Online Outlier Exploration Over Large Datasets

Lei Cao[†], Mingrui Wei [†], Di Yang [‡], Elke A. Rundensteiner[†]
[†]Worcester Polytechnic Institute Worcester, MA 01609, USA
[‡]Oracle Corporation Nashua, NH 03062, USA
lcao|netwrm01|rundenst@cs.wpi.edu,di.yang@oracle.com

ABSTRACT

Traditional outlier detection systems process each individual outlier detection request instantiated with a particular parameter setting one at a time. This is not only prohibitively time-consuming for large datasets, but also tedious for analysts as they explore the data to hone in on the appropriate parameter setting and desired results.

In this work, we present the first online outlier exploration platform, called ONION, that enables analysts to effectively explore anomalies even in large datasets. First, ONION features an innovative interactive anomaly exploration model that offers an "outlier-centric panorama" into big datasets along with rich classes of exploration operations. Second, to achieve this model ONION employs an online processing framework composed of a one time offline preprocessing phase followed by an online exploration phase that enables users to interactively explore the data. The preprocessing phase compresses raw big data into a knowledge-rich ONION abstraction that encodes critical interrelationships of outlier candidates so to support subsequent interactive outlier exploration. For the interactive exploration phase, our ONION framework provides several processing strategies that efficiently support the outlier exploration operations. Our user study with real data confirms the effectiveness of ONION in recognizing "true" outliers. Furthermore as demonstrated by our extensive experiments with large datasets, ONION supports all exploration operations within milliseconds response time.

Categories and Subject Descriptors

H.2 [**Information Systems**]: Database Management

Keywords

Outlier; Online Exploration; Parameter Setting

1. INTRODUCTION

This big data era provides tremendous opportunities for extracting insights from big datasets via advanced analytics. Among these data analytical tasks, understanding "abnormalities" in the data is one of the fundamental services essential for applications ranging from credit fraud prevention, climate change analysis, to financial strategy planning. They all rely on effective outlier detection techniques to discover suspicious card usage and potential identity theft, to forecast disastrous weather phenomena, and to predict market changes and trade opportunities, respectively [7].

In this context, we focus on one well-established abnormality definition [13] called "distance-based outlier", that effectively captures "outliers" [7, 16] − data points behaving significantly differently from others in a dataset.

Limitations of Traditional One-At-A-Time Query Approach. Traditional outlier detection systems require the analyst to select a fixed set of parameter values, most notably a distance threshold r and a count threshold k [13], and then to submit this instantiated request to attempt to detect outliers of interest. This request is then executed from scratch as a one time query to compute the outliers from the target dataset that match that specification. This one-at-a-time query approach suffers from severe limitations.

First, similar to many other data analytical tasks, a good input parameter setting (in this case a pair of appropriate values for k and r parameters) is the key for the analysts to gain insight in data and identify the "true" outliers. However using the current systems, to achieve this the analyst has to continuously re-submit individual requests with different parameter settings in a trial-and-error fashion and *interactively* analyze the respective results. This is extremely ineffective and would be a taxing process for the analysts because of the infinite number of possible parameter settings.

Second, although optimization strategies for executing such requests have been proposed [1, 4, 11, 5], mining outliers according to a particular parameter setting from scratch on large data still tends to take hours as confirmed in our experiments (Sec. 4.3.1). This is clearly not matching the stringent response time of seconds or so required by interactive systems − thus risks losing the attention of the analysts during the process.

Worst yet, if the system only supports such one-at-a-time queries, each individual query would independently generate an outlier set as answers. However without establishing an explicit connection among these "isolated outlier views", it is challenging for the analyst to compare and contrast the outlier sets produced by the different queries over time − especially when working with a big dataset and in turn a large outlier base.

Furthermore important insights, such as how stable the outlier status of each point is, how the detected outlier set migrates across different parameter settings, or what the relationship among different outlier points is (for example, whether some points are "stronger" outliers than others), might be missed during this tedious yet expensive exploration process. This information is critical for the analysts to interpret the characteristics of the outliers hidden in the

KDD'15, August 10-13, 2015, Sydney, NSW, Australia.
© 2015 ACM. ISBN 978-1-4503-3664-2/15/08 ...$15.00.
DOI: http://dx.doi.org/10.1145/2783258.2783387.

dataset. In short, this one-at-a-time approach is neither effective nor efficient for online interactive analytics.

Proposed Solution. We propose a novel online outlier exploration platform, called ONION, that addresses these problems.

ONION Model. ONION offers users an innovative "outlier-centric panorama" into the outliers present within the raw dataset by establishing an interactive anomaly exploration *model*. The ONION model is composed of a comprehensive knowledge base supported by powerful outlier exploration operations.

First, the ONION knowledge base explicitly models the distribution of the outliers with respect to their associated parameter settings by abstracting the *k*, *r* parameters and the points *p* of a dataset D into a three dimensional space called *ONION space* (in short *O-Space*). Further higher level *abstractions* (*P-Space* and *D-Space*) including the stability of the parameter settings in recognizing outliers and the hierarchial domination relationships in abnormality among the outlier candidates independent of any particular parameter setting are extracted and modeled.

Second, rich classes of outlier exploration operations beyond the traditional concept of outlier detection are proposed that not only allow the analysts to understand how parameter changes would impact the captured outliers, but also offer analysts a "parameter-free" approach to identify outliers based on their domain knowledge. Together these operations provide a powerful yet flexible methodology for analysts to explore and interpret the outliers within a large dataset with respect to an infinite parameter space. This enables them to quickly approach the true outliers even in a completely unknown dataset with zero knowledge about the appropriate parameter settings.

ONION Framework. We develop a novel outlier exploration *framework* that implements this ONION model. The ONION framework consists of two components, namely offline one-time ONION knowledge base construction and subsequent interactive outlier exploration. The offline component constructs the ONION knowledge base, while leveraging this knowledge base the interactive component achieves true sense making by supporting the outlier exploration operations with real time responsiveness.

Our offline component rests upon the key observation that given any outlier candidate oc_i in a dataset D there exists a small set of data points in D whose distances to oc_i delimit the entire parameter space $\mathbb{P}$ composed of all possible parameter settings into two segments, so called space delimiter. The parameter settings that fall into the same segment classify p_i to be the same outlier status. By collecting the space delimiters of all outlier candidates utilizing a quadratic complexity algorithm and leveraging their relative positions we successfully represent the key components of outlier analytics, namely the distribution characteristics of outliers, the property of parameter space, and the linkage between outliers and input parameter settings into a compact hierarchical structure.

This structure is proven to be sufficient yet necessary to support all outlier exploration operations, while being compact enough to be stored in the main memory of a standard configuration PC even for a big dataset. Therefore the DISK I/O costs, confirmed to be the dominating costs of outlier detection, are completely avoided by our interactive explorer. In particular by leveraging the relative abnormality of the outlier candidates and the relative strictness of the parameter stable regions in recognizing outliers, we are able to answer all outlier exploration operations in logarithmic time on the cardinality of the outlier candidates.

Our user study with real GMTI data [9] confirms the effectiveness of our ONION platform in quickly honing in on the appropriate parameter settings and in turn approaching the "true" outliers. Furthermore ONION consistently outperforms its state-of-the-art competitors at least *five orders of magnitude* in the processing times for the traditional outlier detection queries in a rich diversity of scenarios tested with big real geolocation datasets. Better yet, it supports a rich set of outlier exploration operations not previously supported – all in milliseconds response time.

Contributions. The contributions of this work include:

1) We propose the first interactive outlier analytics platform that enables analysts to pinpoint appropriate parameter settings and explore outliers in a systematic way.

2) We establish for the analysts an "outlier-centric panorama" into big datasets by integrating the input data and parameter space into a comprehensive multi-space ONION knowledge base.

3) We design logarithmic-complexity algorithms for the processing of each outlier exploration operations with realtime responsiveness by leveraging the compact ONION knowledge base.

4) We confirm the superiority of ONION compared to the traditional mining platform in effectiveness of recognizing true outliers by conducting a user study with real GMTI dataset.

5) Our experimental performance study demonstrates that ONION is at least five orders of magnitude faster than its state-of-the-art competitors for traditional outlier detection queries.

2. ONION MODEL

We propose the online outlier exploration model or in short ONION for modeling and exploring the characteristics of distance-based outliers in a dataset D. We first introduce the concept of distance-based outlier proposed in [13]. We use the term data point or point to refer to a multi-dimensional tuple. Let D be a set with n points $p_1, p_2, p_3, ... \, p_n$. The function $d(p_i, p_j)$ denotes the distance between data points p_i and p_j in D.

Definition 1. ***Distance-Based Outlier.*** *Given a dataset D, a range threshold r ($r \geq 0$) and a count threshold k ($k \geq 1$), a point $p_i \in D$ is an outlier if fewer than k points p_j exist in D whose distance to p_i $d(p_i, p_j)$ is no larger than r.*

Fig. 1 sketches a high-level view of the ONION model. It is composed of the *multi-space* abstraction capturing the key characteristics and interrelationships of outliers and a rich set of *outlier exploration operations*.

