Appendix

A. Proof of Proposition 3

Proof. For a point p in bucket $B[u]$, its r-neighbors must locate in $[p.v-r, p.v+r]$. Referring to Definition 8, the points in bucket $B[u]$ must fall into $[v_{\min}+(u-1)\gamma, v_{\min}+u\gamma)$, i.e., $v_{\min}+(u-1)\gamma \leq p.v < v_{\min}+u\gamma$.

Since $(\lambda-\frac{1}{2})\gamma < r \leq \lambda\gamma$, the following inequality can be derived,

$$\begin{aligned} v_{\min}+(u-\lambda-1)\gamma \leq p.v-r < v_{\min}+(u-\lambda+\frac{1}{2})\gamma, \\ v_{\min}+(u+\lambda-\frac{3}{2})\gamma < p.v+r < v_{\min}+(u+\lambda)\gamma. \end{aligned}$$

That is,

$$[p.v-r, p.v+r] \subseteq [v_{\min}+(u-\lambda-1)\gamma, v_{\min}+(u+\lambda)\gamma),$$

which corresponds to the union of the ranges of buckets $B[u-\lambda], B[u-\lambda+1], \ldots, B[u+\lambda]$, referring to Definition 8. Thus we derive the upper bound $|N(p,r)| \leq \sum_{i=u-\lambda}^{u+\lambda} |B[i]|$. □

B. Proof of Proposition 4

Proof. Similar to the proof of Proposition 3, we also have $v_{\min}+(u-1)\gamma \leq p.v < v_{\min}+u\gamma$. Since $(\lambda-1)\gamma < r \leq (\lambda-\frac{1}{2})\gamma$, we have,

$$v_{\min}+(u-\lambda-\frac{1}{2})\gamma \leq p.v-r < v_{\min}+(u-\lambda+1)\gamma.$$

Intuitively, with the constraint $(2\lambda-2)\gamma < 2r \leq (2\lambda-1)\gamma$, $N(p,r)$ at most overlaps with 2λ consecutive buckets.

Thus, for the case with $v_{\min}+(u-\lambda-\frac{1}{2})\gamma \leq p.v-r < v_{\min}+(u-\lambda)\gamma$, the r-neighbor of p overlaps with 2λ buckets $B[u-\lambda], B[u-\lambda+1]\ldots, B[u+\lambda-1]$, leading to

$$|N(p,r)| \leq \sum_{i=u-\lambda}^{u+\lambda-1} |B[i]|.$$

Otherwise, for the case with $v_{\min}+(u-\lambda)\gamma \leq p.v-r < v_{\min}+(u-\lambda+1)\gamma$, the r-neighbor of p overlaps with 2λ buckets $B[u-\lambda+1], B[u-\lambda+2]\ldots, B[u+\lambda]$, leading to

$$|N(p,r)| \leq \sum_{i=u-\lambda+1}^{u+\lambda} |B[i]|.$$

Combining the above cases, the upper bound can be obtained as the maximum, i.e.,

$$|N(p,r)| \leq \max\left(\sum_{i=u-\lambda}^{u+\lambda-1} |B[i]|, \sum_{i=u-\lambda+1}^{u+\lambda} |B[i]|\right).$$

□

C. Proof of Proposition 5

Proof. For a point p in bucket $B[u]$, its r-neighbors must locate in $[p.v-r, p.v+r]$. Referring to Definition 8, the points in bucket $B[u]$ must fall into $[v_{\min}+(u-1)\gamma, v_{\min}+u\gamma)$, and we thus have $v_{\min}+(u-1)\gamma \leq p.v < v_{\min}+u\gamma$. Since $\ell\gamma \leq r < (\ell+1)\gamma$, we can derive

$$\begin{aligned} v_{\min}+(u-\ell-2)\gamma < p.v-r < v_{\min}+(u-\ell)\gamma, \\ v_{\min}+(u+\ell-1)\gamma \leq p.v+r < v_{\min}+(u+\ell+1)\gamma. \end{aligned}$$

