A APPENDIX

A.1 Single File Query Algorithm

Now, we present the pseudo-code of querying outliers in a file via sliding window in Algorithm 2. Following the convention of query processing in data stream [18], we also consider the current sliding window W, with a number of points expired and some others new compared to the previous window. Thereby, Lines 7 and 9 update the window statistics for each bucket $B[u]$, referring to the aggregation property in Proposition 3.3 in Section 3.2.

The pruning of points in bucket $B[u]$ is then applied, i.e., cases 1 and 2 in Section 3.4. Specifically, we prune the entire bucket $B[u]$ as inliers in Line 10, according to the lower bound in Proposition 3.7 in Section 3.3.2. Likewise, we directly output the points in bucket $B[u]$ as outliers in Lines 12 and 14, referring to Propositions 3.5 and 3.6, in Section 3.3.1.

Otherwise, we need to load at most 4 additional buckets to determine the r-neighbors of points p in bucket $B[u]$, i.e., case 3 in Section 3.4. Lines 19 and 22 aggregate the r-neighbor count according to Proposition 3.9, to determine the outlier.

Example A.1 (Example 3.10 continued). Figure 13 shows the next sliding window of Figure 4, where a segment S_1 is expired and a new segment S_3 comes. The corresponding bucket statistics are updated for the new window $W_{11:00:10}$ starting from time 11:00:10.

Algorithm 2 Outlier Detection Algorithm on Single File

Input: A window W_t at time t with size w and slide s, neighbor distance threshold r and count threshold k

Output: outlier set O_t of current window W_t

1: $\lambda := \lceil r/\gamma \rceil$
2: $\ell := \lfloor r/\gamma \rfloor$
3: initialize outlier set $O_t := \emptyset$
4: initialize buckets $\mathcal{B}$ if algorithm runs for the first window
5: **for** each bucket $B[u]$, $u := 1$ to β **do**
6:   **for** each expired segment S_g **do**
7:     $|B[u]| := |B[u]| - |B_g[u]|$
8:   **for** each new segment S_g **do**
9:     $|B[u]| := |B[u]| + |B_g[u]|$
10:   **if** $\sum_{i=u-\ell+1}^{u+\ell-1} |B[i]| \geq k$ **then**
11:     continue
12:   **else if** $\lceil r/\gamma \rceil - r/\gamma < \frac{1}{2}$ **and** $\sum_{i=u-\lambda}^{u+\lambda} |B[i]| < k$ **then**
13:     $O_t := O_t \cup B[u]$
14:   **else if** $\lceil r/\gamma \rceil - r/\gamma \geq \frac{1}{2}$ **and** $\max\left(\sum_{i=u-\lambda}^{u+\lambda-1} |B[i]|, \sum_{i=u-\lambda+1}^{u+\lambda} |B[i]|\right) < k$ **then**
15:     $O_t := O_t \cup B[u]$
16:   **else**
17:     load additional buckets $B[u \pm \lambda]$, $B[u \pm \ell]$
18:     **for** each point p in $B[u]$ **do**
19:       $cnt := \sum_{i=u-\ell+1}^{u+\ell-1} |B[i]|$
20:       **for** each point p' in $B[u \pm \lambda] \cup B[u \pm \ell]$ **do**
21:         **if** $dist(p, p') \leq r$ **then**
22:           $cnt := cnt + 1$
23:       **if** $cnt < k$ **then**
24:         $O_t := O_t \cup \{p\}$
25: **return** O_t

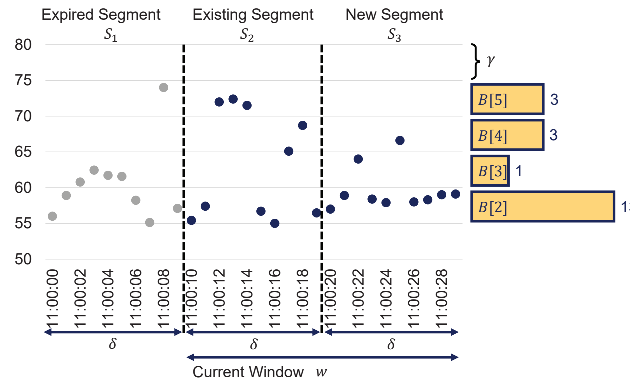

Figure 13: Query in sliding window with bucket statistics

For instance, we have $|B[2]| = |B[2]| - |B_1[2]| + |B_3[2]| = 10 - 5 + 8 = 13$, where $|B_1[2]|$ and $|B_3[2]|$ are the 2nd bucket sizes in segments S_1 and S_3, respectively. Once all the bucket statistics are updated, similar to the previous window in Example 3.10, it checks whether the pruning is applicable. If not, i.e., case 3 in Section 3.4, the algorithm loads the additional buckets for evaluating r-neighbors.