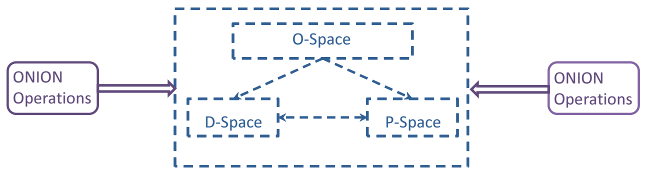

Figure 1: ONION Model

2.1 Multi-Space Abstraction

Our *multi-space* abstraction is composed of three interlinked spaces that we will now define below.

ONION Space. ONION space or in short *O-Space* is a three-dimensional space that models the distribution of the outliers with respect to their associated parameter settings.

Definition 2. ***O-Space*** *denoted as $\mathbb{O}^S$ (Dim_k, Dim_r, Dim_d) is a three-dimensional space with the possible settings of parameters r, k and data points p in dataset D being its three dimensions. The dimension Dim_k ranges over the values that the parameter k can take in the universe of natural number $U_k : [k_{min}, k_{max}]$, where k_{min} and k_{max} are the user-specified lower and upper bounds*

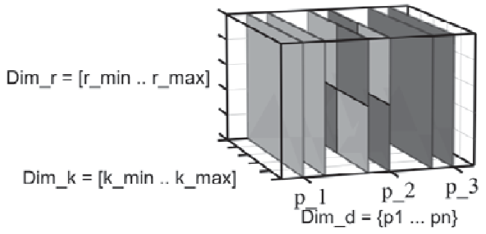

Figure 2: ONION Space

of the k values. Similarly the dimension Dim_r corresponds to the domain of real numbers U_r : $[r_{min}, r_{max}]$ with r_{min} and r_{max} the lower and upper bounds of the values of parameter r. Lastly the dimension Dim_d represents all points $p \in D$ randomly organized into a linear order. Each point is assigned a position in [1,| D |]. Each coordinate $(k_i, r_i, p_i) \in \mathbb{O}^S$ maps to a boolean value $v \in$ {0,1} indicating whether point p_i is an outlier with respect to parameter values k_i and r_i.

In this O-Space any combination of k and r values on the dimensions Dim_k and Dim_r forms a parameter setting ps_i denoted by $ps_i(k_i, r_i)$. Conceptually O-Space encodes the outlier status of all points in *D* with respect to all possible parameter settings.

Since dimension Dim_d represents all data points in dataset D, Dim_d corresponds to a discrete domain of positions. In other words the three-dimensional O-Space can be thought as a sequence of two dimensional slices as shown in Fig. 2. Each slice models the outlier status distribution with respect to all possible parameter settings for one particular point p_i in dataset D.

Based on this O-Space, we further design two additional *higher level* abstractions called *parameter space* and *data space* respectively as shown below.

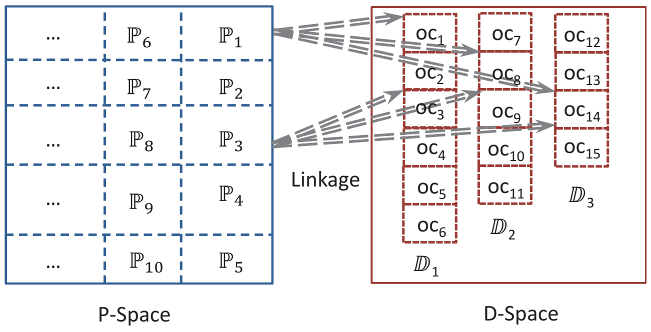

Figure 3: P-Space & D-Space

Parameter Space. Parameter space or in short *P-Space* is based on the observation that despite the infinite number of possible parameter settings, a large range of continuous parameter settings often generate the same set of outliers.

Definition 3. ***P-Space*** *$\mathbb{P} = \mathbb{P}_1 \bigcup \mathbb{P}_2 ... \bigcup \mathbb{P}_m$, such that:*
(1) given any two parameter setting subsets $\mathbb{P}_i$ and $\mathbb{P}_j$ of $\mathbb{P}$ ($1 \leq i, j \leq m$), $\mathbb{P}_i \bigcap \mathbb{P}_j = \emptyset$;
(2) given any two parameter settings ps_j and ps_l in the same $\mathbb{P}_i$ ($1 \leq i \leq m$), ps_j and ps_l generate the same set of outliers.

In other words P-Space divides the two-dimensional space formed by the set of all possible values on the Dim_k, Dim_r axes into a set of disjoint regions. Within each region no matter how the parameter settings are adjusted, the set of outliers generated from dataset D remains unchanged. Each such region is called a *stable region*.

P-Space, partitioning the infinite number of parameter settings into finite number of stable regions, explicitly reveals the influence of the parameter setting adjustment. This offers the analysts an opportunity to determine the appropriate parameter settings using a systematic methodology instead of a random trial and error process.

Data Space. Data space or in short *D-Space* leverages the key abnormality properties demonstrated in the points of dataset D, namely *outlier candidacy* and *domination relationship*.

Outlier Candidacy. Despite the infinite cardinality of P-Space $\mathbb{P}$, given a point p_i in dataset D, its outlier status might be constant with respect to all parameter settings in $\mathbb{P}$. In other words, some points are guaranteed to be outliers in the entire P-Space so called *const outliers*, while some other points are guaranteed to be permanent inliers through the entire P-Space so called *const inliers*. In our O-Space these points would thus correspond to a slice that is all 1's for constant outlier or all 0's for constant inlier. For example, as shown in Fig. 2 p_1 is a const inlier, while p_3 is a const outlier.

Any point p_i in D, that is neither a const outlier nor const inlier, is called an *outlier candidate oc* with respect to $\mathbb{P}$, meaning p_i has opportunity to be classified as outlier for at least some of the parameter setting ps_i in $\mathbb{P}$. In O-Space, an outlier candidate would have at least one cell in its corresponding slice that is 0 (white) and one that is 1 (black). In Fig. 2 p_2 is an outlier candidate.

In practice as confirmed by our experiments (Sec. 4.2), outlier candidates tend to be a strict minority among all points in D. This important *outlier candidacy observation* allows us to significantly reduce the number of data points to be maintained in D-Space. By this, ONION can concentrate the resource utilization on strictly serving these minority outlier candidates, rather than on computing and recording neighborhoods for the general and much larger data population when exploring outliers. Therefore ONION is able to efficiently explore outliers over even big datasets.

Domination Relationship. In dataset D some outlier candidates demonstrate a much *stronger abnormality* than others independent of any particular parameter setting in $\mathbb{P}$. In other words, some data points *dominate* others in abnormality as defined below.

Definition 4. *Given a P-Space $\mathbb{P}$, outlier candidate oc_i in dataset D dominates oc_j if for* ***all*** *parameter settings in $\mathbb{P}$ oc_j is guaranteed to be outlier when oc_i is classified as outlier.*

By Def. 4 if outlier candidate oc_i dominates oc_j, we say that the abnormality of oc_i is stronger than oc_j.

Revealing the domination relationships among outlier candidates ONION offers the analysts an opportunity to better understand several characteristics of the detected outliers from sensitivity to stability. Without such understanding the detected outliers might only be some abstract points indistinguishable from each other for the analysts instead of some true unique abnormal phenomena.

Now we are ready to define our *data space* or in short *D-Space*.

Definition 5. ***D-Space*** *$\mathbb{D} = \mathbb{D}_1 \bigcup \mathbb{D}_2 ... \bigcup \mathbb{D}_m$, such that:*
(1) $\forall$ $oc_i \in \mathbb{D}$, oc_i is an outlier candidate;
(2) given any two outlier candidate subsets $\mathbb{D}_i$ and $\mathbb{D}_j$ of $\mathbb{D}$ ($1 \leq i, j \leq m$), $\mathbb{D}_i \bigcap \mathbb{D}_j = \emptyset$.
(3) the outlier candidates in the same group $\mathbb{D}_i$ are sorted into a linear structure. Given two points oc_j and oc_l with j, l representing their positions in $\mathbb{D}_i$, oc_j dominates oc_l if $j < l$.

In general, leveraging the outlier candidacy and domination relationship properties, D-Space partitions all outlier candidates into multiple disjoint groups. Within each group the domination relationship holds among all members of the group, so called *domination group*. Furthermore the outlier candidates falling in the same domination group are ordered based on the strongness of their abnormality. For example, in Fig. 3, D-Space $\mathbb{D}$ contains three subspaces $\mathbb{D}_1$, $\mathbb{D}_2$, and $\mathbb{D}_3$. In $\mathbb{D}_1$ candidates oc_1 to oc_6 are ordered by the domination relationship. That is, oc_1 dominates other members in $\mathbb{D}_1$. oc_2 is dominated by oc_1, but dominates oc_3 to oc_6.

