The CSI Journal on Computer Science and Engineering

Journal homepage: www.csionjcse.ir

Efficient Geometric Clustering via t-Spanners and Local Adaptive MST Edge Cutting

Mohammad Heydari[1*]

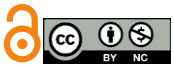
[1] Department of Computer Science, Khansar Campus, University of Isfahan, Isfahan, Iran

*** Corresponding author email address**: m.heydari@khc.ui.ac.ir

A r t i c l e I n f o

Article type:
Original Research

How to cite this article:
Heydari, M. (2026). Efficient Geometric Clustering via t-Spanners and Local Adaptive MST Edge Cutting. *The CSI Journal on Computer Science and Engineering, 20*(2), 76-89.
http://dx.doi.org/10.22034/jcse.2026.577923.1078



A B S T R A C T

Clustering a set of points in the Euclidean plane is a fundamental problem in computational geometry and data analysis, with applications spanning image segmentation, spatial data mining, and pattern recognition. Minimum Spanning Tree (MST)-based clustering methods offer a geometrically intuitive approach, but suffer from the $O(n^2)$ cost of constructing the complete Euclidean graph for large point sets. We propose an efficient $O(n \log n)$ algorithm for clustering n points in the Euclidean plane. Our method constructs a t-spanner with stretch factor $t = 1 + \varepsilon$ for any user-chosen $\varepsilon > 0$ via a Well-Separated Pair Decomposition (WSPD), yielding a sparse graph of $O(n/\varepsilon^2)$ edges whose MST approximates the true Euclidean MST within a factor of $1 + \varepsilon$. For any fixed $\varepsilon > 0$ this is $O(n)$ edges, though the constant grows as $\varepsilon \to 0$. Clustering is then performed by a novel local adaptive edge-cutting criterion: an edge (a_i, b_i) is removed if its weight exceeds α times $\lambda_i = \max(\delta(a_i), \delta(b_i))$, where $\delta(p)$ is the local scale at p (approximated in practice by its k-nearest-neighbour distance). This makes the criterion scale-aware and sensitive to local point density. When $\delta(p)$ is taken as the FST leaf cell diameter, all quantities needed for cutting are available as byproducts of the Fair Split Tree construction at no extra asymptotic cost. Experiments on synthetic Gaussian datasets and standard 2D clustering benchmarks (R15, Aggregation) confirm that the method achieves perfect clustering on well-separated point sets and competitive NMI on denser configurations.

Keywords: *clustering, minimum spanning tree, t-spanner, well-separated pair decomposition, local adaptive cutting*

Article history:
Published online 07 August 2026

1 Introduction

Clustering is one of the most fundamental tasks in data analysis, machine learning, and computational geometry. Given a set of n points in the Euclidean plane, the goal of clustering is to partition them into meaningful groups such that points within the same group are geometrically close, while points in different groups are well-separated. This problem arises in a wide range of applications, including image segmentation [1], spatial data mining [2], geographic information systems, pattern recognition [3], and bioinformatics, making efficient and accurate clustering algorithms of broad practical importance.

Among the many clustering paradigms proposed over the decades, Minimum Spanning Tree (MST)-based methods hold a distinctive geometric appeal. The Euclidean MST of a point set is a sparse graph that faithfully encodes the proximity structure of the data: short edges connect nearby points, and long edges bridge distant regions. Zahn [3] was the first to exploit this structure for clustering, proposing to remove inconsistent edges—those that are anomalously long relative to their local neighborhood—to reveal natural clusters. A key strength of MST-based clustering is its ability to discover clusters of arbitrary shape, overcoming the convexity assumptions inherent in centroid-based methods such as k-means [4]. Furthermore, as shown by Gower and Ross [5], single linkage hierarchical clustering is mathematically equivalent to removing the $k - 1$ longest edges of the Euclidean MST, demonstrating the central role of the MST in clustering theory.

Despite their geometric appeal, MST-based clustering algorithms face a critical computational bottleneck. Computing the exact Euclidean MST on n points traditionally requires constructing the complete Euclidean graph, which has $O(n^2)$ edges, making it impractical for large datasets. Several works have attempted to address this by constructing sparse proxy graphs. Jothi et al. [6, 7] proposed the Local Neighborhood Graph (LNG), a sparse graph with $O(n^{3/2})$ edges that can be constructed in $O(n^{3/2})$ time, from which an approximate MST is derived in O(n3/2 log n) time. While this is a significant improvement over $O(n^2)$, it still falls short of the optimal O(n log n) bound, and crucially, the LNG does not provide a provable worst-case approximation guarantee on the weight of the resulting MST. More recent sparse graph methods [8] follow a similar paradigm but share this limitation. A theoretically grounded, O(n log n) approach with a proven MST approximation guarantee has remained elusive.

In this paper, we propose an efficient clustering algorithm that closes this gap by leveraging geometric t-spanners as a sparse and theoretically principled proxy for the complete Euclidean graph. A t-spanner of a point set S is a graph G on S such that for any two points $u, v \in S$, the shortest path distance in G satisfies $dG(u, v) \leq t \cdot \|uv\|$, where $t \geq 1$ is the stretch factor [9]. Using the Well-Separated Pair Decomposition (WSPD) of Callahan and Kosaraju [10], a t-spanner with $O(n/\varepsilon^2)$ edges can be constructed in O(n log n) time for any fixed $\varepsilon > 0$ by setting $t = 1 + \varepsilon$ and separation constant $s = 2/\varepsilon$ (with $O(n/\varepsilon^2)$ being O(n) for each fixed ε, but growing unboundedly as $\varepsilon \to 0$). A key theorem from [9, 10] guarantees that the MST of the spanner has total weight at most t times the weight of the true Euclidean MST, providing a rigorous quality certificate absent from prior sparse graph approaches. Crucially, in our WSPD-based spanner, every spanner edge corresponds to exactly one WSPD pair, making the pairing between edges and local geometric information immediately available.

Beyond the spanner construction, we introduce a novel local adaptive edge-cutting criterion for partitioning the spanner's MST into clusters. A fundamental weakness of existing MST-cutting strategies is their reliance on global statistics—such as average or standard deviation of edge weights—which fail to adapt to local density variations in the point set. In contrast, our criterion exploits the local geometric information encoded in the Fair Split Tree: for each MST edge (ai, bi), we cut the edge if its weight exceeds $\alpha \cdot \lambda i$, where $\lambda i = \max(\delta(ai), \delta(bi))$ is the maximum FST leaf cell diameter of its endpoints and $\alpha > 0$ is a parameter. Since $\delta(p)$ reflects the local point density specifically around p, the threshold $\alpha \cdot \lambda i$ adapts to local density: tight in dense regions and relaxed in sparse ones. All quantities are computed as byproducts of the Fair Split Tree construction at no extra asymptotic cost.

The main contributions of this paper are as follows:

- We propose an end-to-end geometric clustering algorithm that runs in O(n log n) time and $O(n/\varepsilon^2)$ space for any fixed $\varepsilon > 0$, matching the optimal time complexity for Euclidean MST construction.
- **We establish a provable approximation guarantee:** the MST computed by our algorithm has weight at most $1 + \varepsilon$ times the true Euclidean MST weight, for any user-chosen $\varepsilon > 0$.
- We introduce a novel WSPD-based local adaptive edge-cutting criterion that is scale-aware and sensitive to local

density, overcoming the key limitation of global threshold methods.

• The Fair Split Tree underlies both the WSPD construction and the local scale computation λi, making the algorithm unified around a single data structure with no extra overhead.

• We validate the proposed method through experiments on synthetic Gaussian datasets and standard 2D clustering benchmarks (R15, Aggregation), demonstrating its effectiveness on well-separated point configurations.

The remainder of this paper is organized as follows. Section 2 reviews related work on clustering algorithms and geometric spanners. Section 3 introduces the necessary background on WSPD, t-spanners, and MSTs. Section 4 presents the proposed algorithm in detail. Section 5 provides the theoretical analysis of correctness and complexity. Section 6 reports experimental results, and Section 7 concludes the paper.