That is,

$$[p.v-r, p.v+r] \supseteq [v_{\min}+(u-\ell)\gamma, v_{\min}+(u+\ell-1)\gamma),$$

which leads to the lower bound $|N(p,r)| \geq \sum_{i=u-\ell+1}^{u+\ell-1} |B[i]|$. □

D. Proof of Proposition 6

Proof. Referring to the proofs of Propositions 3 and 5, for a point p in bucket $B[u]$, we have

$$\begin{aligned} [v_{\min}+(u-\ell)\gamma, v_{\min}+(u+\ell-1)\gamma) \subseteq [p.v-r, p.v+r] \\ \subseteq [v_{\min}+(u-\lambda-1)\gamma, v_{\min}+(u+\lambda)\gamma), \end{aligned}$$

Combining with Definitions 1 and 8, we can further derive

$$\bigcup_{i=u-\ell+1}^{u+\ell-1} B[i] \subseteq N(p,r) \subseteq \bigcup_{i=u-\lambda}^{u+\lambda} B[i].$$

With the lower bound obtained, we only need to further check the points contained in $\bigcup_{i=u-\lambda}^{u+\lambda} B[i]$ but not contained in $\bigcup_{i=u-\ell+1}^{u+\ell-1} B[i]$, that is $\bigcup_{i=u-\lambda}^{u-\ell} B[i] \cup \bigcup_{i=u+\ell}^{u+\lambda} B[i]$. In particular, since

$$\ell = \lfloor r/\gamma \rfloor \leq \lceil r/\gamma \rceil = \lambda \leq \lfloor r/\gamma \rfloor + 1 = \ell + 1,$$

we have $\lambda = \ell$ or $\ell+1$. For $\lambda = \ell+1$, only the points in 4 additional buckets $B[u-\lambda], B[u-\ell], B[u+\ell]$ and $B[u+\lambda]$ need to be checked. For $\lambda = \ell$, only the points in 2 additional buckets $B[u-\lambda], B[u+\lambda]$ need to be checked. □

E. Proof of Proposition 7

Proof. If all the points in $F_1, \ldots, F_\rho$ do not overwrite each other, then the bucket sizes can be simply aggregated when merging, i.e., the maximum size $\sum_{h=1}^{\rho} |B^{(h)}[u]|$. On the other hand, if all the points in $F_1, \ldots, F_{\rho-1}$ are overwritten by points in F_ρ, then the bucket size $|B[u]|$ should be the same as $|B^{(\rho)}[u]|$, the minimum size. □

F. Proof of Propositions 8, 9, 10

Proof. Propositions 8, 9, 10 can be easily derived by combining Propositions 3 and 7, Propositions 4 and 7, Propositions 5 and 7, respectively. □

G. 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 [7], 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 2 in Section IV-B.

The pruning of points in bucket $B[u]$ is then applied, i.e., cases 1 and 2 in Section IV-D. Specifically, we prune the entire bucket $B[u]$ as inliers in Line 10, according to the lower bound in Proposition 5 in Section IV-C2. Likewise, we directly output the points in bucket $B[u]$ as outliers in Lines 12 and 14, referring to Propositions 3 and 4, in Section IV-C1.

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 IV-D. Lines 19 and 22 aggregate the r-neighbor count according to Proposition 6, to determine the outlier.