Linkage between P-Space and D-Space. Furthermore as shown in Fig. 3, our ONION model explicitly establishes *linkages* between P-Space and D-Space, or in short *PD-linkage*.

Definition 6. ***PD-Linkage.*** *Given a stable region $\mathbb{P}_i$ in P-Space $\mathbb{P}$ and a domination group $\mathbb{D}_j$ in D-Space $\mathbb{D}$, there exists a link $l(i, j)$ connecting $\mathbb{P}_i$ to an outlier candidate $oc_t \in \mathbb{D}_j$ such that:*
(1) $\forall$ parameter setting ps_i in $\mathbb{P}_i$, oc_t is classified as outlier;
(2) $\forall$ parameter setting ps_i in $\mathbb{P}_i$, oc_{t-1} is classified as inlier.

By the domination relationship definition in Def. 4, if oc_t is an outlier with respect to ps_i, then any outlier candidates listed behind oc_t in $\mathbb{D}_j$ (oc_{t+1}, oc_{t+2}, ...) are guaranteed to be outliers. Therefore the PD-Linkage explicitly connects the stable regions with their generated outliers. In Fig. 3, stable region $\mathbb{P}_1$ is linked to oc_1 of $\mathbb{D}_1$, oc_8 of $\mathbb{D}_2$, and oc_{14} of $\mathbb{D}_3$. Based on the links we immediately get the outlier set $\mathbb{O}_1$ generated by $\mathbb{P}_1$, that is $\{oc_1, ..., oc_6, oc_8, ..., oc_{11}, oc_{14}, oc_{15}\}$.

Overall the multi-space abstraction explicitly models the distribution of the outliers over all parameter settings, the relationships among the parameter settings, the stability and uniqueness of the outlier candidates. It establishes an innovative "outlier-centric panorama" into the outliers within dataset D.

2.2 ONION Operations

Based on the multi-space abstraction we further envision a rich classes of outlier exploration operations that allow users to explore and interpret outliers as well as pinpoint appropriate parameters.

Definition 7. ***Comparative Outlier Analytics (CO).*** *Given an outlier set $\mathbb{O}_{in}$ as input, we report set of outliers $\mathbb{O}_D$ from dataset D, such that:*
(1) $\forall$ point $p_i \in \mathbb{O}_{in}$, $p_i \in \mathbb{O}_D$; and
(3) $\forall$ point $p_i \in \mathbb{O}_D - \mathbb{O}_{in}$, if any $p_j \in \mathbb{O}_{in}$ is classified as outlier with respect to one $ps_l \in \mathbb{P}$, p_i is guaranteed to be classified as outlier by ps_l.

Leveraging the domination relationship in D-Space, CO operation returns all outliers dominated by the outliers specified in the input set $\mathbb{O}_{in}$. CO offers users a "parameter-free" approach to identify outliers based on their domain knowledge about the dataset. More specifically this CO operation helps analysts to identify outliers in a dataset based on sampling some typical outliers.

Definition 8. ***Outlier-Centric Parameter Space Exploration (PSE).*** *Given an outlier set $\mathbb{O}_{in}$ and a δ ($-1 < \delta < 1$) as input, report all parameter settings $ps_j \in$ P-Space $\mathbb{P}$, such that:*
(1) if $\delta \geq 0$, ps_j identifies an outlier set $\mathbb{O}_j \subseteq \mathbb{O}_{in}$ where $\mid \mathbb{O}_j \mid$ = (1 - δ) $\mid \mathbb{O}_{in} \mid$;
(2) if $\delta \leq 0$, ps_j identifies an outlier set $\mathbb{O}_j \supseteq \mathbb{O}_{in}$ where $\mid \mathbb{O}_j \mid$ = (1 - δ) $\mid \mathbb{O}_{in} \mid$.

PSE leverages the stable region property of P-Space and allows analysts to conveniently evaluate the stability of a given outlier set $\mathbb{O}_{in}$. This is one important indicator of how significant the observed abnormal phenomena is. For example, if we set the δ as 0, PSE will return all the parameter settings that are guaranteed to generate the outliers identical to $\mathbb{O}_{in}$, namely a stable region of $\mathbb{P}$. The scope of the returned parameter settings (the size of the stable region) represents how stable the outlier set is across P-Space.

Furthermore PSE provides a tool for analysts to examine how changes in parameter settings may impact the resulting outliers. PSE achieves this, for example, by allowing the analysts to apply PSE to ask for the parameter settings that would return around (1 - δ)% of $\mathbb{O}_{in}$ as the results and then compare them against the parameter settings that generate $\mathbb{O}_{in}$.

Definition 9. ***Outlier Detection (OD).*** *Given a dataset D and a parameter setting ps_i as input, outlier detection returns:*
(1) all outliers $p_j \in$ D with respect to ps_i if $ps_i \in$ P-Space $\mathbb{P}$; or
(2) all points $p_j \in$ D that are classified as outliers with respect to any parameter setting $\in \mathbb{P}$ if ps_i = NULL.

As shown in Def. 9 unlike the traditional distance-based outlier definition, OD leverages the *outlier candidacy* observation of D-Space to allow the input parameter set as NULL. This will return all points that are guaranteed to be outliers with respect to the entire P-Space, that is, the *constant outliers*.

Use Case. Those operations in combination provide a powerful tool for analysts to quickly approach the parameter settings appropriate for her application. For example, when facing a new dataset recording stock market transactions, an analyst may not have any experience to be able to appropriately determine values for parameters k and r. However, given her domain expertise, she may be aware that certain records are abnormal (outliers). Then a CO operation can be applied to help her identify all outliers satisfying her intuition. If the analyst finds the volume of the outliers $\mathbb{O}$ returned by the CO request too overwhelming, then she could apply PSE to ask for the parameter settings that would return, for example, around 60% of $\mathbb{O}$ as the result by setting δ as 0.4. Eventually the OD operation is applied to catch the true outliers to her interest.

Note that the above running example we gave here is just one of many combinative usages of our proposed operations. Those operations can be used individually or in other combinations to serve the ever changing outlier analysis demands.

3. ONION FRAMEWORK

To achieve the ONION model we designed the novel ONION framework. As shown in Fig. 4 ONION framework consists of two phases (a) offline multi-space abstraction construction and (b) online exploration operation processing using the corresponding ONION spaces.

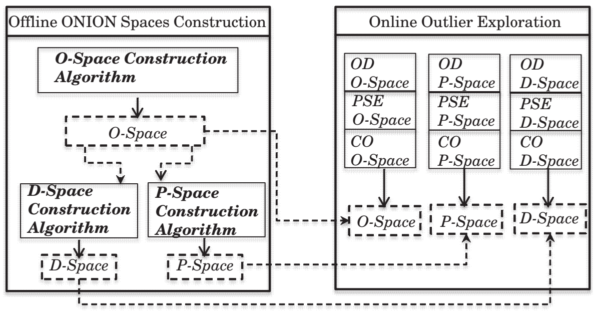
Offline ONION Spaces Construction
O-Space Construction Algorithm
O-Space
D-Space Construction Algorithm
P-Space Construction Algorithm
D-Space
P-Space
Online Outlier Exploration
OD O-Space
PSE O-Space
CO O-Space
OD P-Space
PSE P-Space
CO P-Space
OD D-Space
PSE D-Space
CO D-Space
O-Space
P-Space
D-Space

Figure 4: ONION Framework

3.1 O-Space

3.1.1 *Offline O-Space Construction*

As shown in Fig. 2, the three-dimensional O-Space can be decomposed into a set of two dimensional slices. Each slice corresponds to the outlier status of one point p_i in dataset D with respect to all parameter settings ps_i in the two-dimensional space $\mathcal{P}$ formed by the dimensions Dim_k and Dim_r. Therefore O-Space can be established by modeling the outlier status distribution in $\mathcal{P}$ for each point p_i in dataset *D*, called O-Space(p_i).

The key insight here is that given a point p_i in dataset *D*, it is not necessary to establish O-Space(p_i) by evaluating p_i for each possible ps_i in $\mathcal{P}$. In fact the outlier status of p_i with respect to any ps_i in $\mathcal{P}$ can be correctly determined by collecting only a small amount of meta information.

We first introduce our *k-distance* observation. Generally speaking given a set of outlier detection requests with the same parameter value k for Dim_k, but random values for Dim_r, the outlier status of any point p_i for any of those requests can be determined by checking the distance of p_i towards *one single point* in D. This observation is formally defined in Lemma 1.