2 Related Work

Clustering is one of the most fundamental problems in data analysis and pattern recognition. Over the decades, a wide variety of algorithms have been proposed, which can broadly be categorized into partitional, hierarchical, density-based, grid-based, and graph-based methods. We survey the most relevant approaches, with emphasis on graph-based and MST-based techniques that are most closely related to our work, and conclude with a discussion of geometric spanners and WSPD, which form the algorithmic foundation of our approach.

2.1 Partitional and Centroid-Based Methods

Partitional methods divide a dataset into k disjoint clusters by optimizing an objective function. The most widely used representative is k-means [4], which iteratively assigns points to the nearest centroid and recomputes centroids until convergence. While efficient in practice, k-means assumes spherical, equally sized clusters and is sensitive to initialization. Variants such as k-medoids (PAM) [11], k-medians, and Fuzzy c-Means [12] address some of these limitations but retain the assumption of convex cluster shapes. Gaussian Mixture Models (GMM) solved via the EM algorithm [13] offer a probabilistic generalization, allowing clusters of different shapes and sizes, but require the number of components to be specified in advance. All these methods are illsuited for detecting clusters of arbitrary, non-convex shapes in Euclidean space, a task for which MST-based methods are naturally better equipped.

2.2 Hierarchical Methods

Hierarchical clustering algorithms produce a dendrogram representing nested cluster structures. Agglomerative methods such as single linkage, complete linkage, and average linkage (UPGMA) [14] differ in how inter-cluster distances are computed. Notably, single linkage clustering is algorithmically equivalent to removing the $k - 1$ longest edges from the Euclidean MST [5], establishing a deep connection between hierarchical clustering and spanning trees. Ward's method [15] minimizes total within-cluster variance at each merge step, producing more compact clusters. BIRCH [16] supports incremental, scalable hierarchical clustering for large databases using a compact summary structure. While hierarchical methods reveal multi-scale structure, they are generally expensive for large point sets and do not directly exploit geometric structure.

2.3 Density-Based Methods

Density-based approaches define clusters as regions of high point density separated by sparse regions, making them naturally suited for arbitrary-shaped clusters and noise detection. DBSCAN [2] is the seminal method in this category, requiring two parameters: a neighborhood radius ε and a minimum point count. OPTICS [17] extends DBSCAN by producing a reachability plot that captures cluster structure at multiple density levels. HDBSCAN [18] further generalizes this to a hierarchical framework that is more robust to varying densities and requires only a single parameter. The Density Peaks Clustering algorithm [19] identifies cluster centers as points that are both locally dense and distant from other high-density points. While density-based methods handle irregular shapes and outliers well, their performance is sensitive to parameter choices and degrades under highly non-uniform density distributions.

2.4 Grid-Based and Model-Based Methods

Grid-based methods such as CLIQUE [20] and STING [21] partition the data space into cells and identify dense cells as clusters. Mean Shift [22] performs gradient ascent toward local density modes and is parameter-free in principle, though its computational cost can be high. Affinity Propagation [23] uses message passing between data points

to identify exemplars without requiring k to be specified. These methods, while useful in specific contexts, do not exploit the geometric structure of the Euclidean plane as directly as graph-based approaches.

2.5 Graph-Based and MST-Based Methods

Graph-based clustering methods model the dataset as a weighted graph and extract cluster structure from its topology. The Euclidean MST is a particularly compelling basis for clustering: it is a sparse graph that captures the neighborhood structure of the point set, and its edges naturally reflect proximity relationships.

The foundational work in MST-based clustering is due to Zahn [3], who proposed building the MST of a point set and removing edges that are significantly longer than their local neighborhood edges, a notion he termed inconsistent edges. Zahn's method captures clusters of arbitrary shape but does not yield a fixed number of clusters, and the definition of inconsistency is heuristic in nature. This limitation has motivated decades of follow-up work.

Single linkage clustering, noted to be equivalent to MST-based divisive clustering by Gower and Ross [5], removes the $k - 1$ longest edges from the MST to produce k clusters. While elegant, this approach suffers from the chaining effect, where elongated, spurious clusters form due to sensitivity to outlier edges.

To address the chaining problem and improve robustness, numerous MST-cutting strategies have been proposed. Grygorash et al. [24] introduced HEMST, which removes edges to achieve the best possible reduction in the standard deviation of edge weights, and MSDR, a related variant. Ma et al. [25] proposed CTCEHC, which combines MST-based partitioning with centroid-guided merging using geodesic distances between cluster centroids. Zhong et al. [26] proposed a method using two rounds of minimum spanning trees to improve robustness against outliers and density variation, and later extended this line of work into a hierarchical split-and-merge framework [27]. Zhong et al. [28] further proposed FMST, a divide-and-conquer algorithm that partitions the dataset into $O(\sqrt{n})$ clusters via k-means and merges local MSTs, achieving $O(n^{3/2})$ time with no provable approximation bound. Farag et al. [29] introduced a critical distance methodology for identifying inconsistent edges, showing improved performance on datasets with outliers. Du et al. [30] proposed R-MST, which builds the MST on a set of representative points selected based on density and nearest neighbor distance, using mutual neighbor relationships to adaptively detect inconsistent edges without requiring prior knowledge of k.

A parallel line of research focuses on reducing the $O(n^2)$ complexity of building the complete Euclidean graph from which the MST is computed. Jothi et al. [6, 7] proposed building a sparse Local Neighborhood Graph (LNG) using a centroid-based nearest neighbor rule. The LNG has $O(n^{3/2})$ edges, and the approximate MST constructed from it runs in O(n3/2 log n) time, a significant improvement over $O(n^2)$ while maintaining cluster quality. Akhter et al. [8] further improved this to a two-level partitioning and merging scheme, achieving even sparser graphs. Spectral clustering variants [1, 31] also benefit from sparse graph constructions, using graph Laplacians to find cluster boundaries via eigenvector analysis.

Despite these advances, none of the existing sparse graph approaches achieve the theoretically optimal O(n log n) time with $O(n/\varepsilon^2)$ space while providing a provable approximation guarantee on the MST weight. Our method closes this gap by leveraging geometric t-spanners.

2.6 Geometric Spanners and WSPD

A t-spanner of a point set S is a sparse graph G on S such that for any two points $u, v \in S$, the shortest path in G satisfies $d\_G(u, v) \leq t \cdot \|uv\|$, where $t \geq 1$ is the stretch factor [33]. Geometric spanners with $O(n/\varepsilon^2)$ edges and O(n log n) construction time can be built using the Well-Separated Pair Decomposition (WSPD), introduced in the seminal work of Callahan and Kosaraju [34]. The WSPD decomposes the $n(n - 1)/2$ pairwise distances into $O(n/\varepsilon^2)$ well-separated pairs (A_i, B_i) (for a fixed separation constant $s = 2/\varepsilon$), where each pair satisfies $dist(A_i, B_i) \geq s \cdot \max(diam(A_i), diam(B_i))$ for a separation constant $s > 0$. It can be constructed in O(n log n) time using a fair split tree or compressed quadtree.

A key consequence, proved in [9, 10], is that any t-spanner with O(n) edges yields a t-approximate Euclidean MST: the MST of the spanner has total weight at most t times the weight of the true Euclidean MST. This result directly motivates our approach. Alternative spanner constructions include the Θ-graph (Theta graph), introduced independently by Clarkson [32] and Keil [33], and Yao graphs [34]. Both achieve O(n) edges and O(n log n) construction time, and provide bounded stretch factors when the number of cones is sufficiently large.

To the best of our knowledge, no prior clustering algorithm has exploited the WSPD simultaneously for spanner-based MST approximation while using the

underlying Fair Split Tree's local cell diameters as a density-sensitive cutting criterion. Our method unifies these two threads in a single O(n log n) algorithm with a provable MST approximation guarantee and a geometrically motivated, scale-aware cutting criterion.

