. Mai, J. Jacobsen, S. Amer-Yahia, I. Spence, N.-P. Tran, I. Assent, and Q. V. H. Nguyen, "Incremental density-based clustering on multicore processors," *TPAMI*, vol. 44, no. 3, pp. 1338–1356, 2020.

[12] J.-H. Kim, J.-H. Choi, K.-H. Yoo, and A. Nasridinov, "Aa-DBSCAN: an approximate adaptive DBSCAN for finding clusters with varying densities," *J SUPERCOMPUT*, vol. 75, no. 1, pp. 142–169, 2019.

[13] W.-T. Wang, Y.-L. Wu, C.-Y. Tang, and M.-K. Hor, "Adaptive density-based spatial clustering of applications with noise (DBSCAN) according to data," in *ICMLC*, vol. 1. IEEE, 2015, pp. 445–451.

[14] P. Liu, D. Zhou, and N. Wu, "VDBSCAN: varied density based spatial clustering of applications with noise," in *ICSSSM*. IEEE, 2007, pp. 1–4.

[15] S. Mitra and J. Nandy, "KDDclus: A simple method for multi-density clustering," in *SCAKD*. Citeseer, 2011, pp. 72–76.

[16] K. Diao, Y. Liang, and J. Fan, "An improved DBSCAN algorithm using local parameters," in *Artificial Intelligence*. Springer, 2018, pp. 3–12.

[17] H. Chang and D.-Y. Yeung, "Robust path-based spectral clustering," *Pattern Recognition*, vol. 41, no. 1, pp. 191–203, 2008.

[18] A. Smiti and Z. Elouedi, "DBSCAN-gm: An improved clustering method based on gaussian means and DBSCAN techniques," in *INES*. IEEE, 2012, pp. 573–578.

[19] Z. Akbari and R. Unland, "Automated determination of the input parameter of DBSCAN based on outlier detection," in *AIAI*. Springer, 2016, pp. 280–291.

[20] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *NeurIPS*, ser. NIPS'11. Red Hook, NY, USA: Curran Associates Inc., 2011, p. 2546–2554.

[21] A. Karami and R. Johansson, "Article: Choosing DBSCAN parameters automatically using differential evolution," *IJCA*, vol. 91, no. 7, pp. 1–11, April 2014, full text available.

[22] S. Lessmann, R. Stahlbock, and S. Crone, "Optimizing hyperparameters of support vector machines by genetic algorithms." in *IC-AI*, 01 2005, pp. 74–82.

[23] W. Lai, M. Zhou, F. Hu, K. Bian, and Q. Song, "A new DBSCAN parameters determination method based on improved mvo," *IEEE Access*, vol. 7, pp. 104 085–104 095, 2019.

[24] X. Li and D. Li, "Discovery of rules in urban public facility distribution based on DBSCAN clustering algorithm," in *MIPPR*, vol. 6790. SPIE, 2007, pp. 639–645.

[25] A. Kanervisto, C. Scheller, and V. Hautamäki, "Action space shaping in deep reinforcement learning," in *2020 IEEE Conference on Games (CoG)*. IEEE, 2020, pp. 479–486.

[26] G. Dulac-Arnold, R. Evans, H. van Hasselt, P. Sunehag, T. Lillicrap, J. Hunt, T. Mann, T. Weber, T. Degris, and B. Coppin, "Deep reinforcement learning in large discrete action spaces," *arXiv preprint arXiv:1512.07679*, 2015.

[27] A. Li and Y. Pan, "Structural information and dynamical complexity of networks," *TIT*, vol. 62, no. 6, pp. 3290–3339, 2016.

[28] A. Li, X. Yin, B. Xu, D. Wang, J. Han, Y. Wei, Y. Deng, Y. Xiong, and Z. Zhang, "Decoding topologically associating domains with ultra-low resolution hi-c data by graph structural entropy," *Nature communications*, vol. 9, no. 1, p. 3265, 2018.

[29] Y. Liu, J. Liu, Z. Zhang, L. Zhu, and A. Li, "REM: From structural entropy to community structure deception," in *NeurIPS*, vol. 32, 2019.

[30] J. Wu, X. Chen, K. Xu, and S. Li, "Structural entropy guided graph hierarchical pooling," in *ICML*. PMLR, 2022, pp. 24 017–24 030.

[31] D. Zou, H. Peng, X. Huang, R. Yang, J. Li, J. Wu, C. Liu, and P. S. Yu, "Se-gsl: A general and effective graph structure learning framework through structural entropy optimization," in *Web Conference*, 2023, p. 499–510.