Lemma 1. *Given a set of parameter settings $\mathcal{P}_k \subset \mathcal{P}$, where $\forall$ two parameter settings $ps_x(k_x, r_x)$, $ps_y(k_y, r_y) \in \mathcal{P}_k$, $k_x = k_y = k$, then the outlier status of p_i with respect to any ps_x in $\mathcal{P}_k$ is determined by the distance between p_i and its kth-nearest neighbor p_j denoted as $D^k_{p_i}$.*

Proof. Given any parameter setting $ps_x(k, r_x) \in \mathcal{P}_k$, if $D^k_{p_i} > r_x$, then by the definition of the kth-nearest neighbor, there are at most k-1 other points $p_j \in$ D whose distance towards p_i is not larger than r_x. In other words, p_i has at most k-1 neighbors. By Def. 1, p_i is an outlier. On the other hand, if $D^k_{p_i} \leq r_x$, then there are at least k points p_j with $d(p_i, p_j) \leq r_x$, namely p_j are all neighbors of p_i. p_i is then classified as an inlier by Def. 1. Therefore $\forall$ $ps_x(k, r_x)$ $\in \mathcal{P}_k$, the outlier status of p_i can be correctly determined by comparing r_x against $D^k_{p_i}$. Lemma 1 is proven. ■

Now we are ready to introduce the *space delimiter* insight as the foundation for building O-Space.

Lemma 2. *Given a dataset D and parameter setting space $\mathcal{P}$, $\forall$ $p_i \in D$ the distance set $\mathbb{DS}(p_i) = \{D^{k_x}_{p_i} | k_{min} \leq k_x \leq k_{max}\}$ is sufficient to determine the outlier status of p_i with respect to any parameter setting ps $\in \mathcal{P}$.*

Proof. $\mathcal{P} = \mathcal{P}_{k_{min}} \cup \mathcal{P}_{k_{min+1}} \cup \mathcal{P}_{k_{min+2}} ... \cup \mathcal{P}_{k_j} ... \cup \mathcal{P}_{k_{max-1}}$ $\cup$ $\mathcal{P}_{k_{max}}$, where $\mathcal{P}_{k_j}$ is composed by any $ps_x(k_x, r_x) \in \mathcal{P}$ with $k_x = k_j$ ($k_{min} \leq k_j \leq k_{max}$). Therefore given any $ps \in \mathbb{P}$ ps is guaranteed to be covered by some $\mathcal{P}_{k_j}$. By Lemma 1, $\forall ps \in \mathcal{P}_{k_j}$ the status of p_i can be determined by examining $D^{k_j}_{p_i}$. Since $D^{k_j}_{p_i}$ $\in \mathbb{DS}(p_i)$, therefore $\mathbb{DS}(p_i)$ is sufficient to determine the status of p_i with respect to any $ps \in \mathcal{P}$. Lemma 2 is proven.■

As shown in Fig. 5 this distance set $\mathbb{DS}(p_i)$ delimits $\mathcal{P}$ into two segments. The parameter settings in different segments will classify p_i to different outlier status. Therefore $\mathbb{DS}(p_i)$ is called *space delimiter* of p_i. The set of space delimiters $\{$ $\mathbb{DS}(p_i)$ $|p_i \in D\}$ effectively represents the three dimensional *O-Space*.

Furthermore the space delimiter structure also provides us an approach to quickly discover constant inliers and constant outliers.

Lemma 3. *A point p_i is a const inlier if $D^{k_{max}}_{p_i} \leq r_{min}$.*

Proof. If the distance to p_i's k_{max}th nearest neighbor is $\leq r_{min}$, then p_i has at least k_{max} neighbors or more even under most restricted neighbor criteria, namely Dim_r = r_{min}. Then p_i is an inlier for *ps*(k_{max},r_{min}) that is the most restricted parameter setting in $\mathcal{P}$ in terms of recognizing outlier. If p_i is not an outlier in the most restricted setting, then of course it cannot be outlier in any part of $\mathcal{P}$. Therefore p_i is a const inlier. ■

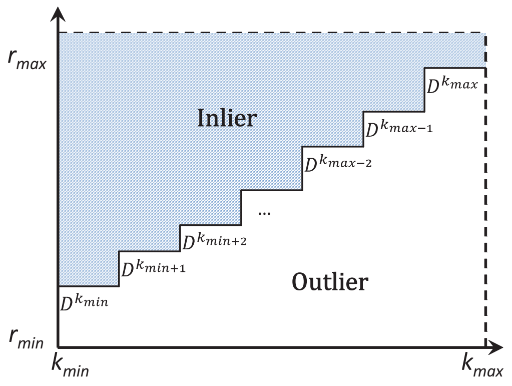

Figure 5: Space Delimiter

Lemma 4. *A point p_i is a const outlier if $D^{k_{min}}_{p_i} > r_{max}$.*

Lemma 4 can be proven in the similar way of proving Lemma 3. Due to space limitation, the proof is omitted.

Naturally any point that is not a const outlier nor a const inlier, is an *outlier candidate oc*. Among all points only for *oc* it is necessary to maintain its *space delimiter*.

Therefore constructing O-Space has two tasks, namely: (1) discovering all ocs and (2) collecting the space delimiter $\mathbb{DS}(oc)$ for each *oc*. Intuitively this can be done by first collecting $\mathbb{DS}(p_i)$ for each point p_i, then locating the constant inliers and outliers by applying Lemmas 3 and 4. Collecting $\mathbb{DS}(p_i)$ is straightforward. We can acquire the k nearest neighbors (kNN) of p_i by applying any kNN algorithm with k set as k_{max}. Since we only care for the range k_{min} to k_{max}, we then discard the k_{min-1} nearest neighbors.

However to discover const inliers it is not necessary to acquire the actual k_{max} nearest neighbors. Once p_i acquires k_{max} neighbors whose distance to p_i is not larger than r_{min}, p_i is guaranteed to be const inlier. Then the kNN search can be terminated immediately. Since const inliers are typically the majority of the dataset, this optimization significantly speeds up the preprocessing process.
Space Complexity. The O-Space data structure is composed of a set of arrays. Each of the arrays contains $(k_{max} - k_{min} + 1)$ float values (distance) corresponding to the space delimiter of one outlier candidate. Therefore the space complexity is linear in the number of outlier candidates $| \mathbb{OC} |$. More precisely it is $O(| \mathbb{OC} | (k_{max} - k_{min} + 1))$.

As confirmed by our experiments (Fig. 9, Sec. 4.3.1), only a small fraction of points is classified as outlier candidates. Most of the points are recognized as const inliers. Therefore *L* is much smaller than the actual cardinality *n* of the input dataset. Hence the O-Space structure is found to be rather compact and in fact small enough to be accommodated in the main memory of a standard PC even when handling a fairly large dataset in order of 10GB.
Time Complexity. The time complexity of constructing O-Space is $O(n^2)$ because of the potential KNN search on each point. Here *n* represents the cardinality of the input dataset D. Furthermore it is worth to emphasize that in fact the cost of building O-Space is similar to the cost of answering one single outlier detection request as confirmed in our experiments (Figures 7, 8, Sec. 4.2.1). In ONION, the expensive exact KNN search is only conducted on outlier candidates. This significantly speeds up the construction of O-Space.

3.1.2 Online Outlier Exploration

O-Space is sufficient to support all three classes of outlier exploration operations. In particular by intelligently maintaining the space delimiter information of each outlier candidate, we are able to drive down the time complexity of supporting online outlier detection (OD) operation from quadratic to linear.

Outlier Detection (OD). For each outlier candidate *oc* we maintain its space delimiter $\mathbb{DS}$ in an array structure by the order of its k_{min}th neighbor at the head and the k_{max}th neighbor at the end. Then for any parameter $ps \in \mathcal{P}$, the outlier status of *oc* can be immediately determined by applying the following *examination rule*.

Definition 10. *Given an outlier candidate oc and its $\mathbb{DS}$ structure, $\forall$ parameter setting $ps(k_x, r_x)$ in $\mathcal{P}$, oc is an outlier if $\mathbb{DS}[k_x - k_{min}] > r_x$. Otherwise p_i is an inlier.*

Therefore to answer OD we only need to perform one scan on the outlier candidate set $\mathbb{OC}$ and sequentially apply the examination rule in Def. 10 on each *oc*. Hence the time complexity is linear to the cardinality of $\mathbb{OC}$.

Outlier-Centric Parameter Space Exploration (PSE). Given an outlier set $\mathbb{O}_{in}$, the parameter settings that recognize $\mathbb{O}_{in}$ as outliers is the intersection of a set of parameter space segments $\mathbb{S}_i$ with respect to each point p_i in $\mathbb{O}_{in}$. All parameter settings in $\mathbb{S}_i$ with respect to p_i will classify p_i as outlier. By Lemma 2 this can be done by checking and comparing the space delimiters $\mathbb{DS}$ of all points in $\mathbb{O}_{in}$. The time complexity is O(| $\mathbb{O}_{in}$ |(k_{max} - k_{min})).

Comparative Outlier Analytics (CO). Similar to PSE, given an outlier set $\mathbb{O}_{in}$, CO can be answered by checking the parameter space segment $\mathbb{S}_i$ with respect to each outlier candidate oc_i in $\mathbb{OC}$ - $\mathbb{O}_{in}$. Point oc_i is dominated by all points o_j in $\mathbb{O}_{in}$ if $\mathbb{S}_i \supseteq \mathbb{S}_j$ for all oc_j. The time complexity is O(| $\mathbb{OC}$ |(k_{max} - k_{min})).