3 Preliminaries

In this section we introduce the mathematical background and key results that underpin our algorithm. We cover basic geometric definitions, the Fair Split Tree and its local cell diameter, the Well-Separated Pair Decomposition, geometric t-spanners constructed via the WSPD, and the central theorem relating the MST of a spanner to the true Euclidean MST.

3.1 Notation and Basic Definitions

Let $S = \{p_1, p_2, \ldots, p_n\}$ be a set of n points in the Euclidean plane $\mathbb{R}^2$. For two points u, v $\in \mathbb{R}^2$, we write $\|uv\|$ to denote the Euclidean distance $\|u - v\|_2$.

Definition 3.1 (Diameter). The diameter of a non-empty set $A \subseteq \mathbb{R}^2$ is

$$\mathrm{diam}(A) = \max_{u,v \in A} \|uv\|.$$

Definition 3.2 (Inter-set Distance). The distance between two non-empty sets $A, B \subseteq \mathbb{R}^2$ is

$$\mathrm{dist}(A, B) = \min_{u \in A, v \in B} \|uv\|.$$

Definition 3.3 (Complete Euclidean Graph). The complete Euclidean graph K(S) is the complete weighted graph on S where the weight of each edge (u, v) equals the Euclidean distance $\|uv\|$. K(S) has $n(n-1)/2 = O(n^2)$ edges.

Definition 3.4 (Minimum Spanning Tree). A Minimum Spanning Tree (MST) of a weighted graph G = (V, E, w) is a spanning tree $T \subseteq E$ that minimizes the total edge weight $w(T) = \sum_{e \in T} w(e)$. The Euclidean MST of S, denoted MST(S), is the MST of K(S).

3.2 Fair Split Tree

The Fair Split Tree (FST), closely related to the compressed quadtree, is the fundamental data structure underlying the WSPD construction.

Definition 3.5 (Fair Split Tree [10]). A Fair Split Tree of S is a rooted binary tree T where:

• Each node v is associated with a subset $S(v) \subseteq S$ and an axis-aligned bounding box R(v).

• The root is associated with S and its bounding box.

• Each internal node v splits R(v) along its longest side into two equal halves, assigning points to child nodes vl and vr accordingly, such that $S(vl) \cup S(vr) = S(v)$ and $S(vl) \cap S(vr) = \emptyset$.

• Each leaf node contains exactly one point of S.

Theorem 3.6 ([10]). A Fair Split Tree of n points in $\mathbb{R}^2$ can be constructed in O(n log n) time and O(n) space.

The Fair Split Tree is the prerequisite for the WSPD: each WSPD pair (Ai, Bi) corresponds to two nodes in T, and the bounding boxes of those nodes directly yield diam(Ai) and diam(Bi) in O(1) per pair.

Definition 3.7 (Local Cell Diameter). During FST construction, alongside the point-set bounding box R(v), each node v also tracks the spatial cell Cell(v): the axis-aligned rectangular region assigned to v by the recursive bisection of the root bounding box. For internal nodes, Cell(v) = R(v); for leaf nodes, Cell(v) retains the full spatial extent of the region even though R(v) degenerates to a single point (since a leaf holds exactly one point of S). For each point $p \in S$, let $\ell(p)$ denote the leaf node of T containing p. The local cell diameter of p is

$$\delta(p) = \mathrm{diam}(\mathrm{Cell}(\ell(p))).$$

Intuitively, δ(p) reflects the local point density around p: in dense regions, cells are subdivided finely so δ(p) is small; in sparse regions, cells remain large and δ(p) is large. Since Cell(ℓ(p)) is maintained during the tree traversal, δ(p) is available in O(1) per point at no extra asymptotic cost.

3.3 Well-Separated Pair Decomposition (WSPD)

Definition 3.8 (Well-Separated Pair [10]). Two non-empty sets A, B ⊆ S are well-separated with respect to a separation constant s > 0 if

$$\mathrm{dist}(A, B) \geq s \cdot \max(\mathrm{diam}(A), \mathrm{diam}(B)).$$

Definition 3.9 (Well-Separated Pair Decomposition [10]). A Well-Separated Pair Decomposition (WSPD) of S with separation constant s > 0 is a sequence of pairs $W = \{(A_i, B_i)\}_{i=1}^{m}$ of non-empty subsets of S such that:

1. (Well-separation) For each i, A_i and B_i are well-separated with respect to s.

2. (Coverage) For every pair of distinct points u, v ∈ S, there exists exactly one index i such that either (u ∈ A_i and v ∈ B_i) or (u ∈ B_i and v ∈ A_i).

Theorem 3.10 ([10]). For any set S of n points in $\mathbb{R}^2$ and any constant separation constant s > 0, a WSPD of S with separation constant s exists and has O(n) pairs. It can be constructed from a Fair Split Tree in O(n log n) time and O(n) space.

An important byproduct of the WSPD construction (Definition 3.9) is that, since each pair (A_i, B_i) corresponds

to two nodes in the Fair Split Tree, the values diam(A_i) and diam(B_i) are available in O(1) per pair with no additional computation.

3.4 Geometric t-Spanners via WSPD

Definition 3.11 (t-Spanner [9]). A graph G = (S, E) with edge weights equal to Euclidean distances is a t-spanner of S if for every pair of points u, v ∈ S,

$$d_G(u, v) \leq t \cdot \|uv\|,$$

where dG(u, v) denotes the shortest path distance between u and v in G, and t ≥ 1 is called the stretch factor (or dilation).

A fundamental result guarantees that for any t > 1, a sparse t-spanner exists and can be built efficiently:

Theorem 3.12 ([9]). For any finite point set S ⊂$\mathbb{R}^2$ and any t > 1, there exists a t-spanner of S with O(n) edges that can be constructed in O(n log n) time.

Theorem 3.12 confirms that the O(n)-edge, O(n log n)-time spanner we construct is optimal in both edge count and construction time up to constants. We construct our t-spanner directly from the WSPD, following the standard construction of Callahan and Kosaraju [10]. For each WSPD pair (A_i, B_i) ∈ W, we select one representative point a_i ∈ A_i and b_i ∈ B_i (typically the point closest to the centroid of its set), and add the edge (a_i, b_i) to the spanner.

Theorem 3.13 ([9, 10]). Let W be a WSPD of S with separation constant s > 0. The graph G obtained by connecting one representative pair per WSPD pair is a t-spanner of S with stretch factor

$$t = 1 + \frac{2}{s}.$$

Therefore, to achieve a desired stretch factor t = 1 + ε for any fixed ε > 0, we set the separation constant to

$$s = \frac{2}{\varepsilon}.$$

Since ε is a fixed constant, s = 2/ε is also a fixed constant, and by Theorem 3.10, the WSPD has O(s^2n) = O(n/ε^2) pairs and is constructed in O(n log n) time. The resulting spanner thus has O(n/ε^2) edges and is built in O(n log n) time. For any fixed ε this edge count is O(n); the constant grows as O(1/ε2) as ε → 0.

Remark 3.14 (Key structural property). In our WSPD-based spanner, every spanner edge (ai, bi) is in exact one-to-one correspondence with its generating WSPD pair (Ai, Bi). This means no additional lookup or traversal is needed during cutting. Furthermore, since ai and bi are leaves in the Fair Split Tree T, their local cell diameters δ(ai) and δ(bi) (Definition 3.7) are immediately available, providing a truly local, density-sensitive scale for the cutting criterion in Section 4.

3.5 MST of a t-Spanner: The Key Theorem

The following theorem is the theoretical cornerstone of our algorithm. It guarantees that the MST of a t-spanner provides a near-optimal approximation to the true Euclidean MST.

Theorem 3.15 ([9, 10]). Let G be a t-spanner of a point set S ⊂$\mathbb{R}^2$. Then

$$w(\mathrm{MST}(G)) \leq t \cdot w(\mathrm{MST}(S)).$$

Proof Sketch. Let T* = MST(S) be the true Euclidean MST. For each edge (u, v) ∈ T*, the t-spanner G contains a path π(u, v) of total length at most t · ‖uv‖. Replacing each edge of T* with the corresponding spanner path yields a connected spanning subgraph H of G with total edge weight at most t · w(T*). Since H is a connected spanning subgraph of G, it contains a spanning tree T' of G with w(T') ≤ w(H) ≤ t · w(T*). Since MST(G) is the minimum-weight spanning tree of G, w(MST(G)) ≤ w(T') ≤ t · w(T*).

