Fast Euclidean OPTICS with Bounded Precision in Low Dimensional Space

Junhao Gan
University of Queensland
j.gan@uq.edu.au

Yufei Tao*
Chinese University of Hong Kong
taoyf@cse.cuhk.edu.hk

ABSTRACT

OPTICS is a popular method for visualizing multidimensional clusters. All the existing implementations of this method have a time complexity of $O(n^2)$ — where n is the size of the input dataset — and thus, may not be suitable for datasets of large volumes. This paper alleviates the problem by resorting to approximation with guarantees. The main result is a new algorithm that runs in $O(n \log n)$ time under any fixed dimensionality, and computes a visualization that has provably small discrepancies from that of OPTICS. As a side product, our algorithm gives an index structure that occupies linear space, and supports the cluster group-by query with near-optimal cost. The quality of the cluster visualizations produced by our techniques and the efficiency of the proposed algorithms are demonstrated with an empirical evaluation on real data.

CCS CONCEPTS

• **Information systems → Clustering**;

KEYWORDS

OPTICS; Density-Based Clustering; Visualizations

ACM Reference Format:
Junhao Gan and Yufei Tao. 2018. Fast Euclidean OPTICS with Bounded Precision in Low Dimensional Space. In *SIGMOD'18: 2018 International Conference on Management of Data, June 10–15, 2018, Houston, TX, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3183713.3196922

1 INTRODUCTION

DBSCAN [10], a well-known *density-based clustering* method, is able to discover clusters of arbitrary shapes. Let P be a set of multi-dimensional points. Define the *vicinity* of a point $p \in P$ as the ball $B(p, \epsilon)$ centered at p with radius ϵ. DBSCAN enforces two rules:

- A point p is an *ϵ-core point* if $B(p, \epsilon)$ covers at least *minPts* points, where *minPts* is a constant parameter.
- If p is an ϵ-core point, all the points in $B(p, \epsilon)$ should appear in the same cluster as p.

*The research of Yufei Tao was partially supported by a direct grant (Project Number: 4055079) from CUHK and by a Faculty Research Award from Google.

SIGMOD'18, June 10–15, 2018, Houston, TX, USA

ACM ISBN 978-1-4503-4703-7/18/06…$15.00
https://doi.org/10.1145/3183713.3196922

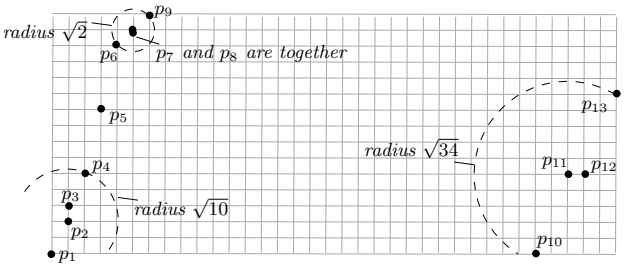


Figure 1: Dataset for our running example

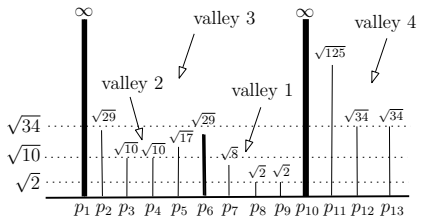


Figure 2: An OPTICS diagram

To form a cluster C, start with an empty C, and add to it an arbitrary ϵ-core point. Then, grow C as follows: for every ϵ-core point $p \in C$, add all the points in $B(p, \epsilon)$ to C (this may bring new ϵ-core points into C), and repeat this until the size of C no longer increases.

Multiple ϵ values can give meaningful clusters. Consider the dataset in Figure 1. Assuming *minPts* = 4, DBSCAN returns

- For $\epsilon = \sqrt{2}$: only one cluster $\{p_6, p_7, p_8, p_9\}$, while all the other points are treated as noise because their vicinities are not dense enough at this ϵ;
- For $\epsilon = \sqrt{10}$: two clusters $\{p_6, p_7, p_8, p_9\}$ and $\{p_1, p_2, p_3, p_4\}$, while p_{10}, p_{11}, p_{12}, and p_{13} are noise;
- For $\epsilon = \sqrt{34}$: also two clusters but different from above: $\{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9\}$ and $\{p_{10}, p_{11}, p_{12}, p_{13}\}$.

Which of the above clusterings make sense? The answer is: all of them. These ϵ values produce clusterings at different "zoom levels". Intuitively, if we imagine each point as a residential area, the clustering of $\epsilon = \sqrt{2}$ could be thought of being at the "city" granularity, that of $\epsilon = \sqrt{10}$ at the "country" granularity, while that of $\epsilon = \sqrt{34}$ at the "continent" granularity.

Also worth noting is the fact that, $\sqrt{2}$, $\sqrt{10}$, and $\sqrt{34}$ are "watershed values" for the parameter ϵ in the following sense: at all other values of ϵ, the clustering is either highly similar to one of the three clusterings shown, or hardly meaningful (e.g., $\epsilon = 0$ gives no clusters at all, and $\epsilon = \infty$ puts everything into a single cluster). It is, therefore, important to identify such watershed values. This, unfortunately, is not easy because those values are highly data-dependent. For small P, it would be feasible to try out numerous ϵ values for a direct comparison, but such a brute-force approach fails to scale with $|P|$.

A dominant approach to tackle the above challenge is to produce a visualization that reveals key information on the *structure* of the clusters, so that humans can use their perceptive instincts to identify interesting parameter values. For DBSCAN, the most popular method towards that purpose is OPTICS [3], which allows one to visualize the forming of clusters at all scales of ϵ simultaneously. Deferring a detailed introduction about OPTICS to Section 2.2, next we illustrate its main features with Figure 2, which is an *OPTICS diagram* for the dataset in Figure 1.

The horizontal axis enumerates the points in a certain order, while a vertical bar is displayed for each point. These bars differ in heights (the number above a bar gives its height), thus creating *valleys*: the four most conspicuous valleys have been indicated in the figure. Every valley reflects a cluster in the dataset; furthermore, the heights of the bars at the bottom of the valley indicate the lowest value of ϵ that will discover the cluster. From Figure 2, one can see the "morphing of clustering" as ϵ increases. At $\epsilon = \sqrt{2}$, Valley 1 (i.e., the first cluster) shows up. At $\epsilon = \sqrt{10}$, Valley 2 (the second cluster) appears. As ϵ grows further, Valleys 1 and 2 merge into Valley 3 (the two clusters are combined into one cluster). When ϵ reaches $\sqrt{34}$, Valley 4 surfaces (recall that there are two clusters at $\epsilon = \sqrt{34}$, as indicated by Valleys 3 and 4).

The morphing behavior illustrated is reminiscent of the *hierarchical clustering* approach as exemplified by HDBSCAN [8]. Assuming a fixed *minPts*, HDBSCAN encodes compactly the clusterings of DBSCAN at all possible ϵ values (precisely speaking, the clusterings encoded in HDBSCAN are slightly different from those of DBSCAN, but the differences are minor). Upon receiving a specific value of ϵ from a user, HDBSCAN simply extracts the corresponding (pre-computed) clusters. In other words, HDBSCAN is a space-economical method for storing the clustering results at multiple ϵ values; it, however, does not eliminate the need to *supply* a suitable ϵ value — without which no *actual* clustering can be returned. As a remedy, the article [8] also described how to produce a visualization diagram (which can be thought of as an OPTICS diagram with minor differences) based on the encoding obtained.

As surveyed in Section 2.2, all known algorithms for computing an OPTICS diagram run in $O(n^2)$ time (where $n = |P|$), though some heuristics have been designed to reduce the running time by constant factors. This includes the HDBSCAN approach which takes $O(n^2)$ time to generate its tree encoding. The quadratic time complexity becomes a serious issue on massive datasets.

1.1 Our Results

This paper's main contribution is a fast algorithm for computing an approximate OPTICS diagram which bears provably small visual differences from the exact OPTICS diagram. Our techniques are based on two central ideas:

- Idea 1: Formalization of the concept of *valley*. Somewhat surprisingly, this has never been done prior to this work. Instead, the term "valley" has been used on an intuitive basis throughout the literature. We rectify this, and make it possible to reason rigorously on the resemblance of two valleys. This lays down the foundation for our proposition: *ρ-approximate OPTICS* — where ρ is a real value satisfying $0 < \rho < 1$ — for generating a diagram where the valleys have guaranteed similarity to those of OPTICS, with the discrepancies continuously fading away as ρ decreases.
- Idea 2: Bridging the ρ-approximate OPTICS problem with the geometric technique of *well-separated pair decomposition*. This yields an algorithm that outputs a ρ-approximate OPTICS diagram on a set P of n points in $O(n \log n + (1/\rho)^{d/2} \cdot n)$ expected time for any fixed dimensionality d.

As a side product, the proposed algorithm generates a data structure of $O(n)$ space that allows one to obtain a ρ-approximate DBSCAN clustering [14] at an *arbitrary* value of ϵ in $O(n)$ time. This time complexity *does not depend on the dimensionality d and the approximation factor ρ at all*, and thus, compares favorably to computing the clusters using the algorithm of [14], which (after applying a patch in [9]) requires $O((1/\rho)^{d/3} \cdot n)$ time in expectation. In fact, our structure supports a more powerful operation called *cluster-group-by* (C-group-by) [13], which groups the points of a query set $Q \subseteq P$ by the clusters they belong to. For example, given $Q = \{p_1, p_5, p_{11}\}$ and $\epsilon = \sqrt{34}$, under the exact DBSCAN semantics, the query must break Q into two groups: $\{p_1, p_5\}$ and $\{p_{11}\}$ because, at this value of ϵ, (as mentioned earlier) points p_1, p_5 belong to the same cluster, while p_{11} is in a different cluster.

The approach of [13] supports C-group-by queries under the ρ-approximate DBSCAN semantics at only one value of ϵ.[1] Our structure does so at arbitrary values of ϵ, that is, the query is free to specify the desired ϵ. The query time is guaranteed to be $O(|Q|)$, plus the time of sorting $O(|Q|)$ integers in the domain of $[1, n]$. The query time is not affected by ρ and d, and is near-optimal because $\Omega(|Q|)$ time is needed even just to look at Q.

It is worth pointing out that, the C-group-by query endows our structure with the "cluster-extraction" functionality of HDBSCAN. As mentioned earlier, given an arbitrary ϵ value, HDBSCAN is able to output right away the (pre-computed) clustering at that value. A C-group query achieves the same by simply setting $Q = P$.

1.2 Paper Organization

The rest of the paper is organized as follows. Section 2 introduces the background knowledge required for our discussion. Section 3 formalizes the concept of valley, and proves some properties on the valleys of OPTICS. Section 4 presents the details of the proposed approximation techniques. Section 5 explains our index structures for answering C-group-by queries. Section 6 contains an experimental evaluation of the proposed solutions with real data. Finally, Section 7 concludes the paper with a summary of findings.

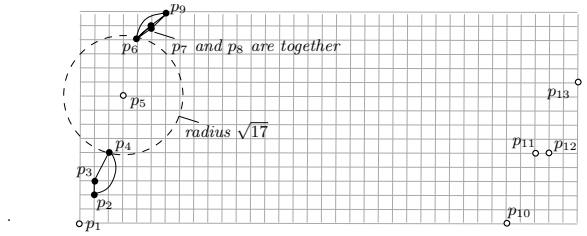
2 RELATED WORK

2.1 DBSCAN and Its ρ-Approximation

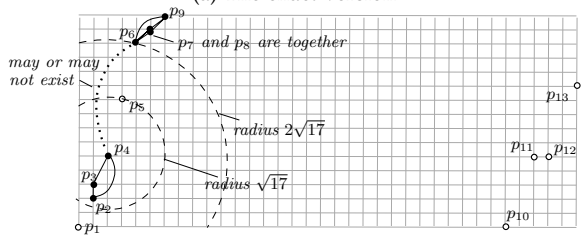
DBSCAN. Denote by $dist(p_1, p_2)$ the Euclidean distance between two points p_1, p_2 in $\mathbb{R}^d$. As before, let P be the input set of n points in $\mathbb{R}^d$. DBSCAN takes two parameters:

- *Vicinity radius ϵ*, which can be any positive real value;
- *Density threshold minPts*, a constant integer at least 1.

[1] For fairness, it should be noted that [13] considered maintaining a clustering in the *dynamic* scenario where updates to the dataset are allowed. We consider the static scenario in this paper.



(a) The exact version



(b) The ρ-approximate version ($\rho = 1$)

Figure 3: Illustration of DBSCAN ($minPts = 4, \epsilon = \sqrt{17}$)

As mentioned, a point p is an *ϵ-core point* if the ball $B(p, \epsilon)$ covers at least *minPts* points of P.

The clusters output by DBSCAN can be conveniently described from a graph's perspective [13]. Imagine an undirected graph $G(\epsilon)$ where each vertex is a distinct ϵ-core point in P. There is an edge between two ϵ-core points p_1, p_2 if and only if $dist(p_1, p_2) \leq \epsilon$. Break $G(\epsilon)$ into connected components (CC): $C_1, C_2, ..., C_m$, where m is the number of CCs. Each C_i ($1 \leq i \leq m$) is taken as a cluster on the ϵ-core points.

To illustrate, suppose that we perform DBSCAN clustering on the data of Figure 1 with $\epsilon = \sqrt{17}$ and $minPts = 4$. Figure 3a shows the $\sqrt{17}$-core points in black, and the non-$\sqrt{17}$-core points in white. It also demonstrates the edges in $G(\sqrt{17})$, whose CCs are $\{p_2, p_3, p_4\}$ and $\{p_6, p_7, p_8, p_9\}$ (non-$\sqrt{17}$-core points are not part of $G(\sqrt{17})$).

Next, each non-ϵ-core point p' is assigned to one, more than one, or no clusters as follows. If $B(p', \epsilon)$ contains no ϵ-core points, p' does not belong to any clusters (it is classified as noise). Otherwise, for every ϵ-core point p in $B(p', \epsilon)$, p' is added to the (only) cluster that contains p. This may put p' into $O(1)$ different clusters as $B(p', \epsilon)$ covers at most $minPts - 1$ points (by definition of p').

Continuing the earlier example, let us focus first on the non-$\sqrt{17}$-core point p_5. There are two $\sqrt{17}$-core points p_4, p_6 in $B(p_5, \sqrt{17})$. We thus add p_5 into the cluster (i.e., the aforementioned CC) of p_4 and the cluster of p_6. Similarly, the non-$\sqrt{17}$-point p_1 is added to the cluster of p_2. This gives the final clusters $\{p_1, p_2, p_3, p_4, p_5\}$ and $\{p_5, p_6, p_7, p_8, p_9\}$. Note that $p_{10}, p_{11}, p_{12}, p_{13}$ are noise at $\epsilon = \sqrt{17}$.