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 $\mathscr{B}$ if algorithm runs for the first window

 5: for each bucket $B[u]$, $u := 1$ to $\beta$ 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 $O_t$
```

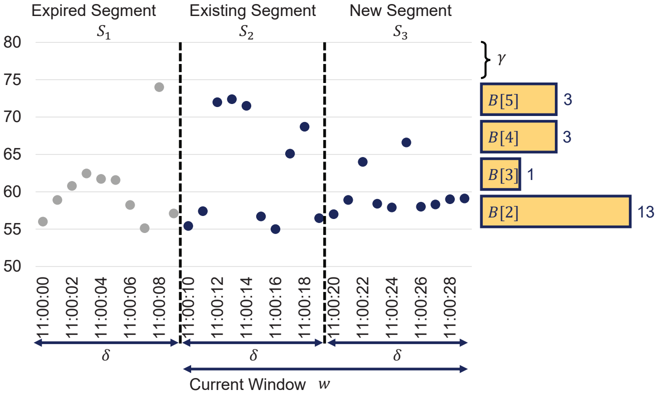

Fig. 15. Query in sliding window with bucket statistics

Example 9 (Example 7 continued)**.** *Figure 15 shows the next sliding window of Figure 6, 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. 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 7, it checks whether the pruning is applicable. If not, i.e., case 3 in Section IV-D, the algorithm loads the additional buckets for evaluating r-neighbors.*

Complexity Analysis: We have β buckets as introduced in Definition 8. Consider all the ω segments in a window as in Definition 2. 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 $\mathscr{O}(\beta\omega)$ 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 $\mathscr{O}(n)$ 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 $\mathscr{O}(\frac{n^2}{\beta})$ time. To sum up, Algorithm 2 runs in $\mathscr{O}(\beta\omega + \frac{n^2}{\beta})$ time, and $\mathscr{O}(\beta\zeta)$ 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.

H. Supporting Various Queries

