



GP-DPC: Genetic Programming-Driven Nonlinear Feature Projection for Scalable Community Detection in Attributed Networks

Chaoyi Ding¹, Jianfeng Wang², Haonan Li³ and Xingsi Xue^{4,*}

¹ College of Computer Science and Technology (College of Big Data), Taiyuan University of Technology, Taiyuan 030024, China

² College of Software, Taiyuan University of Technology, Taiyuan 030024, China

³ Faculty of Computer Science and Information Technology, University Malaya, Kuala 50603, Malaysia

⁴ Fujian Provincial Key Laboratory of Big Data Mining and Applications, Fujian University of Technology, Fuzhou 350118, China

* Correspondence: jack8375@gmail.com

How To Cite: Ding, C.; Wang, J.; Li, H.; et al. GP-DPC: Genetic Programming-Driven Nonlinear Feature Projection for Scalable Community Detection in Attributed Networks. *AI Engineering* **2026**, *2*(2), 18. <https://doi.org/10.53941/aieng.2026.100018>

Received: 29 June 2026

Revised: 14 September 2026

Accepted: 29 September 2026

Published: 13 October 2026

Abstract: Attributed networks are widely used to model engineering systems in which both relational structure and node-specific information are important, such as social platforms, transportation networks, and knowledge graphs. Community detection in such networks requires methods that can jointly exploit graph topology and heterogeneous node attributes while remaining computationally feasible. To address this problem, this paper proposes GP-DPC, a genetic programming-driven framework for projection-guided community detection in attributed networks. Instead of evolving pairwise similarity functions, GP-DPC uses a multi-tree GP representation to learn three node-level nonlinear projection functions, which map attributed nodes into a compact low-dimensional space. Fast density peaks clustering is then performed in the learned space through an approximate k NN strategy, avoiding full pairwise distance computation. A community-aware fitness function combines structural modularity and projection-space separability to guide the evolutionary search. Experiments cover synthetic Handl and LFR benchmarks and five real-world attributed networks. Direct comparisons on the Handl 10c500d and 24c1800d benchmarks and on Cora, Citeseer, and ACM show competitive detection accuracy, while runtime analyses indicate practical inference efficiency. These results support GP-DPC as a promising framework for engineering-oriented attributed network community detection.

Keywords: attributed networks; community detection; genetic programming; density peaks clustering; projection learning

1. Introduction

Community detection is a prototypical object grouping problem that primarily addresses the graph-structured processing of large-scale discrete or partially discrete data networks [1]. It emphasizes the connectivity and modular structures within the graph, partitioning network nodes into multiple subgroups such that intra-group connections are relatively dense while inter-group connections remain sparse [2]. This task holds a fundamental position in network science [3] and finds widespread applications in social network analysis [4,5], bioinformatics [6], recommender systems [7], and other engineering domains. For instance, in power systems, community detection enables controlled islanding and distributed control by partitioning grids into independent subregions [8,9]. In transportation and logistics, it identifies traffic bottlenecks and optimizes warehouse product allocation [10]. In communication networks, it improves data transmission efficiency in mobile edge computing and self-organizing vehicular systems [11,12]. In smart manufacturing, it supports key process identification, social manufacturing community formation, and assembly sequence planning [13,14]. These applications underscore the practical value of community detection. In engineering domains, networks typically exhibit both topological connectivity and node



attributes; thus, community detection must account for both structural modularity and attribute similarity. With the exponential growth of network scale and the increasing complexity of network structures, the dual challenges of efficiency and accuracy in community detection algorithms have become more pronounced [3, 15], driving a continuous surge of research efforts in this field.

Current community detection algorithms face three major technical challenges. First, topology-driven approaches rely exclusively on network structure and cannot incorporate heterogeneous node attributes such as semantic, literal, and structural features, leading to suboptimal community quality on attributed networks [16]. Second, clustering-based methods including Density Peaks Clustering (DPC) require a suitable low-dimensional representation of the data, yet manually designed feature projections are inflexible and fail to generalize across networks with different characteristics. This creates a demand for adaptive feature construction [17]. Third, while genetic programming (GP) offers a promising solution by automatically constructing nonlinear feature combinations that fuse heterogeneous attributes, its conventional application requires exhaustive pairwise distance computations using the evolved similarity functions, incurring prohibitive $O(n^2)$ overhead that severely limits its scalability in engineering practice [18].

Based on this, this paper proposes genetic programming-driven fast density peaks clustering community detection algorithm (GP-DPC), a community detection framework that integrates GP with FastDPC to achieve a favorable balance between accuracy and efficiency. Specifically, GP automatically constructs a low-dimensional projection that supplies the representation required by FastDPC, while FastDPC replaces the exhaustive pairwise-distance traversal in conventional GP-based feature construction, substantially reducing computational overhead. The main contributions are threefold:

- We propose a GP-driven nonlinear projection mechanism for attributed network community detection. The proposed mechanism constructs node-level feature combinations automatically and maps heterogeneous attributes into a compact projection space, reducing the dependence on manually designed representations.
- We design a multi-tree GP representation in which three GP trees jointly encode a three-dimensional projection function. The evolution is guided by a composite fitness function that combines modularity and silhouette coefficient, allowing the learned projection to preserve graph community structure while improving cluster separability in the projected space.
- We integrate approximate k NN-based FastDPC into the GP evaluation process. Compared with conventional GP-based similarity learning that requires $O(n^2)$ pairwise evaluation, the proposed framework avoids full distance-matrix construction and improves the efficiency of projection-based community detection.

The remainder of this paper is organized as follows. Section II formulates the problem and reviews related work on attributed network community detection, evolutionary feature construction, and density-based clustering. Section III presents the proposed GP-DPC framework, including the GP representation, terminal and function sets, nonlinear projection, fitness design, and FastDPC-based community assignment. Section IV describes the experimental study, including datasets, evaluation metrics, and baseline methods. Section V reports the experimental results, ablation studies, and parameter sensitivity analysis. Section VI discusses an engineering-oriented application scenario. Section VII concludes the paper and outlines future research directions.

2. Background

2.1. Community Detection Problem

This paper considers community detection in attributed networks. Let an attributed network be denoted as an undirected graph $\mathcal{G} = (V, E, \mathbf{X})$, where $V = \{v_1, v_2, \dots, v_n\}$ is the node set, $E \subseteq V \times V$ is the edge set, and $\mathbf{X} \in \mathbb{R}^{n \times d}$ is the node attribute matrix. The i -th row $\mathbf{x}_i \in \mathbb{R}^d$ represents the attribute vector of node v_i . Different from topology-only community detection, this setting requires the partition to reflect both graph connectivity and attribute consistency.

Given $\mathcal{G}$, the objective is to divide the node set into K disjoint and non-empty communities, denoted as $\mathcal{C} = \{C_1, C_2, \dots, C_K\}$. A desirable partition should satisfy two requirements. First, nodes in the same community should be densely connected in the original graph. Second, they should also be close or coherent in the attribute space. In contrast, nodes from different communities are expected to have weaker topological connections and lower attribute similarity. Formally, the partition $\mathcal{C}$ satisfies

$$\bigcup_{k=1}^K C_k = V, \quad C_k \cap C_l = \emptyset \quad (k \neq l), \quad K \in \mathbb{Z}^+. \quad (1)$$

In the proposed framework, GP-DPC approaches this problem by learning a low-dimensional projection of

the node attributes while evaluating the resulting partition with respect to the original graph structure. Specifically, the learned projection is used to support density-based community assignment, whereas the structural quality of the partition is assessed on $\mathcal{G}$. This formulation allows the method to combine attribute-driven separability with topology-driven community quality.

2.2. Related Work

Classical community detection algorithms are predominantly topology-driven. The Louvain algorithm [19] iteratively optimizes modularity gain to detect communities, achieving near-linear time complexity $O(n \log n)$. Subsequent parallel variants [20] and GPU-accelerated implementations such as GALA [21] further improve scalability. The Leiden algorithm [22] improves upon Louvain by guaranteeing well-connected communities while maintaining comparable efficiency. However, despite their efficiency, these methods rely exclusively on topological structure and modularity-based objective functions. They cannot incorporate heterogeneous node attributes, such as semantic, literal, or structural features, which are abundant in real-world attributed networks. Consequently, their community detection accuracy remains limited on such networks. Overall, topology-driven methods lack the fusion of node attributes.

Evolutionary algorithms have been widely adopted for community detection, among which Genetic Algorithms (GA) and GP are the most representative paradigms. GA-based methods typically encode community assignments as chromosomes and optimize a fitness function through iterative selection, crossover, and mutation which is often modularity or normalized cut. However, the feature construction capability of conventional GA is limited. They primarily operate on pre-defined feature spaces and lack the flexibility to adaptively combine heterogeneous node attributes. GP, by contrast, offers a more powerful framework for automatic feature construction. It evolves tree-structured expressions that nonlinearly combine multiple feature types into a tailored similarity function. This flexibility enables GP-based methods to capture complex relationships among nodes, often outperforming GA-based counterparts in accuracy. Nevertheless, both GA and GP suffer from a common scalability bottleneck: the evolved similarity function must be evaluated on every pair of nodes to construct the distance or similarity matrix, incurring $O(n^2)$ cost per fitness evaluation. This renders them impractical for large-scale attributed networks. Overall, evolutionary algorithms can construct features but suffer from prohibitively high pairwise evaluation overhead.

DPC algorithm [23] is a representative density-based clustering method. It assumes that cluster centers are local density maxima surrounded by lower-density neighbors and located at relatively large distances from any higher-density points. For each data point, DPC computes two quantities: local density ρ and distance to the nearest point with higher density δ . Cluster centers are then identified as points with anomalously large ρ and δ values. DPC is conceptually simple and requires no iterative optimization. However, the performance of DPC critically depends on the quality of the distance metric and the feature representation in the input space. In attributed networks where nodes carry heterogeneous features, directly applying DPC to the raw high-dimensional attribute space suffers from the well-known distance concentration effect [17]. Alternatively, projecting nodes into a low-dimensional space reduces the number of features that can be preserved, making it difficult to fully exploit the rich multi-source information carried by nodes. FastDPC [24] addresses the efficiency bottleneck of original DPC by replacing exhaustive pairwise distance computation with approximate k -nearest neighbor (k NN) search, reducing the time complexity from $O(n^2)$ to $O(n \log n)$. However, FastDPC does not resolve the feature representation challenge: the clustering quality remains limited by the quality of the low-dimensional projection or distance metric used as input. Overall, FastDPC achieves high efficiency yet relies heavily on the quality of the input representation.

In recent years, graph neural networks (GNNs) have been widely adopted for attributed network analysis. Unlike traditional methods, GNN-based approaches learn node embeddings by aggregating both structural and attribute information. Representative works include GAE, VGAE, ARGAE, and DAEGC, which combine graph autoencoders with adversarial or attentional mechanisms, as well as DGI and SDCN, which leverage self-supervised learning for community-oriented representation learning. In addition, spectral and matrix factorization methods also remain competitive alternatives. OA2H-SP performs hypergraph spectral clustering with adaptive anchor selection [25], dual-view and relative entropy-based NMF incorporate attribute information via entropy regularization [26,27], and SDAC-DA extends deep clustering to semi-supervised scenarios [28]. Numerous studies have demonstrated that these methods achieve strong performance on standard benchmarks and can uncover attribute-feature relationships that traditional methods fail to detect, leading to improved community detection quality. However, these methods typically require substantial computational resources, careful hyperparameter tuning, and offer limited interpretability.

These observations motivate the proposed GP-DPC framework. Instead of using GP to learn a pairwise similarity function, GP-DPC evolves node-level projection functions that map attributed nodes into a compact low-dimensional space. FastDPC is then performed in this learned space to obtain the community partition efficiently.

In this way, the proposed framework links adaptive feature construction with density-based community assignment: GP improves the representation used by FastDPC, while FastDPC avoids the full pairwise-distance evaluation that limits conventional GP-based similarity learning.

3. GP-DPC for Projection-Guided Community Detection

3.1. Overall Framework

Figure 1 illustrates the overall workflow of GP-DPC, and Algorithm 1 gives its main procedure. The framework follows a projection-guided community detection strategy. Instead of evolving a pairwise similarity function and applying it to all node pairs, GP-DPC evolves three node-level projection functions. These functions map each attributed node into a compact three-dimensional space. FastDPC is then applied in this learned space to generate a community partition. During evolution, each candidate projection is evaluated by a community-aware fitness function that combines structural modularity on the original graph and spatial separability in the projected space. In this way, GP-DPC connects adaptive representation construction with efficient density-based community assignment, while avoiding the full pairwise distance computation required by conventional GP-based similarity learning.

Following the framework outlined above, the computational cost of GP-DPC can be decomposed into offline training and online inference. During the offline evolutionary training, each generation evaluates N_p individuals. For each individual, projecting n nodes costs $O(nL)$, where L is the average tree size, and the subsequent FastDPC-based community assignment costs $O(n \log n)$ via approximate kNN search. The per-generation cost is therefore $O(N_p \cdot (nL + n \log n))$. Over $G_{\max}$ generations, the total training complexity is $O(G_{\max} \cdot N_p \cdot (nL + n \log n))$. For online inference, once the three projection functions are learned, projecting a newly arrived node costs only $O(L)$, and assigning it to an existing community via the kNN index requires $O(\log n)$. This separation of expensive offline training from lightweight online inference means that the offline computational cost can be amortized over repeated inference. Once the projection is learned, GP-DPC can support timely online assignment in engineering scenarios where periodic offline retraining is acceptable, such as traffic zone partitioning or network monitoring. A detailed empirical analysis of running time and memory consumption across different network scales is presented in Section 5.

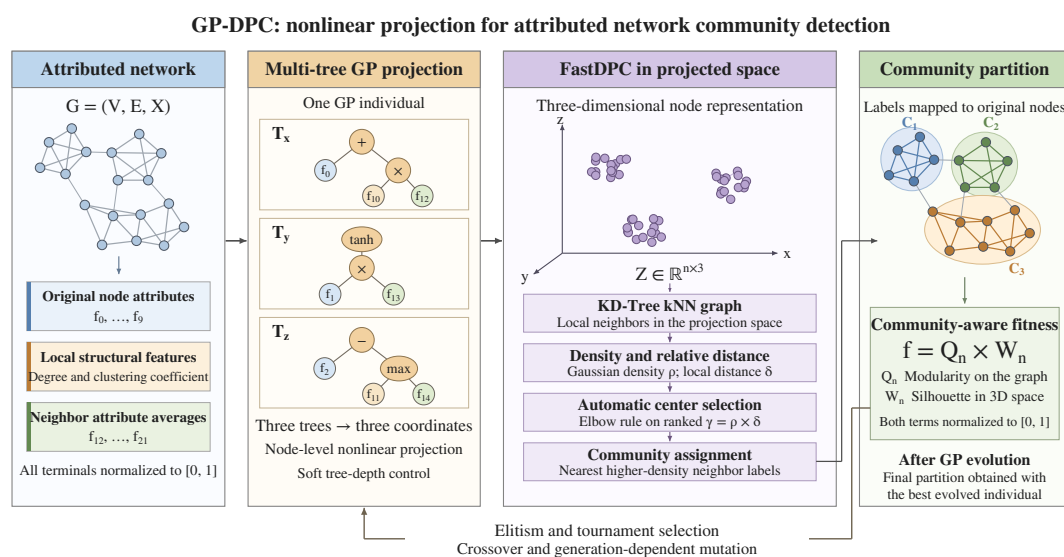


Figure 1. Overall workflow of the GP-DPC framework.