3.2 P-Space

3.2.1 Offline P-Space Construction

To construct the P-Space we first introduce the concept of ***k*-domination** between two outlier candidates.

Definition 11. *Given two outlier candidates oc_i and oc_j and a k value of $Dim_k \in [k_{min}, k_{max}]$, if $D^k_{oc_i} \leqslant D^k_{oc_j}$, then oc_i* ***k-dominates*** *oc_j .*

The following *monotonic property* holds if the k-domination relationship holds between oc_i and oc_j.

Lemma 5. *Given two outlier candidates oc_i and oc_j with oc_i k-dominating oc_j, then for any parameter setting $ps(k, r_x) \in \mathcal{P}$ ($r_{min} \leq r_x \leq r_{max}$), if oc_i is classified as outlier by ps, then oc_j is guaranteed to be outlier with respect to ps.*

Proof. If oc_i is an outlier with respect to $ps(k, r_x)$, $D^k_{oc_i} > r_x$. Since $D^k_{oc_j} \geq D^k_{oc_i}$ by the k-domination definition in Def. 11, $D^k_{oc_j} > r_x$. Therefore oc_j is an outlier with respect to *ps*. ■

In other words, if one parameter setting $ps(k, r_x)$ classifies p_i as an outlier, then any point k-dominated by p_i is guaranteed to also be an outlier. On the other hand, if one parameter setting classifies p_i as an inlier, then any point that k-dominates p_i is also guaranteed to be an inlier as well.

It is straightforward to prove that the k-domination relationship also satisfies the *transitive property*.

Lemma 6. *Given three candidates oc_h, oc_i, and oc_j, if oc_h k-dominates oc_i and oc_i k-dominates oc_j, then oc_h k-dominates oc_j.*

The above properties of the k-domination relationship now enable us to divide the infinite parameter setting space $\mathcal{P}$ into a finite number of *stable parameter regions*.

Lemma 7. *Given the outlier candidate set $\mathbb{OC} \subset$ dataset D and $\mathcal{P}_{k_i} \subset \mathcal{P}$, where | $\mathbb{OC}$ | = n and $\mathcal{P}_{k_i}$ is composed by any parameter setting ps in $\mathcal{P}$ sharing the same Dim_k value k_i, then $\mathcal{P}_{k_i}$ can be divided into n+1 stable regions $\mathbb{P}^j_{k_i}$, where Dim_r of $\mathbb{P}^1_{k_i} \in [r_{min}, D^{k_i}_{oc_1})$, Dim_r of $\mathbb{P}^2_{k_i} \in [D^{k_i}_{oc_1}, D^{k_i}_{oc_2})$, ..., Dim_r of $\mathbb{P}^{j+1}_{k_i} \in [D^{k_i}_{oc_j}, D^{k_i}_{oc_{j+1}})$,...., Dim_r of $\mathbb{P}^{n+1}_{k_i} \in [D^{k_i}_{oc_n}, r_{max}]$ ($D^{k_i}_{oc_1} < D^{k_i}_{oc_2}$, ..., $< D^{k_i}_{oc_j} < D^{k_i}_{oc_{j+1}}$, ..., $< D^{k_i}_{oc_n}$). The identical set of outliers are guaranteed to be generated for all $ps \in \mathbb{P}^j_{k_i}$.*

Proof. $\forall$ $ps(k_i, r_x) \in \mathbb{P}^j_{k_i}$, since $D^{k_i}_{oc_{j-1}} \leq r_x < D^{k_i}_{oc_j}$, $ps(k_j, r_x)$ will classify oc_{j-1} as inlier, while oc_j would be classified as outlier. Since $D^{k_i}_{oc_{j-2}} < D^{k_i}_{oc_{j-1}}$ and $D^{k_i}_{oc_j} < D^{k_i}_{oc_{j+1}}$, we get oc_{j-2} k-dominates oc_{j-1} and oc_j dominates oc_{j+1}. Based on the monotonic property of k-domination, oc_{j-2} will also be classified as an inlier, while oc_{j+1} remains as outlier. Furthermore by the transitive property of k-domination, $\forall$ $ps(k_i, r_x) \in \mathcal{P}^{j+1}_{k_i}$, $oc_1, oc_2, ..., oc_{j-2}$, oc_{j-1} are guaranteed to be inliers, while $oc_j, oc_{j+1}, ..., oc_n$ are guaranteed to be outliers. Therefore the identical set of outliers will be generated for any $ps \in \mathbb{P}^{j+1}_{k_i}$. Lemma 7 has thus been proven.■

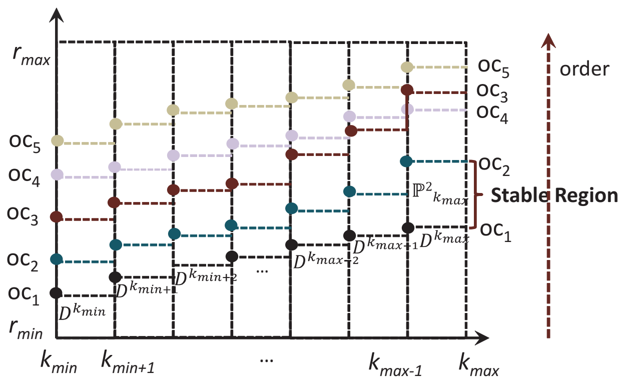

Figure 6: Stable Region

Leveraging Lemma 7 we design an light-weight algorithm (Alg. 1) to build P-Space $\mathbb{P}$. We first define the *parameter node* structure.

Definition 12. *A parameter node, or in short pn, is a data structure composed of the following three elements:*
-pn.obj: an outlier candidate oc in $\mathbb{OC}$;
-pn.k: k parameter value ($k \in [k_{min}, k_{max}]$);
-pn.r. r parameter value, pn.r = $D^{pn.k}_{oc}$;

By Def. 12, each outlier candidate *oc* in $\mathbb{OC}$ will be mapped to *m* nodes, where m = $k_{max} - k_{min} + 1$. Each of the nodes corresponds to one element in the space delimiter $\mathbb{DS}$ of *oc*, that is $D^{k_i}_{oc}$ ($k_{min} \leq k_i \leq k_{max}$). Then we organize the parameter nodes based on *pn.k* into m array lists. Each array list called *kthList* contains the nodes with the same *pn.k* value. Therefore each outlier candidate *oc* is represented by exactly one node *pn* in each *kthList*.

The key idea behind Alg. 1 is to sort the parameter nodes in each *kthList* in ascending order based on their *pn.r* values. By this each subspace $\mathcal{P}_{k_i}$ ($k_{min} \leq k_i \leq k_{max}$) of $\mathcal{P}$ is divided into multiple stable regions $\mathbb{P}^j_{k_i}$ by $D^{k_i}_{oc}$ in $\mathbb{DS}(oc)$ with respect to each *oc* in $\mathbb{OC}$. For example as shown in Fig. 6, $\mathbb{P}^2_{k_{max}}$ is a stable region of $\mathcal{P}_{k_{max}}$ bounded by $D^{k_{max}}_{oc_1}$ of oc_1 and $D^{k_{max}}_{oc_2}$ of oc_2 ($[D^{k_{max}}_{oc_1}, D^{k_{max}}_{oc_2})$). All parameter settings in $\mathbb{P}^2_{k_{max}}$ classify oc_2, ..., oc_5 as outliers.

P-Space then is represented by a hash map with pn.k as the *key* and the corresponding *kthList* as *value*.

Algorithm 1 constructPSpace

```
1: P-Space = $\emptyset$;
2: for each k from $k_{min}$ to $k_{max}$ do
3:     kthList = $\emptyset$;
4:     for each $oc \in ocs$ do
5:         $kthList$.add(new pn(oc, oc.$\mathbb{DS}$, k));
6:     end for
7:     $kthList.sort()$;
8:     P-Space.put(k, kthList);
9: end for
10: return P-Space;
```

3.2.2 Online Outlier Exploration

Outlier Detection (OD). Given a parameter setting $ps_i(k_i, r_i)$ to detect the outliers we only need to locate a particular parameter node pn_{min} in P-Space where $pn_{min}.k = k_i$ and $pn_{min}.r = min(\{pn.r \mid pn.r > r_i\})$. Then the outliers for ps_i will be the outlier candidates corresponding to the parameter nodes in the $k_i thList$ of P-Space and listed behind pn_{min}.

Complexity Analysis. Since each array list is sorted by the pn.r value, pn_{min} can be located in $O(log(\mid \mathbb{OC} \mid))$ time using a binary search style algorithm [8].

