CoBLOC - Continuous Block Level Fairness on Data Streams

Subhodeep Ghosh
NJIT, USA
sg2646@njit.edu

Zhihui Du
NJIT, USA
zhihui.du@njit.edu

Angela Bonifati
Lyon 1 University & IUF, France
angela.bonifati@univ-lyon1.fr

Manish Kumar
IIT Ropar, India
manishk@iitrpr.ac.in

David Bader
NJIT, USA
david.bader@njit.edu

Senjuti Basu Roy
NJIT, USA
senjutib@njit.edu

ABSTRACT

We present CoBLOC, the first interactive system for enforcing *continuous group fairness* over sliding windows in data streams. CoBLOC introduces *block-level fairness*, a fine-grained fairness model that exposes short-term disparities missed by window-level guarantees, and supports efficient real-time monitoring using compact sketch-based summaries. When violations occur, CoBLOC applies theoretically grounded stream reordering algorithms to restore fairness within the current window, while maintaining low-latency, high-throughput performance suitable for real-world streaming analytics. Through interactive demonstrations, CoBLOC also surfaces serendipitous moments where fairness adjustments become critical, provides intuitive explanations of its decisions, and recommends landmark sizes that best balance reordering cost and fairness in dynamic streaming settings.

PVLDB Reference Format:
Subhodeep Ghosh, Zhihui Du, Angela Bonifati, Manish Kumar, David Bader, and Senjuti Basu Roy. **CoBLOC** - Continuous Block Level Fairness on Data Streams. PVLDB, 19(12): 4662 - 4665, 2026.
doi:10.14778/3827998.3828091

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Subhodeep01/CoBLOC.

1 INTRODUCTION

Many real-world applications generate data as *streams* rather than static datasets, including financial market tickers, network monitoring systems, server logs, recommendation platforms, and user click streams [1, 6]. These applications rely on *continuous queries* over *sliding windows* to support real-time analytics on evolving data. In parallel, *fairness* in algorithmic decision-making has become a critical concern in data management, especially in high-stakes domains such as hiring, lending, healthcare, and law enforcement [2, 11, 12]; however, most existing work assumes static datasets and enforces group fairness only once on a fixed input [3, 9, 13], leaving fairness in streaming environments largely unexplored. This gap is particularly evident in real-time content recommendation platforms, where items arrive continuously and are presented in ordered blocks (e.g., pages): even if a sliding window is fair in aggregate, early blocks may exhibit severe representation skew, leading to unequal visibility due to attention decay, and thus motivating the need for fairness guarantees at both the window and block levels.

In this demonstration, we present CoBLOC (refer to Figure 2), the first ever novel interactive system for enforcing *continuous group fairness* over data streams. CoBLOC introduces a *block-level fairness* model that partitions each sliding window into application-defined *blocks* (e.g., pages, days, or weeks) and enforces fairness independently within each block. This enables fine-grained control over representation while naturally generalizing classical window-level group fairness as a special case.

To motivate the fairness model underlying CoBLOC, consider an online content recommendation platform where items arrive continuously from creators belonging to different ethnic groups (e.g., Caucasian, Asian, Hispanic). The system displays recommendations in a paginated format. Concretely, each sliding window contains 15 items, which are shown to the user across 3 pages, each page corresponding to a block of 5 items. Even when the overall window includes a balanced mix of content from all groups, the page-wise distribution may still be skewed — for example, the first few pages may predominantly feature Caucasian creators, while later pages contain more balanced or diverse representation (Refer to Figure 1). Since users' attention typically drops sharply across pages — with most engagement concentrated on the first one or two — this ordering bias results in unequal visibility and exposure. Thus, ensuring proportional representation across an entire window is insufficient; fairness must also be enforced within and across pages, accounting for both the presentation order and attention decay patterns. This is the fairness model CoBLOC is designed to enforce.

B_1:	$[1, C]$	$[2, C]$	$[3, A]$	$[4, H]$	$[5, H]$
B_2:	$[6, C]$	$[7, A]$	$[8, C]$	$[9, H]$	$[10, H]$
B_3:	$[11, A]$	$[12, A]$	$[13, C]$	$[14, H]$	$[15, H]$

Figure 1: A sliding window of 15 items, partitioned into three blocks of size 5, from three groups: *C* (Caucasian), *A* (Asian), *H* (Hispanic).