3.2. Multi-Tree GP for Nonlinear Projection Learning

GP-DPC uses a multi-tree GP representation to learn the projection space required by density-based community assignment. Each GP individual consists of three trees, denoted as (T_x, T_y, T_z) . These trees jointly define a node-level projection function from the original attribute space to a compact three-dimensional space. Given the attribute vector $\mathbf{x}_v \in \mathbb{R}^d$ of node v , the three trees produce three scalar outputs:

$$f_x(\mathbf{x}_v), \quad f_y(\mathbf{x}_v), \quad f_z(\mathbf{x}_v), \quad (2)$$

which form the projected representation

$$\mathbf{z}_v = (f_x(\mathbf{x}_v), f_y(\mathbf{x}_v), f_z(\mathbf{x}_v))^T. \quad (3)$$

Algorithm 1 GP-DPC: projection-guided community detection

Require: Attributed graph $\mathcal{G} = (V, E, \mathbf{X})$ with n nodes; GP parameters: population size N_p , maximum generations $G_{\max}$, elite size E , crossover rate C_X , initial mutation probability M_{init} , final mutation probability M_{final} , and soft tree-depth control parameter $D_{\max}$; FastDPC parameter: number of nearest neighbors k ; Terminal set $\mathcal{T}$ and function set $\mathcal{F}$.

Ensure: Community partition $\mathcal{C} = \{C_1, C_2, \dots, C_K\}$.

1: **Initialization**

2: Generate an initial population $\mathcal{P}$ with N_p individuals.

3: Each individual $I \in \mathcal{P}$ contains three GP trees (T_x, T_y, T_z) , which encode the three coordinate functions of the projection space.

4: Use $D_{\max}$ as a soft reference depth during tree generation to guide the population toward compact tree structures.

5: **for** each individual $I \in \mathcal{P}$ **do**

6: Evaluate its fitness $f(I)$ using the community-aware fitness function.

7: **end for**

8: **Evolutionary Projection Learning**

9: **for** $gen = 1$ **to** $G_{\max}$ **do**

10: Update the generation-dependent mutation probability as

$$M_{\text{gen}} = M_{\text{init}} - (M_{\text{init}} - M_{\text{final}}) \frac{gen}{G_{\max}},$$

where $M_{\text{init}} = 0.6$ and $M_{\text{final}} = 0.1$.

11: Select the top E individuals from $\mathcal{P}$ as elites and copy them into the new population $\mathcal{P}'$.

12: **while** $|\mathcal{P}'| < N_p$ **do**

13: Select two parents I_a and I_b by tournament selection.

14: Generate offspring by applying subtree crossover to the corresponding trees of I_a and I_b with probability C_X .

15: Apply mutation to each offspring individual with probability M_{gen} .

16: If mutation is triggered, randomly select one of the three GP trees and apply either uniform subtree mutation or node-replacement mutation with equal probability.

17: During crossover and mutation, $D_{\max}$ is used as a soft reference depth rather than a strict rejection threshold; offspring whose realized tree depth exceeds $D_{\max}$ are not automatically discarded and remain eligible for fitness evaluation and subsequent evolutionary selection.

18: Add the generated offspring to $\mathcal{P}'$ until $|\mathcal{P}'| = N_p$.

19: **end while**

20: Replace $\mathcal{P}$ with $\mathcal{P}'$.

21: **for** each individual $I \in \mathcal{P}$ **do**

22: Evaluate its fitness $f(I)$ using the community-aware fitness function.

23: **end for**

24: **end for**

25: **Final Community Assignment**

26: Select the best individual $I^* = \arg \max_{I \in \mathcal{P}} f(I)$.

27: Compile the three GP trees of I^* into coordinate functions (F_x, F_y, F_z) .

28: **for** each node $v \in V$ **do**

29: Compute its projected representation:

$$\mathbf{z}_v = (F_x(\mathbf{x}_v), F_y(\mathbf{x}_v), F_z(\mathbf{x}_v))^{\top}.$$

30: **end for**

31: Construct the projection matrix $\mathbf{Z} \in \mathbb{R}^{n \times 3}$.

32: Apply FastDPC to $\mathbf{Z}$ to obtain the community labels.

33: Map the labels of projected points back to the original nodes in V .

34: **return** $\mathcal{C}$

The projected points are then used by FastDPC to generate a community partition. In this way, GP does not learn a pairwise similarity function. Instead, it learns reusable node-level coordinate functions that can be applied to

all nodes without evaluating every node pair.

The three trees in an individual share the same terminal set and function set, but they are allowed to evolve different expressions. Selection is performed at the individual level, since the quality of a projection depends on the three coordinate functions as a whole. Crossover and mutation are applied to the corresponding tree components, so that each coordinate function can be refined while the individual still represents a complete projection model. The fitness of an individual is computed after applying FastDPC in the learned projection space. Therefore, the evolutionary search is guided by the quality of the resulting community partition rather than by the reconstruction of the original attributes.

The function set includes arithmetic operators $(+, -, \times, \div)$, protected division, and nonlinear operators such as max, min, sigmoid, and tanh. These operators allow GP to construct nonlinear combinations of node attributes and local structural statistics. Each tree takes a single node attribute vector $\mathbf{x}_v$ as input and outputs one scalar coordinate.

The terminal set contains node-level features rather than pairwise similarity values. As summarized in Table 1, the terminals include original node attributes, normalized degree, local clustering coefficient, and neighbor-aggregated attributes. This design supports attributed network community detection because it allows the projection to use both attribute information and local topological context. All terminal features are normalized to $[0, 1]$ before GP evolution to reduce scale imbalance among different feature types.

Table 1. Terminal set used in the multi-tree GP representation.

Variable	Feature	Calculation Method	Range
f_0-f_9	Original node attributes	Normalized raw attributes associated with each node. In the LFR experiments, these attributes are generated from community centers with Gaussian noise; in real attributed networks, they correspond to application-specific node features.	$[0, 1]$
f_{10}	Normalized degree	Degree of the node divided by the maximum degree in the graph.	$[0, 1]$
f_{11}	Clustering coefficient	Local clustering coefficient, computed as the ratio of existing edges among the neighbors of a node to all possible edges among them.	$[0, 1]$
$f_{12}-f_{21}$	Neighbor attribute average	Element-wise average of the original attributes of neighboring nodes. If a node has no valid neighbor attribute value, its own attribute value is used as a fallback.	$[0, 1]$

This node-level representation distinguishes GP-DPC from conventional GP-based similarity learning. Conventional designs often evolve a similarity or distance function and then apply it to all node pairs, which requires a full pairwise evaluation. GP-DPC avoids this step by evolving coordinate functions for individual nodes. After the projection functions are learned, the community assignment is performed in the projected space by FastDPC. This design preserves the adaptive feature construction ability of GP while reducing the dependence on exhaustive pairwise distance computation.

Besides, to control structural complexity during evolution, the GP framework employs a soft tree-depth control parameter $D_{\max}$. Unlike a strict depth constraint, $D_{\max}$ defines a preferred tree-complexity range rather than an absolute upper bound. During initialization and genetic variation, tree generation is guided toward this reference depth to alleviate excessive structural growth. However, individuals whose realized tree depth slightly exceeds $D_{\max}$ are not directly removed; instead, they remain eligible for fitness evaluation and evolutionary selection. Therefore, the final tree complexity is adaptively determined by the interaction between the depth control mechanism and the fitness optimization process. For an individual, the realized aggregate tree depth is defined as the sum of the depths of its three GP trees, i.e., $D_{\text{agg}} = D(T_x) + D(T_y) + D(T_z)$. This quantity is used to monitor the structural complexity of the evolved projection model. This dataset-specific depth-control setting allows the evolutionary search to accommodate different levels of representational complexity while limiting excessive tree growth. Elitism preserves the top E individuals, while the generation-dependent mutation strategy maintains exploration early and enables fine-grained exploitation later. The mutation probability follows a linear decay schedule from M_{init} to M_{final} , as described above. These mechanisms are designed as search-space control strategies to mitigate common

GP pathologies. We acknowledge that a formal schema theorem or convergence-rate guarantee is not established in this work. Instead, empirical verification is provided through population diversity and aggregate tree-depth and tree-size monitoring in the experimental section, where we observe stable diversity and bounded tree growth on the tested datasets.

Specifically, the mutation probability is dynamically adjusted according to the current evolutionary generation. Let M_{init} and M_{final} denote the initial and final mutation probabilities, respectively. At generation gen , the mutation probability is updated as

$$M_{\text{gen}} = M_{\text{init}} - (M_{\text{init}} - M_{\text{final}}) \frac{gen}{G_{\text{max}}}. \quad (4)$$

In the experiments, $M_{\text{init}} = 0.6$ and $M_{\text{final}} = 0.1$ are used. Therefore, the mutation probability gradually decreases during evolution, maintaining stronger exploration in early generations and promoting local refinement in later generations. This linear decay strategy is applied only to the mutation probability; offspring generation and replacement remain governed by the standard fitness-based evolutionary process.

3.3. Node Projection into a Low-Dimensional Space

After the multi-tree GP has evolved the three coordinate functions, each node is projected from the original attribute space into a low-dimensional space. Let $\mathbf{x}_v \in \mathbb{R}^d$ be the attribute vector of node v . The learned functions f_x , f_y , and f_z transform $\mathbf{x}_v$ into a three-dimensional representation:

$$\mathbf{z}_v = (f_x(\mathbf{x}_v), f_y(\mathbf{x}_v), f_z(\mathbf{x}_v))^T \in \mathbb{R}^3. \quad (5)$$

Applying Equation (5) to all nodes produces the projection matrix

$$\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n]^T \in \mathbb{R}^{n \times 3}. \quad (6)$$

This projection serves two purposes. First, it compresses heterogeneous node attributes and local structural statistics into a compact representation that can be directly processed by density-based clustering. Second, because the projection is learned by GP under a community-aware fitness function, the resulting space is expected to reflect both attribute-level similarity and graph-level community structure. Nodes that are likely to belong to the same community should be mapped closer to each other, while nodes from different communities should be more separable in the projected space.

For a single node, evaluating the three GP trees costs $O(L)$, where L denotes the average tree size. Therefore, projecting all n nodes costs $O(nL)$. This avoids the full pairwise evaluation required by conventional GP-based similarity learning and provides the input representation for the subsequent FastDPC-based community assignment. Figure 2 gives an illustrative example of the nonlinear node projection process.

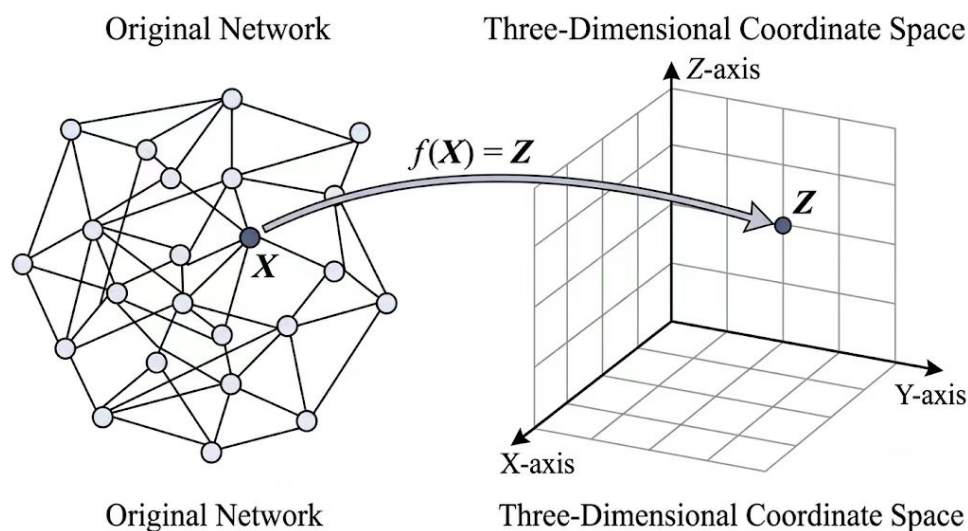


Figure 2. Illustration of nonlinear node projection into a low-dimensional space.

3.4. FastDPC-Based Community Assignment

After the nodes are projected into the learned low-dimensional space, GP-DPC applies FastDPC to generate the community partition. Algorithm 2 gives its main procedure. The input of this stage is the projection matrix

$$\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_n]^\top \in \mathbb{R}^{n \times 3}, \quad (7)$$

where each row $\mathbf{z}_i$ corresponds to the original node v_i . This one-to-one correspondence allows the labels obtained in the projection space to be directly assigned back to the original graph nodes. The role of FastDPC is therefore to identify density-based groups in the learned representation space, while the structural quality of these groups is further considered through the fitness function defined in the next subsection.

To avoid constructing a full pairwise distance matrix, GP-DPC builds a k -nearest neighbor graph in the projection space. Since the projected representation is low-dimensional, a KD-Tree index is used to query the nearest neighbors efficiently. For each projected point $\mathbf{z}_i$, we store its neighbor set $\mathcal{N}_i$ and the corresponding Euclidean distances d_{ij} , where $j \in \mathcal{N}_i$. In the experiments, k is set to 10 according to the parameter sensitivity analysis. This setting provides a local neighborhood large enough for density estimation while keeping the search cost low.

The local density of each point is estimated with a Gaussian kernel over its k nearest neighbors. Instead of using a fixed cutoff distance, GP-DPC computes an adaptive scale parameter from the neighbor distances:

$$\sigma = \text{median} \left(\bigcup_{i=1}^n \{d_{ij} \mid j \in \mathcal{N}_i\} \right) + \epsilon, \quad (8)$$

$$\rho_i = \sum_{j \in \mathcal{N}_i} \exp \left(- \left(\frac{d_{ij}}{\sigma} \right)^2 \right), \quad (9)$$

where ϵ is a small positive constant used to avoid division by zero. The Gaussian kernel provides a smooth density estimate and reduces the sensitivity to a manually specified cutoff distance.