In 2D space, the DBSCAN clustering can be computed in $O(n)$ time, provided that P has been properly sorted [14]; otherwise, standard sorting time is needed in addition. Regarding higher constant dimensionalities d, the fastest existing algorithm [14] runs in $O((n \log n)^{4/3})$ time for $d = 3$, and $O(n^{2-\delta})$ time for $d > 3$, where δ is slightly less than $2/d$. On the lower bound side, it was proved that (under mild assumptions) any algorithm must entail $\Omega(n^{4/3})$ time when $d \geq 3$ [14]. Consequently, approximation is necessary for $d \geq 3$ if one wants to bring down the time complexity even just to $O(n \text{ polylog } n)$.

***ρ*-Approximate DBSCAN.** This approximate version takes the same parameters ϵ and *minPts* (as exact DBSCAN), but in addition also an approximation factor $\rho \geq 0$. The approximate clusters bear a graph interpretation as well, as explained below.

point p	p_1	p_2	p_3	p_4	p_5	p_6	p_7
core-dist(p)	$\sqrt{29}$	$\sqrt{10}$	$\sqrt{10}$	$\sqrt{17}$	$\sqrt{29}$	$\sqrt{8}$	$\sqrt{2}$
point p	p_8	p_9	p_{10}	p_{11}	p_{12}	p_{13}	
core-dist(p)	$\sqrt{2}$	$\sqrt{8}$	$\sqrt{125}$	$\sqrt{34}$	$\sqrt{34}$	$\sqrt{125}$	

Figure 4: Core distances ($minPts = 4$)

Imagine an undirected graph $\tilde{G}(\epsilon, \rho)$ where each vertex is still a distinct ϵ-core point on P. For each pair of ϵ-core points p_1, p_2, whether $\tilde{G}(\epsilon, \rho)$ has an edge between them is decided as:

- If $dist(p_1, p_2) \leq \epsilon$, definitely yes;
- If $dist(p_1, p_2) > (1 + \rho)\epsilon$, definitely no;
- Otherwise, *don't care* (either decision is acceptable).

The CCs of $\tilde{G}(\epsilon, \rho)$ constitute the clusters on the ϵ-core points. The non-ϵ-core points are then assigned to the clusters (i.e., the CCs) in the same manner as in DBSCAN.

For a comparison with Figure 3a, still choose $\epsilon = \sqrt{17}$ and $minPts = 4$, but additionally, set $\rho = 1$. The graph $\tilde{G}(\sqrt{17}, 1)$ is almost the same as $G(\sqrt{17})$, except that flexibility is permitted on (p_4, p_6). Note that $dist(p_4, p_6) = \sqrt{68}$, which falls into the "don't-care case". Hence, there may or may not be an edge between p_4 and p_6. If the edge exists, there is only one CC: $\{p_2, p_3, p_4, p_6, p_7, p_8, p_9\}$; otherwise, there are two CCs $\{p_2, p_3, p_4\}$, and $\{p_6, p_7, p_8, p_9\}$. In either case, non-$\sqrt{17}$-core point p_1 will be added to the cluster of p_2, while non-$\sqrt{17}$-core point p_5 to the clusters of p_4 and p_6.

In general, $\tilde{G}(\epsilon, \rho)$ is "sandwiched" between $G(\epsilon)$ and $G(\epsilon(1 + \rho))$: $\tilde{G}(\epsilon, \rho)$ contains all the edges of $G(\epsilon)$, and has all its edges contained in $G(\epsilon(1 + \rho))$. This leads to a *sandwich guarantee* on the ρ-approximate clustering returned: it must be enclosed between the exact DBSCAN clusterings obtained with vicinity radii ϵ and $(1 + \rho)\epsilon$, respectively (see [14] for the formal description of the guarantee). Hence, as ρ decreases, the approximate clusters become increasingly similar to the precise clusters.

For $\rho > 0$, under any fixed d, a ρ-approximate DBSCAN clustering can be computed in $O((1/\rho)^{d/3} \cdot n)$ expected time[2], which is $O(n)$ for any constant ρ. By integrating this algorithm with fine-tuned heuristics, one can efficiently compute clusters at ρ as low as 0.001 on real-world data [14].

2.2 OPTICS

OPTICS Diagrams. We formalize the OPTICS method proposed in [3] with a graph approach. The input is a set P of n points with ids 1, 2, ..., n, respectively. An OPTICS diagram is parameterized by:

- *minPts*: A constant integer at least 1;
- ϵ_{max}: A sufficiently large positive real value.

Each point $p \in P$ has a *core distance*:

$$core\text{-}dist(p) \quad = \quad dist(p, NN_p(minPts)) \tag{1}$$

where $NN_p(i)$ is the i-th nearest neighbor (NN) of p in P. Note that *core-dist*(p) is the smallest value of ϵ at which p qualifies as an ϵ-core-point for DBSCAN. Figure 4 shows the core distances of all the points in our example, assuming $minPts = 4$. For example,

[2]The bound $O((1/\rho)^d \cdot n)$ was proved in [14]. As pointed out in [9], the power d drops to $d/3$ by simply replacing the "approximate range counting" operation of [14] with the "approximate bichromatic closest pair" algorithm of [4].

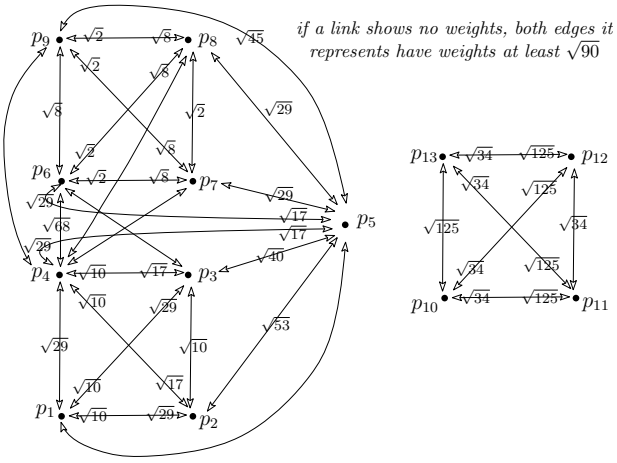


Figure 5: The base graph (*minPts* = 4, $\epsilon_{max} = \sqrt{125}$)

core-dist$(p_1) = \sqrt{29}$ is the distance between p_1 and its 4-th NN p_4 in P (note that p_1 is the NN of itself).

Definition 2.1. The ***base graph*** $\mathcal{G}$ *is a directed graph where*

- *The vertices are the points of P;*
- *For two distinct points (i.e., vertices)* p_1, p_2, $\mathcal{G}$ *has a directed edge* (p_1, p_2) *if and only if:*

$$\max\{\textit{core-dist}(p_1), \textit{dist}(p_1, p_2)\} \quad \leq \quad \epsilon_{max}.$$

If the edge exists, its weight — denoted as $w(p_1, p_2)$ *— equals the left hand side of the above inequality.* □

Figure 5 shows the base graph for our example dataset, under *minPts* = 4 and $\epsilon_{max} = \sqrt{125}$. For example, the edge from point p_1 to p_2 has weight $w(p_1, p_2) = \sqrt{29}$, which is the larger between *core-dist*$(p_1) = \sqrt{29}$ and *dist*$(p_1, p_2) = \sqrt{5}$. Conversely, the edge from p_2 to p_1 has weight $w(p_2, p_1) = \sqrt{10}$, the larger between *core-dist*$(p_2) = \sqrt{10}$ and *dist*$(p_1, p_2) = \sqrt{5}$. Figure 5 represents the two edges with one bidirectional link, but puts each weight near the corresponding arrow. Sometimes there is only one number on a bidirectional link (e.g., the link between p_2 and p_3); this means that both edges represented by the link have the same weight (i.e., $w(p_2, p_3) = w(p_3, p_2) = \sqrt{10}$). Some weights are omitted for clarity. Note that no edge goes, for instance, from p_5 to p_{10} because $\max\{\textit{core-dist}(p_5), \textit{dist}(p_5, p_{10})\} > \epsilon_{max}$.

Definition 2.2. The ***OPTICS procedure*** *is the following program which produces a set of directed trees from* $\mathcal{G}$*:*

OPTICS

1. $V_{unseen} = P$
2. **while** $V_{unseen} \neq \emptyset$
   /* create a new directed tree */
3. $V_{tree} = \{s\}$ where s is the vertex in V_{unseen} with the smallest id
4. delete s from V_{unseen}
5. **while** $\mathcal{G}$ has an edge from V_{tree} to V_{unseen}
6. (p, o) = the lightest edge in $\mathcal{G}$ with $p \in V_{tree}$ and $o \in V_{unseen}$
7. make o a child of p
8. delete o from V_{unseen}, and add it to V_{tree} □

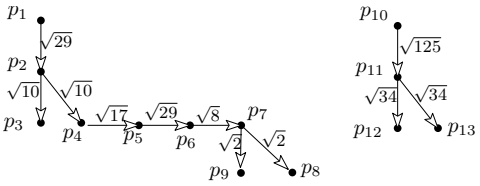


Figure 6: The tree output by the OPTICS procedure (*minPts* = 4, $\epsilon_{max} = \sqrt{125}$)

Let us illustrate the OPTICS procedure with the base graph of Figure 5. At the beginning, $V_{unseen} = P$. Point p_1 is the first removed from V_{unseen} because it has the smallest id, giving $V_{tree} = \{p_1\}$. Each of (p_1, p_2), (p_1, p_3), (p_1, p_4) has weight $\sqrt{29}$, and could be used as the lightest edge at Line 6. Suppose that we break ties by favoring a smaller id; this chooses (p_1, p_2) at Line 6, and makes p_2 a child of p_1 (see Figure 6) in the tree being generated. After moving p_2 from V_{unseen} to V_{tree}, (p_2, p_3) is the next lightest edge at Line 6, thus growing the tree by adding p_3 as a child of p_2. The process continues in the same fashion until p_9 has been moved from V_{unseen} to V_{tree}. At this moment, the procedure has created the left tree in Figure 6, with $V_{tree} = \{p_1, p_2, ..., p_9\}$ and $V_{unseen} = \{p_{10}, p_{11}, ..., p_{13}\}$. As there are no more edges from V_{tree} to V_{unseen}, the procedure starts growing a new tree with p_{10} as the root (it has the smallest id in V_{unseen}). The two final trees are shown in Figure 6.

Definition 2.3. The ***OPTICS sequence*** *is a sequence of point-value pairs* $(o_1, h_1), (o_2, h_2), ..., (o_n, h_n)$ *such that:*

- *Point* o_i $(1 \leq i \leq n)$ *is the i-th point that the OPTICS procedure deletes from* V_{unseen}.
- *If* o_i *is the root of a directed tree,* $h_i = \infty$. *Otherwise,* $h_i = w(p, o_i)$ *where* p *is the parent of* o_i. □

The OPTICS diagram is simply a plot of the OPTICS sequence $(o_1, h_1), ..., (o_n, h_n)$; specifically, it plots the function $f(i) = h_i$ for $i \in [1, n]$. On the base graph of Figure 5, points are deleted from V_{unseen} in ascending order of id. The weights of the edges in the resulting trees are shown in Figure 6. This spawns the OPTICS diagram of Figure 2.

Computation. A naive method to obtain an OPTICS diagram is to first create $\mathcal{G}$ in $O(n^2)$ time, and then perform the OPTICS procedure in $O(n^2)$ time using the Fibonacci heap. This is still the best time complexity known.

Practical efficiency can be improved by heuristics. The number of edges in $\mathcal{G}$ depends on ϵ_{max}, and can be $o(n^2)$ on real data for some ϵ_{max}. In this case, it would be faster to generate the edges of a point $p \in P$ with range reporting — fetching the points covered by $B(p, \epsilon_{max})$ — through a spatial index (e.g., the R-tree) on P [3]. In fact, the edges of $\mathcal{G}$ do not even need to be explicitly generated. By looking at the OPTICS procedure closely, one sees that the only edges required are the lightest edges at Line 6, each of which corresponds to the *bichromatic closest pair* (BCP; recall that, in general, for two sets U, V of points, the BCP is the pair $(u, v) \in U \times V$ with the smallest distance) between V_{tree} and V_{unseen} under the distance metric in Definition 2.1. Retrieval of the BCPs can also be accelerated by a spatial index [1].

Research has also been done to create visualizations that are in the "OPTICS style", but offer no quality guarantees on resemblance (see [5, 8, 9, 19–21] and the references therein).

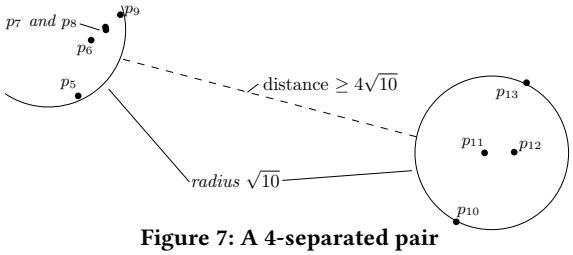


Figure 7: A 4-separated pair

2.3 Well-Separated Pair Decomposition

This subsection gives an introduction to a geometric tool named *well-separated pair decomposition* (WSPD). Let U and V be two sets of points in $\mathbb{R}^d$. For a value $\lambda \geq 1$, U and V are said to be *λ-separated* if there exists a value $r \geq 0$ that validates the following:

- U can be covered by a ball B_1 with radius r. This implies that any two points in U are within distance $2r$.
- V can be covered by a ball B_2 with radius r.
- The smallest distance between B_1 and B_2 is at least $\lambda \cdot r$.

Figure 7 shows a λ-separated pair with $\lambda = 4$, where $U = \{p_5, p_6, p_7, p_8, p_9\}$ and $V = \{p_{10}, p_{11}, p_{12}, p_{13}\}$. Both U and V are covered by a circle with radius $r = \sqrt{10}$, respectively, while the two circles have distance at least $\lambda \cdot r = 4\sqrt{10}$.

Let P be a set of n points in $\mathbb{R}^d$. A *λ-well-separated pair decomposition* (WSPD) of P is a collection of m pairs $(U_1, V_1), (U_2, V_2), ..., (U_m, V_m)$ with three properties:

(1) For every $i \in [1, m]$, $U_i \subseteq P$, $V_i \subseteq P$, and $U_i \cap V_i = \emptyset$.
(2) For any two distinct points p, q in P, there is a unique $i \in [1, m]$ satisfying: between the two sets U_i and V_i, one contains p, and the other contains q.
(3) For every $i \in [1, m]$, U_i and V_i are λ-separated.

The dataset in Figure 2 admits a 4-WSPD that consists of $m = 20$ pairs, as detailed in Figure 8. As demanded by Property (1), U and V are disjoint for each pair (U, V) in the WSPD. Property (2) states that, for every two points p, q ($p \neq q$), there is exactly one pair (U, V) that separates them — that is, either $p \in U, q \in V$ or $p \in V, q \in U$. For example, p_5, p_1 are separated in $(\{p_5\}, \{p_1, p_2, p_3\})$. Finally, Property (3) says that every pair (U, V) must be 4-separated (Figure 7 demonstrated this for one of the pairs).