CoBLOC supports user-specified window size, block granularity, and group fairness constraints over a protected attribute, and continuously monitors, reorders, and explains fairness behavior over the stream. When violations occur, CoBLOC intervenes by reordering items within the current window. If necessary, the system may

Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828091

temporarily delay output by incorporating a small number of additional items—referred to as landmark items—to restore fairness across both current and newly formed windows.

Technical Novelty. CoBLOC is built around two core problems: **Continuous Fairness Monitoring** and the **Continuous Stream Reordering Problem**. The monitoring module continuously verifies block-level fairness within each sliding window (Figure 2). Upon detecting a violation, the system first attempts to restore fairness via reordering within the current window. If this fails, processing pauses briefly to ingest a small number of additional items, called *landmark items*, which extend the stream and induce new sliding windows, representing a "bounded buffering delay" (Figure 2). For example, appending 5 landmark items to a window of 15 induces 5 new sliding windows; the system then jointly reorders all 20 items to maximize fair blocks across all affected windows, with landmark size bounding the buffering delay. This is formalized as the *Continuous Stream Reordering Problem*. Once reordering is completed, the monitoring process proceeds. To support real-time interaction, CoBLOC uses compact sketch-based data structures to incrementally maintain attribute distributions with low storage and update overhead. These sketches enable low-latency fairness monitoring under high-throughput streams, while violations are resolved using efficient reordering algorithms with theoretical guarantees on fairness maximization under temporal constraints. [4] has further details.

2 SYSTEM ARCHITECTURE OF CoBLOC

CoBLOC (Figure 2) is implemented using Confluent Kafka Python, the official Python client for Apache Kafka, to continuously monitor fairness over streaming data and perform reordering when permitted by the application. Data from producers is incrementally pushed into dedicated Kafka *topics*, which act as online data stores. CoBLOC continuously polls these topics, retrieves items in order, and maintains a sliding window buffer of size $|W|$ (by default slide size is 1 item). Each incoming item is appended, and if the buffer is full, the oldest item is evicted. The buffered items are then processed for fairness evaluation and potential reordering.

User-Provided Query Inputs. User provides key input parameters, such as the input dataset, as well as query parameters, i.e. the window length, block size, fairness constraints (attribute and proportion), landmark size.

Continuous Fairness Monitoring. Given user inputs, CoBLOC fetches the first sliding window and constructs a compact sketch to evaluate fairness.

Process next window. CoBLOC interactively processes subsequent windows, incrementally updating the sketch each slide. When the user ends the session, a session summary with per-window query processing times is reported.

Continuous Stream Reordering. If a window violates the fairness constraints, the user may choose to reorder it. Upon a *Yes* response, CoBLOC first attempts an in-window reordering to satisfy the query. If this is insufficient, it retrieves a user-defined number of additional items (landmark elements) and jointly reorders the window and landmarks to maximize the number of *Yes* outcomes for the current and subsequent sliding windows.

Explanations and Recommendations. CoBLOC highlights *serendipitous moments* [8] at the end of each user session, instances where fairness adjustments become particularly critical. Additionally, it recommends *an appropriate landmark size for the session and provides explanations, guiding users toward an effective balance between reordering cost and fairness* in dynamic streaming environments.

3 PROBLEMS AND ALGORITHMS

CoBLOC solves two core technical problems [4], that we present below. As an example, refer to Figure 1, where window length is 15, block size is 5. Imagine there are two different fairness constraints given: Fairness Constraints C1 (analogously C2): ensure each block has 0.3 (0.5) proportion of Caucasian ($\mathcal{C}$), 0.3 (0.2) proportion of Asian ($\mathcal{A}$), and 0.4 (0.3) proportion of Hispanic ($\mathcal{H}$).

Continuous Fairness Monitoring. For every sliding window in the stream and a given fairness constraint, the window is deemed *fair* if each block in the window satisfies the constraint; otherwise, it is deemed *unfair*.

The window in Figure 1 is *fair* under C1 but *unfair* under C2.

Continuous Stream Reordering using Landmark Items. Given a window and additional landmark items, for a given fairness constraint, identify a reordering for data items in the window and the landmark, so that it maximizes the total number of unique fair blocks in the current and the induced sliding windows generated by landmark items.

Given the *Constraints-2*, the minimum number of $\mathcal{C}$ instances required in window of Figure 1 is calculated as $\lfloor 0.5 \times 5 \rfloor \times 3 = 6$. Since the window contains only 5 instances of $\mathcal{C}$, an additional 5 landmark items are read (as shown in Figure 2). These additional 5 items result in 5 more sliding windows. The objective is to perform reordering among the 20 available items (from window and landmarks) such that the maximum possible number of these 16 blocks satisfy fairness constraints.