For each point, FastDPC also needs the distance δ_i to a point with higher density. The original DPC computes this value by searching over all higher-density points, which requires global pairwise distance information. GP-DPC restricts this search to the local neighbor set $\mathcal{N}_i$:

$$\delta_i = \begin{cases} \max_{j \in \mathcal{N}_i} d_{ij}, & \text{if } \rho_j \leq \rho_i \text{ for all } j \in \mathcal{N}_i, \\ \min_{\substack{j \in \mathcal{N}_i \\ \rho_j > \rho_i}} d_{ij}, & \text{otherwise.} \end{cases} \quad (10)$$

If no higher-density point exists in $\mathcal{N}_i$, point i is treated as a local density peak and its δ_i is set to the farthest neighbor distance within $\mathcal{N}_i$. This local back-off rule avoids an additional global search and keeps the density-peak estimation consistent with the k NN-based approximation.

The decision value of each point is then computed as

$$\gamma_i = \rho_i \times \delta_i. \quad (11)$$

Points with large γ_i values are likely to be local density peaks. GP-DPC sorts all points by γ_i in descending order and applies an elbow rule to determine the number of centers. The selected centers are used as initial community representatives. The remaining points are processed in descending order of ρ_i and assigned to the same label as their nearest higher-density neighbor in $\mathcal{N}_i$. This produces a label vector $\mathbf{c} \in \mathbb{N}^n$ in the projection space.

Finally, because the i -th row of $\mathbf{Z}$ corresponds to node v_i , the label c_i is directly assigned to v_i . The resulting community partition is

$$\mathcal{C} = \{C_1, C_2, \dots, C_K\}, \quad C_k = \{v_i \mid c_i = k\}. \quad (12)$$

Thus, FastDPC provides an efficient assignment mechanism in the GP-learned projection space, while the full GP-DPC framework uses the community-aware fitness function to ensure that the learned projection remains consistent with both attribute separability and graph structure.

A theoretical concern regarding approximate kNN in the projection space is the potential distortion of local neighborhood structures. Strictly deriving a Johnson-Lindenstrauss type bound for the GP-evolved projection is intractable, as the GP trees are nonlinear and dynamically changing during evolution. More importantly, GP-DPC is designed as a discriminative projection that maximizes community separability, rather than an isometric

embedding that preserves original distances. The projection therefore intentionally alters local neighborhoods to pull same-community nodes together and push different-community nodes apart. The Gaussian kernel used in density estimation further reduces sensitivity to individual neighbor misalignments by averaging over the entire kNN set. Detailed empirical validation of these properties is provided in the experimental section.

Algorithm 2 Adaptive FastDPC in the projected 3D space

Require: Projected coordinate matrix $\mathbf{Z} \in \mathbb{R}^{n \times 3}$ (row i corresponds to original node v_i); Number of nearest neighbors $k = 10$.

Ensure: Community label vector $\mathbf{c} \in \mathbb{N}^n$ (directly maps back to original graph nodes).

```

1: Step 1: Build  $k$ NN Graph
2: Build a KD-Tree index on  $\mathbf{Z}$ .
3: for  $i = 1$  to  $n$  do
4:   Query  $k$ -nearest neighbors of point  $i$ , store indices in  $\mathcal{N}_i$  and distances in  $\mathcal{D}_i$ .
5: end for
6: Step 2: Compute Adaptive Cutoff Scale
7: Let  $\mathcal{D}_{all} = \bigcup_{i=1}^n \mathcal{D}_i$ .
8: Compute  $\sigma = \text{median}(\mathcal{D}_{all}) + \epsilon$ , where  $\epsilon$  is a small constant (e.g.,  $10^{-6}$ ).
9: Step 3: Estimate Local Density  $\rho$  (Gaussian Kernel)
10: for  $i = 1$  to  $n$  do
11:    $\rho_i = \sum_{j \in \mathcal{N}_i} \exp(-(d_{ij}/\sigma)^2)$ 
12: end for
13: Step 4: Compute Distance to Higher Density  $\delta$  ( $k$ NN-Only Search)
14: for  $i = 1$  to  $n$  do
15:   Let  $\mathcal{H}_i = \{j \mid j \in \mathcal{N}_i \text{ and } \rho_j > \rho_i\}$ .
16:   if  $\mathcal{H}_i \neq \emptyset$  then
17:      $\delta_i = \min_{j \in \mathcal{H}_i} d_{ij}$  (nearest higher-density neighbor)
18:      $\text{nneigh}[i] = \arg \min_{j \in \mathcal{H}_i} d_{ij}$ 
19:   else
20:      $\delta_i = \max_{j \in \mathcal{N}_i} d_{ij}$  (back-off: treat  $i$  as local density peak)
21:      $\text{nneigh}[i] = \arg \max_{j \in \mathcal{N}_i} d_{ij}$ 
22:   end if
23: end for
24: Step 5: Automatic Center Selection (Elbow Rule)
25: Compute decision value  $\gamma_i = \rho_i \times \delta_i$  for all  $i$ .
26: Sort  $\gamma$  in descending order:  $\gamma_{(1)} \geq \gamma_{(2)} \geq \dots \geq \gamma_{(n)}$ .
27: Truncate to  $M = \min(150, \lfloor 2.5\sqrt{n} \rfloor)$ .
28: Find elbow point  $t^* = \arg \max_{t \in [2, M-1]} \text{dist}((t, \gamma_{(t)}), \text{line}((1, \gamma_{(1)}), (M, \gamma_{(M)})))$ .
29: Set number of communities  $K = t^*$ .
30: Select centers  $\{C_1, \dots, C_K\}$  as the top- $K$  points with largest  $\gamma$ .
31: Step 6: Label Assignment (Descending  $\rho$  Traversal)
32: Initialize labels for centers:  $\mathbf{c}_{C_k} = k$ .
33: Sort all points by  $\rho$  in descending order: sorted_idx.
34: for  $i$  in sorted_idx do
35:   if  $\mathbf{c}_i$  is unassigned then
36:      $\mathbf{c}_i = \mathbf{c}_{\text{nneigh}[i]}$  (follow the chain to the nearest higher-density neighbor)
37:   end if
38: end for
39: Step 7: Post-Processing (Optional Refinement)
40: Refine labels using  $k$ -nearest neighbor voting (with  $k = 5$ ) to smooth boundary noise.
41: return  $\mathbf{c}$  (where  $\mathbf{c}[i]$  directly maps back to original graph node  $v_i$ )

```

3.5. Community-Aware Fitness Function

The GP search requires a fitness function to evaluate whether an evolved projection is useful for community detection. In GP-DPC, a candidate individual first maps nodes into the learned projection space, and FastDPC then produces a partition $\mathcal{C}$. The quality of this partition is assessed from two complementary perspectives. Modularity

evaluates whether the partition is consistent with the edge structure of the original graph, while the silhouette coefficient evaluates whether the projected points are well separated in the learned space. Combining these two criteria allows the evolutionary process to favor projections that are meaningful for both graph structure and attribute-driven separability.

3.5.1. Modularity Q

Modularity measures the structural quality of a community partition by comparing the observed density of intra-community edges with the expected density under a random configuration model. It is computed on the original graph $\mathcal{G}$ rather than on the projected point set:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j), \quad (13)$$

where A_{ij} is the adjacency matrix, k_i and k_j are the degrees of nodes i and j , m is the number of edges, and $\delta(c_i, c_j) = 1$ if nodes i and j are assigned to the same community and 0 otherwise. A larger Q indicates that more edges fall inside communities than would be expected by chance. However, modularity alone may favor partitions that are structurally dense but not necessarily well separated in the projected attribute space.

3.5.2. Silhouette Coefficient W

To evaluate the separability of the learned projection, GP-DPC also uses the silhouette coefficient. For a projected point i , the silhouette score is defined as

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}, \quad (14)$$

where $a(i)$ is the average distance from point i to other points in the same community, and $b(i)$ is the average distance from point i to points in the nearest different community. In this framework, these distances are measured in the GP-evolved projection space. The average silhouette value over all points is denoted as W . A larger W means that the partition has stronger intra-community compactness and clearer inter-community separation in the learned representation.

3.5.3. Combined Fitness

Since Q and W are defined on different spaces, they are normalized before combination. Modularity is clipped to the interval $[0, 1]$, and the silhouette coefficient, whose range is $[-1, 1]$, is linearly mapped to $[0, 1]$:

$$Q_n = \begin{cases} 0, & Q \leq 0, \\ Q, & 0 < Q < 1, \\ 1, & Q \geq 1, \end{cases} \quad W_n = \frac{W + 1}{2}. \quad (15)$$

The final fitness is defined as

$$f = Q_n \times W_n. \quad (16)$$

The multiplicative form is used to encourage a balanced optimization of the two criteria. If a projection produces high modularity but poor spatial separability, or clear spatial clusters that are inconsistent with the original graph structure, one of the two normalized terms will be small and the overall fitness will be penalized. This prevents the evolutionary search from relying solely on either topology or projected-space compactness.

For comparison, we also evaluate alternative fitness formulations in the parameter sensitivity analysis, including Q_n only, W_n only, and a weighted sum $f = \alpha Q_n + (1 - \alpha) W_n$. These variants help examine whether the proposed multiplicative form provides a better balance between structural community quality and projection-space separability. Furthermore, the behavior of the proposed multiplicative fitness function is empirically examined in the experimental section through objective-aggregation visualization and evolutionary-dynamics analysis. The results provide empirical evidence that the multiplicative form encourages balanced optimization of both modularity and silhouette, effectively discouraging solutions that excel in only one objective.

4. Experimental Study

4.1. Experiment Configuration

The experiments are designed to evaluate GP-DPC from multiple perspectives, including detection quality, scalability, component contribution, robustness to graph perturbations, convergence behavior of the evolutionary search, and the effectiveness of the projection space learned by the multi-tree GP. These analyses collectively demonstrate that GP-DPC achieves competitive clustering accuracy while producing effective projections with limited observed linear redundancy guided by the proposed fitness function. Moreover, the evolutionary dynamics and fitness behavior observed on the tested datasets indicate that the observed evolutionary dynamics show no obvious premature stagnation under the tested settings. To this end, we use three types of benchmark data. The Handl synthetic datasets are used to examine whether the learned projection can preserve separable group structures under controlled data distributions. The LFR benchmark networks are used to evaluate GP-DPC on graph-structured data with heterogeneous degree distributions and community size distributions. In addition, real-world attributed networks such as Cora are included to validate the practical applicability of GP-DPC in engineering-oriented scenarios where node attributes and graph topology coexist with complex noise patterns. This three-tier design allows us to test both the projection ability of GP-DPC and its applicability to attributed network community detection across synthetic and real-world settings.

4.1.1. Datasets and Graph Construction

Following previous work [18], we first use the synthetic generators introduced by Handl et al. These generators can produce datasets with different cluster shapes, feature dimensions, sample sizes, and numbers of clusters. The Gaussian generator creates clusters with diverse shapes in low-dimensional settings, whereas the elliptical generator produces non-hyper-spherical clusters and is therefore more challenging for projection-based methods. Although these datasets are not graph datasets by origin, they provide controlled ground-truth labels for evaluating projection quality and density-based assignment. To use them for community detection, each data point is treated as a node, the original feature vector is used as the node attribute, and the edge set is constructed from feature-space neighborhood relations. The ground-truth labels are used only for evaluation and are not used during graph construction or GP evolution.

Six Handl-based datasets are generated to examine the behavior of GP-DPC under different scales and numbers of communities:

- 10 communities with 500 nodes, denoted as 10c500d;
- 10 communities with 1000 nodes, denoted as 10c1000d;
- 15 communities with 1000 nodes, denoted as 15c1000d;
- 24 communities with 1800 nodes, denoted as 24c1800d;
- 40 communities with 5000 nodes, denoted as 40c5000d;
- 80 communities with 10000 nodes, denoted as 80c10000d.

Although the Handl and LFR benchmarks both include a setting denoted as 10c500d, they are generated independently: the former is a Handl point dataset subsequently converted into an attributed graph, whereas the latter is a network generated directly using the LFR benchmark. We further use the Lancichinetti–Fortunato–Radicchi (LFR) benchmark to evaluate GP-DPC on graph-structured data. LFR networks are widely used for community detection because they contain heterogeneous degree distributions and community size distributions. The mixing parameter controls the proportion of edges connecting nodes from different communities; therefore, larger mixing values correspond to more difficult community detection tasks. In the LFR experiments, each node is associated with an attribute vector. The original attributes, normalized degree, local clustering coefficient, and neighbor-aggregated attributes form the terminal features used by GP-DPC. The ground-truth community labels generated by LFR are used only for performance evaluation.

We also evaluate GP-DPC on five real-world attributed networks, namely Cora, Citeseer, PubMed, DBLP, and ACM. Cora and Citeseer are citation networks with bag-of-words attributes and research area labels [29,30]. PubMed is a larger citation network of medical articles with three publication categories [31]. DBLP is a co-authorship network where author nodes are described by publication keywords and labeled by research communities [32]. ACM is a computer science citation network with titles, abstracts, and author keywords as attributes [33], and conference or journal venues as labels. These datasets differ from synthetic benchmarks in noise levels, feature distributions, and structural complexity, and are included to assess the empirical applicability of GP-DPC beyond synthetic data and to demonstrate its practical utility in engineering-oriented scenarios with heterogeneous and noisy attributes.

4.1.2. Compared Methods

GP-DPC is compared with several representative community detection methods, including topology-based and density-based approaches. Louvain and Leiden are selected as topology-driven baselines, while DPC, FastDPC, and Random Projection + FastDPC are employed to evaluate the effectiveness of density-based clustering and the contribution of the proposed nonlinear projection. Normalized Spectral Clustering, Graph AutoEncoder (GAE), and Variational Graph AutoEncoder (VGAE) are further introduced as representative spectral and graph representation learning methods for additional comparison.

The compared methods are summarized as follows:

- Louvain [19]: a modularity-maximization method widely used for graph community detection.
- Leiden [22]: an improved variant of Louvain that produces better-connected communities.
- DPC: the original Density Peaks Clustering algorithm applied to the node representation.
- FastDPC: the approximate k NN-based DPC variant applied without GP-based projection learning.
- Random Projection + FastDPC: a projection-based baseline in which the GP-learned projection is replaced by a random projection with the same output dimensionality.
- Normalized Spectral Clustering [34]: a spectral clustering method based on the eigenvectors of the normalized graph Laplacian for node representation learning.
- GAE [35]: a graph representation learning method that learns low-dimensional node embeddings through graph reconstruction.
- VGAE [36]: a probabilistic extension of GAE that learns latent node representations using variational inference.