Lemma 2.4 ([7]). *For any $\lambda \geq 1$, all of the following are true:*

- *P has a λ-WSPD $(U_1, V_1), ..., (U_m, V_m)$ with $m = O(\lambda^d n)$.*
- *In $O(n \log n + \lambda^d n)$ time, we can compute a data structure that allows us to extract, for any $i \in [1, n]$, the points of U_i in $O(|U_i|)$ time, and the points of V_i in $O(|V_i|)$ time.*

The *Euclidean bichromatic closest pair* (EBCP) of two sets U, V of points is the pair (u^*, v^*) minimizing $dist(u, v)$ among all $(u, v) \in U \times V$. We say that a pair $(u, v) \in U \times V$ is an *α-approximate EBCP* of U, V if $dist(u, v) \leq (1 + \alpha) \cdot dist(u^*, v^*)$.

Lemma 2.5 ([6]). *Suppose that we have obtained the data structure of Lemma 2.4 on a λ-WSPD $(U_1, V_1), ..., (U_m, V_m)$ of P. In $O(\lambda^{d-1} n)$ extra time, we can obtain $2m$ representative points $rep(U_1), ..., rep(U_m)$ and $rep(V_1), ..., rep(V_m)$ such that all the following hold on every $i \in [1, m]$:*

- *$rep(U_i) \in U_i$ and $rep(V_i) \in V_i$.*
- *$(rep(U_i), rep(V_i))$ is an $\frac{8}{\lambda^2}$-approximate EBCP of U_i and V_i.*

U	V
$\{\boldsymbol{p_1}\}$	$\{\boldsymbol{p_4}\}$
$\{\boldsymbol{p_2}\}$	$\{\boldsymbol{p_3}\}$
$\{\boldsymbol{p_5}\}$	$\{\boldsymbol{p_6}\}$
$\{\boldsymbol{p_5}\}$	$\{\boldsymbol{p_4}\}$
$\{\boldsymbol{p_6}\}$	$\{\boldsymbol{p_9}\}$
$\{\boldsymbol{p_7}\}$	$\{\boldsymbol{p_8}\}$
$\{\boldsymbol{p_{10}}\}$	$\{\boldsymbol{p_{13}}\}$
$\{\boldsymbol{p_{11}}\}$	$\{\boldsymbol{p_{12}}\}$
$\{\boldsymbol{p_1}\}$	$\{p_2, \boldsymbol{p_3}\}$
$\{\boldsymbol{p_4}\}$	$\{p_2, \boldsymbol{p_3}\}$

U	V
$\{\boldsymbol{p_6}\}$	$\{p_7, \boldsymbol{p_8}\}$
$\{\boldsymbol{p_9}\}$	$\{p_7, \boldsymbol{p_8}\}$
$\{\boldsymbol{p_{10}}\}$	$\{p_{11}, \boldsymbol{p_{12}}\}$
$\{\boldsymbol{p_{13}}\}$	$\{p_{11}, \boldsymbol{p_{12}}\}$
$\{\boldsymbol{p_5}\}$	$\{p_7, \boldsymbol{p_8}, p_9\}$
$\{\boldsymbol{p_5}\}$	$\{p_1, p_2, \boldsymbol{p_3}\}$
$\{\boldsymbol{p_4}\}$	$\{p_6, p_7, \boldsymbol{p_8}, p_9\}$
$\{p_1, p_2, \boldsymbol{p_3}\}$	$\{p_6, p_7, \boldsymbol{p_8}, p_9\}$
$\{\boldsymbol{p_5}, p_6, p_7, p_8, p_9\}$	$\{p_{10}, p_{11}, \boldsymbol{p_{12}}, p_{13}\}$
$\{p_1, p_2, p_3, \boldsymbol{p_4}\}$	$\{p_{10}, p_{11}, \boldsymbol{p_{12}}, p_{13}\}$

Figure 8: A 4-WSPD for our running example (points in bold are representatives)

- *For any point $u \in U_i$ and any point $v \in V_i$:*

$$dist(u, rep(V_i)) \leq (1 + 8/\lambda^2) \cdot dist(u, v)$$
$$dist(rep(U_i), v) \leq (1 + 8/\lambda^2) \cdot dist(u, v).$$

In Figure 8, for each pair of (U, V), the representative points are shown in bold. For example, the representatives of the pair $(U, V) = (\{\boldsymbol{p_1}\}, \{p_2, \boldsymbol{p_3}\})$ are p_1 and p_3. Note that the EBCP between U and V is (p_1, p_2) whose distance is $\sqrt{5}$. The pair of representatives (p_1, p_3) has distance $\sqrt{10}$, and thus, makes a $(\sqrt{2} - 1)$-approximate EBCP for U and V. Recall that here $\lambda = 4$; the approximation ratio $\sqrt{2} - 1$ is less than $8/\lambda^2 = 0.5$, as is consistent with Lemma 2.5.

The next lemma states a useful property for $\lambda > 2$:

Lemma 2.6. *Let (U, V) be an arbitrary λ-separated pair with $\lambda > 2$. The distance between any points $p_1, p_2 \in U$ (or $p_1, p_2 \in V$) must be shorter than the distance between any point $u \in U$ and any point $v \in V$, i.e., $dist(p_1, p_2) < dist(u, v)$.*

Proof. As (U, V) is λ-separated, there exists a real value r such that any two points within U (or V) are within distance $2r$, while any point in U must be at least distance $\lambda \cdot r$ away from any point in V. Hence, $dist(p_1, p_2) \leq 2r < \lambda \cdot r \leq dist(u, v)$. □

3 FORMALIZATION OF VALLEYS

In this section, we formalize the concept of *valley*, and analyze the properties of the valleys of OPTICS. Again, let P be the input set of n points in $\mathbb{R}^d$.

*Definition 3.1. A sequence of point-value pairs $(o_1, h_1), (o_2, h_2), ..., (o_n, h_n)$ is a **height-augmented permutation** of P if*

- *$h_i \geq 0$ for all $i \in [1, n]$, and*
- *$(o_1, o_2, ..., o_n)$ is a permutation of the points in P.* □

We refer to h_i as the *height* of o_i. Clearly, the OPTICS sequence is a height-augmented permutation. Note that, in general, any height-augmented permutation corresponds to a diagram that plots the function $f(i) = h_i$ for $i \in [1, n]$.

*Definition 3.2. Fix a height-augmented permutation $(o_1, h_1), (o_2, h_2), ..., (o_n, h_n)$. An **$H$-valley**, for some positive value $H \leq \epsilon_{max}$, is a subsequence $(o_x, h_x), (o_{x+1}, h_{x+1}), ..., (o_y, h_y)$ satisfying:*

- *$y > x$;*
- *$h_x > H$, and $h_i \leq H$ for all $i \in [x + 1, y]$;*
- *If $y < n$, then $h_{y+1} > H$.* □

Note that an H-valley includes its "left wall" — point o_x (with height over H) — but not its "right wall" (h_{y+1} is not part of the H-valley). Furthermore, all the H-valleys are disjoint. For instance, consider the OPTICS diagram in Figure 2. There are two $\sqrt{10}$-valleys: the first one is $(p_2, \sqrt{29})$, $(p_3, \sqrt{10})$, $(p_4, \sqrt{10})$ while the second one $(p_6, \sqrt{29})$, $(p_7, \sqrt{8})$, $(p_8, \sqrt{2})$, $(p_9, \sqrt{2})$.

The rest of the section aims to establish an important fact: *there is a one-one correspondence between (i) the set of H-valleys, and (ii) the set of DBSCAN clusters obtained with $\epsilon = H$.* For this purpose, a key step is to answer the question: how does an H-valley capture H-core-points (recall that DBSCAN clusters are formed by using core points as the "backbone")? This brings out the next definition:

Definition 3.3. The ***backbone*** *of an H-valley (o_x, h_x), (o_{x+1}, h_{x+1}), ..., (o_y, h_y) is the set $\{o_i \mid i \in [x, y]$ and core-dist$(o_i) \leq H\}$.* □

In the $\sqrt{10}$-valley $(p_2, \sqrt{29})$, $(p_3, \sqrt{10})$, $(p_4, \sqrt{10})$, the backbone includes p_2 and p_3, but not p_4 (as shown in Figure 4, core-dist$(p_4) = \sqrt{17} > \sqrt{10}$). In the $\sqrt{10}$-valley $(p_6, \sqrt{29})$, $(p_7, \sqrt{8})$, $(p_8, \sqrt{2})$, $(p_9, \sqrt{2})$, all the points are in the backbone. We are ready to establish formal links between OPTICS and DBSCAN (proof in Appendix A):

Lemma 3.4. *Fix minPts, ϵ_{max}, and a positive $H \leq \epsilon_{max}$.*

- *There is a one-one mapping between the set of H-valleys and the set of DBSCAN clusters obtained with $\epsilon = H$.*
- *Consider any H-valley Υ; let C be its corresponding DBSCAN cluster under $\epsilon = H$. The backbone of Υ is precisely the set of H-core points in C.*

To illustrate, consider again our example dataset in Figure 3a. As explained in Section 2.1, for *minPts* $= 4$ and $\epsilon = \sqrt{17}$, DBSCAN returns two clusters $C_1 = \{p_1, p_2, p_3, p_4, p_5\}$ and $C_2 = \{p_5, p_6, p_7, p_8, p_9\}$. They correspond to the two $\sqrt{17}$-valleys in the OPTICS diagram (Figure 2): the first $(p_2, \sqrt{29})$, $(p_3, \sqrt{10})$, $(p_4, \sqrt{10})$, $(p_5, \sqrt{17})$, and the second $(p_6, \sqrt{29})$, $(p_7, \sqrt{8})$, $(p_8, \sqrt{2})$, $(p_9, \sqrt{2})$. The backbone of the first valley is $\{p_2, p_3, p_4\}$, where the points are precisely the three $\sqrt{17}$-core-points inside C_1. Likewise, the backbone of the second valley is $\{p_6, p_7, p_8, p_9\}$, i.e., the set of $\sqrt{17}$-core-points inside C_2.

4 ρ-APPROXIMATE OPTICS

We first define an approximate version of OPTICS in Section 4.1, and then present a fast algorithm for its computation in Section 4.2.

4.1 Formalization

Besides *minPts* and ϵ_{max}, we take another parameter $0 < \rho < 1$ to control how similar approximate OPTICS should be to its exact counterpart:

Definition 4.1. A height-augmented permutation σ of P is a ***ρ-approximate OPTICS sequence*** *if both of the following conditions are satisfied for any real value $H \leq \epsilon_{max}/(1+\rho)$:*

(1) *If two points are in the backbone of the same H-valley in the OPTICS sequence, they must be in the backbone of the same H-valley in σ.*
(2) *If two points are in the backbone of the same H-valley in σ, they must be in the backbone of the same $H(1+\rho)$-valley in the OPTICS sequence.* □

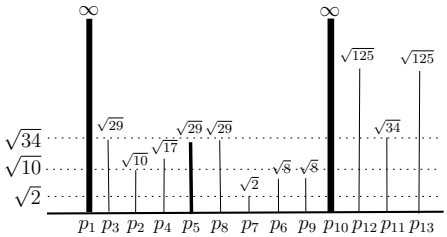


Figure 9: A 0.001-approximate OPTICS diagram

Let σ^* be an OPTICS sequence. A ρ-approximate OPTICS sequence σ has a strong valley-preservation guarantee (proof in Appendix B):

Lemma 4.2 (Valley Preservation Lemma). *Both the following are true for any subset $S \subseteq P$ satisfying $|S| \geq 2$ and any $H \leq \epsilon_{max}/(1+\rho)$:*

- *If σ^* has an H-valley having S in the backbone, σ must also have an H-valley having S in the backbone.*
- *If σ has an H-valley having S in the backbone, σ^* must have an $H(1+\rho)$-valley having S in the backbone.*

Figure 9 plots a 0.001-approximate OPTICS diagram σ on the dataset of Figure 1. Let us compare it with the OPTICS sequence σ^* in Figure 2. The observations below illustrate Lemma 4.2 with representative examples:

- Pick, somewhat arbitrarily, a set $S = \{p_3, p_7, p_8\}$. Notice that σ^* has a $\sqrt{29}$-valley containing S in the backbone (the valley starts from p_2 and ends at p_9 in Figure 2). The 1st bullet of the lemma asserts that σ must also have a $\sqrt{29}$-valley with S in the backbone; indeed, this valley is from p_1 to p_9 in Figure 9. Note that the value $\sqrt{29}$ is not a must: the above statement holds for any value of $H \geq \sqrt{29}$.
- Consider, instead, a set $S = \{p_6, p_7, p_9\}$. Notice that σ has a $\sqrt{8}$-valley with S in the backbone (starting from p_8 and ending at p_9 in Figure 9). The 2nd bullet of Lemma 4.2 declares that σ^* must have a $(1.001 \cdot \sqrt{8})$-valley enclosing S in the backbone. Indeed, this value goes from p_6 to p_9 in Figure 2. Again, the value $\sqrt{8}$ is not a must: the above holds for any value $H \geq \sqrt{8}$.
- We now look at examples of valley *absence*. Set $S = \{p_2, p_8, p_{12}\}$. Observe that σ^* (Figure 2) does not have any $\sqrt{50}$-valleys containing S in the backbone. The 2nd bullet implies that σ (Figure 9) cannot have any $(\sqrt{50}/1.001)$-valleys containing S in the backbone. Similarly, since σ (Figure 9) does not have any $\sqrt{50}$-valleys containing S in the backbone, the 1st bullet implies that σ^* (Figure 2) cannot have any $\sqrt{50}$-valleys containing S in the backbone, either.

4.2 The Algorithm

We will explain how to obtain a ρ-approximate OPTICS sequence with $O(n \log n + (1/\rho)^{d/2} \cdot n)$ expected time.

Algorithm. There are three steps.

Step 1: Graph Creation. Obtain a λ-WSPD of P with

$$\lambda = \sqrt{8/\rho}. \quad (2)$$

Suppose that the λ-WSPD has pairs (U_1, V_1), (U_2, V_2), ..., (U_m, V_m) with $m = O(\lambda^d n)$ (Lemma 2.4). Then, apply Lemma 2.5 to obtain a

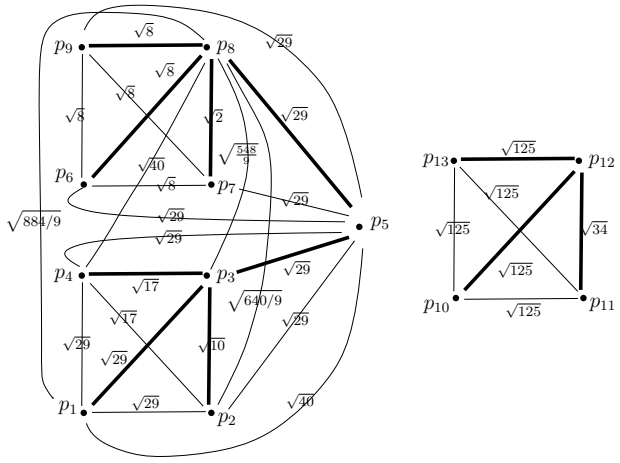


Figure 10: The undirected graph of our algorithm; the bold edges give an MSF ($minPts = 4, \epsilon_{max} = \sqrt{125}, \rho = 0.5$)

representative point $rep(U_i)$ for each U_i, and similarly, $rep(V_i)$ for each V_i where $i \in [1, m]$.