As introduced in Section I, outliers are often queried with different parameters for various interests. Same as the existing methods, our LSMOD also supports queries with different (1) neighbor distance threshold r, (2) neighbor count threshold k, (3) window size w, and (4) slide size s. We evaluate our proposal and other baselines under different query parameters, and present the results on the dataset WH-Chemistry in Figure 16. Owing to the limited space, similar results on other datasets can be found in the supplementary [20].

When neighbor distance threshold r increases in Figure 16(a), each data point has more neighbors in a window, and more inliers can be directly pruned by the index structures of NETS, CPOD and MCOD. Therefore, the time costs of all the baselines decrease. Similarly, for LSMOD, it may aggregate

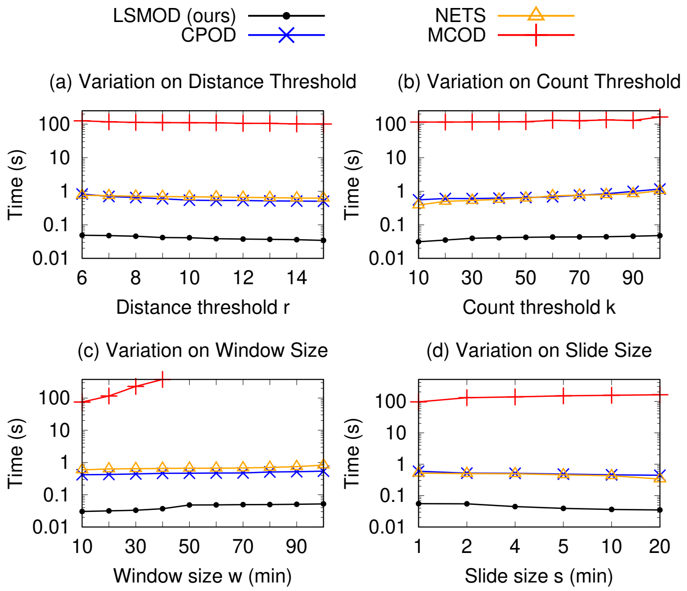

Fig. 16. Varying query parameters on WH-Chemistry

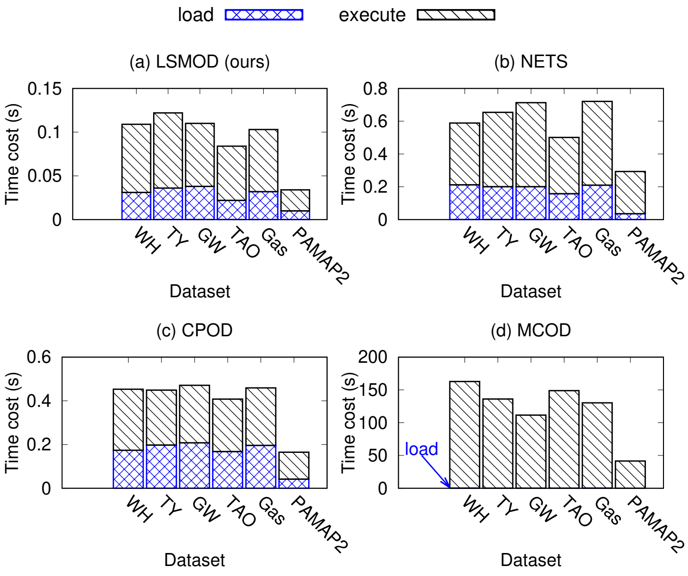

Fig. 17. Running time breakdown evaluation

more buckets with the increase of r, and the lower bounds for the neighbor count of each point become larger. Thus, more inliers can be pruned by comparing the lower bounds with the fixed k, also leading to the decrease of time cost.

When varying k in Figure 16(b), the baselines need to check more points to determine the point status no matter by which index structure, resulting in a slight increasing tendency. For our LSMOD, when k is small, inliers are easy to be determined by Proposition 5. However, fewer inliers can be pruned with the increase of k, and the time cost thus increases as well.