4.1.3. Ablation Settings

To examine the contribution of the main components of GP-DPC, we design four ablation variants:

- Without GP projection: FastDPC is directly applied to the raw node features. This variant tests whether nonlinear projection learning improves the representation used for density-based assignment.
- Random projection: the GP-learned projection is replaced by a random projection. This variant tests whether GP provides a more task-oriented projection than a generic low-dimensional mapping.
- Original DPC instead of FastDPC: the learned GP projection is retained, but FastDPC is replaced by the original DPC algorithm. This variant evaluates the effect of the approximate k NN acceleration.
- GP-based pairwise similarity learning: GP is used to learn a pairwise distance or similarity function without the proposed node-level projection. This variant is included to compare GP-DPC with a conventional GP-based design and to examine the accuracy–efficiency trade-off.

4.1.4. Parameter Settings and Sensitivity Analysis

Unless otherwise specified, the same default parameter settings are used across datasets to ensure a fair comparison. The maximum number of generations $G_{\max}$ is selected according to the parameter sensitivity analysis and is kept fixed across all datasets. In contrast, the soft tree-depth control parameter $D_{\max}$ is treated separately because the representational complexity required by GP may vary across datasets. The sensitivity analysis is used to identify a practical reference range for tree depth rather than a single globally fixed value. Guided by this range, $D_{\max}$ is selected separately for each dataset in the final experiments, while the realized tree depth is adaptively determined by the evolutionary process under the corresponding soft depth-control mechanism rather than being strictly fixed by $D_{\max}$. The same soft-control rule is used for all datasets: $D_{\max}$ biases tree generation and variation toward the reference depth, but offspring exceeding this value are not automatically removed and remain subject to the same fitness-based selection process. The sensitivity analysis further examines the influence of key parameters, including the number of nearest neighbors k , the GP population size N_p , the maximum number of generations $G_{\max}$, the soft tree-depth control parameter $D_{\max}$, the crossover rate C_X , the mutation rate M , and the elite size E . We also compare different fitness formulations, including the proposed multiplicative form $f = Q_n \times W_n$, the weighted-sum form $f = \alpha Q_n + (1 - \alpha)W_n$, and the single-metric variants using only Q_n or only W_n . The purpose of this analysis is to identify stable parameter ranges rather than only the best-performing configuration.

4.1.5. Parameter Sensitivity Analysis

To assess the stability of GP-DPC, we analyze the influence of key parameters from both the GP and FastDPC components:

- the number of nearest neighbors k used in FastDPC;
- the GP population size N_p ;

- the maximum number of generations $G_{\max}$;
- the soft tree-depth control parameter $D_{\max}$;
- the crossover rate C_X and mutation rate M ;
- the elite size E retained in each generation;
- the fitness formulation, including the proposed multiplicative form $f = Q_n \times W_n$, the weighted-sum form $f = \alpha Q_n + (1 - \alpha) W_n$, and the single-metric variants using only Q_n or only W_n .

Except for the dataset-specific tree-complexity setting described above, the remaining default parameters, including $G_{\max}$, are kept fixed across datasets. The sensitivity analysis is used to identify stable parameter settings and reference ranges rather than only the best-performing single configuration.

4.1.6. Projection Redundancy Analysis

To investigate whether the three GP-generated projection axes contain redundant information, we analyze the covariance matrix, pairwise correlation coefficients, pairwise mutual information, and effective rank of the projected coordinates. These complementary statistics characterize the linear and nonlinear dependencies among the three projection axes and assess whether the learned representation exhibits substantial redundancy or dimensional collapse.

4.2. Evaluation Metrics

The performance of GP-DPC is evaluated from three aspects: clustering accuracy, graph-structural quality, and projection characteristics. For datasets with ground-truth labels, external validation metrics are employed to measure the consistency between the detected communities and the reference partitions. Following previous studies [18], Compactness, Sparsity, Separability, Connectedness, and Adjusted Rand Index (ARI) are adopted as clustering evaluation metrics. Compactness evaluates the cohesion of nodes within the same community, while Sparsity provides a complementary characterization of the dispersion of samples within the detected communities. Separability measures the distinction between different communities in the learned representation space, and Connectedness reflects the local consistency of community assignments. ARI measures the agreement between the detected partition and the reference partition after correcting for random agreement, where values closer to 1 indicate higher clustering consistency. Normalized Mutual Information (NMI) is further considered for labeled datasets. It quantifies the mutual information between the detected communities and the reference labels and provides a complementary measurement to ARI.

For evaluating graph-structural properties, modularity and conductance are reported. Modularity measures the difference between the observed intra-community connections and those expected in a random graph with the same degree distribution. Higher modularity values indicate stronger community structures. Conductance evaluates the connectivity between a community and the remaining graph, where lower values represent clearer community boundaries.

Since GP-DPC learns a three-dimensional nonlinear projection through multiple GP trees, the properties of the generated embedding space are further investigated. Let $\mathcal{N}_k^X(i)$ and $\mathcal{N}_k^Z(i)$ denote the k -nearest-neighbor sets of node i in the original attribute space $\mathbf{X}$ and the learned projection space $\mathbf{Z}$, respectively. The Neighborhood Preservation Ratio is defined as:

$$\text{NPR@}k = \frac{1}{n} \sum_{i=1}^n \frac{|\mathcal{N}_k^X(i) \cap \mathcal{N}_k^Z(i)|}{k} \quad (17)$$

where $k = 10$ is used in all experiments. Trustworthiness@10 is the standard rank-based measure that penalizes neighbors introduced in the evaluated space that are not close neighbors in the corresponding reference space. Graph-NPR@10 measures the average overlap between the graph-based reference neighborhoods and the ten nearest neighbors in the projection space. LCC-MAE measures the mean absolute difference between the node-wise local clustering coefficients of the reference graph and those of the k NN graph constructed from the projected coordinates, while LCC-Corr is the Pearson correlation between these two sets of coefficients. Edge Reconstruction AUC uses negative pairwise distance in the projection space as an edge score and evaluates its ability to distinguish graph edges from non-edges. The NPR variants reported in the noise experiments use different reference spaces. Projection-NPR@10 in Table 2 compares the clean projection $\mathbf{Z}$ with its noise-perturbed version $\mathbf{Z}_\eta$, whereas Attribute-NPR@10 in Table 3 compares the original attribute space $\mathbf{X}$ with the projection $\mathbf{Z}_\eta^X$ obtained from noise-perturbed attributes. Consequently, Projection-NPR@10 equals 1.0000 when $\eta = 0$, while Attribute-NPR@10 at $\eta = 0$ reduces to NPR@10 in Table 4. The covariance matrix and correlation coefficients among different projection dimensions are calculated to measure the dependency between axes. Effective Rank is employed to evaluate the dimensional diversity of the learned representation and provide an indicator of whether the projection suffers from

substantial dimensional collapse. The computational efficiency of GP-DPC is evaluated by recording the running time and memory consumption during the optimization process.

Table 2. Projection neighborhood distortion under varying noise levels on the 24c1800d dataset.

Noise Level	Projection-NPR@10	Trustworthiness@10	K Error	ARI	NMI
0.00	1.0000	1.0000	0	1.0000	1.0000
0.05	0.2741	0.9929	1	0.9890	0.9955
0.10	0.1747	0.9893	1	0.9912	0.9950
0.20	0.1391	0.9834	20	0.1084	0.4661
0.30	0.1079	0.9615	18	0.1233	0.4978

Table 3. Attribute noise sensitivity on the 24c1800d dataset.

Noise Level	Attribute-NPR@10	K Error	ARI	NMI
0.00	0.2314	0	1.0000	1.0000
0.05	0.1923	19	0.1538	0.5646
0.10	0.1891	18	0.2464	0.6575
0.20	0.1804	19	0.1433	0.5487
0.30	0.1598	20	0.1699	0.5511

Table 4. Projection quality metrics on Handl datasets.

Dataset	NPR@10	Trust@10	Graph-NPR@10	LCC-MAE	LCC-Corr	Edge Recon AUC
10c500d	0.2704	0.9671	0.3638	0.1576	0.2085	0.9803
10c1000d	0.1664	0.9612	0.2120	0.2687	0.0415	0.9697
15c1000d	0.2446	0.9772	0.3851	0.1252	0.2614	0.9848
24c1800d	0.2314	0.9877	0.3799	0.1192	0.2395	0.9926

4.3. Parameter Sensitivity Analysis

In this experiment, we conduct sensitivity analysis on the following parameters: the number of nearest neighbors k in the k NN search, the truncation parameter M in the elbow rule, the GP population size N_p , the maximum number of generations $G_{\max}$, the soft tree-depth control parameter $D_{\max}$, the crossover rate, the mutation rate, the elite number E , and the fitness function formulation. The sensitivity analysis results for these parameters are illustrated in Figure 3. Using ARI as the evaluation metric and the 24c1800d dataset ($n = 1800$, true $K = 24$) as the experimental basis, the sensitivity analysis is used to identify effective reference ranges rather than to impose a single globally fixed configuration. The results indicate favorable performance when $k = 10$, the population size is greater than 70, the reference per-tree depth setting is around 5, the crossover rate is approximately 0.95, the mutation rate is around 0.5, and $E = 10$ within the tested ranges. The depth sensitivity result is used to identify a practical reference range rather than to impose a globally fixed setting, and the final $D_{\max}$ is therefore selected on a dataset-specific basis with reference to this sensitivity analysis. Since the structural complexity required by different datasets may vary, the soft tree-depth control parameter $D_{\max}$ is selected on a dataset-specific basis in the final experiments, guided by the stable range identified in the sensitivity analysis. In contrast, the maximum number of generations $G_{\max}$ is fixed according to the value selected from the parameter sensitivity analysis and is kept unchanged across datasets. It should be noted that the sensitivity experiment varies individual parameters under a controlled reference configuration, whereas the final experiments use dataset-specific $D_{\max}$ settings, while the realized tree complexity emerges from the evolutionary process under the corresponding soft depth-control setting. Therefore, the effective tree complexity observed in the final runs is not expected to coincide exactly with the single-factor setting that yields favorable performance in the sensitivity analysis.

Since the sensitivity analysis is conducted on the 24c1800d dataset, the resulting tree-depth setting is treated as a practical reference rather than a universally optimal value for all datasets. To account for differences in network scale and structural complexity, we therefore make a limited dataset-specific adjustment to $D_{\max}$ in the final experiments. Specifically, $D_{\max} = 5$ is used for the Handl datasets 10c500d, 10c1000d, 15c1000d, 24c1800d, and 40c5000d, as well as for the LFR datasets 10c500d and 20c1000d. In contrast, $D_{\max} = 8$ is used for the

larger Handl 80c10000d dataset and the five real-world attributed networks, namely Cora, Citeseer, PubMed, DBLP, and ACM. These settings are guided by the sensitivity trends identified on 24c1800d, with a modest extension to $D_{\max} = 8$ for larger or more structurally heterogeneous networks. Here, $D_{\max}$ serves as a soft depth-control parameter rather than a strict cutoff, and the realized tree depth may vary during evolution.

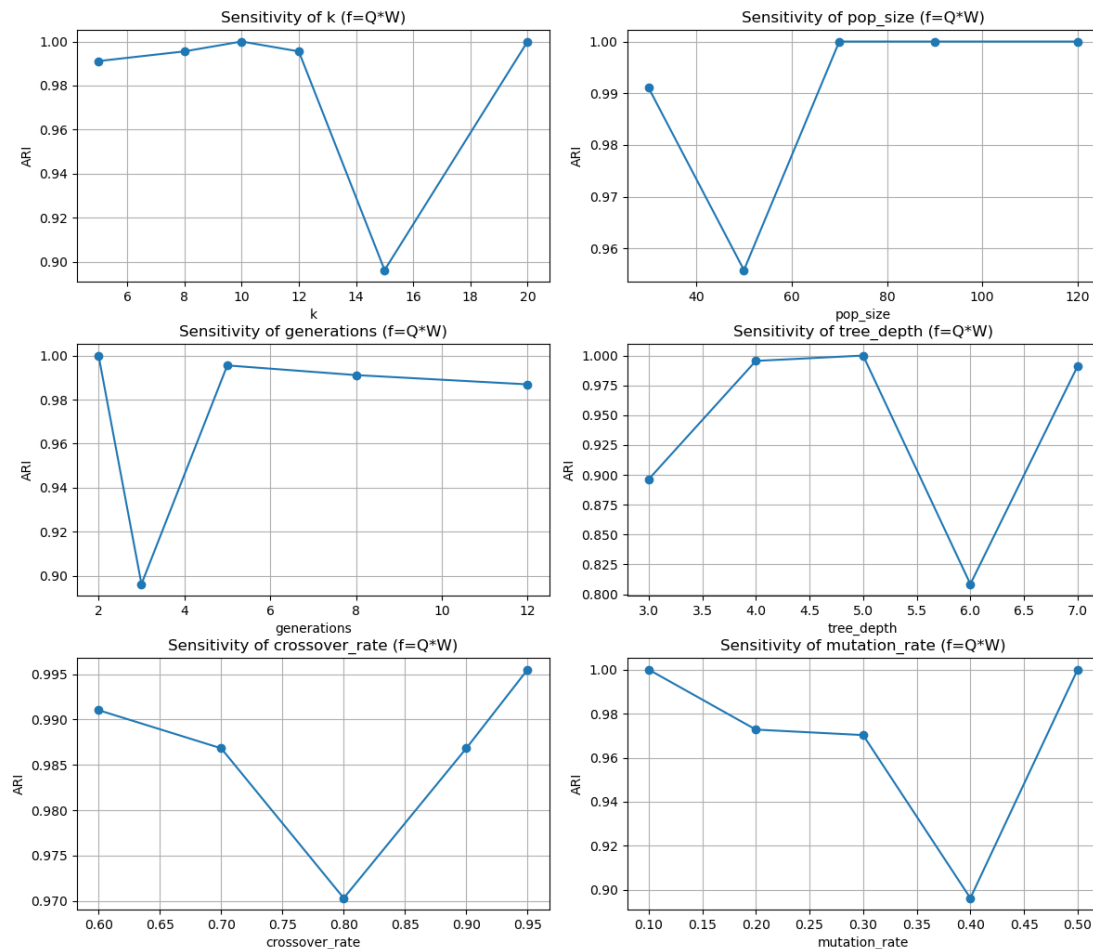


Figure 3. Parameter sensitivity analysis of key hyperparameters.

For the truncation parameter M , which determines how many top- γ candidates are considered in the elbow rule, our experiments reveal a clear threshold effect in Figure 4. When $M < 60$, the estimated number of communities $\hat{K}$ is consistently below the ground-truth value of 24, because some potential density peaks may be excluded from the candidate pool. For $M > 70$, $\hat{K}$ stabilizes near the ground-truth value, typically between 24 and 25, and remains stable throughout the remainder of the tested range. The corresponding ARI values rise from approximately 0.5 to near 1.0 once M exceeds 70, and maintain a high level thereafter. Only a minor one-community overestimation is occasionally observed, indicating that the elbow detector remains relatively stable to the inclusion of additional low- γ candidates. The default heuristic $M = \min(150, \lfloor 2.5\sqrt{n} \rfloor)$ yields approximately 106 for $n = 1800$, which lies safely in the stable high-performance region where both $\hat{K}$ and ARI remain close to their favorable values. This supports the practical reliability of the heuristic design within the tested range.