[32] A. Biedenkapp, H. F. Bozkurt, T. Eimer, F. Hutter, and M. Lindauer, "Dynamic algorithm configuration: Foundation of a new meta-algorithmic framework," in *ECAI*, 2020, pp. 427–434.

[33] S. Adriaensen, A. Biedenkapp, G. Shala, N. Awad, T. Eimer, M. Lindauer, and F. Hutter, "Automated dynamic algorithm configuration," *JAIR*, vol. 75, pp. 1633–1699, 2022.

[34] K. Xue, J. Xu, L. Yuan, M. Li, C. Qian, Z. Zhang, and Y. Yu, "Multi-agent dynamic algorithm configuration," *NeurIPS*, vol. 35, pp. 20 147–20 161, 2022.

[35] A. Li, X. Yin, and Y. Pan, "Three-dimensional gene map of cancer cell types: Structural entropy minimisation principle for defining tumour subtypes," *Scientific reports*, vol. 6, no. 1, pp. 1–26, 2016.

[36] R. Zhang, H. Peng, Y. Dou, J. Wu, Q. Sun, Y. Li, J. Zhang, and P. S. Yu, "Automating DBSCAN via deep reinforcement learning," in *ACM CIKM*, 2022, pp. 2620–2630.

[37] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *JMLR*, vol. 13, no. 2, 2012.

[38] M. Mundhenk, J. Goldsmith, C. Lusena, and E. Allender, "Complexity of finite-horizon markov decision process problems," *JACM*, vol. 47, no. 4, pp. 681–720, 2000.

[39] K. Zheng, H. Li, R. C. Qiu, and S. Gong, "Multi-objective reinforcement learning based routing in cognitive radio networks: Walking in a random maze," in *ICNC*. IEEE, 2012, pp. 359–363.

[40] L. Bom, R. Henken, and M. Wiering, "Reinforcement learning to train Ms. Pac-Man using higher-order action-relative inputs," in *ADPRL*. IEEE, 2013, pp. 156–163.

[41] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *NeurIPS*, 2000, pp. 1008–1014.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *NeurIPS*, vol. 30, 2017.

[43] P. A. Estévez, M. Tesmer, C. A. Perez, and J. M. Zurada, "Normalized mutual information feature selection," *Trans. Neural Networks*, vol. 20, no. 2, pp. 189–201, 2009.

[44] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *ICML*. PMLR, 2018, pp. 1587–1596.

[45] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *ICLR*, pp. 1–14, 2015.

[46] Y. Xian, P. Li, H. Peng, Z. Yu, Y. Xiang, and P. S. Yu, "Community detection in large-scale complex networks via structural entropy game," in *Proceedings of the ACM on Web Conference 2025*, 2025, pp. 3930–3941.

[47] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *NeurIPS*, vol. 24, 2011.

[48] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[49] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *EP*. Springer, 1998, pp. 591–600.

[50] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE TEVC*, vol. 13, no. 2, pp. 398–417, 2008.

[51] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *TKDD*, 2007.

[52] C. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *IEEE Trans. Comput.*, vol. C-20, no. 1, pp. 68–86, 1971.

[53] H. Chang and D.-Y. Yeung, "Robust path-based spectral clustering," *Pattern Recognition*, vol. 41, no. 1, p. 191–203, 2008.

[54] C. Veenman, M. Reinders, and E. Backer, "A maximum variance cluster algorithm," *TPAMI*, vol. 24, no. 9, pp. 1273–1280, 2002.

[55] M. Rezaei and P. Fränti, "Can the number of clusters be determined by external indices?" *IEEE Access*, vol. 8, pp. 89 239–89 257, 2020.

[56] T. Barton, "Clustering benchmarks," 2019. [Online]. Available: https://github.com/deric/clustering-benchmark/

[57] X. Zhu, "Stream data mining repository," 2010. [Online]. Available: http://www.cse.fau.edu/~xqzhu/stream.html

[58] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *JMLR*, vol. 11, pp. 2837–2854, 2010.

[59] X. Zeng, H. Peng, and A. Li, "Effective and stable role-based multi-agent collaboration by structural information principles," in *AAAI*, 2023.

[60] A. Li, Q. Hu, J. Liu, and Y. Pan, "Resistance and security index of networks: structural information perspective of network security," *Scientific reports*, vol. 6, no. 1, pp. 1–24, 2016.