In Figure 16(c), with window size w increasing, the time cost of MCOD increases rapidly, since the points and micro-clusters in a window become more time-costly to maintain. For others, their time costs slightly increase due to the more costly initialization phases, while our LSMOD remains 1 order of magnitude faster than other baselines.

When varying slide size s in Figure 16(d), NETS, CPOD and our LSMOD have lower time costs under large slide sizes. This is because a larger slide size s means fewer windows to check, for a time series with fixed length. However, MCOD takes more time to maintain its micro-clusters, with more points expired and arrived for each slide.

I. Running Time Breakdown

We conduct breakdown evaluation for running time in Figure 17 to show whether LSM-tree designs impede the streaming methods. For LSMOD, we measure the time cost for loading bucket statistics, outliers and suspicious points from disk, taking about 20% of the total time. For the baselines, they use the same series readers as aforesaid, and thus they all have almost the same time costs of loading data from the LSM-tree store. It takes about 20%, 40% and 0.1% of their total time costs, for NETS, CPOD and MCOD, respectively. That is, the execution time costs of the baselines make up the major part of their overall time costs, much higher than the running time of LSMOD. In other words, the baselines are still more time-consuming than our proposal even if not considering the loading costs in LSM-tree storage, i.e., LSM-tree designs are not the bottleneck that impedes the streaming methods.