In addition, to determine the most appropriate fitness function, we analyze four formulations: using only Q , using only W , the weighted sum $\alpha Q + (1 - \alpha)W$, and the multiplicative form $Q \times W$. Experimental analysis in Figure 5 shows that, under the tested setting, the first three formulations exhibit earlier stagnation or inferior search behavior compared with the multiplicative formulation. When a single metric is used as the fitness function, the evolutionary process is guided solely by that objective, which may reduce the incentive to improve the complementary criterion and lead to less balanced search behavior. The weighted-sum formulation is also sensitive to the choice of α ; as α approaches 1, the contribution of W becomes progressively smaller, causing the search to be increasingly dominated by Q . The ARI performance as a function of the weight coefficient is shown in the figure below. By contrast, the multiplicative form $Q \times W$ imposes a strong penalty property: when either metric approaches zero, the overall fitness rapidly decreases, encouraging the evolutionary search to improve both objectives simultaneously. This mechanism penalizes solutions dominated by a single objective and encourages the

evolutionary search to consider both objectives simultaneously. However, it does not guarantee the absence of local optima. Furthermore, to avoid the scenario where a single metric dominates the product by tending toward infinity or zero, we normalize both fitness components to the range $[0, 1]$, ensuring that neither metric unduly influences the final fitness value.

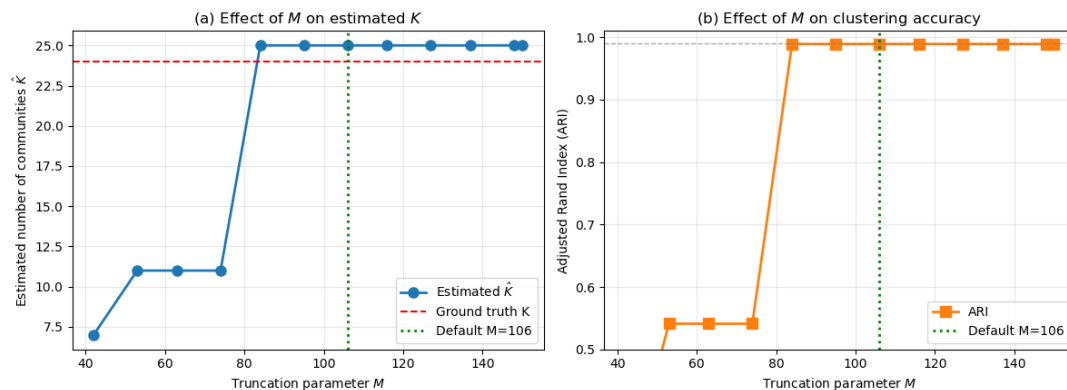


Figure 4. Parameter sensitivity analysis of M .

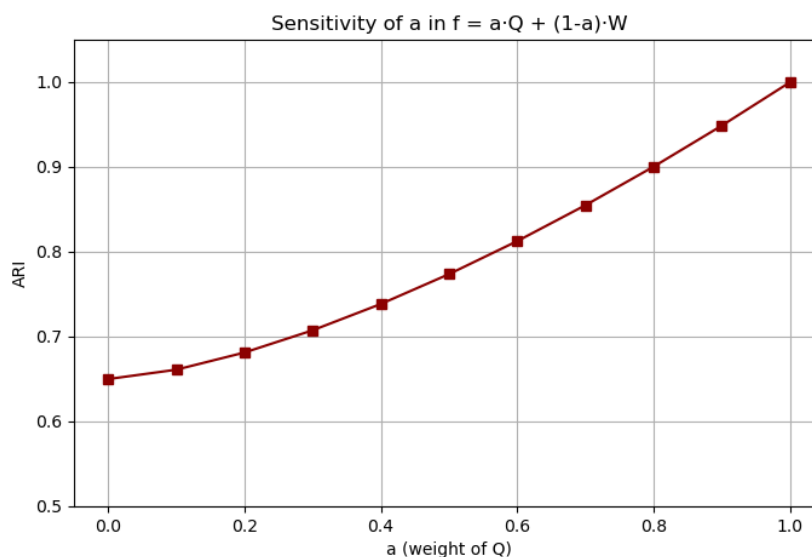


Figure 5. Parameter sensitivity analysis of the weight of Q .

5. Results and Discussion

5.1. Performance on Synthetic Clustering Benchmarks

As shown in Table 5, GP-DPC identifies the expected number of communities on the first four datasets and on 40c5000d. On the first four datasets, the ARI and NMI reach 1.0000, indicating that the detected partitions are fully consistent with the reference labels. On the 40c5000d dataset, the ARI decreases to 0.8280 and the NMI to 0.8719, while the number of communities is correctly identified. On the largest dataset, 80c10000d, the ARI drops to 0.6475 and the NMI to 0.7464, with the estimated number of communities deviating from the ground truth by four. This progressive decline reflects the increasing difficulty of clustering as both the number of nodes and the number of communities grow. Nevertheless, the results demonstrate that GP-DPC maintains a reasonable level of partition accuracy even under substantially larger scales. The modularity values remain high across all datasets, exceeding 0.91 even on the largest 80c10000d dataset, showing that the detected communities are structurally coherent on the constructed graphs. Conductance is 0 on the first four datasets and remains close to 0 on 40c5000d and 80c10000d, indicating weak inter-community connectivity in the detected partitions.

The changes in the clustering-oriented metrics also reflect the increasing difficulty of the datasets. Connectedness remains at a high level, which means that locally related nodes are generally assigned to the same community. Separability gradually decreases as the dataset scale and the number of communities increase. This trend is expected because larger datasets with more communities usually contain more complex boundaries and more ambiguous

neighboring regions in the feature space. Therefore, the decrease in Separability and ARI on larger datasets should be interpreted together with NMI, modularity, and conductance, rather than as an isolated failure of the method.

Table 5. GP-DPC performance on Handl datasets.

Scale	K	Conn.	Spar.	Sep.	ARI	NMI	Modularity	Conductance
10c500d	10	95.3107	0.1184	0.7434	1.0000	1.0000	0.9000	0.0000
10c1000d	10	98.3460	0.1109	0.7232	1.0000	1.0000	0.9000	0.0000
15c1000d	15	97.6587	0.1062	0.6421	1.0000	1.0000	0.9333	0.0000
24c1800d	24	98.3947	0.1080	0.5114	1.0000	1.0000	0.9583	0.0000
40c5000d	40	99.2837	0.1039	0.2980	0.8280	0.8719	0.9718	0.0001
80c10000d	76	99.5246	0.1239	0.2751	0.6475	0.7464	0.9171	0.0012

For the Handl datasets, the original data are point sets in a feature space rather than graph-structured networks. To make them suitable for attributed community detection, we construct an attributed graph $\mathcal{G} = (V, E, \mathbf{X})$ before applying GP-DPC. Each data point is treated as a node $v \in V$, and its feature vector is used as the node attribute $\mathbf{x}_v$. Edges are established according to neighborhood relations in the feature space. Specifically, two nodes are connected if one lies within the ϵ -neighborhood of the other, where ϵ is selected as the minimum threshold that keeps the constructed graph connected. The reference labels are used only for external evaluation and are not used during graph construction, GP evolution, or FastDPC assignment. Therefore, the modularity and conductance values reported in Table 5 are computed on the constructed graph using the partitions produced by GP-DPC.

Beyond the clustering accuracy results, we further analyze the evolutionary dynamics and projection quality of GP-DPC on the Handl datasets. Table 6 reports the convergence behavior, tree complexity, and projection-space properties across the five Handl benchmarks. Here, Mean Agg. Depth denotes the mean realized aggregate depth of the evolved individuals, where the aggregate depth of an individual is the sum of the depths of its three GP trees (T_x, T_y, T_z). This metric reflects the realized tree complexity under the dataset-specific $D_{\max}$ setting and should not be interpreted as the dataset-specific depth-control parameter itself.

Table 6. Evolutionary dynamics and projection quality metrics on Handl datasets.

Data	Stag.	Div.	Mean Agg. Depth	Size	Mean Corr	Max Corr	Pairwise MI	Eff. Rank
10c500d	8	0.8733	14.00	68.00	0.2007	0.2978	1.2437	2.6386
10c1000d	4	0.8467	13.00	205.00	0.3369	0.4850	1.3600	2.3711
15c1000d	11	0.7600	9.67	47.33	0.3102	0.4060	1.8913	2.5453
24c1800d	6	0.8867	14.33	111.67	0.0936	0.1406	2.1599	2.8120
80c10000d	4	0.7800	11.33	28.00	0.3824	0.8758	1.5451	2.1703

The maximum stagnation length and final population diversity provide empirical evidence of the evolutionary behavior of GP-DPC across the tested datasets. The maximum stagnation length remains relatively limited. The final population diversity remains above 0.76 in all cases, suggesting sufficient exploration capability throughout evolution. For the first four datasets, the mean and maximum absolute correlation coefficients remain low, suggesting that no substantial linear redundancy is observed among the projected dimensions. However, on the largest 80c10000d dataset, the maximum absolute correlation increases to 0.8758 and the effective rank decreases to 2.17, suggesting that the three projection axes exhibit stronger linear correlation on large-scale data. This observation is consistent with the increased difficulty of learning a projection with limited redundancy when both the number of nodes and communities grow substantially.

To evaluate the computational efficiency of GP-DPC across different network scales, we conducted experiments on Handl datasets with network sizes ranging from 500 to 10,000 nodes. Table 7 summarizes the training time, inference time, and peak memory consumption across these datasets.

On the smallest dataset with 500 nodes and 10 communities, the training completes in approximately 207.67 s with a peak memory of 184.68 MB, while the inference time is merely 0.0196 s. As the network scale increases to 10,000 nodes with 80 communities, the training time extends to approximately 64,507 s, equivalent to about 17.9 h, and the peak memory rises to 869 MB. The training time grows substantially as both network size and the number of communities increase. This behavior is expected because larger networks and more complex community structures increase the computational cost of projection evaluation, density-peak estimation, and fitness calculation. The inference time remains consistently low across all datasets, indicating the efficiency of the projection-based online assignment. Once the three GP trees are learned, assigning a new node to a community requires only evaluating the

projection functions and querying the kNN index. The peak memory remains under 900 MB on all tested datasets, suggesting that GP-DPC maintains a moderate memory footprint for attributed networks of practical scale. Although the offline training time grows substantially with network size, this cost can be amortized over repeated inference, since the learned projection is reusable. These results complement the fine-grained time breakdown on the LFR dataset shown in Figure 6.

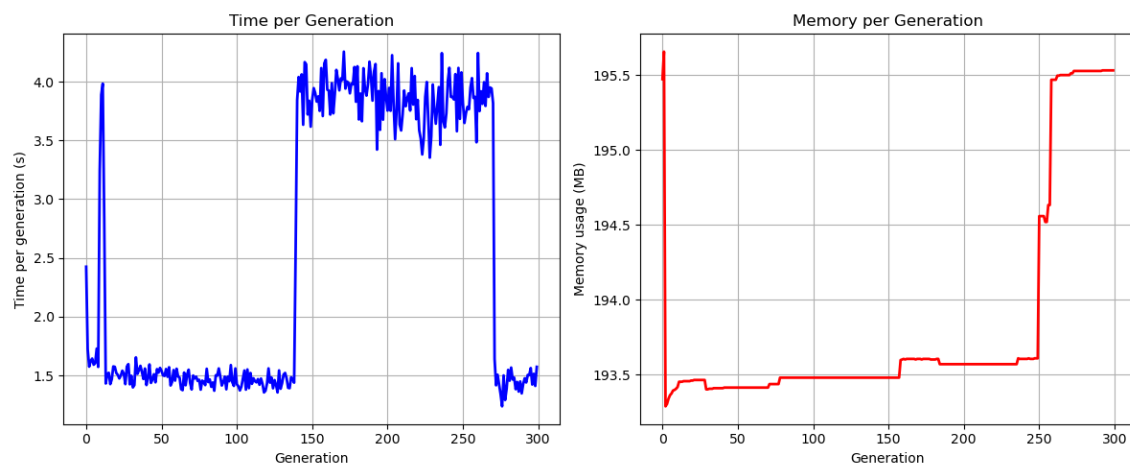


Figure 6. The running time and memory footprint of the GP-DPC algorithm on the 10c500d LFR dataset.

Table 7. Runtime and memory consumption of GP-DPC on Handl datasets of increasing scale.

Dataset	Nodes (n)	Communities	Training (s)	Inference (s)	Memory (MB)
10c500d	500	10	207.67	0.0196	184.68
10c1000d	1000	10	679.88	0.0732	195.15
15c1000d	1000	15	1383.73	0.1558	191.75
24c1800d	1800	24	5828.81	0.3820	211.66
40c5000d	5000	40	28734.86	1.3052	302.54
80c10000d	10000	80	64507.35	4.7937	869.13

To further compare GP-DPC with topology-driven and representation learning-based community detection methods, we evaluate Louvain, Leiden, Normalized Spectral Clustering, GAE, and VGAE on two representative Handl-based graphs. The results are shown in Table 8. Among these methods, Normalized Spectral Clustering requires the eigen-decomposition of the graph Laplacian matrix. For dense graphs, exact eigen-decomposition may require $O(n^3)$ operations, though for sparse graphs iterative solvers such as the Lanczos algorithm can reduce the practical runtime significantly. GAE and VGAE, as GNN-based methods, have a time complexity of $O(L \cdot (E \cdot F' + N \cdot F \cdot F'))$, where L is the number of GNN layers, E is the number of edges, N is the number of nodes, F is the input feature dimension, and F' is the hidden dimension. These methods serve as representative baselines from spectral and deep learning paradigms, respectively.

The comparison in Table 8 reveals several key observations. On the 10c500d dataset, NS, GAE, VGAE, and GP-DPC all achieve ARI and NMI values of 1.0000, whereas Louvain and Leiden obtain lower ARI values of 0.9781 and 0.9824, respectively. Although NS, GAE, and VGAE achieve the same label-level clustering accuracy on this dataset, their Conn., Spar., and Sep. values differ, reflecting differences in the learned representation spaces even when the resulting partitions are highly consistent with the reference labels. GP-DPC further achieves the highest Connectedness and Separability among the compared methods on 10c500d. On the more challenging 24c1800d dataset, the performance differences become more pronounced. VGAE achieves an ARI of 0.9295 and an NMI of 0.9744, making it the most competitive among the graph representation-learning baselines. GAE follows with an ARI of 0.7909, while NS obtains a lower ARI of 0.4135, suggesting that the spectral baseline is less effective under this more challenging attributed-network setting. GP-DPC matches the best ARI and NMI performance on 10c500d and achieves the strongest overall performance on 24c1800d, together with the highest modularity and the lowest conductance. Leiden identifies the expected number of communities on 24c1800d, while Louvain produces 25 communities instead of 24. Although Leiden achieves ARI and NMI values close to 1.0000, its modularity is substantially lower and its conductance considerably higher than those of GP-DPC. This indicates that high agreement with the reference labels does not necessarily correspond to equally favorable

graph-structural quality under the constructed attributed-network graph. In contrast, the projection-guided design of GP-DPC jointly considers projected-space separability and graph structural quality during evolution, leading to partitions that achieve both high label consistency and favorable structural metrics.