Outlier-Centric Parameter Space Exploration (PSE). Utilizing P-Space to support PSE operation is straightforward. We can traverse through each $kthList$ of P-Space to locate the stable regions that return the outlier set $\mathbb{O}_{in}$ specified in the input. Given one particular array list $k_i thList$, we first locate the parameter node pn_{1st} of oc_j corresponding to the first outlier in $\mathbb{O}_{in}$. Then we compare the outliers in $\mathbb{O}_{in}$ with the objects listed behind pn_{1st} in $k_i thList$ one by one. If all objects match, one stable region $\mathbb{P}^j_{k_i}$: $[D^{k_i}_{oc_{j-1}}, D^{k_i}_{oc_j})$ will be returned.

Complexity Analysis. The cost of supporting PSE relies on the number of $kthList$ and the outliers in O_{in}. Therefore the time complexity is $O(m \mid \mathbb{O}_{in} \mid)$, where m = $k_{max} - k_{min} + 1$.

Comparative Outlier Analytics (CO). Given an outlier set $\mathbb{O}_{in}$, CO operation can be answered by checking each outlier candidate oc_i in $\mathbb{OC}$ - $\mathbb{OC}_{in}$. oc_i is dominated by all points o_j in $\mathbb{O}_{in}$ if oc_i is listed behind all o_j in every $kthList$.

Complexity Analysis. The time complexity is $O(m \mid \mathbb{OC} \mid)$, where m = $k_{max} - k_{min} + 1$.

3.3 D-Space

3.3.1 Offline D-Space Construction

By Def. 5, to construct D-Space $\mathbb{D}$, we have to divide all outlier candidates *oc* into multiple domination groups $\mathbb{D}_i$. The domination relationship holds among all *oc*s falling in the same group $\mathbb{D}_i$.

Next we introduce the *domination rule* in Lemma 8 to evaluate whether the domination relationship holds between two outlier candidates based on their space delimiters in O-Space.

Lemma 8. *Given two outlier candidates oc_i and oc_j, oc_i dominates oc_j if $\forall$ $k_l \in [k_{min}, k_{max}]$, $D^{k_l}_{oc_i} \leq D^{k_l}_{oc_j}$*

Proof. By Def. 11, given one $k \in [k_{min}, k_{max}]$, if $D^k_{oc_i} \leq D^k_{oc_j}$, then oc_i k-dominates. By Lemma 5 given any parameter setting *ps* $\in$ parameter subspace $\mathcal{P}_k \subset \mathcal{P}$, oc_j is guaranteed to be outlier if oc_i is classified as outlier by *ps*. Since $D^{k_l}_{oc_i} \leq D^{k_l}_{oc_j}$ holds for any k_l $\in [k_{min}, k_{max}]$, then oc_i k_l-dominates oc_j for any k_l. Therefore if oc_i is classified as an outlier by any parameter setting *ps* in $\mathcal{P}$, then oc_j is guaranteed to be an outlier. By the definition of domination relation in Def. 4, Lemma 8 is proven. ■

As shown in Fig. 6, oc_1 dominates oc_2, because $D^k_{oc_1} < D^k_{oc_2}$ for any k $\in [k_{min}, k_{max}]$.

It is straightforward to prove that domination relationship satisfies the transitivity property.

Lemma 9. *Given three outlier candidates oc_h, oc_i, and oc_j, if oc_h dominates oc_i and oc_i dominates oc_j, then oc_h dominates oc_j.*

Next we propose a graph-based solution that successfully constructs D-Space. First we construct an undirected graph based on the domination relationships among all outlier candidates.

Definition 13. ***Domination Graph***. *The domination graph of the outlier candidate set $\mathbb{OC}$ is a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, such that (1) a node v_i exists in $\mathcal{V}$ to represent a point oc_i in $\mathbb{OC}$, and (2) an edge e_{ij} = (v_i, v_j) exists in $\mathcal{E}$ if domination relationship does not hold between oc_i and $oc_j \in \mathbb{OC}$ corresponding to nodes v_i and v_j in $\mathcal{V}$.*

This domination graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ tends to be a sparse graph, because the domination relationship tends to hold among most points in $\mathbb{OC}$. This is the case because the distance of oc_i towards its kNN usually does not dramatically change. If $D^{k_i}_{oc_i}$ is smaller than $D^{k_i}_{oc_j}$, then $D^{k_i+1}_{oc_i}$ also tends to be smaller than $D^{k_i+1}_{oc_j}$.

Given a domination graph $\mathcal{G}$, a completely disjointed graph with zero edge can always be derived by removing some nodes and the corresponding edges. This indicates by removing a small number of points corresponding to these nodes, we can get a subset of $\mathbb{OC}$ such that the domination relationship holds among all points in it. If we could determine the *minimal number of nodes* whose removal will completely isolate the remaining nodes, then we could build the largest domination group out of $\mathbb{OC}$. We now note that the problem of finding the minimal number of nodes to remove so that no edge remains in $\mathcal{G}$ can be mapped to the minimum vertex cover problem $-$ a classical *NP-complete* problem.

Clearly any minimum vertex cover algorithm can be applied here. Then D-Space can be built by recursively applying the minimum vertex cover algorithm on the removed nodes as shown in Alg. 2. The domination group built in each iteration is guaranteed to be the largest at that round. Therefore this process concurrently also minimizes the number of the domination groups. Since the domination graph tends to be a sparse graph, the number of the domination groups generated is small. As confirmed in our experiments, usually two or three trees are sufficient to cover all outlier candidates.

Algorithm 2 $construct\_DForest$

```
Input: $\mathbb{OC}$ // outlier candidates
Output: $domination\_forest$ // constructed domination forest;
1: if ($\mathbb{OC}$ == $\emptyset$) then
2:     return $\emptyset$;
3: else
4:     $domination\_tree$ = $\emptyset$;
5:     removed = minVertexCover($\mathbb{OC}$);
6:     $\mathbb{OC}$ = $\mathbb{OC}$ - removed;
7:     $domination\_tree$ = buildDtree($\mathbb{OC}$);
8:     return $domination\_forest$ + $domination\_tree$ +
       $Construct\_DForest$(removed);
9: end if
```

Given a domination group $\mathbb{D}_i$ a *domination tree tree*$_i$ can be constructed by sorting the outlier candidates in the ascending order based on the distance to their kth nearest neighbors, where k can be any element in $[k_{min}, k_{max}]$. In this domination tree, each *oc* will dominate the points listed behind it, while it in turn will be dominated by the points listed in front of it by the transitive property of the domination relationship. Therefore D-Space is represented by a *domination forest* composed of multiple *domination trees*.

Furthermore *domination forest* also incorporates the *stable region* concept of P-Space along its linkage to D-Space.

Lemma 10. *Given two adjacent points oc_i and oc_{i+1} in domination tree $tree_l$, any parameter setting $ps_x(k_x, r_x)$ with $k_{min} \leq k_x \leq k_{max}$ and $D^{k_x}_{oc_i} \leq r_x < D^{k_x}_{oc_{i+1}}$ will classify the same set of points oc_j in $tree_l$ as outliers, where $j > i$.*

Proof. Since $D^{k_x}_{oc_i} \leq r_x < D^{k_x}_{oc_{i+1}}$, by Lemma 1 ps_x will classify oc_{i+1} as outlier and oc_i as inlier. Since oc_{i+1} dominates oc_j, all oc_js are outliers. Any other point oc_h in $tree_l$ will be classified as inlier because oc_h dominates inlier oc_i. Lemma 10 is proven.■

As shown in Fig. 6, the parameter settings bounded by lines of oc_1 and oc_2 generate the same set of outliers: oc_2, ..., oc_5.

3.3.2 Online Outlier Exploration

The domination forest can efficiently support all classes of outlier exploration operations.

Comparative Outlier Analytics (CO). CO can be supported by locating the first point p_{1st} in each domination tree dominated by the weakest outlier o_i in the outlier input set $\mathbb{O}_{in}$ using a binary search style algorithm. Then all points listed behind p_{1st} in each of the domination trees are guaranteed to be outliers.

Outlier Detection (OD). Similar to CO, OD can be supported by applying the binary search style algorithm on each domination tree to locate the first outlier candidate classified as outlier by the input parameter setting ps_i.

The time complexity of processing CO and OD is $O(log \mid tree_1 \mid + log \mid tree_2 \mid + ... + log \mid tree_n \mid)$. It relies on the size of each tree and the number of the trees.

Outlier-Centric Parameter Space Exploration (PSE). By Lemma 10, the parameters that generate the same outliers $\mathbb{O}_{in}$ can be located by examining the strongest outlier oc in $\mathbb{O}^i_{in}$ and the first point in front of oc in each domination tree $tree_i$. Here $\mathbb{O}^i_{in} = \mathbb{O}_{in} \cap tree_i$. The intersection of the parameters returned from each tree will be the final result of PSE. The time complexity is $O(n + \mid \mathbb{O}_{in} \mid)$, where n is the number of the trees.

In summary the time complexity of the online phase relies on the size of each tree and the number of the trees. It is easy to see that the smaller the number of the trees is, the lower the costs will be.