Corollary 3.16. Using the WSPD-based spanner with separation constant s = 2/ε and stretch factor t = 1 + ε, the MST of the spanner satisfies

$$w(\mathrm{MST}(G)) \leq (1 + \varepsilon) \cdot w(\mathrm{MST}(S)).$$

For small ε, this is a near-optimal approximation to the true Euclidean MST, obtained in O(n log n) time without constructing the O(n^2)-edge complete Euclidean graph.

4 Proposed Algorithm

In this section we present our clustering algorithm in full detail. The algorithm takes as input a set S of n points in $\mathbb{R}^2$, a stretch parameter ε > 0, and a cutting parameter α > 0, and outputs a partition of S into clusters. The algorithm proceeds in four main phases: (1) construction of the WSPD, (2) construction of the t-spanner from the WSPD, (3) computation of the MST of the spanner, and (4) local adaptive edge cutting to extract clusters.

4.1 Phase 1: WSPD Construction

Given the input point set S and separation constant s = 2/ε, we construct a WSPD W = $\{(A_i, B_i)\}_{i=1}^{m}$ of S using the algorithm of Callahan and Kosaraju [34]. This proceeds in two steps: first a Fair Split Tree T is built over S in O(n log n) time, then W is extracted from T by a recursive traversal. The output is a set of m = O(n/ε^2) well-separated pairs, each

stored as a pair of Fair Split Tree nodes (v_i, w_i) representing A_i and B_i respectively.

For each pair (Ai, Bi), the following quantities are computed and stored:

• **Representative points ai ∈Ai and bi ∈Bi:** any fixed representative from each set, e.g. the point stored at the FST node.

• **δ(ai) and δ(bi):** the local cell diameters (Definition 3.7) of each representative, read from their FST leaf nodes in O(1).

• **λi = max(δ(ai), δ(bi)):** the local scale of edge (ai, bi), used as the reference length in the cutting criterion.

Remark 4.1 (All points appear as vertices). By the coverage property of the WSPD, every point p ∈ S appears in at least one WSPD pair. Since we pick one representative per set, some points may never be selected and would be isolated in the spanner G. To avoid this, whenever a point p is the only member of a set (|Ai| = 1 or |Bi| = 1) it is always selected as that set's representative. For any remaining uncovered points, the second loop of Phase 2 adds one edge to their nearest neighbor in S \ {p}, tagged with λ = max(δ(p), δ(q)) where q is the nearest neighbor. This adds at most O(n) edges and O(n log n) time, preserving overall complexity.

4.2 Phase 2: Spanner Construction

From the WSPD W, we construct the spanner G = (S, E) in two steps. First, for each pair (Ai, Bi) ∈W, we add the edge (ai, bi) with weight ‖aibi‖, tagged with λi = max(δ(ai), δ(bi)), and record both ai and bi as covered. This one-to-one correspondence between spanner edges and WSPD pairs (Remark 3.14) means no additional lookup is needed during cutting. Second, for any point p ∈ S not covered as a representative in the first step, we find its nearest neighbor q ∈ S and add the edge (p, q) tagged with λ = max(δ(p), δ(q)). This guarantees all n points are vertices of G, as required by Lemma 5.5.

By Theorem 3.13 and Lemma 5.6, G is a (1 + ε)-spanner of S with O(n/ε²) edges. The construction takes O(n log n) time (dominated by the nearest-neighbor queries for uncovered points).

4.3 Phase 3: MST of the Spanner

$$w(T) \leq (1+\varepsilon) \cdot w(\mathrm{MST}(S)).$$

4.4 Phase 4: Local Adaptive Edge Cutting

For each edge ei = (ai, bi) ∈T with local scale λi = max(δ(ai), δ(bi)), we remove ei from T if and only if:

$$\|a_i b_i\| > \alpha \cdot \lambda_i.$$

The intuition is as follows. $\delta(a_i)$ is the diameter of the FST spatial cell containing a_i, reflecting the local point density around a_i specifically. By the well-separation condition, every WSPD-derived spanner edge already satisfies $\|a_i b_i\| \geq s \cdot \lambda_i$ (Lemma 5.8), so when using FST cell diameters, α must be set strictly greater than s (see Section 4.6 and Corollary 5.9). Within-cluster MST edges tend to have lengths close to $s \cdot \lambda_i$, while between-cluster edges bridge well-separated regions and are significantly longer. The formal cutting guarantee is characterised by Proposition 5.10. This criterion adapts to local density: in dense regions λ_i is small, so the threshold $\alpha \cdot \lambda_i$ is tight and discriminates fine cluster boundaries; in sparse regions λ_i is large, raising the threshold and preventing over-segmentation. Since $\delta(a_i)$ and $\delta(b_i)$ are precomputed during FST construction at no extra cost, this phase runs in O(n) time. After cutting, the connected components of the remaining forest are returned as clusters.

4.5 Complete Algorithm and Pseudocode

Algorithm 1 summarizes the complete procedure.

4.6 Choice of Parameters

Stretch parameter ε. The parameter ε controls MST approximation quality. Smaller ε gives a tighter bound but increases s = 2/ε and the number of WSPD pairs (which is O(n/ε²), i.e. O(n) for fixed ε but growing as ε → 0). In practice, ε = 1 (s = 2, t = 2) or ε = 0.5 (s = 4, t = 1.5) are reasonable defaults.

Algorithm 1 WSPD-Based Geometric Clustering

Require: Point set S = {p₁, …, pₙ} ⊂ ℝ², stretch parameter ε > 0, cutting parameter α > 0 (see Section 4.6 for constraints on α)
Ensure: A partition of S into clusters C₁, C₂, …
1: s ← 2/ε ▷ Separation constant for stretch t = 1 + ε
2: T ← BUILDFAIRSPLITTREE(S) ▷ O(n log n); stores δ(p) for each p ∈ S
3: W ← BUILDWSPD(T, s) ▷ O(n/ε²) pairs in O(n log n) total
4: G ← empty graph on vertex set S
5: covered ← ∅ ▷ Track which points have been selected as representatives
6: for each pair (A_i, B_i) ∈ W do
7: a_i ← REPRESENTATIVE(A_i); b_i ← REPRESENTATIVE(B_i)
8: λ_i ← max(δ(a_i), δ(b_i)) ▷ Local scale from FST leaf cells
9: Add tagged edge (a_i, b_i, ‖$a_i b_i$‖, λ_i) to G
10: covered ← covered ∪ {a_i, b_i}
11: end for

12: for each p ∈ S \ covered do ▷ Connect any point never chosen as a representative
13: q ← NEARESTNEIGHBOR(p, S \ {p})
14: λ ← max(δ(p), δ(q))
15: Add tagged edge (p, q, ‖pq‖, λ) to G
16: end for ▷ G now spans all n points; |E(G)| = O(n/ε²)
17: T ← KRUSKAL(G) ▷ MST of spanner, O(n log n); each edge carries its λ tag
18: C ← UNIONFIND(S) ▷ Initialize each of the n points as its own component
19: for each edge e = (u, v) ∈ T with tag λ_e do
20: if ‖uv‖ ≤ α · λ_e then
21: UNION(C, u, v) ▷ Within-cluster edge: merge the two components
22: end if ▷ Otherwise: between-cluster edge, leave components separate
23: end for
24: return connected components of C

Cutting parameter α. The parameter α controls how aggressively edges are cut. An edge (ai, bi) is cut if its length exceeds α · λi.

The appropriate range of α depends on the definition of δ(p).