Table 8. Comparison of GP-DPC with representative topology-driven, spectral, and graph representation-learning baselines.

Dataset	Method	K	Conn.	Spar.	Sep.	ARI	NMI	Modularity	Conductance
10c500d	Leiden	10	54.0074	0.2577	0.4783	0.9824	0.9893	0.5486	0.3514
	Louvain	10	53.9652	0.2577	0.4774	0.9781	0.9260	0.5484	0.3516
	NS	10	82.1105	0.1936	0.6247	1.0000	1.0000	0.8999	0.0000
	GAE	10	91.6384	0.1204	0.7163	1.0000	1.0000	0.8999	0.0000
	VGAE	10	93.0245	0.1167	0.7420	1.0000	1.0000	0.8999	0.0000
	GP-DPC	10	95.3107	0.1184	0.7434	1.0000	1.0000	0.9000	0.0000
24c1800d	Leiden	24	63.0977	0.2124	0.2549	0.9920	0.9989	0.3154	0.6430
	Louvain	25	63.0897	0.2124	0.2602	0.9908	0.9896	0.3154	0.6430
	NS	24	87.6727	0.1064	0.2429	0.4135	0.8016	0.7614	0.2392
	GAE	24	94.3306	0.1084	0.3216	0.7909	0.9332	0.8766	0.1147
	VGAE	24	97.4395	0.1067	0.5094	0.9295	0.9744	0.9289	0.0464
	GP-DPC	24	98.3947	0.1080	0.5114	1.0000	1.0000	0.9583	0.0000

To further examine the properties of the learned projection on the Handl benchmarks, we report the Neighborhood Preservation Ratio, Trustworthiness, Graph NPR, LCC MAE, LCC Correlation, and Edge Reconstruction AUC in Table 4. These metrics collectively characterize the fidelity of the projection space from multiple perspectives.

The Neighborhood Preservation Ratio ranges from 0.1664 to 0.2704 across the four datasets, indicating that the local neighborhood structure in the projected space differs substantially from that in the original attribute space. This observation does not indicate a failure of the projection. GP-DPC is designed as a discriminative projection that maximizes community separability rather than an isometric embedding that preserves original distances. The projection is intentionally optimized to pull same-community nodes together and push different-community nodes apart, inherently altering local neighborhood structures. Crucially, despite the low NPR, the Edge Reconstruction AUC remains above 0.96 across all datasets, providing empirical evidence that substantial edge-level structural information is retained on the tested datasets. The LCC-based metrics further indicate that local clustering patterns are preserved to varying degrees across the tested datasets.

These results suggest that the advantage of GP-DPC on the Handl benchmarks comes from its projection-guided design. Louvain and Leiden mainly rely on graph topology after graph construction, while NS, GAE, and VGAE learn node embeddings through spectral decomposition or neural networks but still require careful hyperparameter tuning and substantial computational resources. In contrast, GP-DPC jointly uses node attributes, local structural features, and density-based assignment in the learned projection space, achieving perfect or near-perfect clustering. Therefore, the Handl results mainly support the effectiveness of GP-DPC in learning a useful low-dimensional representation for separable attributed data. The results on LFR benchmark networks are further reported to evaluate its behavior on graph-structured community detection tasks.

5.2. Performance on LFR Attributed Networks

As shown in Table 9, GP-DPC obtains an ARI of 0.9449 on 10c500d and 0.8602 on 20c1000d. These values indicate that the detected communities remain largely consistent with the reference communities. The number of detected communities is exactly recovered on 10c500d, while one additional community is produced on 20c1000d. This deviation suggests that the larger LFR network contains more ambiguous local structures, making density peak selection more difficult. Nevertheless, the difference in K is small, and the ARI remains above 0.85, showing that most nodes are still assigned to appropriate communities.

The structural metrics further support this observation. The modularity values are above 0.78 on both LFR datasets, indicating that the detected partitions preserve meaningful graph community structure. Conductance remains below 0.05, which means that the detected communities have relatively clear boundaries and limited inter-community connectivity. Compared with the Handl datasets, the values of Connectedness, Separability, and ARI are lower. This is expected because the LFR benchmark introduces more realistic topological heterogeneity and inter-community mixing, whereas the Handl datasets mainly test separability in a constructed feature-space graph. Therefore, the performance drop does not necessarily indicate a failure of GP-DPC, but reflects the increased difficulty of community detection on graph-structured benchmarks.

The running time and memory footprint of GP-DPC on the 10c500d LFR dataset are illustrated in Figure 6.

Table 9. GP-DPC performance on LFR attributed networks.

Scale	K	Conn.	Spar.	Sep.	ARI	Modularity	Conductance
10c500d	10	22.9001	0.6062	0.3976	0.9449	0.7834	0.0195
20c1000d	21	21.7490	0.7381	0.2903	0.8602	0.7817	0.0374

The results show that GP-DPC can complete projection learning and community assignment with manageable computational cost on this benchmark. Since GP-DPC separates offline evolutionary projection learning from FastDPC-based assignment, its computational burden is mainly concentrated in the training stage, while the learned projection can be reused for subsequent community assignment. Overall, the LFR results provide preliminary evidence that GP-DPC is not limited to idealized synthetic point sets and can also work on attributed networks with more realistic graph structures. The confidence in these findings is further strengthened by the comparative experiments on five real-world attributed networks, which are presented in the following subsection.

5.3. Performance on Five Real-World Attributed Networks

To further validate the practical effectiveness of GP-DPC beyond synthetic benchmarks, we evaluate it on five real-world attributed networks, namely Cora, Citeseer, PubMed, DBLP, and ACM. In addition, we conduct comparative experiments on three representative datasets to assess the effectiveness of GP-DPC against representative baselines, including spectral and GNN-based methods. These experiments demonstrate the empirical applicability of GP-DPC across heterogeneous real-world attributed networks in which node attributes and graph topology coexist with varying levels of noise.

Table 10 reports the performance of GP-DPC on five real-world attributed networks. The predicted number of communities generally aligns with the ground truth on most datasets, with exact recovery achieved on DBLP and ACM. Among the five datasets, ACM yields a relatively higher fitness value of 0.4688, while DBLP achieves a competitive silhouette score of 0.4826. These observations further demonstrate the empirical applicability of GP-DPC across heterogeneous real-world attributed networks with diverse attribute characteristics and varying levels of noise.

To further assess the effectiveness of GP-DPC against representative baselines, we conduct comparative experiments on three representative datasets, namely Cora, Citeseer, and ACM. The selected baselines include two topology-driven methods, Louvain and Leiden, a spectral method (NJW Spectral), an exact spectral method (Dense Exact Laplacian Eigenmap), and a GNN-based method (GAE-GCN with KMeans). Table 11 summarizes the comparison results in terms of community count estimation, clustering accuracy, and graph structural quality.

Table 11 reveals several notable observations. Topology-driven methods, Louvain and Leiden, predict excessively large numbers of communities on Cora with 104 and 106 respectively against the true K of 7, and on Citeseer with 466 and 472 respectively against the true K of 6. This indicates that purely topology-based approaches tend to over-partition real-world attributed networks, as they fail to incorporate node attributes that provide critical constraints on community boundaries. In contrast, GP-DPC produces substantially more accurate community count estimates with only one deviation on Cora and Citeseer, and exact recovery on ACM. In terms of clustering accuracy, GP-DPC achieves the highest ARI on Cora with 0.28698 and on ACM with 0.38677, and outperforms all baselines on Citeseer with an ARI of 0.23898. The GNN-based method GAE-GCN achieves competitive performance on ACM with an ARI of 0.35974, yet it still falls behind GP-DPC. Regarding structural quality, topology-driven methods yield high modularity values but at the cost of grossly incorrect community numbers, confirming that modularity alone is an insufficient criterion for attributed community detection. These results suggest that GP-DPC achieves a favorable balance between topological structure preservation and attribute-aware community partitioning on the tested real-world attributed networks.

Table 10. GP-DPC performance on real-world attributed networks.

Dataset	K(Pred/True)	ARI	NMI	Conn.	Sparsity	Sil.	Fitness
Cora	6/7	0.28698	0.30206	5.88697	1.64590	0.41514	0.34816
Citeseer	5/6	0.23898	0.30840	6.58091	1.67155	0.27872	0.37886
PubMed	4/3	0.13216	0.19495	11.42643	1.19126	0.31846	0.36813
DBLP	4/4	0.21294	0.25395	13.24997	1.14781	0.48260	0.32681
ACM	3/3	0.38677	0.44531	6.19987	1.71776	0.41884	0.46883

Table 11. Comparison of GP-DPC with representative topology-driven, spectral, and graph representation-learning baselines on real-world attributed networks.

Dataset	Method	K		ARI	NMI	Conn.	Spar.	Q	Cond.
		Pred	True						
Cora	Louvain	104	7	0.24076	0.25259	1.81828	1.38810	0.81432	0.02970
	Leiden	106	7	0.25420	0.26145	1.83720	1.38699	0.82346	0.02998
	NJW Spectral	7	7	0.27984	0.36005	5.10889	1.63053	0.66325	0.07016
	Laplacian Eigenmap	7	7	0.00306	0.04208	3.05430	1.67657	0.07480	0.00000
	GAE-GCN	7	7	0.10430	0.14617	3.30971	1.73325	0.53157	0.32369
	GP-DPC	6	7	0.28698	0.30206	5.88697	1.64590	0.49205	0.17494
Citeseer	Louvain	466	6	0.09823	0.12888	1.93521	1.00098	0.88987	0.00447
	Leiden	472	6	0.09658	0.13386	1.98338	1.00104	0.89378	0.00520
	NJW Spectral	6	6	0.17927	0.19530	6.24535	1.49528	0.72232	0.01210
	Laplacian Eigenmap	6	6	0.01876	0.02442	4.12092	1.61289	0.32198	0.00000
	GAE-GCN	6	6	0.07522	0.08668	4.86872	1.52794	0.60644	0.20813
	GP-DPC	5	6	0.23898	0.30840	6.58091	1.67155	0.59255	0.18811
ACM	Louvain	747	3	0.05379	0.26494	0.36215	1.59027	0.82363	0.01626
	Leiden	750	3	0.05621	0.27032	0.37296	1.58538	0.82561	0.01710
	NJW Spectral	3	3	0.13932	0.15534	6.87040	1.78549	0.49557	0.02483
	Laplacian Eigenmap	3	3	0.00885	0.02177	4.17875	1.92739	0.04853	0.00000
	GAE-GCN	3	3	0.35974	0.45663	6.12216	1.83346	0.53008	0.06481
	GP-DPC	3	3	0.38677	0.44531	6.19987	1.71776	0.66116	0.18748

5.4. Evolutionary Dynamics and Projection Quality Analysis

To investigate whether the GP search suffers from premature convergence or code bloat, and whether the three learned projection axes are redundant or collinear, we conduct two sets of analyses on representative datasets. The evolutionary dynamics and tree complexity are examined on the Handl 24c1800d and 10c500d datasets, while the projection-space properties including covariance analysis and objective-aggregation analysis are evaluated on the Handl 24c1800d benchmark and the real-world ACM network. Table 12 summarizes the evolutionary behavior and tree complexity across the representative Handl datasets.

Table 12 summarizes the evolutionary dynamics and tree complexity of GP-DPC on two representative Handl datasets. The maximum stagnation length remains limited on both datasets, while the final population diversity remains above 0.87, suggesting that population diversity is maintained during evolution. The mean aggregate depth of the three GP trees is 14.00 on 10c500d and 14.33 on 24c1800d, whereas the corresponding tree size increases from 68.00 to 111.67. These results indicate that the structural complexity of the evolved individuals varies across datasets rather than following a simple monotonic relationship with dataset scale. No unbounded increase in the reported complexity measures is observed in these runs.

Table 12. Evolutionary dynamics and tree-complexity metrics on representative Handl datasets.

Dataset	Stag.	Diversity	Mean Agg. Depth	Size
10c500d	8	0.8733	14.00	68.00
24c1800d	6	0.8867	14.33	111.67

In addition, we evaluate the projection-space properties on the Handl 24c1800d dataset and the real-world ACM network through covariance analysis and objective-aggregation analysis. The covariance analysis examines the correlation structure among the three projected dimensions to detect potential redundancy or collinearity. The multiplicative fitness function is further examined through its objective aggregation surface, while the convergence behavior is characterized using the stagnation and population-diversity statistics reported above. These analyses are used to empirically characterize the evolutionary search behavior and possible stagnation patterns on Table 13.

Table 13. Projection quality metrics on 24c1800d and ACM.

Dataset	Var(Z1)	Var(Z2)	Var(Z3)	Max Corr	Pairwise MI	Eff. Rank
24c1800d	0.05879	0.04583	0.10009	0.14060	2.15986	2.81202
ACM	0.04652	0.03616	0.04009	0.24142	0.82992	2.84666

The variance values across the three axes are well balanced on both datasets, with no single dimension dominating the others. The maximum absolute correlation coefficients are 0.14060 on 24c1800d and 0.24142 on ACM, indicating limited linear dependency among the projected dimensions. The effective rank values of 2.81202 and 2.84666 are both close to the theoretical maximum of 3, suggesting that no substantial linear redundancy or dimensional collapse is observed among the three projected dimensions on these representative datasets.

To visualize the aggregation behavior of the multiplicative fitness function $f = Q_n \times W_n$, we plot the objective aggregation surface over the two-dimensional domain of normalized modularity Q_n and normalized silhouette W_n . Figure 7 shows this surface on the ACM dataset, where a clear ridge appears along the diagonal where both Q_n and W_n are jointly high. The contour view in Figure 8 on the Handl 24c1800d dataset further reveals that the maximum of the product is located at the intersection of high Q_n and high W_n values, while regions where one metric is low and the other is high receive low scores due to the multiplicative penalty.

This visualization illustrates the design motivation of the multiplicative formulation: it inherently penalizes solutions that are strong in only one objective, thereby encouraging the evolutionary search to consider both structural and attribute-driven quality. The observed convergence behavior and robustness results reported in Section 5.4 and Section 5.5 provide empirical evidence that this formulation is effective in practice, though we do not claim that the surface analysis alone proves the absence of local optima in the GP search space.