3.1 Continuous Fairness Monitoring

We now briefly present our technical framework for monitoring fairness constraints continuously over streaming data. Our approach leverages compact sketches to efficiently track protected attribute distributions within sliding windows and verify block-level fairness in real time. Additional algorithmic details and formal proofs are provided in [4].

For the initial window, the system constructs a compact *Forward Sketch* that summarizes the cumulative distribution of protected attribute values across the window. As the window slides, the sketch is updated incrementally by discarding the oldest entry, shifting the remaining summaries, and appending a new cumulative count vector for the newly arrived item. The Forward Sketch can be built in time and space linear in the window size and the number of protected groups for the first window, and subsequently updated in time linear in the number of protected groups per slide, enabling efficient, low-latency monitoring of block-level fairness in streaming environments.

Using the Forward Sketch, we present an algorithm for monitoring block-level fairness that inspects only the sketch entries corresponding to block boundaries. The algorithm halts immediately upon detecting a violation, and reports the window as fair

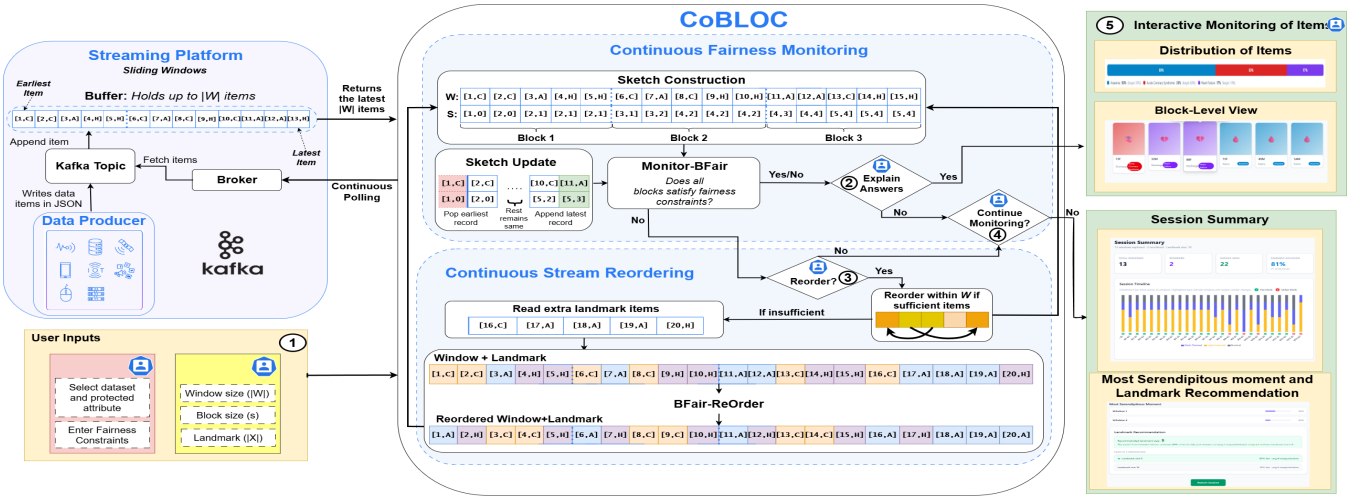

Figure 2: CoBLOC System

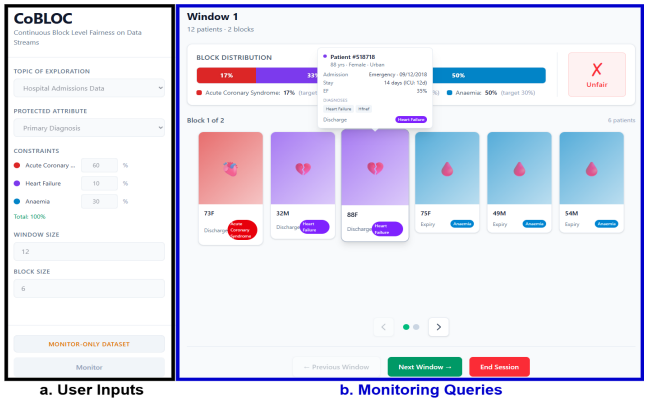

Figure 3: User Inputs and Monitoring Queries

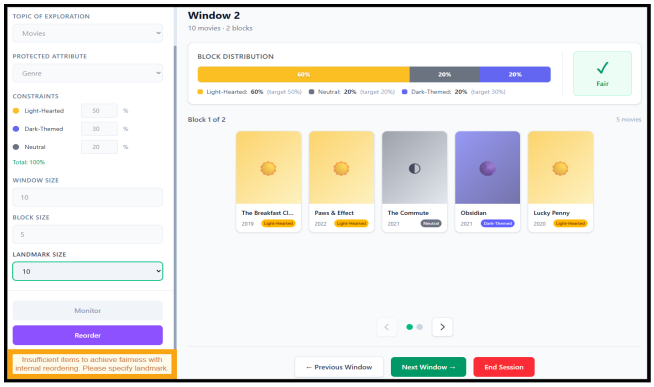

Figure 4: Reordering Queries

only if all blocks satisfy the required constraints. In the worst case, fairness checking runs in time proportional to the number of blocks and the number of protected groups, while in the best case it requires time proportional only to the number of protected groups. Additional algorithmic details and formal proofs are provided in [4].

3.2 Continuous Stream Reordering

The reordering problem is formalized as finding a permutation of the input stream that maximizes the number of unique fair blocks, accounting for multiple valid fairness requirements and the limited availability of items from each protected group. Unlike greedy approaches, the continuous overlapping-window setting is non-trivial: a locally fair block placement can violate fairness in subsequent overlapping windows, making greedy solutions globally suboptimal.