Theoretical (FST leaf cell diameter). When δ(p) is the FST leaf cell diameter, the well-separation condition and Lemmas 5.7–5.8 give:

$$\|a_i b_i\| \geq \mathrm{dist}(A_i, B_i) \geq s \cdot \max(\mathrm{diam}(A_i), \mathrm{diam}(B_i)) \geq s \cdot \lambda_i.$$

‖aibi‖≥ s·λi, so α must satisfy α > s for any such edge to be retained (Corollary 5.9). Within-cluster MST edges tend to have lengths close to s · λi, while between-cluster edges are significantly longer. A practical default of α = c · s for a small constant c > 1 (e.g. c = 1.5 or c = 2) is recommended when using FST cell diameters.

Practical (kNN distance proxy). In implementations where δ(p) is approximated by the distance to the kδ-th nearest neighbour of p, the kNN distance is generally larger than the FST leaf cell diameter: it spans to actual data neighbours rather than reflecting cell boundary geometry. This inflates λi relative to the FST case, effectively rescaling all thresholds upward. As a consequence, the constraint α > s no longer applies in the same form: a smaller α (even α < s) may suffice to retain within-cluster edges while still cutting between-cluster ones. In practice, α = 0.6 · s with kδ = 5 performs well across a range of datasets (Section 6).

Larger α produces fewer, larger clusters; smaller α produces more, finer clusters. The optimal value depends on the expected cluster separation and the choice of δ(p).

5 Analysis

In this section we analyze the correctness and complexity of the proposed algorithm. We prove the time and space complexity, establish the MST approximation guarantee, verify that the spanner correctly spans all n points, and formally characterize the properties of the local adaptive cutting criterion.

5.1 Time Complexity

Theorem 5.1 (Time Complexity). The proposed algorithm runs in O(n log n) time.

Proof. We analyze each phase in turn.

Phase 1 (WSPD Construction). The Fair Split Tree T is constructed in O(n log n) time by Theorem 3.6. During construction, the local cell diameter δ(p) for each point p ∈ S is read from its leaf spatial cell in O(1), contributing O(n) total. The WSPD W is then extracted from T in O(n log n) time by Theorem 3.10, producing m = O(n/ε²) pairs. For each pair, the representative selection and computation of λi take O(1), contributing O(n/ε²) total. Phase 1 therefore runs in O(n log n) time (for any fixed ε).

Phase 2 (Spanner Construction). One edge is added per WSPD pair in O(1) each, giving O(n/ε²) time for the WSPD edges. The coverage pass checks each of the n points against the covered set in O(1) using a boolean array. For any uncovered point, a nearest-neighbor query is performed in O(log n) using the FST (or a pre-built k-d tree), contributing O(n log n) worst-case. Phase 2 therefore runs in O(n/ε² + n log n) = O(n log n) time (for fixed ε).

Phase 3 (MST of the Spanner). The spanner G has |E| = O(n/ε²) edges (for fixed ε). Kruskal's algorithm [35] sorts the edges in O((n/ε²) log n) = O(n log n) time and performs O(n/ε²) union-find operations, each taking O(α_A(n)) amortized time, where α_A denotes the inverse Ackermann function (distinct from the cutting parameter α). For fixed ε this is O(n) total. Phase 3 therefore runs in O(n log n) time.

Phase 4 (Edge Cutting). The MST T has exactly n − 1 edges. For each edge, the cutting criterion evaluates ‖$a_i b_i$‖ > α · λ_i in O(1), since both quantities are precomputed and stored as edge attributes. Phase 4 runs in O(n) time.

Cluster Extraction. The connected components of the remaining forest are read from the union-find structure C in O(n) time.

Summing over all phases, the dominant cost is O(n log n), proving the theorem.

5.2 Space Complexity

Theorem 5.2 (Space Complexity). For any fixed $\varepsilon > 0$, the proposed algorithm uses $O(n/\varepsilon^2)$ space.

Proof. The Fair Split Tree T has O(n) nodes and uses O(n) space. The WSPD W has $O(s^2n) = O(n/\varepsilon^2)$ pairs, each stored as a pair of FST node pointers plus O(1) auxiliary data (ai, bi, λi), contributing $O(n/\varepsilon^2)$ space. The spanner G has $O(n/\varepsilon^2)$ edges. The MST T has n − 1 edges. The union-find structure C uses O(n) space. The total space is therefore $O(n/\varepsilon^2)$, which is O(n) for any fixed ε.

5.3 MST Approximation Quality

Theorem 5.3 (MST Approximation). Let T = MST(G) be the MST of the spanner G produced by the algorithm with stretch parameter $\varepsilon > 0$. Then

$$w(T) \leq (1 + \varepsilon) \cdot w(\mathrm{MST}(S)).$$

Proof. By construction, G is a $(1 + \varepsilon)$-spanner of S, as guaranteed by Theorem 3.13 with separation constant $s = 2/\varepsilon$. The result then follows directly from Theorem 3.15.

Corollary 5.4. For any fixed $\varepsilon > 0$, the approximation factor $1 + \varepsilon$ is a user-controlled constant independent of n.

Choosing smaller ε gives a tighter MST approximation, at the cost of a larger separation constant $s = 2/\varepsilon$. The number of WSPD pairs is $O(s^2n)$ in $\mathbb{R}^2$ (see [34]), which is O(n) for any fixed s (equivalently, fixed ε); however, the constant hidden in the $O(\cdot)$ grows as $O(1/\varepsilon^2)$ as $\varepsilon \to 0$.

5.4 Correctness of the Spanner

We formally verify that the spanner G is a valid $(1 + \varepsilon)$-spanner that spans all n points of S.

Lemma 5.5 (Full Vertex Coverage). Every point $p \in S$ is a vertex of the spanner G.

Proof. The algorithm maintains a boolean array covered[p] for each $p \in S$. In the WSPD loop, every selected representative ai and bi is marked covered. In the second loop of Phase 2, any point p with covered[p] = false is given a nearest-neighbor edge, making it incident to at least one edge in G. Hence every point in S is a vertex of G.

Lemma 5.6 (Spanner Validity and Connectivity). The graph G constructed in Phase 2 is a $(1 + \varepsilon)$-spanner of S, and in particular is connected.

Proof. This follows directly from Theorem 3.13: the WSPD W with separation constant $s = 2/\varepsilon$ yields a spanner with stretch factor $t = 1 + 2/s = 1 + \varepsilon$. The additional nearest-neighbor edges added for uncovered points only reduce shortest path distances in G, preserving the stretch factor. Connectivity follows immediately: a t-spanner by definition satisfies $d_G(u, v) \leq t\|uv\| < \infty$ for every pair $u, v \in S$, so every pair of vertices is connected by a path in G, making G connected. Connectivity is required for Kruskal's algorithm to produce a spanning tree (rather than a spanning forest) in Phase 3.

5.5 Properties of the Cutting Criterion

We now formally establish the key properties of the local adaptive cutting criterion.

Lemma 5.7 (Local Scale Bound). For each WSPD pair (Ai, Bi) with representative edge (ai, bi),

$\lambda_i = \max(\delta(a_i), \delta(b_i)) \leq \max(\mathrm{diam}(A_i), \mathrm{diam}(B_i))$.

Proof. Since $a_i \in A_i$, the leaf node $\ell(a_i)$ is a descendant of the FST node v_i representing A_i. The spatial cell $\mathrm{Cell}(\ell(a_i))$ is therefore a sub-cell of $\mathrm{Cell}(v_i)$, so

$$\delta(a_i) = \mathrm{diam}(\mathrm{Cell}(\ell(a_i))) \leq \mathrm{diam}(\mathrm{Cell}(v_i)).$$

In the Callahan–Kosaraju construction, $\mathrm{diam}(A_i)$ in the well-separation condition refers to $\mathrm{diam}(\mathrm{Cell}(v_i))$ (the diameter of the spatial cell, not merely of the finite point set), so $\delta(a_i) \leq \mathrm{diam}(A_i)$. By the same argument, $\delta(b_i) \leq \mathrm{diam}(B_i)$. Taking the maximum of both sides gives the result.

Lemma 5.8 (Lower Bound on Spanner Edge Length). For each WSPD-derived spanner edge (ai, bi) with local scale λi,

$$\|a_i b_i\| \geq s \cdot \lambda_i.$$

Corollary 5.9 (Necessary Condition on α under FST Cell Diameter). When δ(p) is the FST leaf cell diameter, for the cutting criterion to retain any WSPD-derived MST edge, the parameter α must satisfy $\alpha > s$.