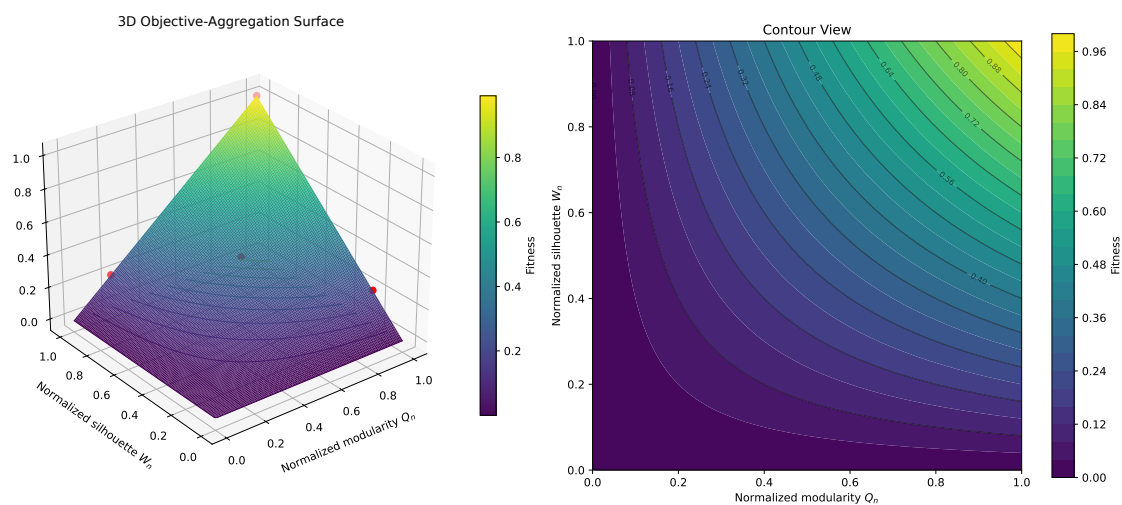


Figure 7. Objective aggregation surface of the multiplicative fitness function on the ACM dataset.

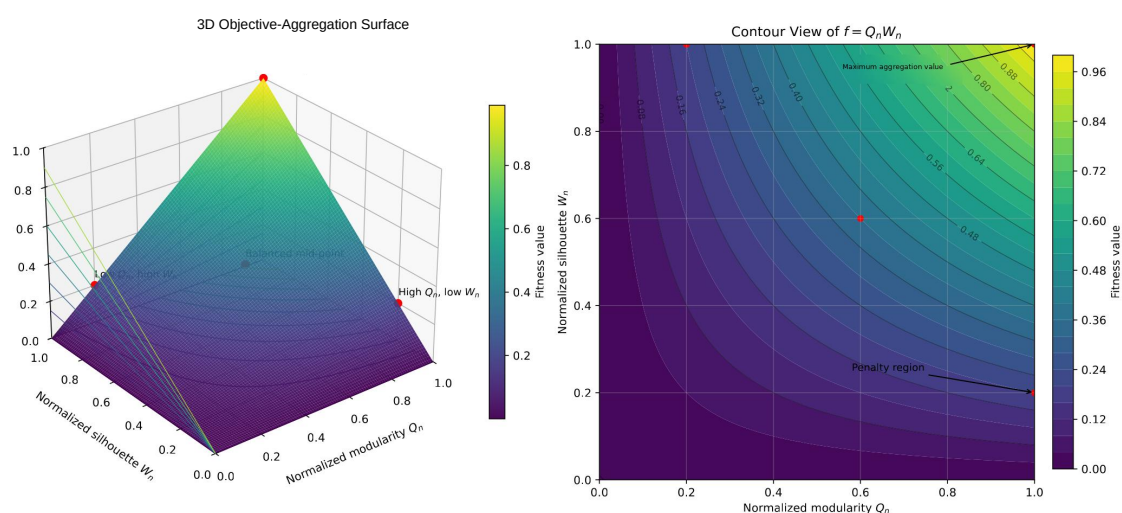


Figure 8. Objective aggregation surface of the multiplicative fitness function on the 24c1800d Handl dataset.

5.5. Robustness to Graph Perturbations and Attribute Noise

To evaluate the robustness of GP-DPC under topological perturbations, we conduct a graph corruption experiment by randomly removing edges from the original attributed graph. We consider four removal ratios, namely 5%, 10%, 15%, and 20%, and perform the experiment on the 24c1800d dataset, which contains 24 communities and 1800 nodes.

This dataset is selected as a representative medium-scale benchmark with sufficiently complex community structure to test robustness under stress, while remaining computationally tractable for repeated experiments across all perturbation settings. After edge removal, the remaining graph is used as the input structure, and the GP-evolved projection learned on the original graph is applied to the perturbed graph for community assignment. For each removal ratio, the graph-derived terminal features, including normalized degree, local clustering coefficient, and neighbor-aggregated attributes, are recalculated from the perturbed graph, while the raw node attributes and the GP projection functions learned on the original graph remain fixed. The resulting partition is evaluated with reference to the original unperturbed graph. Specifically, ARI, NMI, and K Error are evaluated against the unchanged ground-truth community labels, whereas modularity, conductance, Graph-NPR, and LCC-MAE in Table 14 are computed on or against the original unperturbed graph. This setup reflects realistic engineering scenarios where network connectivity may be incomplete due to measurement errors, communication failures, or missing sensor data. Table 14 reports the performance degradation under increasing levels of edge removal.

Table 14. Graph edge perturbation results on the 24c1800d dataset.

Removal Ratio	ARI	NMI	Q	Conductance	K Error	Graph-NPR	LCC-MAE
0.0%	1.0000	1.0000	0.9583	0.0000	0	0.3799	0.1192
5.0%	0.9898	0.9957	0.9453	0.0297	1	0.3743	0.1356
10.0%	0.9889	0.9955	0.9447	0.0285	1	0.3707	0.1498
15.0%	0.9889	0.9955	0.9478	0.0224	1	0.3666	0.1674
20.0%	0.9889	0.9955	0.9484	0.0208	1	0.3677	0.1871

Table 14 reports the performance of GP-DPC under increasing levels of edge removal. At the unperturbed setting, GP-DPC achieves perfect ARI and NMI scores with zero community-count error. When up to 20% of the edges are randomly removed, the ARI remains above 0.988 and the NMI remains above 0.995, indicating that the detected partitions are largely stable under moderate topological perturbations. The modularity values also remain relatively high across all perturbation levels, while the K Error increases only to 1 after edge removal. After an initial increase at 5% edge removal, conductance gradually decreases as the removal ratio further increases. The Graph-NPR remains relatively stable after perturbation, whereas the LCC-MAE gradually increases from 0.1192 to 0.1871, indicating that local clustering patterns are affected to some extent as more edges are removed. Overall, these results suggest that GP-DPC exhibits relatively strong robustness to moderate edge perturbations on the 24c1800d dataset.

Table 2 examines projection-space distortion when Gaussian noise is introduced directly into the projected coordinates after GP projection. For a noise level η , the perturbed projection is denoted by $\mathbf{Z}_\eta$, while the original node attributes, graph topology, and evolved GP trees remain unchanged. Projection-NPR@10 and Trustworthiness@10 in Table 2 use the clean projection $\mathbf{Z}$ as the reference and the noise-perturbed projection $\mathbf{Z}_\eta$ as the evaluated space. Consequently, both metrics equal 1.0000 at the zero-noise setting. The Projection-NPR decreases rapidly from 1.0000 to 0.2741 at a noise level of 0.05 and continues to decline as the noise level increases, indicating that local neighborhood relationships in the projection space are sensitive to perturbations. In contrast, Trustworthiness remains approximately 0.99 up to a noise level of 0.10 and remains above 0.96 even at 0.30, suggesting that the overall neighborhood-ranking quality remains relatively high despite increasing local distortion. The ARI and NMI remain close to 1.0 for noise levels up to 0.10, but both decrease sharply at 0.20, accompanied by a substantial increase in K Error. These results indicate that GP-DPC remains relatively stable under mild projection-space noise, while its clustering performance degrades substantially when the perturbation level increases beyond the range tested at 0.10.

Table 3 evaluates sensitivity to noise introduced directly into the node attributes before projection. For each noise level, the perturbed attributes are processed by the fixed evolved GP trees to obtain the corresponding projected coordinates, while the graph topology remains unchanged and GP is not retrained. Attribute-NPR@10 compares the neighborhoods in the clean original attribute space with those in the projection obtained from the noise-perturbed attributes. At zero noise, the perturbed attributes are identical to the original attributes and the resulting projection is the clean projection $\mathbf{Z}$; therefore, Attribute-NPR@10 reduces to NPR@10 in Table 4, explaining the common value of 0.2314 for the 24c1800d dataset. The Attribute-NPR decreases gradually from 0.2314 to 0.1598 as the noise level increases from 0 to 0.30, indicating increasing distortion of the original attribute-neighborhood relationships. However, the clustering performance is substantially more sensitive to attribute perturbation. The ARI decreases sharply from 1.0000 at the noise-free setting to 0.1538 at a noise level of 0.05, while the K Error increases from 0 to 19. Although the ARI shows some fluctuation at higher noise levels, it remains substantially

below the noise-free result throughout the tested range. These observations indicate that GP-DPC is highly sensitive to attribute noise, which is consistent with the fact that the learned projection functions directly depend on node attributes. In engineering scenarios where attribute quality cannot be guaranteed, attribute pre-processing, denoising, or robustness-enhancing mechanisms should therefore be considered before applying GP-DPC.

5.6. Ablation Study

Four variants are compared with the full GP-DPC framework. The first variant GP uses GP-based pairwise similarity learning without the proposed node-level projection and FastDPC assignment. The second variant DPC, retains the GP-learned projection but replaces FastDPC with the original Density Peaks Clustering algorithm, thereby evaluating the contribution of the approximate kNN-based acceleration strategy. The third variant FastDPC, applies the approximate k NN-based FastDPC directly to the original node representation. The fourth variant Random Projection, replaces the GP-evolved projection with a random projection of the same output dimensionality. These variants are evaluated on two representative Handl datasets, 10c500d and 24c1800d, and the results are reported in Table 15.

The results show that the GP variant achieves strong accuracy on both datasets, with an ARI of 1.0000 on 10c500d and 0.9997 on 24c1800d. This confirms that GP is effective in constructing discriminative feature combinations. However, this variant relies on pairwise similarity learning and therefore requires substantially higher computational cost than the proposed node-level projection design. In contrast, GP-DPC maintains the same or slightly better ARI while obtaining higher modularity and lower conductance. This indicates that the proposed projection-guided design preserves the representation ability of GP while producing structurally cleaner community partitions.

The DPC and FastDPC variants exhibit different forms of performance degradation relative to the full GP-DPC framework. The DPC variant retains the GP-learned projection but replaces FastDPC with the original DPC algorithm. It detects 11 communities on 10c500d and 25 communities on 24c1800d, with ARI values of 0.9149 and 0.8979, respectively. This indicates that, even with the learned projection retained, replacing the proposed FastDPC assignment mechanism leads to lower clustering accuracy and less stable community-count estimation. In contrast, the FastDPC variant operates directly on the original node representation without GP-based projection learning. It detects 12 and 27 communities on the two datasets, with further reductions in ARI. These results suggest that both the GP-learned projection and the FastDPC-based assignment mechanism contribute to the overall performance of GP-DPC, while the projection plays an important role in providing a representation space suitable for density-based community detection.

Table 15. Ablation Study on the Handl 10c500d and 24c1800d datasets.

Dataset	Method	K	Conn.	Spar.	Sep.	ARI	Mod.	Cond.
10c500d	GP	10	94.0142	0.1391	0.7257	1.0000	0.8388	0.0008
	DPC	11	57.7007	0.2476	0.9644	0.9149	0.8736	0.0729
	FastDPC	12	59.2233	0.2429	0.9164	0.8518	0.8035	0.1868
	Random projection	11	98.5199	0.0452	0.5376	0.9711	0.8787	0.0483
	GP-DPC	10	95.3107	0.1184	0.7434	1.0000	0.9000	0.0000
24c1800d	GP	24	99.9318	0.0469	0.5165	0.9997	0.8514	0.0090
	DPC	25	67.5866	0.2261	0.7927	0.8979	0.9439	0.0364
	FastDPC	27	67.6987	0.2203	0.6702	0.8689	0.9074	0.1098
	Random projection	25	98.9733	0.0445	0.2291	0.9901	0.9488	0.0230
	GP-DPC	24	98.3947	0.1080	0.5114	1.0000	0.9583	0.0000

The Random Projection variant obtains competitive results, especially on the two Handl datasets where the underlying group structures are relatively separable. This result indicates that low-dimensional projection itself can be helpful for density-based assignment. However, Random Projection does not consistently recover the correct number of communities and produces lower separability or higher conductance than GP-DPC. For example, on 24c1800d, Random Projection obtains an ARI of 0.9901 but detects 25 communities and has lower separability than GP-DPC. This shows that a generic projection can preserve part of the data structure, but it is less aligned with the community detection objective than the GP-evolved projection.

6. Engineering-Oriented Analysis

The engineering relevance of GP-DPC can be illustrated through attributed network analysis in intelligent transportation systems. In such a scenario, a road network can be modeled as an attributed graph $\mathcal{G} = (V, E, \mathbf{X})$, where each node represents an intersection or traffic sensor, and each edge denotes a physical road segment. The goal is to divide the network into traffic control regions in which intersections show strong internal connectivity and similar traffic patterns. Compared with topology-only partitioning, this task requires the algorithm to consider both road connectivity and node attributes, such as historical flow statistics, peak-hour factors, periodic traffic patterns, and local structural indicators. Therefore, it provides a suitable engineering example for attributed community detection: a useful partition should be structurally meaningful in the road graph and operationally coherent in terms of traffic behavior.

In this setting, the node attribute vector can follow the same general structure as the terminal set used in GP-DPC. The original attributes may include normalized traffic-flow features, while the structural attributes include normalized degree and local clustering coefficient. Neighbor-aggregated attributes can further summarize the traffic behavior of adjacent intersections. GP-DPC first learns three nonlinear projection functions from these node-level features and maps each intersection into a compact low-dimensional space. FastDPC then assigns the projected nodes to density-based communities. This two-stage design is practically useful because the computationally expensive GP evolution can be performed offline using historical traffic data, whereas online assignment only requires evaluating the learned projection functions and querying nearby projected points, thereby amortizing the offline computational cost over repeated online queries. Thus, newly added sensors or intersections can be assigned to existing control regions without recomputing all pairwise distances or retraining the whole model immediately.