As for the size of each tree, suppose two forests ft_1 and ft_2 composed of the same number of trees are derived from outlier candidate set $\mathbb{OC}$. For forest ft_1, $\mid ft_1.tree_1 \mid \gg \mid ft_1.tree_2 \mid ... \gg \mid ft_1.tree_n \mid$, while for forest ft_2, $\mid ft_2.tree_1 \mid \approx \mid ft_2.tree_2 \mid ... \approx \mid ft_2.tree_n \mid$. Then the cost of the binary search amounts to $binary(ft_1) < binary(ft_2)$. For example suppose $\mathbb{OC}$ contains 2^m points. ft_1 consists of two trees including the largest possible tree $\mid ft_1.tree_1 \mid = 2^m - 1$ and the smallest tree $\mid ft_1.tree_2 \mid = 1$, while $\mid ft_2.tree_2 \mid = \mid ft_2.tree_2 \mid = 2^{m-1}$. Then $binary(ft_1)$ = $\log(2^m - 1) + 1 <$ m +1, while $binary(ft_2) = 2\log(2^{m-1})$ = 2(m-1). Obviously when m is reasonably large, $binary(ft_1)$ is far smaller than $binary(ft_2)$. Therefore instead of making each tree equal size, the ideal forest construction algorithm should produce the largest possible trees out of $\mathbb{OC}$.

As shown in Sec. 3.3.1 our graph-based D-Space construction algorithm (Alg. 2) not only minimizes the number of trees created, but also maximizes the size of the trees in the forest. Therefore it effectively optimizes the performance of outlier exploration.

4. EXPERIMENTAL EVALUATION

Environment. All experiments ran on a Linux Server with 8 GB memory 2.6GHz Quad-Core CPU using Java 1.6.0 64bit runtime.

Real Datasets. We utilize the GMTI (Ground Moving Target Indicator) dataset [9] to conduct user study. GMTI contains around 10,000 records regarding the information of soldiers, vehicles, and helicopters deployed in a certain region. The outliers are detected based on targets' latitude and longitude. We use the outliers manually labeled by the experts familiar with the data as ground truth.

We also use the geolocation data from OpenStreetMap (http://download.geofabrik.de/) to evaluate the performance of ONION when handling large dataset. It contains the geolocation information of 50 million buildings (10G) over Australia and Oceania, such as houses, cafes, stations, etc.. A location on the map is considered to be outlier based on their distances to other locations.

Methodology. We evaluate the processing time and scalability of both our offline preprocessing and online mining algorithms by varying the sizes of the dataset D, parameter space $\mathcal{P}$, and the number of mining requests. We compare against the state-of-the-art DOLPHIN [1] in a rich variety of representative use cases.

In particular at the **offline** phase, we evaluate the processing time of constructing O-Space that builds the foundation of ONION in comparison to the index construction cost of DOLPHIN. At the **online** phase, the performance of our online algorithms associated with O-Space, P-Space, and D-Space respectively is evaluated and contrasted for all three outlier exploration types, namely outlier detection (OD), outlier-centric parameter space exploration (PSE), and comparative outlier analytics (CO). The algorithms associated with each ONION abstraction are named in the format of "operation type" + "_" + "Abstraction type". For example the algorithm supporting OD operation on O-Space is named as "OD_OSpace". Furthermore we also compare our ONION against DOLPHIN on the processing time of traditional outlier detection query − the only exploration type that Dolphin supports.

4.1 User Study

We conduct a user study to evaluate the effectiveness of ONION in recognizing outliers contrasting against the traditional one-at-a-time query approach (TRAD) that only supports outlier detection operation. Since TRAD takes hours to process a large dataset (10G) as confirmed in Sec. 4.3.1, it is not acceptable for interactive analytics. Therefore in this study we adopt the relative small dataset (GMTI) − a clear bias to TRAD.

We invited 50 users from both WPI and Yantai University, China. The users are divided into two groups. Each group only evaluates one system. Each user is allowed to continuously submit mining requests supported by the target system until the generated results meet the precision and recall requirement (0.9,0.9) set by us. In each round the precision and recall are automatically calculated and feedbacked to the users. In any case the study will terminate after 15 minutes. Users are provided a distribution plot of GMTI dataset that assists them to initialize the parameter setting. For each user, we count the number of trials (the submitted mining requests) on each exploration operation. Then the trial number is averaged on the users belonging to the same group.

System	Success Rate	Overall	OD	CO	PSE
ONION	1	5.6	1.8	2.6	1.2
TRAD	0.36	16.2	16.2	−	−

Table 1: Statistics

As shown in Table 1, only 36% of the TRAD users are able to eventually meet the precision and recall requirement in 15 minutes, while all users using ONION succeed. In average TRAD takes users 16.2 trials to meet the requirement, while the ONION users only need 5.6. In particular in average the ONION users submit

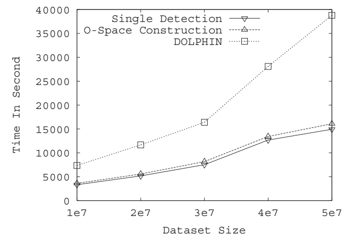

Figure 7: O-Space Construction: Varying Data Size

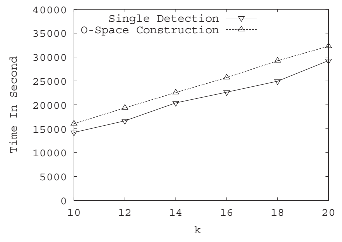

Figure 8: O-Space Construction: Varying *k*

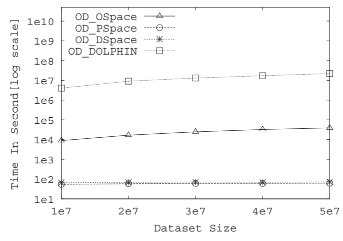

Figure 9: OD: Varying Dataset Size

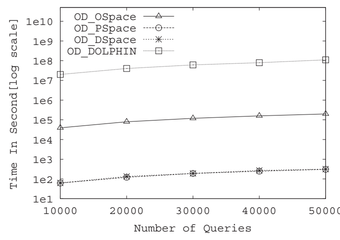

Figure 10: OD: Varying Number Of Requests

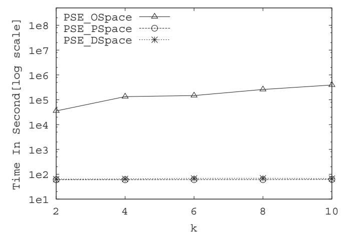

Figure 11: PSE: Varying parameter space size

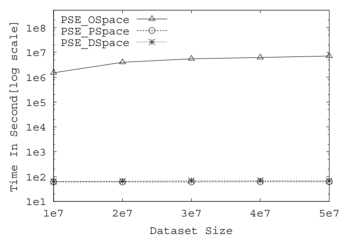

Figure 12: PSE: Varying Dataset Size

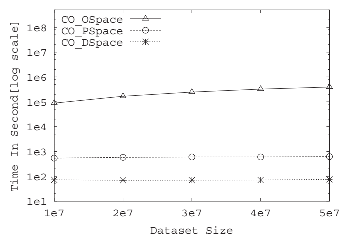

Figure 13: CO: Varying Dataset Size

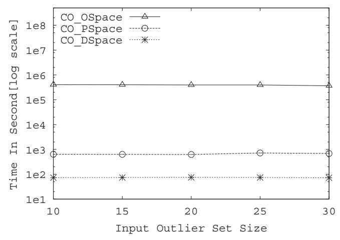

Figure 14: CO: Varying input outlier set size

CO operation 1.8 times, PSE operation 2.6 times, and the traditional outlier detection (OD) 1.2 times. This confirms that our new outlier exploration operations indeed save users significantly effort on pinpointing appropriate parameter settings.

4.2 Offline Preprocessing

4.2.1 O-Space Construction

We first focus on the processing time of constructing O-Space (construct_OSpace) from raw data by varying the parameter space size, as well as the dataset size. The costs of one time outlier detection without employing any index is used as the baseline to evaluate the extra overhead introduced by constructing O-Space.

Varying dataset size. Fig. 7 illustrates the results when dataset size increases from 10 million up to 50 million. We vary the dataset size by including more and more buildings belonging to different regions in Australia. The parameter space is fixed with k_{max} as 10 and r_{max} as 4000. Clearly constructing O-Space has ignorable overhead compared to the cost of one time outlier detection when parameter setting *ps* specified as (k_{max},r_{min}). Both our O-Space construction process and one time detection process need to detect up to k_{max} neighbors within r_{min} radius for each point p_i. The additional overhead of O-Space construction is introduced by having to track and maintain all possible outlier candidates with respect to the entire parameter space. However as shown in Fig. 7 such overhead is small (around 10%). Furthermore constructing O-Space is significantly faster than constructing DOLPHIN index.