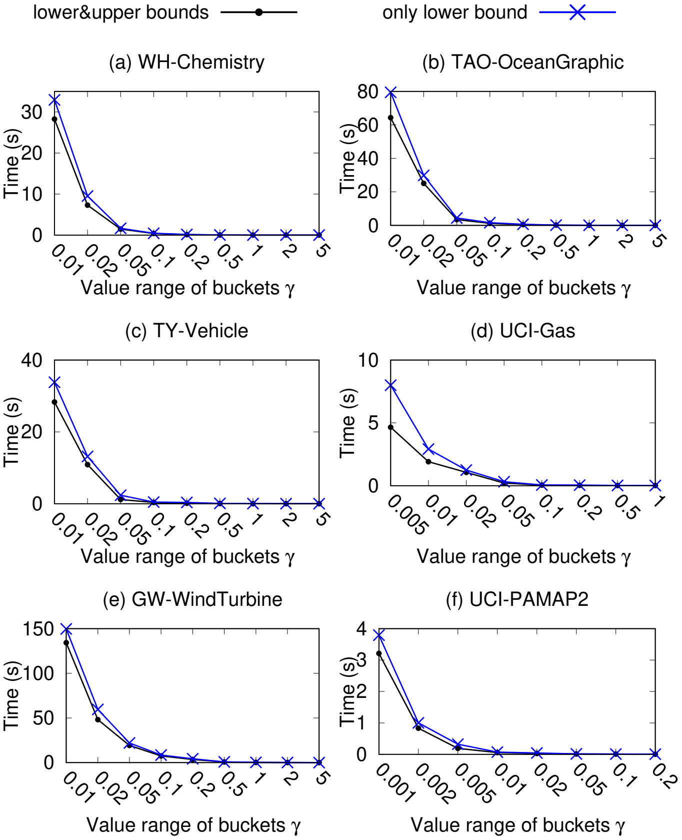

Fig. 18. Varying the width of buckets on value γ

J. Evaluation of Proposed Techniques

Recall that time series T is stored in files F_h and segments S_g with buckets $B[u]$, as illustrated in Figure 4. Therefore, in addition to the scalability in data size in Figure 9, we also evaluate how the numbers of files, segments and buckets affect the performance of LSMOD, by varying the number ρ of overlapping files, the time range δ of splitting segments and the value width γ of buckets, in Figures 10, 19 and 18, respectively. Ablation study is also presented to compare the proposed LSMOD with both upper and lower bounds, and with only lower bound.

1) Varying the Time Range of Segments δ: Figure 19 varies the time range δ for splitting time series into segments in Definition 4. Under the same data sizes, a larger time range δ leads to fewer segments, and thus the processing time cost decreases. Recall that to support query processing in sliding window, the window size w and slide s are usually the integral multiples of δ, as discussed in Section III-B. A larger δ thus means less supported queries.

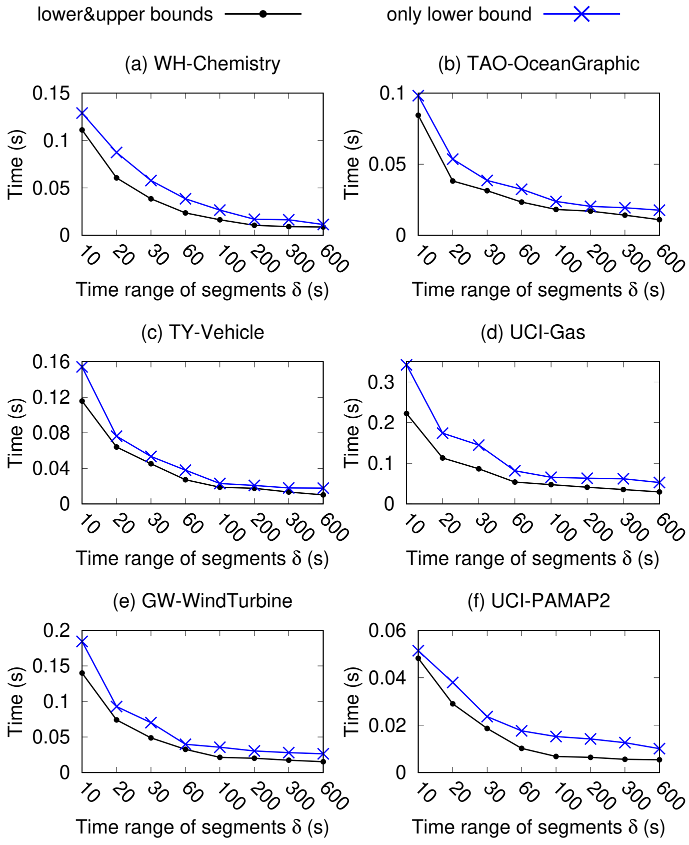

Fig. 19. Varying the range of segments on time δ