We will now construct an undirected weighted graph $\tilde{\mathcal{G}}$ whose vertices are the distinct points in P. Starting from an empty graph, we add edges gradually by processing each pair (U_i, V_i) (where $i \in [1, m]$) in one of the following ways:

- *Case I:* $|U_i| \geq minPts, |V_i| \geq minPts$. Add a single edge $(rep(U_i), rep(V_i))$ to $\tilde{\mathcal{G}}$.
- *Case II:* $|U_i| < minPts, |V_i| \geq minPts$. For every point $u \in U_i$, add an edge $(u, rep(V_i))$ to $\tilde{\mathcal{G}}$.
- *Case III:* $|U_i| \geq minPts, |V_i| < minPts$. For every point $v \in V_i$, add an edge $(rep(U_i), v)$ to $\tilde{\mathcal{G}}$.
- *Case IV:* $|U_i| < minPts, |V_i| < minPts$. For every point $u \in U_i$ and every point $v \in V_i$, add an edge (u, v) to $\tilde{\mathcal{G}}$.

Clearly, we add one edge in Case I, less than $minPts = O(1)$ edges in Cases II and III, and less than $minPts^2 = O(1)$ edges in Case 4. $\tilde{\mathcal{G}}$ therefore has $O(m) = O(\lambda^d \cdot n)$ edges so far.

Finally, for every edge (u, v) in $\tilde{\mathcal{G}}$, calculate its weight as:

$$\tilde{w}(u, v) = \max\left\{core\text{-}dist(u), core\text{-}dist(v), \frac{dist(u, v)}{1 + \rho}\right\}. \quad (3)$$

If $\tilde{w}(u, v) > \epsilon_{max}$, discard it from $\tilde{\mathcal{G}}$. This completes the construction of $\tilde{\mathcal{G}}$. Note that $\tilde{\mathcal{G}}$ has $O(\lambda^d \cdot n)$ edges, all of which have weights at most ϵ_{max}.

Assuming $minPts = 4, \epsilon_{max} = \sqrt{125}$, and $\rho = 0.5$, Figure 10 shows the graph $\tilde{\mathcal{G}}$ based on the 4-WSPD of Figure 8. For Case I, consider the following pair in the 4-WSPD: $(\{p_1, p_2, p_3, \boldsymbol{p_4}\}, \{p_{10}, p_{11}, \boldsymbol{p_{12}}, p_{13}\})$. We first add an edge (p_4, p_{12}) to $\tilde{G}$, but since $\tilde{w}(p_4, p_{12}) = \max\{core\text{-}dist(p_4), core\text{-}dist(p_{12}), \frac{dist(p_4, p_{12})}{1+0.5}\} = \frac{dist(p_4, p_{12})}{1.5} > \sqrt{125}$, the edge is then removed. For Case II, consider the pair $(\{p_1, p_2, \boldsymbol{p_3}\}, \{p_6, p_7, \boldsymbol{p_8}, p_9\})$, for which 3 edges are added to $\tilde{G}$, i.e., (p_1, p_8), (p_2, p_8), (p_3, p_8). Case III is symmetric to Case II. For Case IV, consider the pair $(\boldsymbol{p_1}, \boldsymbol{p_4})$ in the 4-WSPD, which adds edge (p_1, p_4) to $\tilde{G}$.

Step 2: MSF. Find an MSF of $\tilde{\mathcal{G}}$ using the state-of-the-art algorithm for this purpose (we will come back to this later). Denote the MSF by $\mathcal{F}$.

Step 3: Prim. We now invoke Prim's algorithm on $\mathcal{F}$ to produce the final ρ-approximate OPTICS sequence σ. After a CC (connected component) has been exhausted, care is exercised in picking a new source vertex by respecting the id ordering (as in OPTICS). For clarity, we provide the pseudocode in full, which will be referred to as the *approximate OPTICS procedure*:

approximate OPTICS
1. $V_{unseen} = P$
2. **while** $V_{unseen} \neq \emptyset$
   /* traverse a new connected component of $\mathcal{F}$ */
3. $V_{tree} = \{s\}$ where s is the vertex in V_{unseen} with the smallest id
4. delete s from V_{unseen}
5. **while** $\mathcal{F}$ has an edge from V_{tree} to V_{unseen}
6. (p, o) = the lightest edge (weight calculated in (3)) in $\mathcal{F}$ with $p \in V_{tree}, o \in V_{unseen}$
7. delete o from V_{unseen}, and add it to V_{tree}

The final sequence σ is $(o_1, h_1), (o_2, h_2), ..., (o_n, h_n)$ such that:

- Point o_i $(1 \leq i \leq n)$ is the i-th point that the above procedure deletes from V_{unseen}.
- If o_i is deleted at Line 4, $h_i = \infty$. Otherwise, it is deleted at Line 7 after finding an edge (p, o) at Line 6 with $o = o_i$. In this case, $h_i = \tilde{w}(p, o_i)$. □

The bold edges in Figure 10 illustrate an MSF $\mathcal{F}$ of $\tilde{G}$ for our running example. Applying the Prim's algorithm, a.k.a. the approximate OPTICS procedure, on $\mathcal{F}$ generates the following height-augmented sequence: (p_1, ∞), $(p_3, \sqrt{29})$, $(p_2, \sqrt{10})$, $(p_4, \sqrt{17})$, $(p_5, \sqrt{29})$, $(p_8, \sqrt{29})$, $(p_7, \sqrt{2})$, $(p_6, \sqrt{8})$, $(p_9, \sqrt{8})$, (p_{10}, ∞), $(p_{12}, \sqrt{125})$, $(p_{11}, \sqrt{34})$, and $(p_{13}, \sqrt{125})$. This is the sequence plotted by Figure 9. Note that the *actual* approximation quality of the sequence may be better than 0.5 — as explained earlier, Figure 9 is in fact a 0.001-approximate OPTICS diagram.

Running Time. By Lemmas 2.4 and 2.5, the λ-WSPD, as well as $rep(U_i)$ and $rep(V_i)$ for all $i \in [1, m]$, can be computed in $O(n \log n + \lambda^d \cdot n)$ time. In each of Cases I-IV, we need to retrieve at most $minPts = O(1)$ points from U_i and V_i, respectively, which takes $O(1)$ time by the 2nd bullet of Lemma 2.4. Hence, the edges of $\tilde{\mathcal{G}}$ are generated in $O(m) = O(\lambda^d \cdot n)$ time.

For Step 2, one can apply the algorithm of [16], which computes the MSF $\mathcal{F}$ in linear expected time, i.e., $O(m) = O(\lambda^d \cdot m)$. Since $\mathcal{F}$ has at most $n - 1$ edges, Step 3 clearly takes $O(n \log n)$ time.

The algorithm requires the value of $core\text{-}dist(p)$ for every point $p \in P$. This can be fulfilled by finding the $minPts$ nearest neighbors for every point $p \in P$ (i.e., the $minPts$ points in P with the smallest distances to p). This can be done in $O(minPts \cdot n \log n) = O(n \log n)$ time in any d-dimensional space with $d = O(1)$ [22]. The above analysis proves that our entire algorithm runs in $O(n \log n + \lambda^d \cdot n) = O(n \log n + (1/\rho)^{d/2} \cdot n)$ expected time.

We prove the correctness of our algorithm in Appendix C. With this, we have arrived at the first main result of the paper:

THEOREM 4.3. *A ρ-approximate OPTICS sequence can be computed in $O(n \log n + (1/\rho)^{d/2} \cdot n)$ expected time.*

As a remark, if one uses directly Prim's algorithm for Step 2, the total time is $O((1/\rho)^{d/2} \cdot n \log n)$ in the worst case.

5 CLUSTER GROUP-BY QUERIES

We will develop an index structure to support cluster group-by queries based on the techniques of the previous section. First, Section 5.1 defines such queries formally, after elaborating on the inherent connections between ρ-approximate OPTICS sequences and ρ-approximate DBSCAN clusters. Then, Section 5.2 describes our structure and its query algorithms, and establishes their performance guarantees.

All the "weights" in this section are computed based on (3).

5.1 From MSF to Clusters

The MSF $\mathcal{F}$ produced in Step 3 of our algorithm in Section 4.2 serves as a succinct way to encode ρ-approximate DBSCAN clusters (reviewed in Section 2.1) at all scales of ϵ. To clarify this, we define:

*Definition 5.1. Given a positive real-value $\epsilon \leq \epsilon_{max}/(1+\rho)$, the **$\epsilon$-forest** — denoted as $\mathcal{F}(\epsilon)$ — is the undirected forest obtained from $\mathcal{F}$ by the steps below:*

- *Remove all the edges of $\mathcal{F}$ with weights greater than ϵ.*
- *Discard the trees with no edges.* □

Figure 11 illustrates $\mathcal{F}(\sqrt{17})$, which is produced by the above steps from the MSF $\mathcal{F}$ in Figure 10 (which in turn was created with $minPts = 4$, $\rho = 0.5$, and $\epsilon_{max} = \sqrt{125}$).

Suppose that we want to find a ρ-approximate DBSCAN clustering under (i) the same value of *minPts* behind Theorem 4.3, and (ii) an arbitrary vicinity radius $\epsilon \leq \epsilon_{max}/(1+\rho)$. Equipped with $\mathcal{F}(\epsilon)$, there is a simple way to achieve the purpose (proof in Appendix D):

Lemma 5.2. *The following steps return a ρ-approximate DBSCAN clustering:*

1. *Take the set of points in every tree of $\mathcal{F}(\epsilon)$ as a cluster.*
2. *If an ϵ-core point p appears in no trees, form a cluster including p itself.*
3. *For each non-ϵ-core point $p' \in P$, add it to the cluster of p for every ϵ-core point $p \in B(p', \epsilon)$.*
4. *Return the clusters.*

We will represent the set of clusters determined by Lemma 5.2 as $C(\epsilon, \rho)$. Let us illustrate the lemma by following its steps to derive $C(\sqrt{17}, 0.5)$ from the forest $\mathcal{F}(\sqrt{17})$ in Figure 11. We first create two clusters $C_1 = \{p_6, p_7, p_8, p_9\}$ and $C_2 = \{p_2, p_3, p_4\}$. Second, check whether all $\sqrt{17}$-core points are already included; the answer is yes (as shown in Figure 4, no ϵ-core points are outside $C_1 \cup C_2$). Third, we insert non-$\sqrt{17}$-core point p_1 to the cluster C_2 of the $\sqrt{17}$-core point $p_2 \in B(p_1, \sqrt{17})$. Likewise, non-$\sqrt{17}$-core point p_5 is added to both C_1 and C_2 (due to $\sqrt{17}$-core points p_4 and p_6 in $B(p_5, \sqrt{17})$, respectively). The final $C_1 = \{p_5, p_6, p_7, p_8, p_9\}$ and $C_2 = \{p_1, p_2, p_3, p_4, p_5\}$ constitute $C(\sqrt{17}, 0.5)$.

We are now ready to define C-group-by queries:

*Definition 5.3. Given a set $Q \subseteq P$ and a positive real value $\epsilon \leq \epsilon_{max}/(1+\rho)$, a **cluster group-by query** returns for each cluster C in $C(\epsilon, \rho)$: $C \cap Q$, if $C \cap Q \neq \emptyset$, or nothing, otherwise.*

For instance, a query with $Q = \{p_1, p_3, p_8, p_{11}\}$ and $\epsilon = \sqrt{17}$ must return $\{p_1, p_3\}$ and $\{p_8\}$ because in $C(\sqrt{17}, 0.5)$: p_1 and p_3 are in the same cluster, Q has no other points in the cluster of p_8, and p_{11} is not in any clusters. As a special case, when $Q = P$, a C-group-by query extracts all the clusters in $C(\epsilon, 0.5)$ at the requested ϵ.

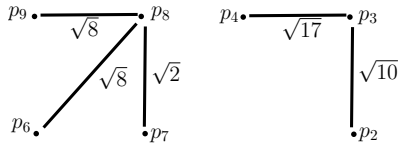


Figure 11: Illustration of $\mathcal{F}(\epsilon)$ ($\epsilon = \sqrt{17}$)

5.2 A C-Group-By Index

Suppose that Theorem 4.3 outputs a ρ-approximate OPTICS sequence $o_1, o_2, ..., o_n$. For each point $p \in P$, define its *sequence rank* as i if $p = o_i$.

Structure. We record for every point $p \in P$:

- Its *core-dist*(p) (see (1)) and *sequence rank*.
- The *minPts* nearest neighbors of p in P.

The above information occupies $O(minPts \cdot n) = O(n)$ space, and can be computed in $O(n \log n + (1/\rho)^{d/2} \cdot n)$ expected time. Recall that the *minPts* nearest neighbors of all points in P can be computed in $O(n \log n)$ time, as discussed in Section 4.2.

We will also need a structure to support the *tree-path* operation:

> Given two nodes u, v in $\mathcal{F}$, return the highest weight of the edges on the path between u and v in $\mathcal{F}$. If such a path does not exist, return ∞.

For example, on the MSF $\mathcal{F}$ of Figure 10, a tree-path query with $u = p_2$ and $v = p_8$ returns $\sqrt{29}$. Tree-path operations are well-understood: they are identical to finding the lowest common ancestor of two nodes [17]. The structure of [11] supports every operation in $O(1)$ time using $O(n)$ space; the structure can be constructed in $O(n \log n)$ time.

Overall, our structure uses $O(n)$ space and can be built in $O(n \log n + (1/\rho)^{d/2} \cdot n)$ time.

Queries with Only Core Points. We now proceed to discuss how to answer a C-group-by query with set Q and vicinity radius ϵ. For each point $q \in Q$, by comparing the pre-computed *core-dist*(q) to ϵ, we can decide in constant time whether q is an ϵ-core point. Let us first consider the case where Q contains only ϵ-core points.

Set $k = |Q|$; and list the points in Q as $q_1, q_2, ..., q_k$ in ascending order of sequence rank. As a naive approach, for any $i, j \in [1, k]$, one can determine whether q_i and q_j are in the same cluster of $C(\epsilon, \rho)$, by performing a tree-path operation to find out if they are connected in $\mathcal{F}$ by a path with only edges of weights at most ϵ.[3] This, however, entails $O(k^2)$ time. We improve the running time by proving in Appendix E:

Lemma 5.4. *For any $i \in [1, k-2]$, if q_i and q_{i+2} are in the same cluster of $C(\epsilon, \rho)$, so are q_i and q_{i+1}.*

The lemma indicates that the aforementioned $O(k^2)$ time can be reduced to $O(k)$ by performing only $k-1$ tree-path operations: one operation on each pair of q_i and q_{i+1} for $i \in [1, k-1]$. This groups $q_1, q_2, ..., q_k$ by cluster.

General Queries. Consider now a query whose set Q contains non-ϵ-core points. As shown below, this can be converted to another query whose query set Q' contains only ϵ-core points, and has size at most $minPts \cdot |Q|$.