If $\alpha \leq s$, then by Lemma 5.8 every WSPD-derived spanner edge satisfies $\|a_ib_i\| \geq s \cdot \lambda_i \geq \alpha \cdot \lambda_i$; the cutting rule removes an edge only when $\|a_ib_i\| > \alpha \cdot \lambda_i$, so every edge with $\|a_ib_i\| > \alpha \cdot \lambda_i$ is cut. Since $\|a_ib_i\| \geq s \cdot \lambda_i > \alpha \cdot \lambda_i$ holds strictly whenever $\alpha < s$, all WSPD-derived edges are cut, leaving all points as singletons. (Nearest-neighbor edges added for uncovered points are not subject to this lower bound and may or may not be cut depending on α, but since they connect points within the same dense region, they are unlikely to determine cluster structure.)

The following proposition characterizes when the cutting criterion correctly separates two clusters under an idealized geometric model.

Proposition 5.10 (Bridge-Edge Cutting under Idealized Model). Suppose S consists of two disjoint subsets C1 and C2 such that every point in Ck is within distance rk of the centroid of Ck (k = 1, 2), and the centroids

are distance D apart. Let rmax = max(r1, r2) and suppose α > s. If

$$\alpha < \frac{D - r_1 - r_2}{2r_{\max}},$$

then the cutting criterion cuts every cross-cluster spanner edge (in particular the unique MST edge bridging C1 and C2). Furthermore, if every within-cluster spanner edge (aj, bj) additionally satisfies ‖ajbj‖≤α · λj, the two clusters are correctly recovered.

Proof Sketch. Bridge edge is cut. Let (ai, bi) be any cross-cluster spanner edge with ai ∈C1, bi ∈C2. By the triangle inequality, ‖aibi‖≥D −r1 −r2. Since the clusters are well-separated (D −r1 −r2 > 0), the FST recursive bisection will place all points of C1 and all points of C2 in separate subtrees before subdividing within each cluster; consequently, the spatial cell Cell(ℓ(ai)) is entirely contained within the spatial region assigned to C1's subtree, whose diameter is at most 2r1. Hence δ(ai) ≤2r1 ≤2rmax; similarly δ(bi) ≤2rmax, giving λi ≤2rmax. Hence

$$\frac{\|a_i b_i\|}{\lambda_i} \geq \frac{D - r_1 - r_2}{2r_{\max}} > \alpha,$$

so ‖aibi‖> α · λi and the edge is cut. The within-cluster retention then follows directly from the stated assumption.

Remark 5.11. The upper bound on α in Proposition 5.10 is (D − r₁ − r₂)/(2r_max). When using the FST cell diameter (so that the constraint α > s applies), a valid α exists whenever this upper bound exceeds s, i.e.

$$\frac{D - r_1 - r_2}{2r_{\max}} > s \iff D > 2sr_{\max} + r_1 + r_2.$$

Since r1, r2 ≤rmax, a sufficient condition is D > (2s + 2) rmax, which is precisely the condition that the two clusters are well-separated at scale s, consistent with the WSPD design.

5.6 Comparison with Prior Work

Table 1 situates our method within the landscape of MST-based clustering algorithms, comparing time complexity, space complexity, whether a provable MST weight guarantee is established, and whether the edge-cutting criterion is density-adaptive. We discuss each entry below.

Classical methods ([3, 24]). These methods build the complete Euclidean graph K(S) with O(n²) edges and compute the exact MST from it, leading to O(n²) time and space. Clusters are extracted by removing edges according to a global heuristic criterion (inconsistency or standard-deviation thresholding), with no provable approximation guarantee and no adaptation to local density.

Divide-and-conquer approximate MST ([28]). Zhong et al. employ k-means to partition the dataset into O(√n) clusters and construct local MSTs within each partition, yielding an approximate MST in O(n³/²) time and space. No provable bound on the MST weight is established, and the cutting criterion remains global.

Table 1: Comparison of MST-based clustering algorithms. n denotes the number of input points in ℝ². MST Bound indicates whether the algorithm provides a provable worst-case bound on the weight of the MST it computes relative to the true Euclidean MST weight. Adaptive Cut indicates whether the edge-cutting criterion adapts to local point density rather than applying a global threshold.

Method	Time	Space	MST Bound	Adaptive Cut
Zahn [18]	O(n²)	O(n²)	None	No
Grygorash et al. [19]	O(n²)	O(n²)	None	No
Zhong et al. [21]	O(n³/²)	O(n³/²)	None	No
Jothi et al. [27]	O(n³/² log n)	O(n³/²)	None	No
Akhter et al. [28]	O(n³/²)	O(n³/²)	None	No
Exact EMST [29, 30]	O(n log n)	O(n)	= OPT	No
Ours	O(n log n)	O(n/ε²)†	≤ (1 + ε) · OPT	Yes

† Space (and edge count) is O(n/ε²) for fixed ε > 0; the constant grows as ε → 0. All other entries assume ε is a fixed constant.

Local Neighborhood Graph methods ([7, 8]). Jothi et al. [7]construct a sparse Local Neighborhood Graph (LNG) of size O(n³/²) using a centroid-based nearest-neighbor rule, and compute the approximate MST from LNG in O(n³/² log n) time. Akhter et al. refine this approach with a two-level partitioning strategy, reducing graph construction to O(n³/²). Neither method provides a worst-case approximation ratio on the MST weight, and both use global cutting criteria.

Exact Euclidean MST via Delaunay triangulation [36, 37]. The Euclidean MST of n points in ℝ² can be computed exactly in O(n log n) time and O(n) space: the Delaunay triangulation, which is the dual of the Voronoi diagram and can be built in O(n log n) time [37], contains the EMST as a subgraph; running Kruskal's or Prim's algorithm on its O(n) edges then yields the exact MST [36]. Our method matches this O(n log n) time complexity and uses O(n/ε²) space (linear for any fixed ε). The key distinction is not speed but architecture: the exact EMST approach provides no

principled clustering mechanism—a separate, typically global, edge-cutting step must be appended. Our method integrates the $(1 + \varepsilon)$-approximate MST and the density-adaptive cutting criterion within a single unified WSPD framework, with the FST serving both steps at no extra cost.

In summary, our algorithm matches the $O(n \log n)$ time of the exact EMST approach and uses $O(n/\varepsilon^2)$ space (which is $O(n)$ for any fixed ε), while simultaneously providing (i) a provable $(1 + \varepsilon)$-approximation guarantee on the MST weight for any user-chosen $\varepsilon > 0$, and (ii) a density-adaptive edge-cutting criterion grounded in the local geometry of the Fair Split Tree. To our knowledge, our method is the only MST-based clustering algorithm that combines all four of: optimal $O(n \log n)$ time complexity, $O(n/\varepsilon^2)$ space, a provable MST approximation guarantee, and a density-adaptive cutting criterion.

6 Experiments

We evaluate the proposed algorithm on two sets of experiments: synthetic Gaussian datasets of varying cluster count and separation (Section 6.1), and standard 2D shape benchmarks from the clustering literature (Section 6.2). All experiments are conducted in Python 3.12 using a pure-Python implementation of the WSPD and Fair Split Tree; a C/C++ implementation would substantially reduce wall-clock time. The implementation of the proposed algorithm is publicly available at https://github.com/drmheydari/clustering.