Complexity Analysis. We have β buckets as introduced in Definition 3.1. Consider all the ω segments in a window as in Definition 3.3. There are at most ω segments expired and ω segments new in the sliding window. The update of bucket statistics in Lines 7 and 9 thus takes $O(\beta\omega)$ time. In the worst case, all the n points in the window may be output as outliers, referring to the upper bounds in Lines 12 and 14, in $O(n)$ time. Otherwise, we need to load point p and check its r-neighbors. Luckily, we only need to load a constant number (up to 4) of additional buckets in Line 17. Referring to the average number of points in a bucket $\frac{n}{\beta}$, it takes $O(\frac{n^2}{\beta})$ time. To sum up, Algorithm 2 runs in $O(\beta\omega + \frac{n^2}{\beta})$ time, and $O(\beta\zeta)$ extra space, where β is the number of buckets, ω is the number of segments in a window, n is the number of points in a window and ζ is the number of segments in a file.

A.2 Evaluation on Supporting Various Queries

Figures 14, 15, 16, 17 presents the evaluation on 6 datasets under different (1) neighbor distance threshold r, (2) neighbor count threshold k, (3) window size w, and (4) slide size s, respectively.

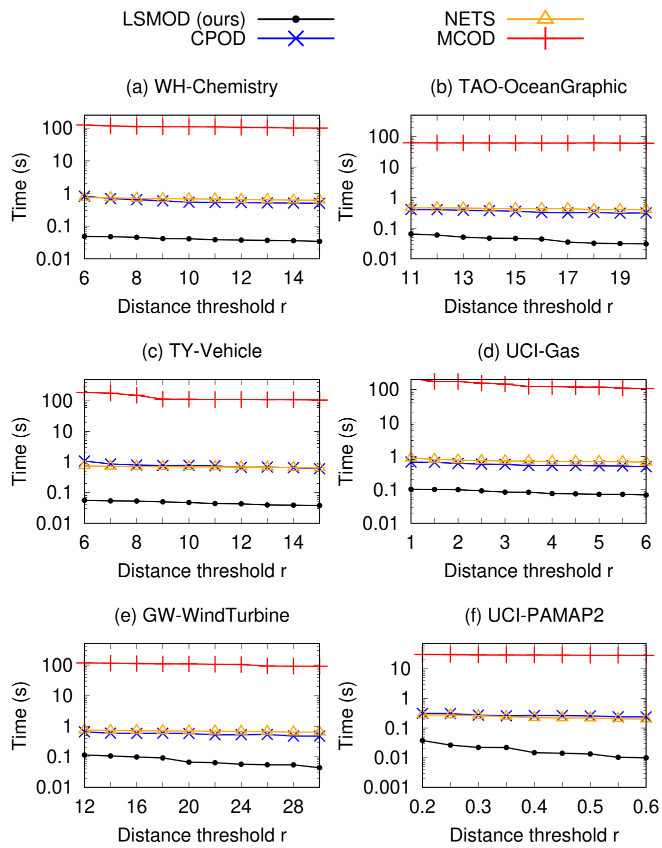

Figure 14: Varying neighbor distance threshold r of query

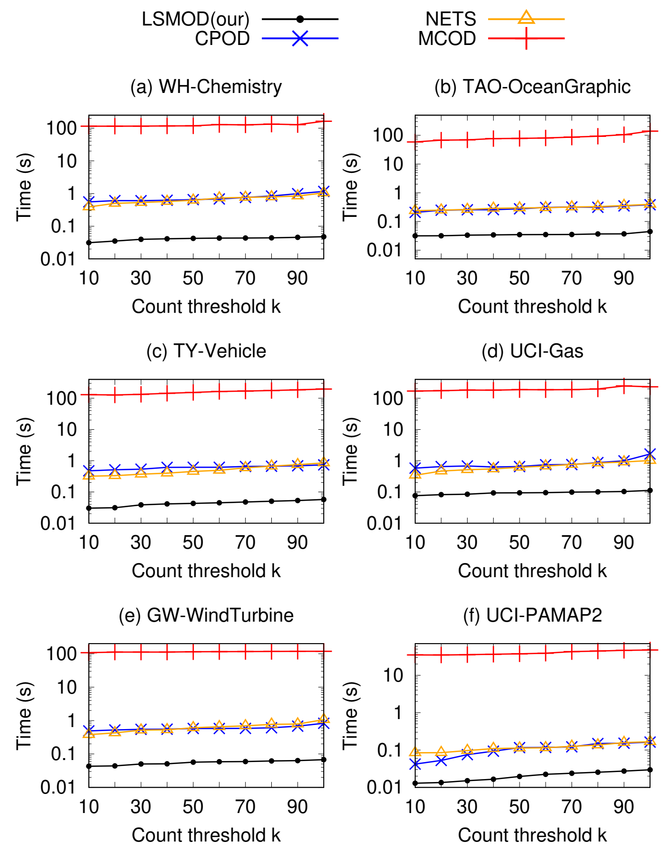

Figure 15: Varying neighbor count threshold k of query

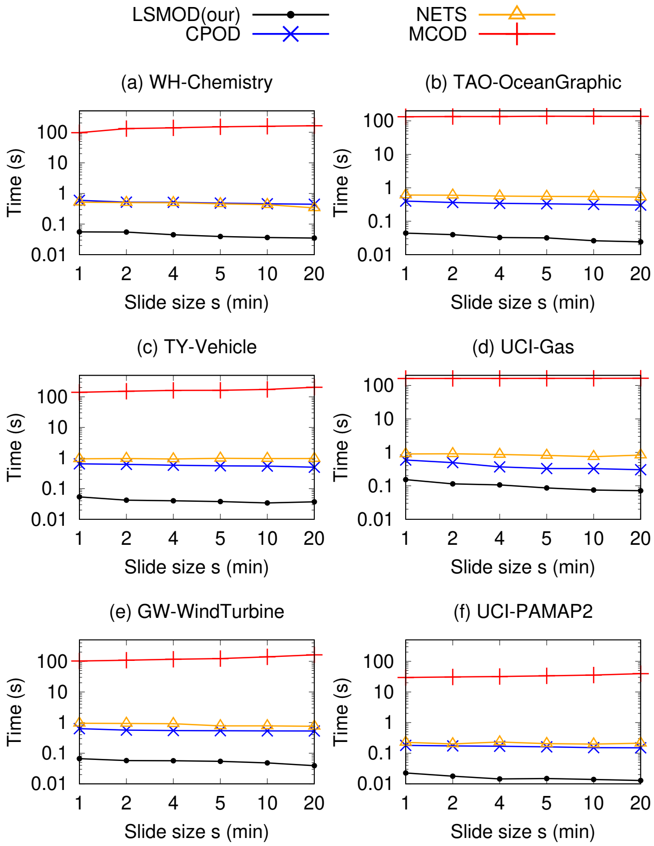

Figure 17: Varying window slide s of query

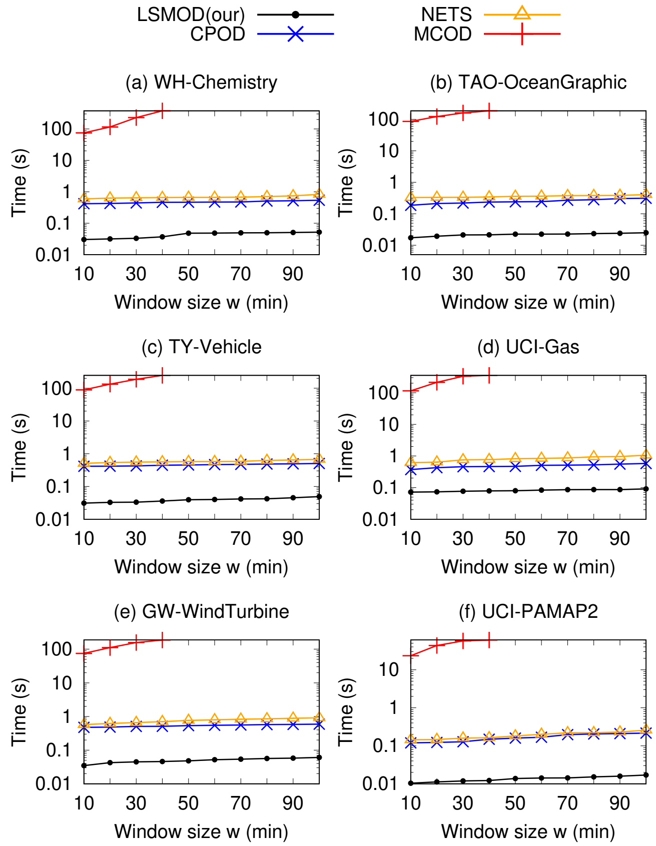

Figure 16: Varying window size w of query