Initialize Q' to include all the ϵ-core points in Q. Then, for each non-ϵ-core point $q' \in Q$, retrieve the (pre-computed) *minPts* nearest

[3] They are, if and only if the operation returns a value at most ϵ.

neighbors of q': $NN_{q'}(1), NN_{q'}(2), ..., NN_{q'}(minPts)$. For every $j \in [1, minPts]$, add $NN_{q'}(j)$ to Q' if $NN_{q'}(j)$ is an ϵ-core point with distance at most ϵ from q'. Such $NN_{q'}(j)$ is said to be a *Q-canonical neighbor* of q'. Now sort Q' by sequence rank, and group its points by cluster using the "core-point only" algorithm explained earlier. Each non-ϵ-core point $q' \in Q$ is added to every group containing any of its Q-canonical neighbors (the number of which is at most *minPts*). The correctness follows directly from Lemma 5.2. The overall query time is $O(minPts \cdot |Q|) = O(|Q|)$.

We now arrive at our second main result:

THEOREM 5.5. *In $O(n \log n + (1/\rho)^{d/2} \cdot n)$ time, we can construct a data structure of $O(n)$ space that answers a C-group-by query with query set Q in $O(|Q|)$ time, plus the time to sort $O(|Q|)$ integers from the domain of $[1, n]$.*

As a final remark, it takes $O(|Q| \log\log |Q|)$ to sort $O(|Q|)$ integers each of which is between 1 and n [2]; furthermore, if $|Q| \geq n^{0.01}$, the sorting time can be reduced to $O(|Q|)$ [18].

6 EXPERIMENTS

6.1 Setup

Datasets. We deployed 5 real datasets. The first three — named *PAMAP2*, *Farm*, *Household*, respectively — were obtained from [14], where they were used for studying ρ-approximate DBSCAN. We refer the reader to [14] for a detailed description of those datasets. The fourth dataset, *HT*, was produced in [15][4], collecting readings of home sensors monitoring temperature, humidity, and the concentration levels of several gases. The last one, *Chem*, was produced in [12][5], and again a collection of sensor readings, but of Ethylene and CO levels in a chemical environment. Their key meta information of all these datasets can be found in Table 1.

From each dataset, we also obtained sample sets of various sizes. These sample sets were used to study the scalability of the examined algorithms (to be elaborated next), and the similarity between the OPTICS diagram of a full dataset and that of a sample set.

Competing Methods. We compared the proposed approximate OPTICS solution to 5 methods, each of which is representative in a unique sense, as explained below:

- *ABKS*: The original OPTICS algorithm of [3], named after the initials of the authors.
- *DeliClu*: The state-of-the-art algorithm [1] for computing exact OPTICS diagrams (ideas reviewed in Section 2.2).
- *40-thread*: A *multi-core* algorithm [19] for computing an OPTICS diagram by leveraging the parallelism of 40 threads on 20 CPU cores.
- *HDBSCAN*: The hierarchical version [8] of DBSCAN; as mentioned in Section 1, an OPTICS diagram can be extracted from the clustering encodings.
- *SOPTICS*: An algorithm [20] that (just like our solution) achieves a sub-quadratic time complexity by resorting to approximation.

[4] *Https://archive.ics.uci.edu/ml/datasets/Gas+sensors+for+home+activity+monitoring*.
[5] *Https://archive.ics.uci.edu/ml/datasets/Gas+sensor+array+under+dynamic+gas+mixtures*.

dataset	dim.	number of points	domain of each dim.
PAMAP2	4	3,850,505	$[0, 10^5]$
Farm	5	3,627,086	$[0, 10^5]$
Household	7	2,049,280	$[0, 10^5]$
HT	10	928,991	$[0, 10^6]$
Chem	16	4,208,261	$[0, 10^6]$

Table 1: Meta information of our datasets

Our comparisons with the above methods aimed at different purposes. With respect to the exact algorithms — namely *ABKS*, *DeliClu*, *40-thread*, and *HDBSCAN*[6] — we focused on evaluating the efficiency gains achieved by approximation. It should be noted that, other than *40-thread*, all the other algorithms use a *single* thread. Nevertheless, as we will see, our algorithm (as well as *SOPTICS*) outperformed *40-thread* significantly in most cases anyway, thereby demonstrating the power of approximation.

With respect to *SOPTICS*, we carried out the comparison on both diagram quality and CPU efficiency. Our main objective is to demonstrate two phenomena. First, the approximate OPTICS diagrams we produced were *robustly* accurate (thanks to the valley preservation lemma, i.e., Lemma 4.2), as opposed to those by *SOPTICS* which, although reasonably good overall, were inaccurate in some scenarios. Second, our algorithm, besides its accuracy superiority, also demonstrated a clear performance advantage over *SOPTICS*.

It is worth pointing out that the inclusion of *HDBSCAN* also serves one more purpose. As explained in Section 1, C-group-by queries trivially realize the functionality of extracting clusters at specific ϵ values (but with respect to ρ-approximate DBSCAN). Hence, by showing that our algorithm is much faster than HDBSCAN, we provide practitioners with another attractive option for encoding clusterings.

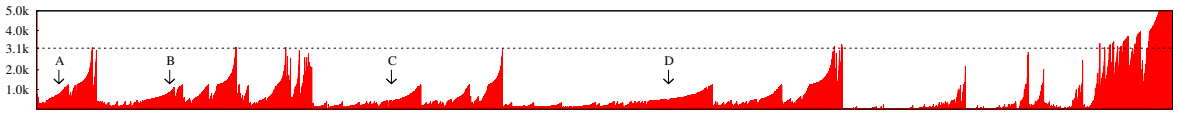
The parameter *minPts* was set to 10 for all the competing methods throughout the experiments.

Approximation Quality Metrics. We will assess the quality of approximate OPTICS diagrams with two approaches. The first one is to present the diagrams in full to enable a direct visual comparison. This approach, however, fails to quantify the degree of similarity. To remedy the defect, we introduce the notion of *valley-preservation ratio* (VP-ratio, for short) as follows.

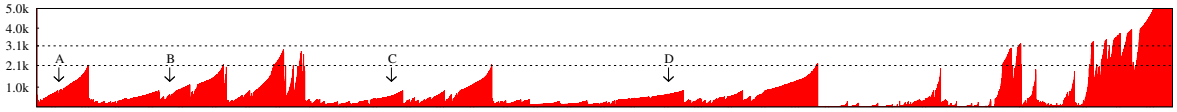
Let σ^* be an (exact) OPTICS sequence (Definition 2.3), and σ be the height-augmented permutation (Definition 3.1) output by an approximation algorithm. Fix a value of ϵ. Intuitively, if σ and σ^* are similar, then (i) the ϵ-valleys in σ^* should be present in σ, and (ii) vice versa. We quantify the preservation in these two directions as ρ_1 and ρ_2 respectively, both of which — as defined below — are at least 1, with higher values indicating *worse* preservation quality:

- To define ρ_1, let V^* be the set of ϵ-valleys in σ^* (all these valleys are disjoint by definition). For each valley $\Upsilon^* \in V^*$, check whether σ has an ϵ-valley that contains the backbone of Υ^*. If so, define $\rho_1(\Upsilon^*) = 1$. Otherwise, let ϵ' be the lowest value such that σ has an ϵ'-valley whose backbone contains Υ^* (it can be verified that ϵ' must be greater than ϵ); accordingly, define $\rho_1(\Upsilon^*) = \epsilon'/\epsilon$. In both cases, $\rho_1(\Upsilon^*)$

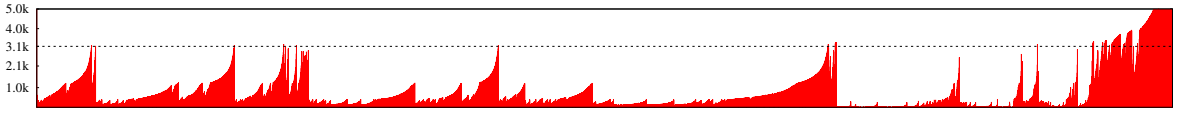
[6] Strictly speaking, the diagrams created by *40-thread* and *HDBSCAN* are not precisely the same as OPTICS. However, the discrepancies are minor and could be ignored in practice.



(a) *Household* (exact)

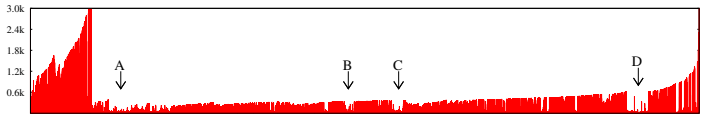


(b) *Household* ($\rho = 0.5$)

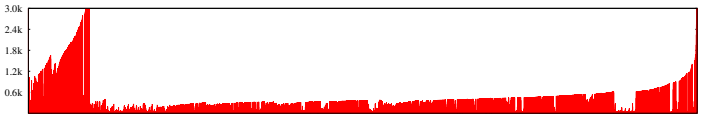


(c) *Household* ($\rho = 0.001$)

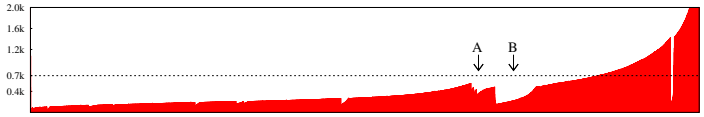
Figure 12: Effects of ρ on the approximation quality of our techniques



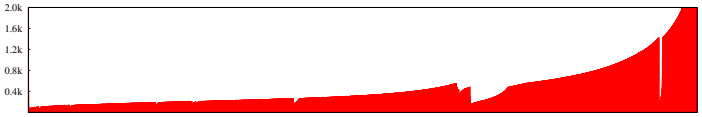
(a) *PAMAP2* (exact)



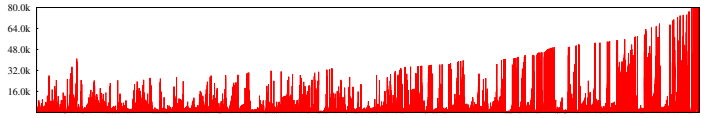
(b) *PAMAP2* (0.001-approximate)



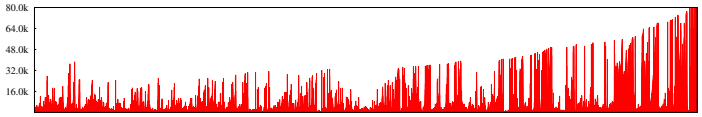
(c) *Farm* (exact)



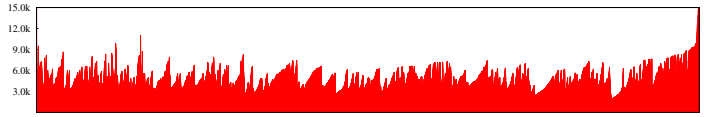
(d) *Farm* (0.001-approximate)



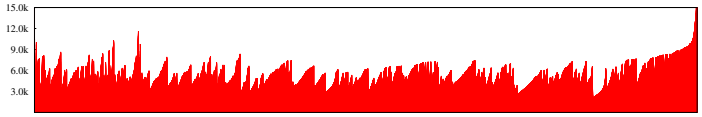
(e) *HT* (exact)



(f) *HT* (0.001-approximate)

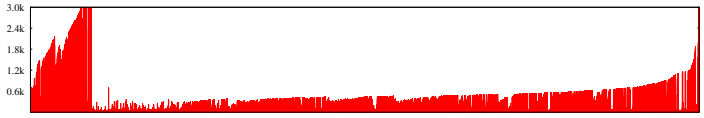


(g) *Chem* (exact)

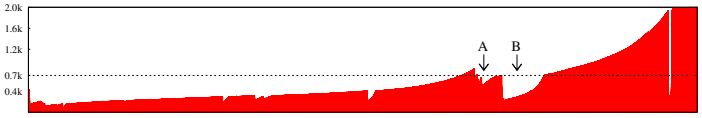


(h) *Chem* (0.001-approximate)

Figure 13: Exact OPTICS diagrams vs. our approximate diagrams



(a) *PAMAP2*



(b) *Farm*

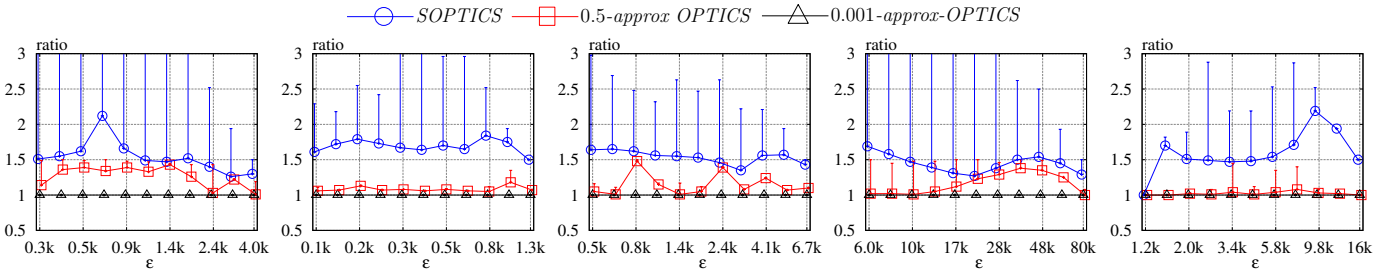
Figure 14: Two approximate diagrams illustrating defects of *SOPTICS*

indicates how well Υ^* is retained in σ. Finally, define ρ_1 as the maximum of $\rho_1(\Upsilon^*)$ for all $\Upsilon^* \in V^*$.

- To define ρ_2, let V be the set of ϵ-valleys in σ. For each ϵ-valley $\Upsilon \in V$, $\rho_2(\Upsilon)$ is defined in the same way as $\rho_1(\Upsilon^*)$, except for replacing σ with σ^*. The value of ρ_2 is the maximum of $\rho_2(\Upsilon)$ for all $\Upsilon \in V$.

The VP-ratio of σ is now finalized as $\max\{\rho_1, \rho_2\}$. Note that the VP-ratio is a function of ϵ: indeed, it indicates the resemblance of ϵ-valleys between σ^* and σ.

The VP-ratio thus defined is determined by a single valley in V^* or V (i.e., the most poorly preserved valley); we therefore refer to it as the *worst-case VP-ratio*. We also define a *weighted* version to measure the overall similarity of ϵ-valleys. For this purpose, redefine ρ_1 as the weighted sum of $\rho_1(\Upsilon^*)$ for all $\Upsilon^* \in V^*$, where the weight of Υ^* equals how much percent of the ϵ-core points (of the whole dataset) appear in Υ^*. With ρ_2 redefined analogously, the weighted VP-ratio of σ is the maximum of the (new) ρ_1 and ρ_2.



(a) *PAMAP2* (b) *Farm* (c) *Household* (d) *HT* (e) *Chem*

Figure 15: VP-ratios vs. ϵ (each dot on a curve represents the weighted VP-ratio, and its bar represents the worst-case VP-ratio)

The proposed ρ-approximate OPTICS technique has the feature that it guarantees a VP-ratio of at most $1 + \rho$ even in the worst case (let alone the weighted case); see Lemma 4.2.