[61] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," *ACM Sigmod record*, vol. 28, no. 2, pp. 49–60, 1999.

[62] A. Bryant and K. Cios, "RNN-DBSCAN: A density-based clustering algorithm using reverse nearest neighbor density estimates," *TKDE*, vol. 30, no. 6, pp. 1109–1121, 2017.

[63] H. Darong and W. Peng, "Grid-based DBSCAN algorithm with referential parameters," *Physics Procedia*, vol. 24, pp. 1166–1170, 2012.

[64] Z. Falahiazar, A. Bagheri, and M. Reshadi, "Determining the parameters of DBSCAN automatically using the multi-objective genetic algorithm." *J. Inf. Sci. Eng.*, vol. 37, no. 1, pp. 157–183, 2021.

[65] A. Sharif, J. P. Li, M. A. Saleem, G. Manogran, S. Kadry, A. Basit, and M. A. Khan, "A dynamic clustering technique based on deep reinforcement learning for internet of vehicles," *J Intell Manuf*, vol. 32, pp. 757–768, 2021.

[66] J. Brehmer, S. Macaluso, D. Pappadopulo, and K. Cranmer, "Hierarchical clustering in particle physics through reinforcement learning," *arXiv preprint arXiv:2011.08191*, 2020.

[67] E. M. Grua and M. Hoogendoorn, "Exploring clustering techniques for effective reinforcement learning based personalization for health and wellbeing," in *SSCI*. IEEE, 2018, pp. 813–820.

Hao Peng is currently a Professor at the School of Cyber Science and Technology at Beihang University. His research interests include representation learning and reinforcement learning. He is the Associate Editor of the IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP), International Journal of Machine Learning and Cybernetics (IJMLC), and Neural Networks.

Xiang Huang is currently a Master's Degree candidate in the School of Cyber Science and Technology at Beihang University. His research interests include Information Theory and Machine Learning.

Shuo Sun is currently a Ph.D. candidate in the School of Cyber Science and Technology at Beihang University. His research interests include Information Theory and Machine Learning.

Ruitong Zhang is currently a Ms.D. candidate in the School of Cyber Science and Technology in Beihang University. Her research interests include deep learning, graph mining and reinforcement learning.

Xizhao Wang is a Professor at the Big Data Institute of Shenzhen University. His current research interests include uncertainty modeling and machine learning for big data. He is the General Co-Chair of the 2002-2017 International Conferences on Machine Learning and Cybernetics, cosponsored by IEEE SMCS. He is a Fellow of the IEEE.

Philip S. Yu is a Distinguished Professor and the Wexler Chair in Information Technology at the Department of Computer Science, University of Illinois at Chicago. Before joining UIC, he was at the IBM Watson Research Center, where he built a world-renowned data mining and database department. He was the EIC of ACM TKDD (2011-2017) and IEEE TKDE (2001-2004). He is a Fellow of the ACM and IEEE.

APPENDIX A
TIME COST ANALYSIS

To compare the time cost of different models, we report the runtime of each model on the largest Unbalance2 and Cure-t2 datasets. The time is calculated by letting each model complete a maximum of 30 search rounds (including the DBSCAN execution time) and allowing the algorithm to determine early stopping. The process was repeated 10 times and the average results are reported in Table 9.

TABLE 9: Average time cost (second).

Dataset	Rand	BO-TPE	Anneal	PSO	GA	DE	KDist	BDE	DRL-DBSCAN	AR-DBSCAN
Unbalance2	22.66	22.10	21.14	17.38	26.98	32.68	19.38	29.12	52.30	55.20
Cure-t2	18.40	14.76	13.46	11.18	23.18	17.16	13.21	16.42	32.52	32.56

APPENDIX B
THE IMPACT OF MAXIMUM RECURSIVE LAYERS L_{max}

In this section, we report the full NMI results of AR-DBSCAN on offline datasets with varing maximum recursive layers in Table 10.