Varying parameter space $\mathcal{P}$. Dolphin is excluded from this case because it does not have the *parameter space* concept. The influence of varying range of k is evaluated. Fig. 8 represents the results when varying k_{max} from 10 to 20, while holding r_{max} at 4000m and dataset size at 50 million. The overhead is still around 10% for the same reason explained above. As k_{max} increases, the cost of O-Space construction grows in the trend similar to one time outlier detection. Varying the range of r shows the similar influence. Due to space constraint, the results are not included.

4.3 Online Outlier Exploration

4.3.1 Online Outlier Detection

We evaluating the processing time of online outlier detection by varying the size of the datasets and the number of the requests.

Varying dataset size. Fig. 9 shows the advantage of ONION for outlier detection. We ran 10,000 requests with randomly chosen parameter settings from the entire parameter space and show the total processing time. P-Space and D-Space methods show very similar performance. Therefore their lines in Fig. 9 are overlapped. In average each request can be processed in milliseconds. Both consistently outperform DOLPHIN 5 orders of magnitude. Furthermore for D-Space and P-Space the detection cost grows only logarithmically in the size of outlier candidates that are the strict minority of the whole dataset (fewer than 10%), while DOLPHIN grows linearly. Therefore, ONION scales to large dataset.

Varying number of request. We increase the number of OD requests from 10,000 up to 50,000, while holding dataset size constant at 50 million. The total detection time is measured. Fig. 10 shows that our algorithms scale linearly in the number of requests. Again P-Space and D-Space algorithms are at least 5 orders of magnitude faster than DOLPHIN. Even our linear complexity O-Space method is 3 order of magnitude faster than DOLPHIN in average.

4.3.2 Outlier-Centric Parameter Space Exploration

We evaluate the performance of processing PSE request not supported by DOLPHIN. Each chart shows the accumulated processing time for 10,000 requests.

Varying parameter space size. Fig. 11 measures the influence to the processing time of PSE when varying the size of the parameter space. This is achieved by increasing k_{max} from 2 to 10. For P-Space and D-Space the cost of supporting PSE relies on the number of domination trees and the parameter node lists. Therefore, the cost of P-Space and D-Space is not sensitive to the change of k_{max}. On the other hand O-Space method has to check all outlier candidates. Since the number of outlier candidates grows as k_{max} increases, the cost of O-Space method will also increase lineally.

Varying dataset size: outlier set as input. Fig. 12 demonstrates the performance of our PSE algorithms. That is, given a PSE request, we use a set of randomly selected outlier candidates as input. The size of the datasets is varied from 10 million up to 50 million. Similar to the experiment that uses parameter settings as input, P-Space and D-Space methods significantly outperform O-Space method 3 orders of magnitude.

4.3.3 Comparative Outlier Analytics

Next we evaluate the performance of supporting CO operation.

Varying dataset size. Fig. 13 illustrates the processing time of supporting CO operation by varying dataset sizes. We use a randomly selected outlier set as input. The operation returns all outlier candidates that are dominated by the input outliers. D-Space supports CO operation by only looking at each domination tree in the domination forest once, while the number of the domination tree is small (at most 3 when the dataset contains all 50 millions buildings). On the other hand, O-Space method has to scan all candidates, while P-Space method has to search the parameter node lists for every possible k value. Therefore D-Space method is about 1 order of magnitude faster than P-Space method, and about 3 to 4 orders of magnitude faster than O-Space method.

Varying size of input outlier set. In Fig. 14 we vary the size of the input outlier set from 10 to 30, while keeping the sizes of dataset and parameter space stable. For each method we only need to check the weakest outlier of the input outlier set. Since the cost of determining the weakest outlier is negligible, all our three methods are not sensitive to the size of the input outlier set.

5. RELATED WORK

Outlier Detection. Outlier detection has been the focus of much research in the statistics literature for over a century [12, 3]. The most common approach is to assume that all points follow a distribution with known distribution parameters (e.g., mean and variance). The points that do not properly fit the model are considered to be outliers. However, such approaches suffer from the serious limitation that the data distribution and underlying parameters must either be explicitly known apriori or be easily inferred.

Approaches that do not rely on data distributions have also been proposed. In [10, 15, 18] all points that are not a core part of any cluster are classified as outliers. In other words the outliers are in this case the by-products of data clustering. However we note here that a point that is not a member of any cluster is not necessarily abnormal. This is so because the goal of clustering is to group together points that are extremely similar to one another. Therefore such approaches lack strong notion of what constitutes an outlier.

To address this limitation, the notion of an outlier based on density (of neighborhood) or based on distance (of neighbors) has been defined. Density-based approaches [6, 17] assign an outlier score to any given point by measuring the density relative to its local neighborhood restricted by a pre-defined threshold. Therefore density-based outliers, regarded as "local outliers", are able to identify outliers often missed by other methods. However it has been observed that such methods do not scale well to large datasets [14].

Furthermore explicit distance-based approaches, based on the well known nearest-neighbor principle, were first proposed by Ng and Knorr [13]. They employ a well-defined distance metric to detect outliers, that is, the greater is the distance of the point to its neighbors, the more likely it is an outlier. The basic algorithm for such distance-based definition, the nested loop (NL) algorithm, calculates the distance between each pair of points and then set as outliers those that are far from most points. The NL algorithm has quadratic complexity with respect to the number of points. Thus it is not suitable for truly large datasets.

As a result, extensive effort has been focusing on identifying practical sub-quadratic algorithms [1, 4, 11, 5]. Several optimization principles have been proposed such as the use of compact data structures [11], of lightweight outlier detection oriented indices [1], and of pruning and randomization [4]. In particular by indexing the possible neighbors of each point p_i in dataset D based on their distances to p_i, [1] is able to approximate whether p_i is an outlier in the time complexity near linear to the cardinality of D. However, while these methods offer improved performance compared to statistical or clustering based approaches, they still suffer from unacceptable response times such that hours or even days for online queries. Furthermore none of these works tackles the important and hard problem of choosing proper parameter setting from the infinite number of possible options. Our work not only successfully satisfies the real time responsiveness requirement, but also saves users the significant effort otherwise spent on parameter tuning.

Parameter Space Exploration in Clustering. In [2] the *OPTICS* algorithm creates an augmented ordering of the dataset to represent the clustering structure corresponding to a set of parameter settings. However, the producing of outliers as by-products of clustering has already been shown to be not effective in capturing abnormal phenomena [16]. Furthermore the ordering information is only effective in representing the clusterings with respect to a small range of parameter settings, that is the parameters with only the neighbor range threshold variable. Our work instead supports a full range of possible parameter settings composed of both range and neighbor count thresholds.

6. CONCLUSION

Interactive outlier exploration over large dataset is an extremely important yet difficult task. Our novel ONION framework achieves this by bridging the data space and parameter space. By extracting the outlier candidates along with their interrelationships and abstracting them into successive more powerful structures, ONION is able to effectively discover outliers with real time responsiveness.

7. REFERENCES

[1] F. Angiulli and F. Fassetti. Dolphin: An efficient algorithm for mining distance-based outliers in very large datasets. *TKDD*, 3(1), 2009.

[2] M. Ankerst, M. M. Breunig, H. Kriegel, and J. Sander. OPTICS: ordering points to identify the clustering structure. In *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA.*, pages 49–60, 1999.

[3] V. Barnet and T. Lewis. Outliers in statistical data. *International Journal of Forecasting*, 12(1):175–176, 1996.

[4] S. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *KDD*, pages 29–38, 2003.

[5] K. Bhaduri, B. L. Matthews, and C. Giannella. Algorithms for speeding up distance-based outlier detection. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 859–867, 2011.

[6] M. M. Breunig and et al. Lof: Identifying density-based local outliers. In *SIGMOD Conference*, pages 93–104, 2000.

[7] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection. *ACM Computing Surveys*, 41(3):1–58, 2009.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.

[9] Entzminger and et al. Jointstars and gmti: past, present and future. *Aerospace and Electronic Systems, IEEE Transactions*, 35(2):748 –761, Apr. 1999.

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

[11] A. Ghoting, S. Parthasarathy, and M. E. Otey. Fast mining of distance-based outliers in high-dimensional datasets. *Data Min. Knowl. Discov.*, 16(3):349–364, 2008.

[12] D. M. Hawkins. *Identification of Outliers*. Springer, 1980.

[13] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *VLDB*, pages 392–403, 1998.

[14] H.-P. Kriegel, P. Kröger, and A. Zimek. Outlier detection techniques. In *In Tutorial of the 13th PAKDD*, 2009.

[15] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *VLDB*, pages 144–155, 1994.

[16] G. H. Orair, C. H. C. Teixeira, Y. Wang, W. M. Jr., and S. Parthasarathy. Distance-based outlier detection: Consolidation and renewed bearing. *PVLDB*, 3(2):1469–1480, 2010.

[17] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. LOCI: fast outlier detection using the local correlation integral. In *Proceedings of the 19th International Conference on Data Engineering, March 5-8, 2003, Bangalore, India*, pages 315–326, 2003.

[18] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *SIGMOD*, pages 103–114, 1996.