Machine and OS. All experiments were run on a server equipped with two CPUs, each being an Intel Xeon Processor (E5-2630 v4 a 2.2Ghz). Each CPU had 10 cores; and each core was able to support 2 threads concurrently. This made 40 the largest number of "real" threads. All the cores shared a memory of 128GB. The operating system was Ubuntu 16.04.

6.2 Approximation Quality of Diagrams

We start by demonstrating the effects of Lemma 4.2 — the valley preservation lemma (our quality guarantee) — under different values of ρ. The best dataset for this purpose is *Household*, for which Figure 12a shows the exact OPTICS diagram, and Figures 12b and 12c give our approximate diagrams under $\rho = 0.5$ and 0.001, respectively. Both approximate diagrams are similar in shape, but differ in a manner that manifests benefits of a smaller ρ. Consider $\epsilon = 3.1$k; and focus on the four valleys marked as A, B, C, and D in the exact diagram. In the $\rho = 0.5$ diagram, even though the four valleys are still there, their heights are lower such that, at $\epsilon = 3.1$k, the four valleys have merged into a single one. This can be problematic in cluster analysis: by relying on Figure 12b, a user obtains wrong information about the number and the sizes of the clusters at $\epsilon = 3.1$k. The issue is avoided by the $\rho = 0.001$ diagram; the aforementioned "height distortion" is visually undetectable in Figure 12c, which looks almost the same as Figure 12a (except for the ordering of some smaller valleys inside Valley D).

Next, we explain the above phenomenon with Lemma 4.2. Notice that each of A, B, C, and D is a 2.1k-valley in Figure 12b. The 2nd bullet of the lemma assures us that these valleys must also be present in the exact diagram, but perhaps with a larger height of $(1 + \rho) \cdot 2.1\text{k} = 3.15\text{k}$ ($\rho = 0.5$ for Figure 12b). In other words, there can be a distortion factor of 1.5, which is indeed what we saw. This immediately implies that Figure 12c has a distortion factor of only 1.001, which is why it can preserve all the valleys to a level that cannot be discerned with naked eyes.

Henceforth, we set $\rho = 0.001$ by default for our techniques. Figure 13 presents the exact and our approximate diagrams produced on *PAMAP2*, *FARM*, *HT*, and *Chem*, respectively.

Let us turn attention to *SOPTICS*. In general, this method produced reasonably accurate approximate diagrams. In particular, its diagrams on *Household*, *HT*, and *Chem* looked as good as our diagrams. The same, however, cannot be said on *PAMAP2* and *Farm*, for which the diagrams of *SOPTICS* are presented in Figure 14. They exhibit two typical defects of *SOPTICS*:

- Valley disappearance: As shown in Figure 13a, *PAMAP2* has four sizable valleys A, B, C, and D, as indicated with arrows. Some of these valleys are absent from the *SOPTICS* diagram in Figure 14a.
- Height distortion: *SOPTICS* suffers from the same issue as our $\rho = 0.5$ diagram in Figure 12b. By looking at Figure 14b, a user would think that *Farm* has two major clusters marked as A and B at $\epsilon = 0.7$k. This is not true: Figure 13c shows that the two clusters in fact have merged at this ϵ.

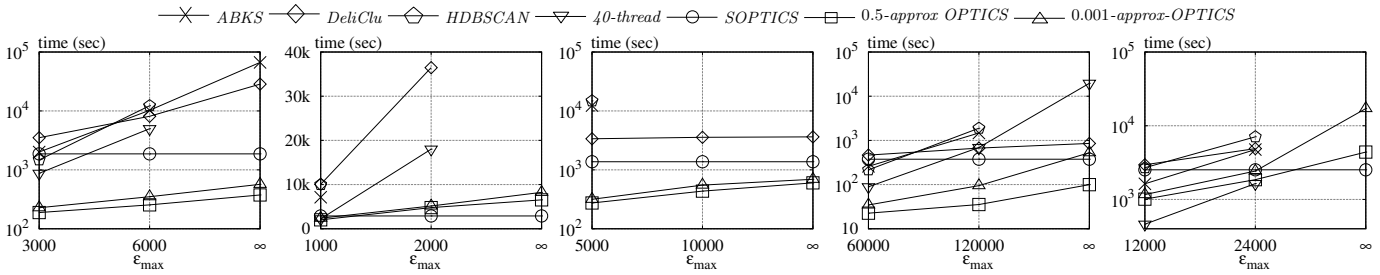
The fundamental reason behind the above is that *SOPTICS* lacks a quality guarantee similar to Lemma 4.2. To establish this argument further, we measured the VP-ratios of the approximate diagrams produced by *SOPTICS* and our algorithm with $\rho = 0.5$ and 0.001, respectively. Figure 15 plots the VP-ratios as a function of ϵ for each dataset. Each dot on a curve gives the weighted VP-ratio, and is associated with a vertical bar that indicates the corresponding worst-case VP-ratio (see Section 6.1 for the definitions of these ratios). For example, for *PAMAP2*, at $\epsilon = 0.3$k our 0.5-approximate diagram had a weighted VP-ratio of around 1.1, and a worst-case VP-ratio of 1.5. As expected, our ρ-approximate diagrams always ensure a worst-case VP-ratio of $1 + \rho$ (explaining the high quality of our $\rho = 0.001$ diagrams). The VP-ratios of *SOPTICS* in general are consistently worse (i.e., greater) in all scenarios. In particular, its worst-case VP-ratio can be very bad (again, because it has no valley-preservation guarantees).

Appendix F contains additional experiments that assess the quality of diagrams obtained from sample sets.

6.3 Computation Efficiency

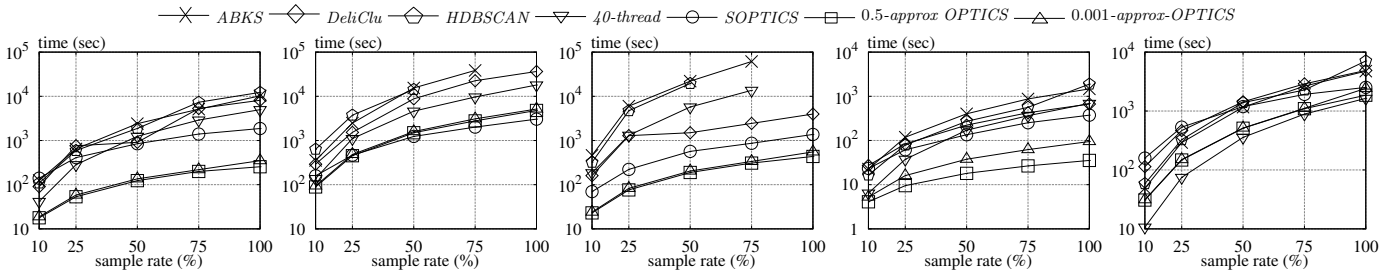
Cost of Diagram Computation. In an exact OPTICS diagram, some points may have ∞ as their heights. The number of such *undefined* points depends on the ϵ_{max} parameter of OPTICS. In general, the smaller ϵ_{max} is, the faster it would be to compute an OPTICS diagram, but at the expense of more undefined points. For each of the 5 real datasets deployed, we identified the smallest value of ϵ_{max} at which there were no more than 1% undefined points. Specifically, these values were 6k, 2k, 10k, 120k, and 24k for *PAMAP2*, *Farm*, *Household*, *HT*, and *Chem* respectively; they were chosen as the default value of ϵ_{max} on the corresponding datasets.

Figure 16 plots the execution time of all the competing methods as a function of ϵ_{max}. For our algorithm, we inspected its cost at both $\rho = 0.5$ and 0.001. We considered three values of ϵ_{max}: half



(a) *PAMAP2* (b) *Farm* (c) *Household* (d) *HT* (e) *Chem*

Figure 16: Running time vs. ϵ_{max}



(a) *PAMAP2* (b) *Farm* (c) *Household* (d) *HT* (e) *Chem*

Figure 17: Running time vs. sampling rate (scalability test)

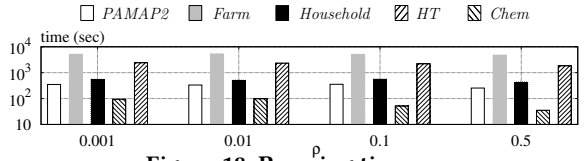


Figure 18: Running time vs. ρ

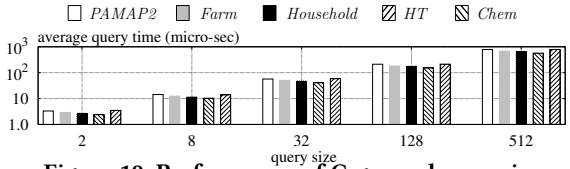


Figure 19: Performance of C-group-by queries

the default, default, and ∞. Note that $\epsilon_{max} = \infty$ generates the "complete" OPTICS diagram (although a part of the diagram may concern uninterestingly high values of ϵ that have been trimmed off in the diagrams in Section 6.2). Some methods have no results in the diagrams of Figure 16; this means that they did not terminate within 10 hours in the corresponding settings.

Except for a single exception (ϵ_{max} = 12k on *Chem*), approximation algorithms (i.e., ours and *SOPTICS*) outperformed the exact algorithms significantly — often by a factor of at least an order of magnitude — confirming the computational benefits of approximation. Compared to *SOPTICS*, we lost out mainly when ϵ_{max} was set to ∞ on the dataset of the highest dimensionality: *Chem* (16 dimensions). The proposed techniques are designed for relatively low dimensionalities, as predicted by its time complexity analysis. We also lost to *SOPTICS* slightly on *Farm* when ϵ_{max} was large. But in the other scenarios, our algorithms were considerably faster.

Figure 17 demonstrates the scalability of each algorithm by plotting the running time as a function of the sample rate. The relative superiority of alternative algorithms follows the patterns in Figure 16. Figure 18 gives the running time of our algorithm under various ρ values. The influence of ρ tended to be more significant when the dimensionality d was higher (as can also been seen in Figure 16), although we did not observe a growth of $(1/\rho)^{d/2}$ (remember that theoretical analysis is conservative, and hence, typically pessimistic on practical data).

Cost of C-Group-By Queries. The last set of experiments evaluated the efficiency of C-group-by queries (see Section 5.1). For this purpose, we generated query *workloads*, each containing 100,000 queries satisfying three conditions: (i) their sets Q all had the same size (hence, $|Q|$ is a workload parameter), (ii) each Q was a random subset (of the specified size) of the underlying dataset, and (iii) their ϵ values were decided uniformly at random from 0 to $\epsilon_{max}/(1+\rho)$. Figure 19 plots, as a function of $|Q|$, the average cost of all queries in the same workload. Note that the y-axis is in micro (i.e., 10^{-6}) seconds. The query time increased linearly with $|Q|$ on each dataset; all queries were answered with negligible cost.

7 CONCLUSIONS

OPTICS is a popular method for visualizing clusterings of multidimensional points. All the existing algorithms for implementing the method suffer from a quadratic time complexity in the worst case, and thus, may not be applicable to massive datasets. This paper has discussed how to alleviate the problem with approximation. Our key proposition is a new concept called *ρ-approximate OPTICS*, which defines a class of clustering visualizations that have strong guarantees in their resemblance to exact OPTICS diagrams. We have described an algorithm that finds such a visualization in $O(n \log n + (1/\rho)^{d/2} \cdot n)$ time, where n is the size of the input, and d is the dimensionality. Experiments have confirmed that high-quality visualizations can be obtained for real datasets with cost significantly lower than that of the state-of-the-art algorithms for computing exact/approximate OPTICS diagrams. Our techniques produce as a side product a linear-space index structure that is capable of answering ad-hoc C-group-by queries efficiently under the semantics of ρ-approximate DBSCAN.

REFERENCES

[1] Elke Achtert, Christian Böhm, and Peer Kröger. 2006. DeLi-Clu: Boosting Robustness, Completeness, Usability, and Efficiency of Hierarchical Clustering by a Closest Pair Ranking. In *Proceedings of Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*. 119–128.

[2] Arne Andersson, Torben Hagerup, Stefan Nilsson, and Rajeev Raman. 1998. Sorting in Linear Time? *Journal of Computer and System Sciences (JCSS)* 57, 1 (1998), 74–93.

[3] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. OPTICS: Ordering Points To Identify the Clustering Structure. In *Proceedings of ACM Management of Data (SIGMOD)*. 49–60.

[4] Sunil Arya and Timothy M. Chan. 2014. Better ϵ-Dependencies for Offline Approximate Nearest Neighbor Search, Euclidean Minimum Spanning Trees, and ϵ-Kernels. In *Proceedings of Symposium on Computational Geometry (SoCG)*. 416.

[5] Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander. 2000. Fast Hierarchical Clustering Based on Compressed Data and OPTICS. In *Proceedings of European Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*. 232–242.

[6] Paul B. Callahan and S. Rao Kosaraju. 1993. Faster Algorithms for Some Geometric Graph Problems in Higher Dimensions. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 291–300.

[7] Paul B. Callahan and S. Rao Kosaraju. 1995. A Decomposition of Multidimensional Point Sets with Applications to k-Nearest-Neighbors and n-Body Potential Fields. *Journal of the ACM (JACM)* 42, 1 (1995), 67–90.

[8] Ricardo J. G. B. Campello, Davoud Moulavi, Arthur Zimek, and Jörg Sander. 2015. Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 10, 1 (2015), 5:1–5:51.

[9] Mark de Berg, Ade Gunawan, and Marcel Roeloffzen. 2017. Faster DB-scan and HDB-scan in Low-Dimensional Euclidean Spaces. *CoRR* abs/1702.08607 (2017).

[10] Martin Ester, Hans-Peter Kriegel, Jorg Sander, and Xiaowei Xu. 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of ACM Knowledge Discovery and Data Mining (SIGKDD)*. 226–231.

[11] Johannes Fischer and Volker Heun. 2006. Theoretical and Practical Improvements on the RMQ-Problem, with Applications to LCA and LCE. In *Proceedings of Annual Symposium on Combinatorial Pattern Matching (CPM)*. 36–48.

[12] Jordi Fonollosa, Sadique Sheik, Ramon Huerta, and Santiago Marco. 2015. Reservoir computing compensates slow response of chemosensor arrays exposed to fast varying gas concentrations in continuous monitoring. *Sensors and Actuators B: Chemical* 215 (2015), 618–629.

[13] Junhao Gan and Yufei Tao. 2017. Dynamic Density Based Clustering. In *Proceedings of ACM Management of Data (SIGMOD)*. 1493–1507.

[14] Junhao Gan and Yufei Tao. 2017. On the Hardness and Approximation of Euclidean DBSCAN. *ACM Transactions on Database Systems (TODS)* 42, 3 (2017), 14:1–14:45.

[15] Ramón Huerta, Thiago Mosqueiro, Jordi Fonollosa, Nikolai F. Rulkov, and Irene Rodríguez-Luján. 2016. Online Decorrelation of Humidity and Temperature in Chemical Sensors for Continuous Monitoring. *CoRR* abs/1608.01719 (2016).

[16] David R. Karger, Philip N. Klein, and Robert Endre Tarjan. 1995. A Randomized Linear-Time Algorithm to Find Minimum Spanning Trees. *Journal of the ACM (JACM)* 42, 2 (1995), 321–328.