Implementation note on $\delta(p)$. The theoretical definition of $\delta(p)$ is the diameter of the FST leaf cell containing p. In the Python implementation we approximate this by the distance from p to its $k\delta$-th nearest neighbour (default $k\delta = 5$), computed in $O(n \log n)$ via a k-d tree. The kNN distance is typically larger than the FST leaf cell diameter: it spans to actual data neighbours rather than reflecting axis-aligned cell boundaries. This inflates λi, which effectively rescales

Baselines. We compare against four established clustering methods: Single-Linkage hierarchical clustering [14], k-Means [4], the Local Neighborhood Graph (LNG) method of Jothi et al. [6, 7] (the closest prior MST-based approach in our comparison table), and DBSCAN [2]. All baselines are implemented using scikit-learn [38] (v1.4) with default settings except where noted. LNG is reimplemented in Python following the description in [7] using a k-d tree for nearest-neighbour queries. DBSCAN parameters (neighbourhood radius ε_db and min_samples) are tuned per dataset via the elbow method on the k-nearest-neighbour distance plot, with $k = 5$. For methods requiring k as input (Single-Linkage, k-Means, LNG), the true k is supplied. DBSCAN requires no k; it detects clusters automatically based on density parameters (neighbourhood radius and minimum point count), which were tuned per dataset.

Metrics. We report three standard external clustering quality indices: Adjusted Rand Index (ARI) [39], Normalised Mutual Information (NMI), and Fowlkes-Mallows Index (FMI). All three lie in $[0, 1]$ (or $[-1, 1]$ for ARI) with higher values indicating better agreement with ground-truth labels. We also report $\hat{k}$, the number of clusters detected by each method.

6.1 *Synthetic Gaussian Datasets*

We generate five 2D datasets of isotropic Gaussian clusters using NumPy with random seed 42 for full reproducibility. Datasets vary in number of clusters k, inter-cluster separation, and within-cluster density structure:

- **Gaussian (k=3, sep=8) (n = 300):** Three equally sized, well-separated Gaussian clusters ($\sigma = 0.5$, centre-to-centre distance 8). This is the canonical setting covered by Proposition 5.10.
- **Gaussian (k=5, sep = 8) (n = 500):** Five clusters at the same separation, testing scalability in k.
- **Gaussian (k=3, sep = 5) (n = 300):** Reduced separation ($\sigma = 0.5$, distance 5), probing robustness near the boundary of our theoretical regime.
- **Varying Density (n = 400):** Three clusters with centre-to-centre separation of 6 units and standard deviations 0.2, 0.8, and 1.5, testing the adaptive cutting criterion against global-threshold methods.
- **Multi-Scale Gaussian (n = 300):** Three well-separated clusters with centre-to-centre separation of 8 units and strongly heterogeneous densities ($\sigma \in \{0.15, 0.50, 1.20\}$), specifically designed to challenge global-threshold baselines.

Results are shown in Table 2. Our method achieves perfect scores (ARI = 1.000) on all three well-separated Gaussian datasets and on the multi-scale dataset, matching the best baselines. The results for Gaussian ($k = 3$) with sep = 5 and sep = 8 are numerically identical across all methods: both separations are large enough relative to $\sigma = 0.5$ that all methods achieve their maximum quality, with LNG obtaining ARI = 0.990 due to minor graph-construction artefacts independent of separation at this scale. On Varying Density, our method obtains ARI = 0.576, which outperforms Single-Linkage and LNG on ARI (ARI = 0.530)

while matching them on NMI; k-Means and DBSCAN perform better here, as they exploit global density information that our purely local criterion does not. This is consistent with the theoretical model: our Proposition 5.10 assumes clusters of comparable density; datasets with extreme density ratios between clusters lie outside this regime.

6.2 Standard 2D Shape Benchmarks

We evaluate on two standard 2D point-set clustering benchmarks that are widely used in the geometric clustering literature [40]:

• **R15 [40] (n = 600, k = 15):** Fifteen Gaussian clusters of equal size arranged in a 5 × 3 grid with uniform inter-cluster spacing and standard deviation 0.6. This benchmark tests scalability in k and the ability to detect many small, closely-packed clusters.

Table 2: Clustering quality on synthetic Gaussian datasets. n = number of points; $\hat{k}$ = detected number of clusters. Bold entries mark the best value in each column per dataset (ties included).

Dataset	Method	ARI ↑	NMI ↑	FMI ↑	$\hat{k}$
Gaussian (k=3, sep=8) n=300	Ours	1.000	1.000	1.000	3
	Single-Link	1.000	1.000	1.000	3
	k-Means	1.000	1.000	1.000	3
	LNG	0.990	0.983	0.993	3
	DBSCAN	0.980	0.963	0.986	3
Gaussian (k=5, sep=8) n=500	Ours	1.000	1.000	1.000	5
	Single-Link	1.000	1.000	1.000	5
	k-Means	1.000	1.000	1.000	5
	LNG	0.990	0.986	0.992	5
	DBSCAN	0.977	0.969	0.982	5
Gaussian (k=3, sep=5) n=300	Ours	1.000	1.000	1.000	3
	Single-Link	1.000	1.000	1.000	3
	k-Means	1.000	1.000	1.000	3
	LNG	0.990	0.983	0.993	3
	DBSCAN	0.980	0.963	0.986	3
Varying Density n=400	Ours	0.576	0.697	0.778	3
	Single-Link	0.530	0.698	0.773	3
	k-Means	0.963	0.934	0.977	3
	LNG	0.530	0.698	0.773	3
	DBSCAN	0.936	0.901	0.960	4
Multi-Scale Gaussian n=300	Ours	1.000	1.000	1.000	3
	Single-Link	1.000	1.000	1.000	3
	k-Means	1.000	1.000	1.000	3
	LNG	0.985	0.977	0.990	3
	DBSCAN	0.903	0.909	0.935	4

• **Aggregation [41] (n = 788, k = 7):** Seven Gaussian clusters of varying size and density, arranged in an irregular layout. The heterogeneity in cluster density makes this a challenging test for global-threshold methods.

Results are shown in Table 3. On Aggregation, our method achieves perfect scores across all three metrics, outperforming LNG and DBSCAN. On R15 our method obtains ARI = 0.815 and NMI = 0.955, both exceeding Single-Linkage and LNG (NMI = 0.935), though k-Means (NMI = 0.994) and DBSCAN (NMI = 0.990) achieve higher NMI by correctly detecting all 15 clusters. The lower ARI on R15 relative to k-Means and DBSCAN is expected: with 15 tightly spaced clusters and kδ = 5 nearest neighbours, the local scale estimate δ(p) for boundary points occasionally includes neighbours from adjacent clusters, slightly inflating λ_i and causing two neighbouring clusters to merge ($\hat{k}$ = 13 instead of 15). The high NMI indicates that most cluster structure is correctly recovered.

Summary. Across all well-separated Gaussian datasets our method matches or exceeds all baselines. On datasets that violate the separation assumption (Varying Density, R15 with 15 closely spaced clusters), performance degrades gracefully: ARI drops but NMI remains competitive, indicating that most cluster structure is still recovered. These results confirm that the WSPD-based local adaptive criterion is effective precisely in the geometric regime for which it is theoretically justified.

7 Conclusion

We presented an O(n log n)-time, O(n/ε²)-space geometric clustering algorithm built on a WSPD-based (1 + ε)-spanner and a local adaptive edge-cutting criterion that uses FST leaf cell diameters as density-sensitive thresholds. The approach provides a provable MST approximation guarantee absent from prior sparse-graph methods, while the Fair Split Tree serves both the spanner construction and the cutting criterion at no extra asymptotic cost. Experiments confirm perfect clustering on well-separated datasets and competitive performance on denser benchmarks. Future work includes extending the correctness analysis to general k-cluster configurations, generalising to $\mathbb{R}^d$, and providing tighter theoretical guarantees for the kNN-distance proxy of δ(p).

Table 3: Clustering quality on standard 2D shape benchmarks. Datasets reproduced from published cluster statistics; coordinates standardised before all methods. Bold entries mark the best value in each column per dataset (ties included).

Dataset	Method	ARI ↑	NMI ↑	FMI ↑	$\hat{k}$
R15 [41] n=600, k=15	Ours	0.815	0.955	0.841	13
	Single-Link	0.764	0.935	0.800	15
	k-Means	0.993	0.994	0.993	15
	LNG	0.764	0.935	0.800	15
	DBSCAN	0.989	0.990	0.990	15
Aggregation [42] n=788, k=7	Ours	1.000	1.000	1.000	7
	Single-Link	1.000	1.000	1.000	7
	k-Means	1.000	1.000	1.000	7
	LNG	0.990	0.988	0.991	7
	DBSCAN	0.982	0.978	0.985	7

Ethical Considerations

All procedures performed in this study were under the ethical standards.