From an engineering perspective, GP-DPC offers three potential advantages. First, it separates offline representation learning from online community assignment, which is important for systems that require timely response but can tolerate periodic offline retraining. Second, it provides a mechanism for combining heterogeneous node attributes with local graph structure, avoiding the limitation of methods that rely only on road connectivity or only on raw traffic features. Third, its memory cost is mainly determined by the projected node representations and the k NN structure, making it suitable for moderate-scale attributed network analysis. Nevertheless, practical deployment still requires further validation on real transportation datasets with dynamic traffic conditions, missing sensor records, and evolving network topology. Therefore, this analysis should be regarded as an engineering-oriented use case that explains how GP-DPC can be integrated into real systems, rather than as a complete deployment study.

7. Conclusions and Future Work

This paper proposes GP-DPC, a projection-guided community detection framework for attributed networks. The core idea is to use GP to learn node-level nonlinear projection functions and then apply FastDPC in the learned low-dimensional space for efficient community assignment. GP provides adaptive feature construction, while FastDPC avoids the full pairwise distance computation required by conventional GP-based similarity learning. The computational complexity is $O(G_{\max} \cdot N_p(nL + n \log n))$ during training and $O(nL + n \log n)$ during full-batch inference, while the online assignment of a newly arrived node requires only $O(L + \log n)$. This means that the offline training cost can be amortized over repeated inference, making GP-DPC potentially applicable to scenarios where periodic offline retraining is acceptable and timely online response is required.

The experimental evaluation covers synthetic HandI and LFR benchmarks, five real-world attributed networks, and various perturbation conditions. Direct baseline comparisons are reported for the HandI 10c500d and 24c1800d benchmarks and for Cora, Citeseer, and ACM, where GP-DPC achieves competitive detection accuracy. Separate runtime analyses indicate low inference cost, although offline training time increases substantially with network scale. The projection quality analysis on the representative 24c1800d and ACM datasets suggests limited linear redundancy among the three evolved axes, with effective rank values close to 3, while the objective-aggregation visualization shows that the multiplicative fitness function penalizes solutions dominated by a single objective without establishing unimodality, smoothness, or the absence of local optima. The robustness experiments further show that GP-DPC maintains high ARI under up to 20% edge removal but remains sensitive to attribute noise, with substantial degradation even at relatively low noise levels. Taken together, these results support the effectiveness and practical potential of GP-DPC within the evaluated settings.

The effectiveness of GP-DPC stems from the integration of three components: multi-tree GP projection, density-based community assignment, and community-aware fitness evaluation. The learned projection transforms heterogeneous node attributes and local structural statistics into a compact space suitable for density estimation. FastDPC then assigns communities efficiently without constructing a global distance matrix. The composite fitness

function further guides the search toward projections that are consistent with both structural modularity in the original graph and separability in the projected space. These properties make GP-DPC a useful framework for attributed network analysis tasks where both node attributes and graph topology need to be considered.

Several directions remain for future work. First, the scalability of GP-DPC should be further examined on larger real-world attributed networks with more than 10,000 nodes, where approximate DPC variants or parallel fitness evaluation could be explored to reduce training time. Second, dynamic attributed networks with evolving node attributes and graph structures call for incremental GP evolution and local fine-tuning mechanisms to adapt to community drift without full retraining. Third, robustness to missing, noisy, or incomplete node attributes should be improved through neighbor-based imputation, multi-view attribute aggregation, or uncertainty-aware projection learning. Finally, beyond synthetic and benchmark networks, GP-DPC should be validated on real semantic graphs and OAEI-derived attributed networks to examine whether the detected communities can support ontology matching, candidate generation, and other knowledge engineering tasks.

Author Contributions

C.D.: conceptualization, methodology, software, writing—original draft preparation; J.W.: project administration, resources, supervision; H.L.: supervision, visualization; X.X.: supervision, software, validation, writing—reviewing, funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding

This research was funded by Natural Science Foundation of Fujian Province Under Grant 2026J0011026, and National Natural Science Foundation of China Under Grant 62172095.

Institutional Review Board Statement

Not applicable.

Data Availability Statement

The Handl synthetic benchmark generators are publicly available at: <https://personalpages.manchester.ac.uk/staff/Julia.Handl/generators.html>. The LFR benchmark generator is available at: https://figshare.com/articles/dataset/Lancichinetti_Fortunato_Radicchi_LFR_benchmark/1149962. The preprocessed Cora, Citeseer, and PubMed datasets are available from the Planetoid repository at: <https://github.com/kimiyoung/planetoid>, while the ACM and DBLP datasets are available through the SDCN data release at: <https://github.com/bdy9527/SDCN>. The synthetic instances generated for this study, the attributed graphs constructed from the Handl datasets, the attributed LFR data, and the code supporting the findings are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare no conflict of interest.

Use of AI and AI-Assisted Technologies

During the preparation of this work, the authors used ChatGPT-5.6 sol to assist with debugging program errors and code review. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

References

1. Li, J.; Lai, S.; Shuai, Z.; et al. A Comprehensive Review of Community Detection in Graphs. *Neurocomputing* **2024**, *600*, 128169. <https://doi.org/10.1016/j.neucom.2024.128169>.
2. Khawaja, F.R.; Zhang, Z.; Memon, Y.; et al. Exploring Community Detection Methods and Their Diverse Applications in Complex Networks: A Comprehensive Review. *Soc. Netw. Anal. Min.* **2024**, *14*, 115. <https://doi.org/10.1007/s13278-024-01274-1>.
3. Jin, D.; Yu, Z.; Jiao, P.; et al. A Survey of Community Detection Approaches: From Statistical Modeling to Deep Learning. *IEEE Trans. Knowl. Data Eng.* **2023**, *35*, 1149–1170. <https://doi.org/10.1109/tkde.2021.3104155>.
4. Kumar, P.; Jain, R.; Chaudhary, S.; et al. Solving Community Detection in Social Networks: A Comprehensive Study. In Proceedings of the 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), Erode, India, 8–10 April 2021; pp. 239–345. <https://doi.org/10.1109/iccmc51019.2021.9418412>.
5. Naik, D.; Ramesh, D.; Gandomi, A.H.; et al. Parallel and Distributed Paradigms for Community Detection in Social Networks: A Methodological Review. *Expert Syst. Appl.* **2022**, *187*, 115956. <https://doi.org/10.1016/j.eswa.2021.115956>.

6. Manipur, I.; Giordano, M.; Piccirillo, M.; et al. Community Detection in Protein-Protein Interaction Networks and Applications. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **2023**, *20*, 217–237. <https://doi.org/10.1109/tcbb.2021.3138142>.
7. Gasparetti, F.; Sansonetti, G.; Micarelli, A. Community Detection in Social Recommender Systems: A Survey. *Appl. Intell.* **2021**, *51*, 3975–3995. <https://doi.org/10.1007/s10489-020-01962-3>.
8. Chen, D.; Su, H.; Zhang, H. Community Detection-Based Renormalization Method for Power Grids Topology. *IEEE Trans. Ind. Inform.* **2026**, *22*, 1916–1926. <https://doi.org/10.1109/tii.2025.3632843>.
9. Safarzadeh, M.; Yousefi, G.R.; Latify, M.A.; et al. Power System Controlled Islanding Using Directed Motif-Based Spectral Clustering: A Novel Complex Network Perspective. *IET Smart Grid* **2025**, *8*, e70007. <https://doi.org/10.1049/stg2.70007>.
10. Hu, H.; Qi, Y.; Kang, N.; et al. Optimizing Warehouse Product Allocation at JD.com: A Community Detection-Driven Approach. *Eur. J. Oper. Res.* **2026**, *334*, 315–335. <https://doi.org/10.1016/j.ejor.2026.04.040>.
11. Wu, J.; Dai, T.; Guan, P.; et al. Opportunistic Routing for Mobile Edge Computing: A Community Detected and Task Priority Aware Approach. *Comput. Netw.* **2025**, *258*, 111000. <https://doi.org/10.1016/j.comnet.2024.111000>.
12. Antosiewicz, M.; Szufel, P.; Kamiński, B.; et al. Clustering Problem in Self-Organising Communication Networks on the Example of Smart Vehicle Transportation System. *Przegląd Stat. Stat. Rev.* **2025**, *72*, 1–24. <https://doi.org/10.59139/ps.2025.04.1>.
13. Hu, J.H.; Sun, Y.N.; Qin, W. A Key Process Identification Framework for Aircraft Assembly Production Based on the Network with Physical Attributes. *J. Manuf. Syst.* **2025**, *80*, 595–609. <https://doi.org/10.1016/j.jmsy.2025.03.024>.
14. Makanda, I.L.D.; Yang, M.; Shi, H.; et al. Leveraging Natural Language Processing and Community Detection for Shaping Manufacturing Communities in Social Manufacturing. *J. Manuf. Syst.* **2024**, *74*, 1091–1105. <https://doi.org/10.1016/j.jmsy.2024.05.020>.
15. El-Moussaoui, M.; Hanine, M.; Kartit, A.; et al. A Systematic Review of Deep Learning Methods for Community Detection in Social Networks. *Front. Artif. Intell.* **2025**, *8*, 1572645. <https://doi.org/10.3389/frai.2025.1572645>.
16. Chunaev, P. Community Detection in Node-Attributed Social Networks: A Survey. *Comput. Sci. Rev.* **2020**, *37*, 100286. <https://doi.org/10.1016/j.cosrev.2020.100286>.
17. Wang, Z.; Yang, J.; Guan, J.; et al. Enhanced Denesity Peak Clustering for High-Dimensional Data. In Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence, Philadelphia, PA, USA, 25 February–4 March 2025; Volume 39, pp. 21411–21419. <https://doi.org/10.1609/aaai.v39i20.35442>.
18. Lensen, A.; Xue, B.; Zhang, M. Genetic Programming for Evolving Similarity Functions for Clustering: Representations and Analysis. *Evol. Comput.* **2020**, *28*, 531–561. https://doi.org/10.1162/evco_a_00264.
19. Blondel, V.D.; Guillaume, J.L.; Lambiotte, R.; et al. Fast Unfolding of Communities in Large Networks. *J. Stat. Mech. Theory Exp.* **2008**, *2008*, P10008. <https://doi.org/10.1088/1742-5468/2008/10/p10008>.
20. Lu, H.; Halappanavar, M.; Kalyanaraman, A. Parallel Heuristics for Scalable Community Detection. *Parallel Comput.* **2015**, *47*, 19–37. <https://doi.org/10.1016/j.parco.2015.03.003>.
21. Wang, Z.; Lin, X.; Li, X.; et al. Swift Unfolding of Communities: GPU-Accelerated Louvain Algorithm. In Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming, Las Vegas, NV, USA, 1–5 March 2025; pp. 441–454. <https://doi.org/10.1145/3710848.3710884>.
22. Traag, V.A.; Waltman, L.; van Eck, N.J. From Louvain to Leiden: Guaranteeing Well-Connected Communities. *Sci. Rep.* **2019**, *9*, 5233. <https://doi.org/10.1038/s41598-019-41695-z>.
23. Rodriguez, A.; Laio, A. Clustering by Fast Search and Find of Density Peaks. *Science* **2014**, *344*, 1492–1496. <https://doi.org/10.1126/science.1242072>.
24. Chen, Y.; Hu, X.; Fan, W.; et al. Fast Density Peak Clustering for Large Scale Data Based on kNN. *Knowl.-Based Syst.* **2020**, *187*, 104824. <https://doi.org/10.1016/j.knosys.2019.06.032>.
25. Berahmand, K.; Sheikhpour, R.; Saberi-Movahed, F.; et al. OA²H-SP: One-Step Anchor-Adaptive Hypergraph Spectral Clustering. In Proceedings of the 2025 IEEE International Conference on Data Mining (ICDM), Washington, DC, USA, 12–15 November 2025; pp. 1048–1054. <https://doi.org/10.1109/icdm65498.2025.00113>.
26. Mohammadi, M.; Berahmand, K.; Forouzandeh, S.; et al. Dual-View Entropy-Regularized Nonnegative Matrix Factorization for Attributed Graph Clustering. *Inf. Sci.* **2025**, *721*, 122566. <https://doi.org/10.1016/j.ins.2025.122566>.
27. Berahmand, K.; Mohammadi, M.; Sheikhpour, R.; et al. Relative Entropy-Based Regularized Non-Negative Matrix Factorization for Attributed Graph Clustering. *ACM Trans. Knowl. Discov. Data* **2025**, *19*, 1–28. <https://doi.org/10.1145/3765742>.
28. Berahmand, K.; Bahadori, S.; Abadeh, M.N.; et al. SDAC-DA: Semi-Supervised Deep Attributed Clustering Using Dual Autoencoder. *IEEE Trans. Knowl. Data Eng.* **2024**, *36*, 6989–7002. <https://doi.org/10.1109/tkde.2024.3389049>.
29. McCallum, A.K.; Nigam, K.; Rennie, J.; et al. Automating the Construction of Internet Portals with Machine Learning. *Inf. Retr.* **2000**, *3*, 127–163. <https://doi.org/10.1023/a:1009953814988>.
30. Giles, C.L.; Bollacker, K.D.; Lawrence, S. CiteSeer: An Automatic Citation Indexing System. In Proceedings of the Third ACM Conference on Digital Libraries, Pittsburgh, PA, USA, 23–26 June 1998; pp. 89–98. <https://doi.org/10.1145/276675.276685>.
31. Sen, P.; Namata, G.; Bilgic, M.; et al. Collective Classification in Network Data. *AI Mag.* **2008**, *29*, 93–106. <https://doi.org/10.1609/aimag.v29i3.2157>.

32. Tang, J.; Zhang, J.; Yao, L.; et al. ArnetMiner: Extraction and Mining of Academic Social Networks. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, NV, USA, 24–27 August 2008; pp. 990–998. <https://doi.org/10.1145/1401890.1402008>.
33. Krapivin, M.; Marchese, M.; Casati, F. Exploring and Understanding Citation-Based Scientific Metrics. *Adv. Complex Syst.* **2010**, *13*, 59–81. <https://doi.org/10.1142/s0219525910002487>.
34. Ng, A.Y.; Jordan, M.I.; Weiss, Y. On Spectral Clustering: Analysis and an Algorithm. In Proceedings of the 15th International Conference on Neural Information Processing Systems: Natural and Synthetic, Vancouver, BC, Canada, 3–8 December 2001; pp.849–856.
35. Hasanzadeh, A.; Hajiramezanali, E.; Narayanan, K.; et al. Semi-Implicit Graph Variational Auto-Encoders. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; Volume 32, pp. 10712–10723.
36. Ding, Y.; Tian, L.P.; Lei, X.; et al. Variational Graph Auto-Encoders for miRNA-Disease Association Prediction. *Methods* **2021**, *192*, 25–34. <https://doi.org/10.1016/j.ymeth.2020.08.004>.