[17] Valerie King. 1997. A Simpler Minimum Spanning Tree Verification Algorithm. *Algorithmica* 18, 2 (1997), 263–270.

[18] David G. Kirkpatrick and Stefan Reisch. 1984. Upper Bounds for Sorting Integers on Random Access Machines. *Theoretical Computer Science* 28 (1984), 263–276.

[19] Md. Mostofa Ali Patwary, Diana Palsetia, Ankit Agrawal, Wei-keng Liao, Fredrik Manne, and Alok N. Choudhary. 2013. Scalable parallel OPTICS data clustering using graph algorithmic techniques. In *Proceedings of International Conference on High Performance Computing, Networking, Storage and Analysis*. 49:1–49:12.

[20] Johannes Schneider and Michail Vlachos. 2017. Scalable density-based clustering with quality guarantees using random projections. *Data Min. Knowl. Discov.* 31, 4 (2017), 972–1005.

[21] Aniko Vagner. 2016. The GridOPTICS clustering algorithm. *Intell. Data Anal.* 20, 5 (2016), 1061–1084.

[22] Pravin M. Vaidya. 1989. An O(n log n) Algorithm for the All-nearest.Neighbors Problem. *Discrete & Computational Geometry* 4 (1989), 101–115.

APPENDIX

A PROOF OF LEMMA 3.4

Denote (as before) by $\mathcal{G}$ the base graph (Definition 2.1) of P. Define a path in $\mathcal{G}$ as an *H-path* if all its edges have weights at most H.

Lemma A.1. Let $p_1, p_2, ..., p_\ell$ be an H-path. If *core-dist*$(p_\ell) \leq H$, then $p_\ell, p_{\ell-1}, ..., p_1$ is also an H-path.

Proof. For each $i \in [1, \ell - 1]$, since $w(p_i, p_{i+1}) \leq H$, *core-dist*$(p_i) \leq H$ and $dist(p_i, p_{i+1}) \leq H$. Combining this with *core-dist*$(p_\ell) \leq H$ shows that $w(p_{i+1}, p_i) \leq H$ for every $i \in [1, \ell - 1]$. □

Given two points $p_1, p_2 \in P$, we say they are *strongly H-connected* if $\mathcal{G}$ has an H-path from p_1 to p_2, and an H-path from p_2 to p_1.

Lemma A.2. *Two points p_1, p_2 are both in the backbone of the same H-valley in the OPTICS sequence if and only if they are strongly H-connected.*

Proof. *The if direction.* Let π_1 be the H-path from p_1 to p_2, and π_2 the H-path from p_2 to p_1. The concatenation of π_1 and π_2 makes a cycle. Let u be the first vertex in the cycle that appears in the OPTICS sequence. Denote by Υ the H-valley where u belongs. Let the H-valley be $(o_x, h_x), (o_{x+1}, h_{x+1}), ...(o_y, h_y)$ where the subscripts indicate their positions in the OPTICS sequence. We will prove that both p_1 and p_2 are in the backbone of Υ.

Let S be the set of points before o_x in the OPTICS sequence, i.e., $S = \{o_1, o_2, ..., o_{x-1}\}$. Recall that o_x can be removed from V_{unseen} at either Line 4 or Line 8 of the OPTICS procedure. In both cases, it must hold that, in $\mathcal{G}$, all the edges from S to $P \setminus S$ have weights greater than H. This is obvious if the removal of o_x happens at Line 4, in which case S has no edges to $P \setminus S$ at all. Suppose that (p, o_x) is the edge at Line 6 that got o_x deleted at Line 8. We thus know that (p, o_x) is the lightest among all the edges from S to $P \setminus S$, while the definition of H-valley says $w(p, o_x) = h_x > H$.

That no edges from S to $P \setminus S$ have weights at most H implies:

- Fact 1: For any vertex v in Υ, there is an H-path from o_x to v passing only the vertices outside S.
- Fact 2: If there is an H-path from o_x to a vertex $v \notin S$ that passes only vertices outside S, v must appear in Υ.

By definition of u, none of the vertices on π_1 and π_2 belong to S. Combining this with Fact 1, we know that o_x has H-paths to p_1 and p_2 respectively: these paths reach u first, and then use the edges of π_1 and π_2; they include only vertices outside S. Hence, both p_1 and p_2 are in Υ according to Fact 2.

The existence of π_1 and π_2 indicates that *core-dist*$(p_1) \leq H$ and *core-dist*$(p_2) \leq H$. Hence, p_1 and p_2 are in the backbone of Υ.

The only-if direction. Let $\Upsilon = ((o_x, h_x), (o_{x+1}, h_{x+1}), ...(o_y, h_y))$ be the H-valley with p_1, p_2 in the backbone. Fact 1 still holds on $S = \{o_1, o_2, ..., o_{x-1}\}$; hence, o_x can reach any of o_{x+1}, o_{x+2}, ..., o_y with an H-path. By Lemma A.1, if o_i ($i \in [x + 1, y]$) is in the backbone of Υ, reversing the H-path from o_x to o_i gives an H-path from o_i to o_x. Thus, any two backbone vertices of Υ can reach each other via H-paths (through o_x), which completes the proof. □

Recall from Section 2.1 that the DBSCAN clusters can be defined over an undirected graph $G(H)$. Two points p_1, p_2 are strongly H-connected in $\mathcal{G}$ if and only if they can reach each other in $G(H)$. Both statements in Lemma 3.4 follow directly from Lemma A.2 and the way that clusters are formed (as explained in Section 2.1).

B PROOF OF LEMMA 4.2

We need the following property of height-augmented permutations:

Lemma B.1. *Let σ be any height-augmented permutation on a set P of points. Consider any distinct points $p_1, p_2, p_3 \in P$. For any*

value H, if we know that (i) p_1, p_2 are in the same H-valley and (ii) p_1, p_3 are in the same H-valley, then p_1, p_2, p_3 must all be in the same H-valley.

Proof. Consider first that the relative ordering of the three points in σ is p_1, p_2, p_3. In this case, the lemma is obviously true, because p_2 is on the subsequence from p_1 to p_3, while the subsequence must be a part of the H-valley covering p_1 and p_3. The same argument applies to any relative ordering where p_1 ranks either the first or last among the three points.

It suffices to discuss on the scenario where the ordering is p_2, p_1, p_3 (as p_3, p_1, p_2 is symmetric). As the subsequence from p_2 to p_1 is a part of an H-valley, the height of p_1 must be at most H. This, in turn, proves that the entire subsequence from p_2 to p_3 must be a part of an H-valley. □

Lemma 4.2 follows from Definition 4.1 and the above lemma.

C CORRECTNESS PROOF FOR OUR ALGORITHM IN SECTION 4.2

We will prove that the sequence returned by our algorithm in Section 4.2 fulfills the two requirements in Definition 4.1. Recall that our algorithm constructs a λ-WSPD $(U_1, V_1), ..., (U_m, V_m)$, about which we have:

Lemma C.1. *Fix an arbitrary (U_i, V_i) ($i \in [1, m]$). Let H be the distance between an arbitrary point in U_i and an arbitrary point in V_i. If $|U_i| \geq$ minPts (or $|V_i| \geq$ minPts), then every point $p \in U_i$ (or V_i, resp.) has core-dist(p) less than H.*

Proof. We consider only $p \in U_i$ due to symmetry. By Lemma 2.6 (and applying $\lambda \geq \sqrt{8} > 2$), any two points in U_i have distance less than H. If $|U_i| \geq$ *minPts*, then p has distance less than H to at least *minPts* points. □

We inherit from Appendix A the definition of *H-paths* in $\mathcal{G}$, and also the definition of two points being *strongly H-connected* in $\mathcal{G}$. Consider the graph $\tilde{\mathcal{G}}$ constructed by Step 1 of our algorithm; remember that all edges are undirected, and that their weights are defined in (3). Similar to H-paths in $\mathcal{G}$, define a path in $\tilde{\mathcal{G}}$ as an *H-path* if all its edges have weights at most H.

Lemma C.2. *Both statements are true for any $H \leq \epsilon_{max}/(1+\rho)$:*

- *If two points are connected by an H-path in $\tilde{\mathcal{G}}$, they are strongly $H(1+\rho)$-connected in $\mathcal{G}$.*
- *If two points are strongly H-connected in $\mathcal{G}$, they are connected by an H-path in $\tilde{\mathcal{G}}$.*

Proof. The 1st bullet is straightforward. Let $u_1, u_2, ..., u_\ell$ be the vertices on an H-path in $\tilde{\mathcal{G}}$ between $p_1 = u_1$ and $p_2 = u_\ell$. By (3),

$$\max\{\textit{core-dist}(u_i), \textit{core-dist}(u_{i+1}), \textit{dist}(u_i, u_{i+1})\} \leq H(1+\rho) \leq \epsilon_{max}$$

holds for every $i \in [1, \ell - 1]$. Hence, in $\mathcal{G}$, both $w(u_i, u_{i+1})$ and $w(u_{i+1}, u_i)$ are at most $H(1+\rho)$, proving that p_1, p_2 are strongly $H(1+\rho)$-connected.

The rest of the proof concentrates on the 2nd bullet. Define $X = \{\{u, v\} \mid u, v \in P, u \neq v, \textit{dist}(u, v) \leq H, \textit{core-dist}(u) \leq H$, and $\textit{core-dist}(v) \leq H\}$. Note that each member of X is an *unordered* pair $\{u, v\}$, and that u and v are strongly H-connected. It suffices[7] to prove the claim:

For any $\{u, v\}$ in X, there is an H-path between u and v in $\tilde{G}$.

Sort all the $\{u, v\}$ in X by $\textit{dist}(u, v)$. We will prove the above claim by induction on $\textit{dist}(u, v)$.

Base Case. Let $\{u, v\}$ be an unordered pair in X with the smallest $\textit{dist}(u, v)$. The combination of $\textit{core-dist}(u) \leq H$, $\textit{core-dist}(v) \leq H$, and $\textit{dist}(u, v) \leq H$ gives $\tilde{w}(u, v) \leq H$. We will prove that the edge (u, v) exists in $\tilde{\mathcal{G}}$, and hence, makes an H-path between u and v in $\tilde{\mathcal{G}}$. Let (U_i, V_i) be the pair in the λ-WSPD of P with $u \in U_i$ and $v \in V_i$. We claim that $|U_i| <$ *minPts* and $|V_i| <$ *minPts*. This means that Step 1 of our algorithm processes (U_i, V_i) as Case IV, and hence, adds the edge (u, v) to $\tilde{\mathcal{G}}$.

Due to symmetry, we prove only $|U_i| <$ *minPts*. Assume that this is not true. Then, Lemma C.1 asserts that $\textit{core-dist}(p) < \textit{dist}(u, v) \leq H$ for any point $p \in U_i$. For any distinct points p_1, p_2 in U_i, Lemma 2.6 tells us that $\textit{dist}(p_1, p_2) < \textit{dist}(u, v) \leq H$. Hence, $\{p_1, p_2\}$ is an unordered pair in X, contradicting the definition of (u, v).

Inductive Case. Fix an arbitrary unordered pair $\{u, v\} \in X$. Assuming that the 2nd bullet holds for any $\{u', v'\} \in X$ satisfying $\textit{dist}(u', v') < \textit{dist}(u, v)$, next we prove that the bullet is also true on $\{u, v\}$, namely, u and v are connected by an H-path in $\tilde{\mathcal{G}}$.

Let (U_i, V_i) be the pair in the λ-WSPD of P with $u \in U_i$ and $v \in V_i$. Recall that Step 1 of our algorithm processes (U_i, V_i) by distinguishing Cases I-IV. We discuss each case in turn.

Case I. By the 2nd bullet in Lemma 2.5:

$$\begin{aligned} \textit{dist}(\textit{rep}(U_i), \textit{rep}(V_i)) &\leq (1 + 8/\lambda^2) \cdot \textit{dist}(u, v) \\ &= (1+\rho) \cdot \textit{dist}(u, v) \leq (1+\rho)H \end{aligned}$$

where the last inequality used (2). Applying Lemma C.1 with $\textit{dist}(u, v) \leq H$, we get $\textit{core-dist}(\textit{rep}(U_i)) < H$ and similarly, $\textit{core-dist}(\textit{rep}(V_i)) < H$. Hence, $\tilde{w}(\textit{rep}(U_i), \textit{rep}(V_i)) \leq \frac{1}{1+\rho} \cdot (1 + \rho)H = H < \epsilon_{max}$. This proves that $\tilde{\mathcal{G}}$ has an edge $(\textit{rep}(U_i), \textit{rep}(V_i))$.

Lemma 2.6 shows that $\textit{dist}(u, \textit{rep}(U_i)) < \textit{dist}(u, v)$. Hence, by the inductive assumption, u and $\textit{rep}(U_i)$ must be connected by an H-path in $\tilde{\mathcal{G}}$.[8] Similarly, this is also true for v and $\textit{rep}(V_i)$. Thus, we have found an H-path to connect u and v in $\tilde{\mathcal{G}}$.

Case II. $\tilde{\mathcal{G}}$ has an edge $(u, \textit{rep}(V_i))$. By the 3rd bullet of Lemma 2.5:

$$\textit{dist}(u, \textit{rep}(V_i)) \leq (1 + 8/\lambda^2) \cdot \textit{dist}(u, v) \leq (1+\rho) \cdot H.$$

As explained in Case I, $\textit{core-dist}(\textit{rep}(V_i)) < H$ which, together with $\textit{core-dist}(u) \leq H$, leads to $\tilde{w}(u, \textit{rep}(V_i)) \leq \frac{1}{1+\rho}(1+\rho)H = H$.

If $v = \textit{rep}(V_i)$, we are done because the edge $(u, \textit{rep}(V_i))$ itself is an H-path in $\tilde{G}$ between u and v. Otherwise, as in Case I, the inductive assumption shows the existence of an H-path in $\tilde{\mathcal{G}}$ between v and $\textit{rep}(V_i)$, thus also giving an H-path between u and v.

Case III. Symmetric to Case II.

[7] To see why, consider any distinct points u', v' that are strongly H-connected, but $\{u', v'\}$ is not in X. Let $p_1, p_2, ..., p_x$ (where $u' = p_1$, $v' = p_x$, and $x \geq 3$) be an H-path from p_1 to p_x. By Lemma A.1, $p_x, p_{x-1}, ..., p_1$ is an H-path. It follows that $\{p_i, p_{i+1}\}$ is in X, for each $i \in [1, x-1]$. Thus, if we can prove that $\tilde{G}$ has an H-path between p_i and p_{i+1} for every i, the concatenation of all these paths is an H-path between u' and v'.

[8] If $u = \textit{rep}(U_i)$, the path is a trivial one with no edges.