Acknowledgments

Authors thank all who helped us through this study.

Conflict of Interest

The authors report no conflict of interest.

Funding

According to the authors, this article has no financial support.

References

[1] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, pp. 888-905, 2000, doi: 10.1109/34.868688.

[2] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, 1996, pp. 226-231.

[3] C. T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *IEEE Transactions on Computers,* vol. C-20, no. 1, pp. 68-86, 1971, doi: 10.1109/T-C.1971.223083.

[4] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory,* vol. 28, no. 2, pp. 129-137, 1982, doi: 10.1109/TIT.1982.1056489.

[5] J. C. Gower and G. J. S. Ross, "Minimum spanning trees and single linkage cluster analysis," *Journal of the Royal Statistical Society: Series C,* vol. 18, no. 1, pp. 54-64, 1969, doi: 10.2307/2346439.

[6] R. Jothi, S. K. Mohanty, and A. Ojha, "Fast minimum spanning tree based clustering algorithms on local neighborhood graph," in *Proc. Graph-Based Representations in Pattern Recognition (GbRPR)*, 2015, vol. LNCS 9069, pp. 292-301, doi: 10.1007/978-3-319-18224-7_29.

[7] R. Jothi, S. K. Mohanty, and A. Ojha, "Fast approximate minimum spanning tree based clustering algorithm," *Neurocomputing,* vol. 272, pp. 542-557, 2018, doi: 10.1016/j.neucom.2017.07.038.

[8] M. M. Akhter, A. A. Khan, R. Maheshwari, R. Jothi, and S. K. Mohanty, "A fast sparse graph based clustering technique using dispersion of data points," *Neurocomputing,* vol. 618, p. 129054, 2025, doi: 10.1016/j.neucom.2024.129054.

[9] G. Narasimhan and M. Smid, *Geometric Spanner Networks*. Cambridge University Press, 2007.

[10] P. B. Callahan and S. R. Kosaraju, "A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields," *Journal of the ACM,* vol. 42, no. 1, pp. 67-90, 1995, doi: 10.1145/200836.200853.

[11] L. Kaufman and P. J. Rousseeuw, "Clustering by means of medoids," in *Statistical Data Analysis Based on the L1 Norm*, 1987, pp. 405-416.

[12] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum Press, 1981.

[13] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society: Series B,* vol. 39, no. 1, pp. 1-38, 1977, doi: 10.1111/j.2517-6161.1977.tb01600.x.

[14] P. H. A. Sneath and R. R. Sokal, *Numerical Taxonomy*. San Francisco: Freeman, 1973.

[15] J. H. Ward, "Hierarchical grouping to optimize an objective function," *Journal of the American Statistical Association,* vol. 58, no. 301, pp. 236-244, 1963, doi: 10.1080/01621459.1963.10500845.

[16] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: An efficient data clustering method for large databases," in *Proc. ACM SIGMOD*, 1996, pp. 103-114, doi: 10.1145/235968.233324.

[17] M. Ankerst, M. M. Breunig, H. P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," in *Proc. ACM SIGMOD*, 1999, pp. 49-60, doi: 10.1145/304182.304187.

[18] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Proc. PAKDD*, 2013, vol. LNCS 7819, pp. 160-172, doi: 10.1007/978-3-642-37456-2_14.

[19] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science,* vol. 344, no. 6191, pp. 1492-1496, 2014, doi: 10.1126/science.1242072.

[20] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," in *Proc. ACM SIGMOD*, 1998, pp. 94-105, doi: 10.1145/276304.276314.

[21] W. Wang, J. Yang, and R. Muntz, "STING: A statistical information grid approach to spatial data mining," in *Proc. VLDB*, 1997, pp. 186-195.

[22] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, no. 5, pp. 603-619, 2002, doi: 10.1109/34.1000236.

[23] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science,* vol. 315, no. 5814, pp. 972-976, 2007, doi: 10.1126/science.1136800.

[24] O. Grygorash, Y. Zhou, and Z. Jorgensen, "Minimum spanning tree based clustering algorithms," in *Proc. IEEE ICTAI*, 2006, pp. 73-81, doi: 10.1109/ICTAI.2006.83.
[25] Y. Ma, H. Lin, Y. Wang, H. Huang, and X. He, "A multi-stage hierarchical clustering algorithm based on centroid of tree and cut edge constraint," *Information Sciences,* vol. 557, pp. 194-219, 2021, doi: 10.1016/j.ins.2020.12.016.
[26] C. Zhong, D. Miao, and R. Wang, "A graph theoretical clustering method based on two rounds of minimum spanning trees," *Pattern Recognition,* vol. 43, no. 3, pp. 752-766, 2010, doi: 10.1016/j.patcog.2009.07.010.
[27] C. Zhong, D. Miao, and P. Fränti, "Minimum spanning tree based split-and-merge: A hierarchical clustering method," *Information Sciences,* vol. 181, no. 16, pp. 3397-3410, 2011, doi: 10.1016/j.ins.2011.04.013.
[28] C. Zhong, M. Malinen, D. Miao, and P. Fränti, "A fast minimum spanning tree algorithm based on K-means," *Information Sciences,* vol. 295, pp. 1-17, 2015, doi: 10.1016/j.ins.2014.10.012.
[29] H. K. Farag, F. S. Salem, A. E. Taha, and F. Murtagh, "A new data clustering algorithm based on inconsistent distance methodology," *Expert Systems with Applications,* vol. 129, pp. 296-310, 2019, doi: 10.1016/j.eswa.2019.03.051.
[30] H. Du, D. Lu, Z. Wang, C. Ma, X. Shi, and X. Wang, "Fast clustering algorithm based on MST of representative points," *Mathematical Biosciences and Engineering,* vol. 20, no. 9, pp. 15830-15858, 2023, doi: 10.3934/mbe.2023705.
[31] A. A. Khan and S. K. Mohanty, "L-ASCRA: A linearithmic time approximate spectral clustering algorithm using topologically-preserved representatives," *IEEE Transactions on Knowledge and Data Engineering,* vol. 36, no. 12, pp. 8643-8654, 2024, doi: 10.1109/TKDE.2024.3483572.
[32] K. L. Clarkson, "Approximation algorithms for shortest path motion planning," in *Proc. 19th ACM Symposium on Theory of Computing (STOC)*, 1987, pp. 56-65, doi: 10.1145/28395.28402.
[33] J. M. Keil, "Approximating the complete Euclidean graph," in *Proc. 1st Scandinavian Workshop on Algorithm Theory (SWAT)*, 1988, vol. LNCS 318, pp. 208-213, doi: 10.1007/3-540-19487-8_23.
[34] A. C. Yao, "On constructing minimum spanning trees in k-dimensional spaces and related problems," *SIAM Journal on Computing,* vol. 11, no. 4, pp. 721-736, 1982, doi: 10.1137/0211059.
[35] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society,* vol. 7, no. 1, pp. 48-50, 1956, doi: 10.1090/S0002-9939-1956-0078686-7.
[36] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*. New York: Springer-Verlag, 1985.
[37] M. I. Shamos and D. Hoey, "Closest-point problems," in *Proc. 16th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, 1975, pp. 151-162, doi: 10.1109/SFCS.1975.8.
[38] F. Pedregosa and et al., "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825-2830, 2011.
[39] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification,* vol. 2, no. 1, pp. 193-218, 1985, doi: 10.1007/BF01908075.
[40] C. J. Veenman, M. J. T. Reinders, and E. Backer, "A maximum variance cluster algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, no. 9, pp. 1273-1280, 2002, doi: 10.1109/TPAMI.2002.1033218.
[41] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation," *ACM Transactions on Knowledge Discovery from Data,* vol. 1, no. 1, pp. 1-30, 2007, doi: 10.1145/1217299.1217303.