TABLE 10: NMI Performance on Offline Datasets with Varying Maximum Recursive Layers ($L_{\max}$)

$L_{\max}$	Aggregation	Compound	Pathbased	D31	Unbalance2	Asymmetric	Skewed	Cure-t1	Cure-t2
1	0.956	0.952	0.768	0.612	0.988	0.833	0.784	0.916	0.728
2	0.975	0.952	0.774	0.728	0.994	0.832	0.815	0.941	0.713
3	0.976	0.950	0.811	0.753	0.994	0.869	0.828	0.981	0.750
4	0.977	0.952	0.830	0.753	0.994	0.874	0.829	0.974	0.736
5	0.976	0.952	0.846	0.788	0.994	0.872	0.852	0.976	0.797

APPENDIX C
LABEL PROPORTION

In our experiments, we uniformly set the label training proportion to 0.2. Here, we examine the impact of a smaller label training proportion. The results are reported in Table 11. Compared to other baselines, AR-DBSCAN consistently keeps its performance and variance advantages across label proportions from 20% to 5%, which demonstrates the adaptability and stability of AR-DBSCAN to changes in the label proportion.

TABLE 11: NMI performance in offline tasks with different label proportions.

Model	Proportion	Aggregation	Compound	Pathbased	D31	Unbalance2	Asymmetric	Skewed	Cure-t1	Cure-t2
Rand	20%	.759±.114	.748±.054	.659±.232	.308±.329	.942±.053	.744±.059	.604±.176	.794±.156	.564±.266
	10%	.748±.255	.775±.081	.626±.233	.281±.244	.946±.040	.721±.063	.568±.203	.816±.139	.599±.233
	5%	.651±.312	.690±.049	.690±.157	.424±.225	.901±.213	.711±.087	.565±.183	.786±.241	.515±.215
	2%	.693±.276	.683±.053	.621±.235	.419±.335	.883±.157	.682±.049	.605±.188	.716±.234	.551±.219
BO-TPE	20%	.716±.141	.702±.241	.784±.069	.229±.243	.954±.005	.777±.034	.472±.253	.808±.173	.708±.115
	10%	.671±.230	.691±.250	.774±.068	.231±.165	.952±.007	.753±.037	.437±.260	.813±.189	.666±.113
	5%	.631±.246	.641±.226	.754±.147	.249±.204	.952±.004	.744±.062	.498±.317	.773±.160	.631±.221
	2%	.669±.243	.657±.224	.654±.235	.266±.259	.954±.006	.735±.139	.426±.306	.760±.272	.598±.301
PSO	20%	.593±.354	.455±.335	.600±.278	.359±.334	.893±.221	.756±.052	.525±.260	.840±.200	.578±.245
	10%	.575±.342	.573±.316	.622±.220	.263±.205	.885±.237	.764±.073	.522±.213	.732±.274	.616±.236
	5%	.530±.342	.562±.297	.552±.227	.294±.260	.861±.246	.734±.028	.528±.207	.839±.203	.493±.303
	2%	.543±.351	.557±.396	.532±.273	.288±.265	.848±.283	.720±.083	.468±.314	.834±.269	.564±.299
GA	20%	.752±.146	.704±.254	.684±.188	.229±.202	.896±.193	.731±.070	.509±.268	.703±.269	.480±.285
	10%	.789±.159	.682±.216	.653±.152	.286±.162	.883±.102	.705±.042	.558±.279	.695±.235	.547±.283
	5%	.748±.178	.679±.212	.641±.224	.273±.216	.878±.209	.700±.094	.517±.179	.657±.234	.445±.281
	2%	.747±.158	.631±.235	.640±.217	.286±.236	.862±.214	.678±.120	.508±.265	.667±.229	.425±.272
BDE	20%	.634±.281	.723±.251	.512±.332	.414±.361	.945±.047	.770±.050	.649±.152	.851±.146	.644±.181
	10%	.610±.230	.714±.284	.576±.316	.558±.346	.936±.052	.730±.047	.618±.175	.794±.148	.634±.229
	5%	.571±.246	.688±.208	.466±.213	.440±.224	.935±.061	.693±.096	.601±.174	.772±.142	.587±.199
	2%	.569±.243	.680±.290	.419±.271	.410±.263	.910±.057	.660±.115	.584±.177	.736±.156	.574±.200
AR-DBSCAN	20%	**.978±.005**	**.951±.005**	**.825±.061**	**.773±.062**	**.995±.004**	**.909±.028**	**.838±.032**	**.987±.013**	**.764±.104**
	10%	.974±.006	.940±.017	.825±.065	.748±.078	.993±.004	.876±.074	.837±.037	.966±.051	.758±.119
	5%	.971±.011	.927±.019	.819±.077	.741±.085	.993±.003	.875±.074	.830±.037	.946±.151	.729±.130
	2%	.946±.038	.906±.070	.809±.103	.722±.142	.991±.004	.850±.080	.823±.042	.921±.118	.708±.157