Case IV. $\tilde{\mathcal{G}}$ has an edge (u, v). The 2nd bullet is true because $\tilde{w}(u, v) = \max\{\textit{core-dist}(u), \textit{core-dist}(v), \frac{dist(u,v)}{1+\rho}\} \leq H$. □

Regarding the MSF $\mathcal{F}$ from Step 2 of our algorithm, we have:

Lemma C.3. *For any $H > 0$, two points are connected by an H-path in $\tilde{\mathcal{G}}$, if and only if they are connected by an H-path in $\mathcal{F}$.*

Proof. The if direction is trivial. Next, we prove the only-if direction. Suppose that there exist points p_1, p_2 violating the lemma. Let $u_1, u_2, ..., u_\ell$ be an H-path in $\tilde{\mathcal{G}}$ between $p_1 = u_1$ and $p_2 = u_\ell$. Let i be the smallest integer such that u_i and u_{i+1} are not connected by an H-path in $\mathcal{F}$. This means that the path from u_i to u_{i+1} in $\mathcal{F}$ contains an edge with weight (as calculated by (3)) greater than H. Removing this edge and adding (u_i, u_{i+1}) will produce another spanning forest with a smaller weight than $\mathcal{F}$, contradicting that $\mathcal{F}$ is an MSF. □

Consider the height-augmented sequence σ produced by Step 3. The valleys in σ have the following pattern:

Lemma C.4. *Two points p_1, p_2 are in the backbone of the same H-valley in σ if and only if there is an H-path between them in $\mathcal{F}$.*

Proof. All the "weights" in this proof are calculated by (3).

The if direction. Let π be an H-path between p_1 and p_2 in $\mathcal{F}$. Let u be the first vertex on π that appears in σ. Denote by Υ the H-valley in σ where u belongs; let Υ be $(o_x, h_x), (o_{x+1}, h_{x+1}), ...$ (o_y, h_y) where the subscripts indicate their positions in σ. We will prove that both p_1 and p_2 are in the backbone of Υ.

Let S be the set of points before o_x in σ, i.e., $S = \{o_1, o_2, ..., o_{x-1}\}$. Recall that o_x can be removed from V_{unseen} at either Line 4 or Line 7 of the approximate OPTICS procedure. In both cases, it must hold that, in $\mathcal{F}$, all the edges from S to $P \setminus S$ have weights greater than H. This is obvious if the removal of o_x happens at Line 4, in which case S has no edges to $P \setminus S$ at all. Suppose that (p, o_x) is the edge at Line 6 that got o_x deleted at Line 7. We thus know that (p, o_x) is the lightest among all the edges from S to $P \setminus S$, while the definition of H-valley says $\tilde{w}(p, o_x) = h_x > H$.

This implies:

- Fact 1: For any vertex v in Υ, $\mathcal{F}$ has an H-path between o_x and v that involves only the vertices outside S.
- Fact 2: For any vertex $v \notin S$, if $\mathcal{F}$ has an H-path between o_x and v that includes only vertices outside S, v appears in Υ.

By definition of u, none of the vertices on π belong to S. Combining this with Fact 1, we know that, in $\mathcal{F}$, o_x has H-paths to p_1 and p_2 respectively: these paths reach u first, and then use the edges of π; they include only vertices outside S. Hence, both p_1 and p_2 are in Υ according to Fact 2.

The existence of π tells us $\textit{core-dist}(p_1) \leq H$ and $\textit{core-dist}(p_2) \leq H$ (otherwise, any edges of p_1, p_2 in $\tilde{G}$ must have weights greater than H). We thus know that p_1 and p_2 are in the backbone of Υ.

The only-if direction. Let $\Upsilon = ((o_x, h_x), (o_{x+1}, h_{x+1}), ...(o_y, h_y))$ be the H-valley containing p_1, p_2 in the backbone. Fact 1 still holds on $S = \{o_1, o_2, ..., o_{x-1}\}$; hence, o_x can reach any of $o_{x+1}, o_{x+2}, ...,$ o_y with an H-path in $\mathcal{F}$. Hence, there is an H-path between any two vertices of Υ, thus completing the proof. □

The correctness of our algorithm becomes obvious:

Corollary C.5. *σ is a ρ-approximate OPTICS sequence.*

Proof. From Lemma A.2, and the 2nd bullet of Lemma C.2, if two points p_1, p_2 are in the backbone of the same H-valley of the OPTICS sequence, they are connected by an H-path in $\tilde{\mathcal{G}}$. By Lemmas C.3 and C.4, p_1, p_2 must be in the backbone of the same H-valley in σ. Hence, σ satisfies the 1st requirement of Definition 4.1.

From Lemmas C.4, C.3, and the 1st bullet of Lemma C.2, if two points p_1, p_2 are in the backbone of the same H-valley in σ, they are strongly $H(1+\rho)$-connected in $\mathcal{G}$. By Lemma A.2, p_1, p_2 must be in the backbone of the same $H(1+\rho)$-valley in the (exact) OPTICS sequence. This proves that σ fulfills the 2nd requirement of Definition 4.1. □

D PROOF OF LEMMA 5.2

Recall from Section 2.1 the definitions of graphs $G(\epsilon)$ and $\tilde{G}(\epsilon, \rho)$ used to formulate exact DBSCAN and ρ-approximate DBSCAN, respectively. Both of them are undirected graphs defined only on ϵ-core points. It suffices to prove the existence of a $\tilde{G}(\epsilon, \rho)$ whose CCs (connected components) correspond to exactly the clusters of ϵ-core points decided by Steps (1) and (2) in the lemma's statement. Once this is done, Step (3) assigns the non-ϵ-core points in the same way as what ρ-approximate DBSCAN does based on the CCs of $\tilde{G}(\epsilon, \rho)$.

We construct such $\tilde{G}(\epsilon, \rho)$ explicitly as follows:

- First, set $\tilde{G}(\epsilon, \rho)$ to $G(\epsilon)$.
- Then, for every edge (u, v) in $\mathcal{F}(\epsilon)$, add an unweighted edge (u, v) to $\tilde{G}(\epsilon, \rho)$ if it is not already there. The 1st bullet of Lemma C.2 asserts that u and v must be strongly $\epsilon(1+\rho)$-connected in $\mathcal{G}$. In turn, this says that the edge (u, v) must belong to $G(\epsilon(1+\rho))$, as desired.

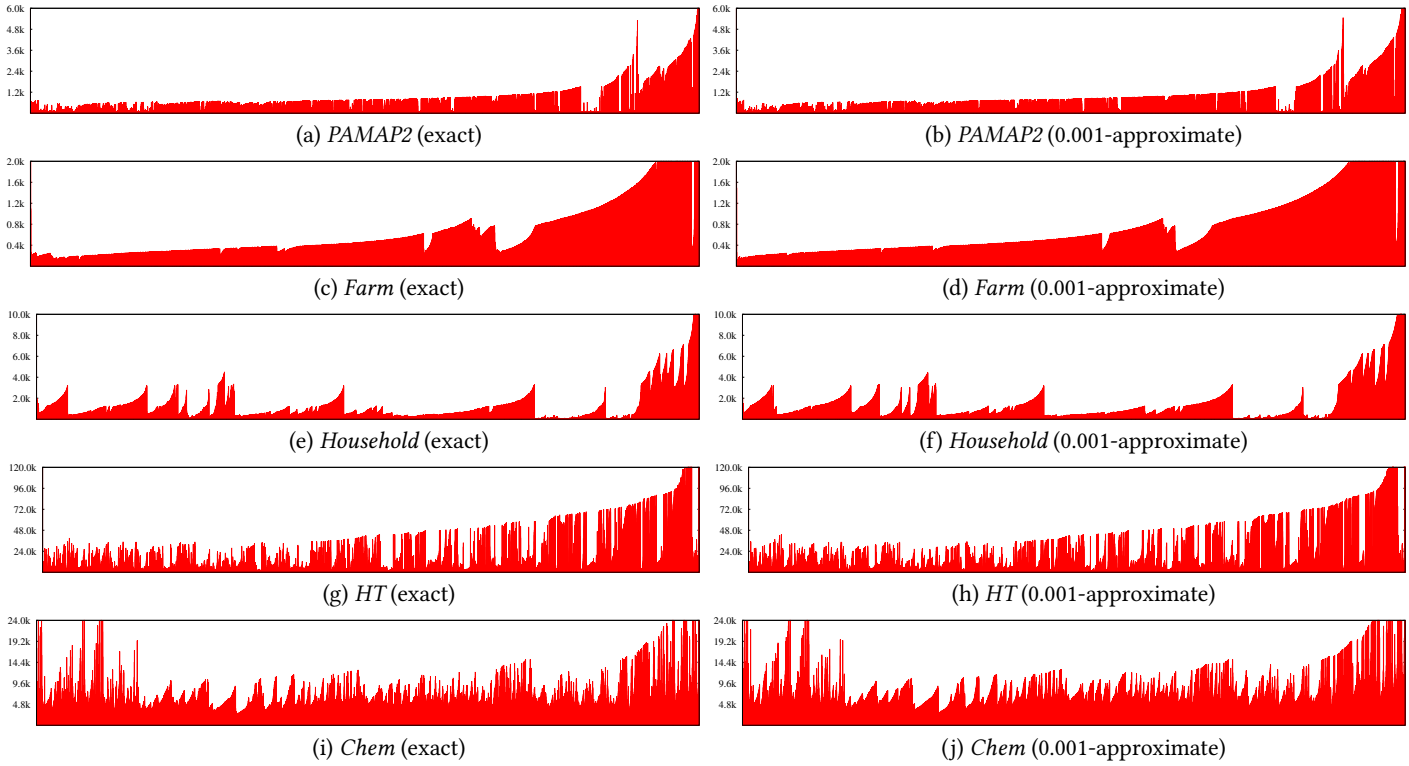
If two ϵ-core points p_1, p_2 are in the same cluster after Step (2), they must belong to the same tree of $\mathcal{F}(\epsilon)$. In this case, p_1, p_2 must be in the same CC of $\tilde{G}(\epsilon, \rho)$ because all the edges of $\mathcal{F}(\epsilon)$ are in $\tilde{G}(\epsilon, \rho)$. To complete the proof, it remains to show that if two ϵ-core points p_1, p_2 are in the same CC of $\tilde{G}(\epsilon, \rho)$, they must be in the same tree of $\mathcal{F}(\epsilon)$, i.e., in the same cluster after Step (2).

If p_1, p_2 are in the same CC of $G(\epsilon)$, they must be strongly ϵ-connected in $\mathcal{G}$. By the 2nd bullet of Lemma C.2 and Lemma C.3, this means that in $\mathcal{F}$ they are connected by a path having edges with weights at most ϵ, and hence, are in the same tree of $\mathcal{F}(\epsilon)$. Consider now the case where p_1, p_2 that are not in the same CC of $G(\epsilon)$. Let π be a path from p_1 to p_2 in $\tilde{G}(\epsilon, \rho)$; denote by $u_1, u_2, ..., u_\ell$ the vertices on π with $p_1 = u_1$ and $p_2 = u_\ell$. We can find a path on $\mathcal{F}(\epsilon)$ from u_1 to u_ℓ as follows. For each $i \in [1, \ell - 1]$, if u_i and u_{i+1} are in the same CC of $G(\epsilon)$, we can definitely travel from u_i to u_{i+1} on $\mathcal{F}(\epsilon)$ as proved earlier. Otherwise, (u_i, u_{i+1}) is not an edge in $G(\epsilon)$, and must have come from $\mathcal{F}(\epsilon)$. This completes the whole proof of Lemma 5.2.

E PROOF OF LEMMA 5.4

Lemma 5.2 directly gives:

Corollary E.1. *For any value $\epsilon \leq \epsilon_{max}/(1+\rho)$, two ϵ-core points p_1, p_2 are in the same cluster in $C(\epsilon, \rho)$ if and only if, in $\mathcal{F}$, they are connected by a path containing only edges of weights at most ϵ.*



(a) *PAMAP2* (exact) (b) *PAMAP2* (0.001-approximate)

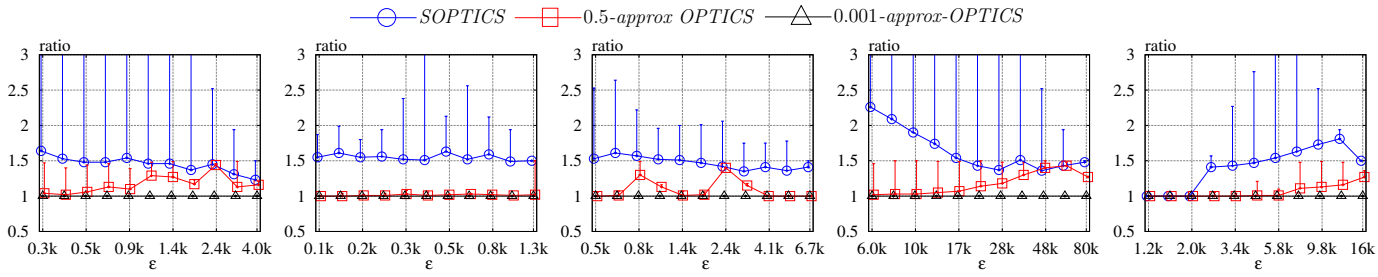
(c) *Farm* (exact) (d) *Farm* (0.001-approximate)

(e) *Household* (exact) (f) *Household* (0.001-approximate)

(g) *HT* (exact) (h) *HT* (0.001-approximate)

(i) *Chem* (exact) (j) *Chem* (0.001-approximate)

Figure 20: Exact OPTICS diagrams vs. our approximate diagrams on 10% sample sets



(a) *PAMAP2* (b) *Farm* (c) *Household* (d) *HT* (e) *Chem*

Figure 21: VP-ratios on 10% sample sets

The corollary says that q_i and q_{i+2} are connected by a path in $\mathcal{F}$ with edges that have weights at most ϵ. By Lemma C.4, they belong to the same ϵ-valley of the ρ-approximate OPTICS sequence that produced $\mathcal{F}$. As q_{i+1} is between q_i and q_{i+2} in sequence rank, q_i and q_{i+1} are also in the same ϵ-valley, and hence, in the backbone of the valley. It thus follows from Lemma C.4 and Corollary E.1 that q_i and q_{i+1} are in the same cluster of $\mathcal{C}(\epsilon, \rho)$.

F EXTRA EXPERIMENTS: SAMPLING AND APPROXIMATION QUALITY

For efficiency reasons, it would be tempting to compute an OPTICS diagram from a sample set, and use this diagram as a substitute for the diagram on the original dataset. To explore this direction, we present, in Figure 20, the exact and our 0.001-approximate OPTICS diagrams on 10% sample sets of the five datasets. These diagrams indeed seem to have similar shapes compared to those in Figures 12 and 13, except that they differ quite considerably in the heights of valleys. The ϵ values where significant valleys are identified in the exact diagrams (see Figures 12 and 13) are almost all off by a rather large margin in these sample-set diagrams.

Even when a user decides to opt for sample-set diagrams, our proposed approximation techniques still provide an appealing alternative choice, because our approximate diagrams are highly accurate in all cases, and yet can be computed much faster. Regarding quality, we complement the approximate diagrams in Figure 20 with the VP-ratio results in Figure 21, which compares our 0.001-approximate diagrams against *SOPTICS* and our own $\rho = 0.5$ diagrams in the same style as in Figure 15. Regarding efficiency, we refer the reader to the scalability experiments (Figure 17) in Section 6.3.