2) Varying the Value Width of Buckets γ: Figure 18 varies the bucket width γ of value range. Again, under the same data sizes, a larger γ leads to less buckets, and thus lower time cost. As also discussed at the end of Section IV-C, to enable the lower bound for pruning, we prefer to set a bucket width $\gamma \leq r$ in practice, and a larger γ means less supported queries. However, if γ is set very small, there will be a large number of buckets, even more than the number of points in a window, leading to fairly high time cost. In practice, a smaller γ supports more queries, while a larger γ decreases the query time cost, again a trade-off.

K. Evaluation on Supporting Various Queries

As introduced in Section I, outliers are often queried with different parameters for various interests. Same as the existing methods, our LSMOD also supports queries with different (1) neighbor distance threshold r, (2) neighbor count threshold k, (3) window size w, and (4) slide size s. We evaluate our proposal and other baselines under different query parameters. Figures 20, 21, 22, 23 present 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.

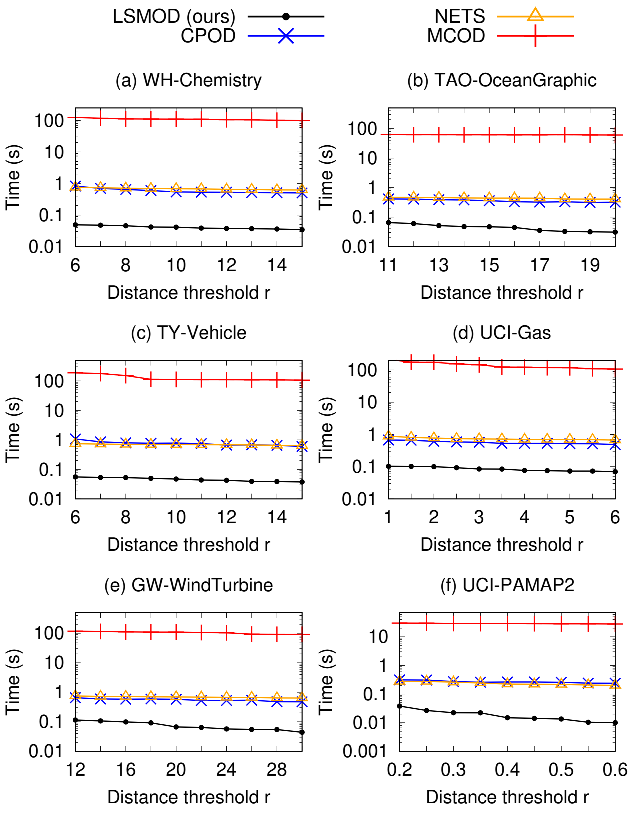

Fig. 20. Varying neighbor distance threshold *r* of query

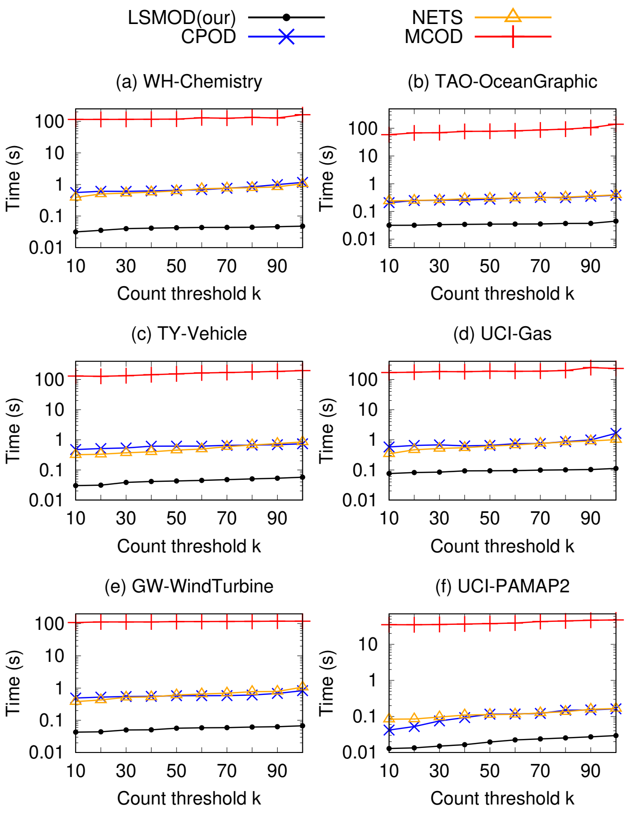

Fig. 21. Varying neighbor count threshold *k* of query

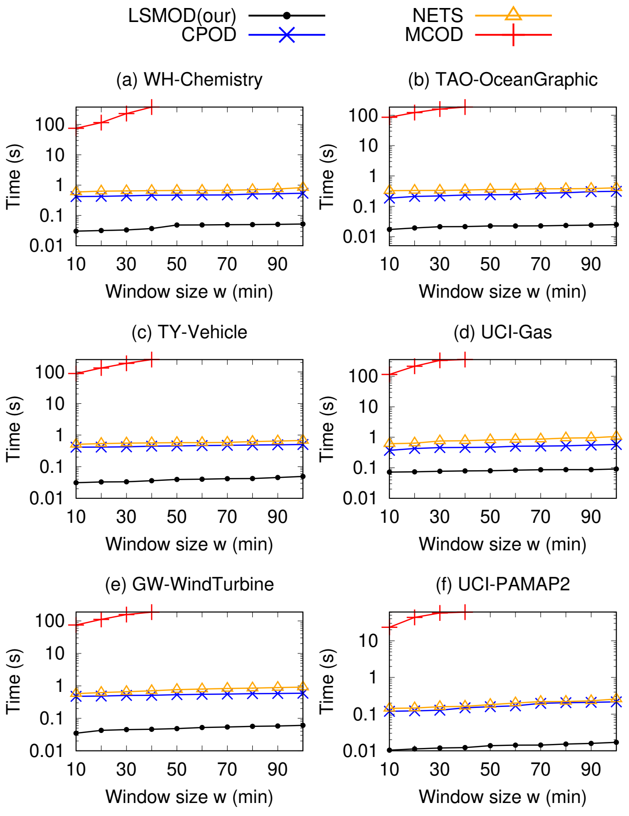

Fig. 22. Varying window size *w* of query

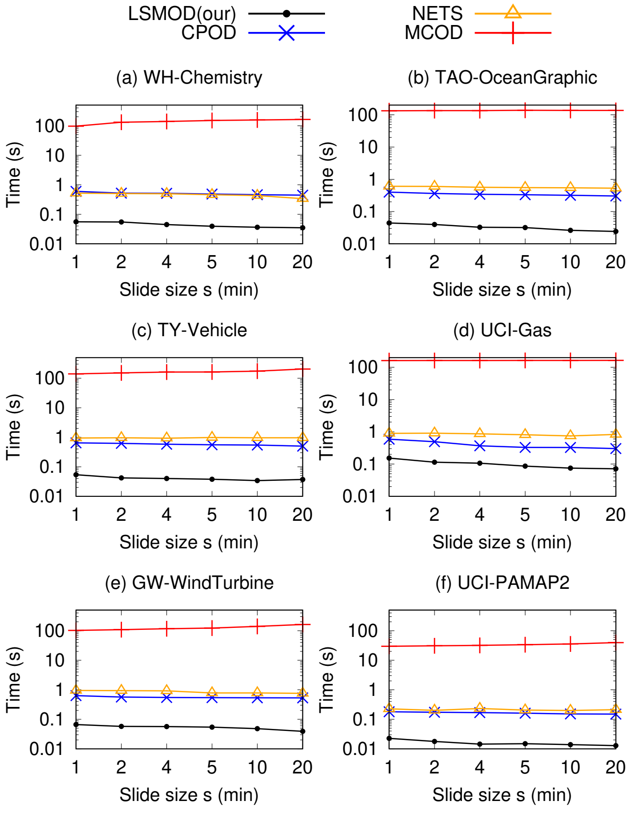

Fig. 23. Varying window slide *s* of query