The key insight behind our solution is that any optimal reordering has a highly regular structure. When a fairness requirement specifies how many items from each protected group must appear in a fixed-size block, the maximum number of disjoint fair blocks that can be formed is limited by the most constrained group—that is, the group with the fewest available items relative to its required count. We design an algorithm that enumerates all valid fairness requirements, and for each one invokes a subroutine that constructs an optimal reordered stream using the isomorphic block framework. The algorithm selects the permutation that yields the maximum number of fair blocks. We formally prove that this approach is optimal and present its running time [4].

4 DEMONSTRATION OVERVIEW

CoBLOC will be demonstrated using two applications - hospital admission where only fairness monitoring is allowed; for movies recommendation using MovieLens [7], where both monitoring and reordering are allowed. In contrast to movie recommendations, the hospital admissions follow reality, and changing the order of admissions according to the best demographic group will delay treatment and violate triage protocols.

User Defined Inputs (Figure 3(a).) Users configure protected attribute, fairness constraints, window size, block size, and (for MovieLens) landmark size.

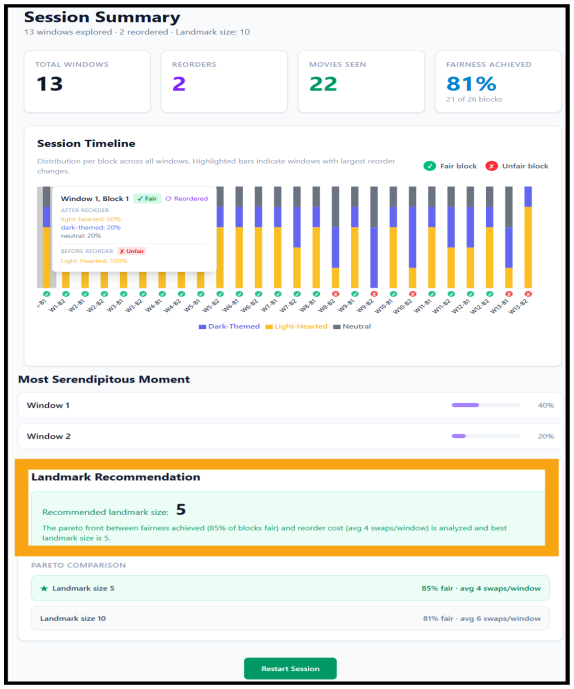

Figure 5: Providing Explanations

Monitoring (Figure 3(b).) CoBLOC continuously evaluates each sliding window, checking whether all blocks satisfy the specified fairness constraints. For example, in the hospital admission setting, users can define desired proportions over primary diagnosis values. Blocks correspond to hourly shifts within a fixed time window in this setting and fairness ensures no patient group is systematically deferred to later, higher-wait, lower-capacity periods within a window. The system verifies block-level fairness in real time and provides a distribution of items in the block for the user to verify whether target fairness has been reached or not. Hovering over an item, shows a panel that gives a brief description about the patient.

Reordering (Figure 4). If the application permits reordering—as in the MovieLens-based recommendation setting—CoBLOC leverages its reordering mechanism to enforce fairness. Given a user-specified protected attribute (e.g., genre), CoBLOC first attempts to reorder the current sliding window to satisfy the fairness constraints. When this is insufficient, it explicitly detects the limitation (see the orange highlighted box in Figure 4) and solicits user input for landmark size. Guided by this input (in this case, landmark size is specified as 10), the system adaptively retrieves a small set of landmark items and performs a joint reordering over both the window and the selected landmarks. This produces a maximally fair arrangement not only for the current window but also for subsequent induced sliding windows, enabling users to interactively observe how fairness is restored over time.

Explanations and Recommendations (Figure 5). The user can continue monitoring subsequent windows and invoke reordering whenever permitted. At the end of the session, CoBLOC presents two key insights. First, it identifies the *serendipitous moment*: under monitoring-only mode, this corresponds to the window with the highest total variation distance [5] revealing where the stream's fairness behavior was most anomalous, a likely indicator of systemic bias at that point; when reordering is enabled, it denotes the window exhibiting the largest divergence between the original and fairness-adjusted distributions marking where intervention had the greatest impact. Users can act on this information by changing constraints, adjusting block granularity, or flagging that window for external audit. Second, the system analyzes the trade-off between reordering cost per window - which is the average number of swaps per window - and the achieved fairness percentage across different landmark sizes, and recommends the size that best balances this trade-off—i.e., one that lies on the Pareto frontier [10]. As illustrated in the orange highlighted box in Figure 5, even though the user selects a landmark size of 10, the system recommends a size of 5, as it provides the most favorable trade-off. An explanation for this recommendation is also provided. Finally, CoBLOC reports the throughput for monitoring and (when applicable) reordering corresponding to different configurations of window size, block size and landmark size [4].

5 CONCLUSION

This demonstration presents CoBLOC, the first interactive system for enforcing continuous group fairness in data streams. Using a block-level fairness model, it enables real-time monitoring, explanation, and corrective reordering to detect and mitigate short-term disparities that window-level guarantees may obscure. The current system assumes ordered arrival with protected attributes immediately available - a deliberate simplification to keep the demo focused on the core fairness problem rather than general stream-processing engineering. Handling out-of-order arrivals, late records, and missing attributes are orthogonal challenges we leave for future work.

ACKNOWLEDGMENTS

Subhodeep Ghosh, Manish Kumar, and Senjuti Basu Roy's work are supported by the following funding agencies: (1) National Science Foundation award number(s): 1942913 (2) Office of Naval Research award number(s): N000141812838, N000142112966, N000142412466.

REFERENCES

[1] Shivnath Babu and Jennifer Widom. 2001. Continuous queries over data streams. *ACM Sigmod Record* 30, 3 (2001), 109–120.
[2] Reuben Binns. 2020. On the apparent conflict between individual and group fairness. In *FATS*. 514–524.
[3] David García-Soriano and Francesco Bonchi. 2021. Maxmin-fair ranking: individual fairness under group-fairness constraints. In *SIGKDD*. 436–446.
[4] Subhodeep Ghosh et al. 2026. Continuous Fairness On Data Streams. arXiv:2601.08976 [cs.LG] https://arxiv.org/abs/2601.08976
[5] Alison L Gibbs and Francis Edward Su. 2002. On choosing and bounding probability metrics. *International statistical review* 70, 3 (2002), 419–435.
[6] Lukasz Golab and M Tamer Özsu. 2003. Processing sliding window multi-joins in continuous queries over data streams. In *VLDB Conference*.
[7] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (Dec. 2015), 19 pages. https://doi.org/10.1145/2827872
[8] Denis Kotkov, Shuaiqiang Wang, and Jari Veijalainen. 2016. A survey of serendipity in recommender systems. *Knowledge-Based Systems* 111 (2016), 180–192.
[9] Caitlin Kuhlman and Elke Rundensteiner. 2020. Rank aggregation algorithms for fair consensus. *Proceedings of the VLDB Endowment* 13, 12 (2020).
[10] Patrick Ngatchou et al. 2005. Pareto multi objective optimization. In *ISAP*. IEEE.
[11] Evaggelia Pitoura et al. 2022. Fairness in rankings and recommendations: an overview. *The VLDB Journal* (2022), 1–28.
[12] Julia Stoyanovich et al. 2020. Responsible data management. *PVLDB* (2020).
[13] Dong Wei et al. 2022. Rank aggregation with proportionate fairness. In *SIGMOD*.