Brief Announcement: Fair Ordering via Streaming Social Choice Theory

Geoffrey Ramseyer
Stanford University
Stanford, California, USA
geoff.ramseyer@cs.stanford.edu

Ashish Goel
Stanford University
Stanford, California, USA
ashishg@stanford.edu

ABSTRACT

How can we order transactions in a replicated state machine "fairly?" In the model of prior work [2, 8, 9, 13], each of n replicas observes transactions in a different order, and the system aggregates these observed orderings into a single order. We argue that this problem is best viewed through the lens of the classic preference aggregation problem of social choice theory, in which rankings on candidates are aggregated into an election result.

Two features make this problem novel and distinct. First, the number of transactions is unbounded, and an ordering must be defined over a countably infinite set. And second, decisions must be made quickly and with only partial information. Additionally, some faulty replicas might misreport their observations; the influence of faulty replicas on the output should be well understood.

Prior work studies a "γ-batch-order-fairness" property, which divides an ordering into contiguous batches. If a γ fraction of replicas receive a transaction τ before another transaction τ', then τ' cannot be in an earlier batch than τ. This definition holds vacuously, so we strengthen it to require that batches have minimal size, while accounting for faulty replicas.

This lens gives a protocol with both strictly stronger fairness and better liveness properties than prior work. We specifically adapt the Ranked Pairs [12] method to this streaming setting. This algorithm can be applied on top of any of the communication protocols (in various network models) of prior work for immediate liveness and fairness improvements. Prior work relies on a fixed choice of γ and a bound on the number of faulty replicas f, but we show that Ranked Pairs satisfies our definition for every $\frac{1}{2} < \gamma \leq 1$ simultaneously and for any f, where fairness guarantees degrade as f increases.

CCS CONCEPTS

• **Computer systems organization** → Dependable and fault-tolerant systems and networks; • **Theory of computation** → **Streaming, sublinear and near linear time algorithms**.

KEYWORDS

social choice, fair ordering, blockchain architecture

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
PODC '24, June 17–21, 2024, Nantes, France
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0668-4/24/06
https://doi.org/10.1145/3662158.3662772

ACM Reference Format:
Geoffrey Ramseyer and Ashish Goel. 2024. Brief Announcement: Fair Ordering via Streaming Social Choice Theory. In *ACM Symposium on Principles of Distributed Computing (PODC '24), June 17–21, 2024, Nantes, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3662158.3662772

1 INTRODUCTION

We study the problem of ordering transactions in a replicated state machine. In the standard setting, each of a set of n distinct *replicas* maintains a copy of a *state machine.* Replicas communicate to agree on a totally-ordered log of transactions $\tau_1 \prec \tau_2 \prec \dots$, and then each replica applies transactions in this order to its state machine.

There are many communication protocols through which replicas can agree on *some* total order (solving the "total order broadcast" problem; Défago et al. [6] gives a survey). Yet in many systems, most notably within today's public blockchains, significant financial value can be derived from ordering transactions in specific ways [5]. These systems must therefore agree on not just any ordering but an "optimal" one, for some notion of optimal.

The key observation is that this problem is a novel streaming variation of the classic preference aggregation problem of social choice theory [1, 3, 4, 7, 10]. Prior protocols broadly run in two phases [2, 8, 9, 13]; first, some network protocol (in varying network models) produces agreement on a reported "vote" from each replica on the ordering of a set of transactions (in social choice parlance, a "ranking" on a set of "candidates"). And second, these reported votes are aggregated into a single, total order. This second phase is the subject of our study, and is agnostic to the communication protocol and network model of the first phase.

Two key differences separate this problem from the classic social choice setting. First, the number of transactions ("candidates") is countably infinite (as clients can continually send new transactions for an unbounded time). And second, the system must produce the output ranking in a streaming fashion. It cannot wait to see the reported orders over the entire (infinite) set of transactions before making a decision on the relative ordering of two transactions. Instead, the system must output an append-only, totally-ordered log of transactions. Furthermore, the system should minimize the delay between when a client sends a transaction and when that transaction is appended to the output.

While we believe this problem to be interesting in its own right, studying it directly through the lens of classic social choice theory, enables us to develop an ordering algorithm with both stronger "fairness" (the key property studied in prior work) and stronger liveness guarantees than the prior work. This algorithm could be deployed on top of any of the network protocols of prior work. The full version [11] includes precise comparisons with the prior work.

As an additional complication, replicas might adjust their reported orderings in response to strategic considerations. One desirable property of an order aggregation process is that the influence of a (potentially colluding) subset of replicas is bounded and precisely quantified. Mirroring the prior work, we write here that it is "faulty" behavior for a replica to report anything other than the order in which it observes transactions arriving over the network.

There is a wide body of social choice literature on this underlying aggregation problem. Our goal is to demonstrate an application for our streaming version of the problem, and to demonstrate the value of using social choice results in this application by targeting the precise desiderata raised in prior work. There are many other natural desiderata, notions of fairness, and aggregation rules that may be practically useful. The application of social-choice style aggregation rules in this streaming setting poses a number of interesting open questions. For example, if replicas have distinct financial motivations, or accept bribes from clients to order transactions in specific ways, is there an aggregation rule that maximizes (perhaps approximately) social welfare? For a reader coming from social choice theory, "transaction" could be replaced by "candidate", and "an ordering vote" by "a ranking."

2 MODEL

We consider a model in which there are *n replicas* cooperating to develop a total ordering of transactions. Transactions are received over the network from *clients*. We say that the ordering in which a replica receives transactions is that replica's observed ordering.

Definition 2.1 (Ordering Preference). An *Ordering Preference* on a (finite or countably-infinite) set of transactions is a total ordering [1] $\hat{\sigma_i} = \left(\tau_{i_1} \prec \tau_{i_2} \prec \tau_{i_3} \prec \dots\right)$

However, at any finite time, each replica can have received only a finite number of transactions. Replicas periodically submit these finite orderings to a ranking algorithm. The ranking algorithm is run repeatedly over time. We assume that that replicas can reliably broadcast their votes to each other, and come to agreement on the votes of all replicas (i.e., agree on the input to a ranking algorithm), but abstract away the communication protocol and network model.

Definition 2.2 (Ordering Vote). A replica i submits to a ranking algorithm an *ordering vote* on a set of k transactions, $\sigma = \left(\tau_{i_1}, \dots \tau_{i_k}\right)$. (where $\tau_{i_{j_1}} \prec \tau_{i_{j_2}}$ for $j_1 < j_2$).

Definition 2.3 (Ranking Algorithm). A deterministic algorithm $\mathcal{A}(\sigma_1, \dots, \sigma_n)$ takes as input an ordering vote from each replica and outputs an ordering σ on a subset of the transactions in the input.

The output need not include every transaction in the input. We say that a replica is *honest* if its observed ordering always extends its ordering vote, and if whenever it submits a vote, it contains all transactions that the replica has observed at that time. Otherwise, the replica is *faulty.*

3 MINIMAL BATCH ORDER FAIRNESS

We focus on the notion of "fairness" proposed and studied in prior work [2, 8, 9], but strengthen it to provide meaningful and achievable guarantees.

[1] Isomorphic to a subset of ω

Definition 3.1 (γ-batch-order-fairness). Suppose that τ and τ' are received by all nodes. If γn nodes received τ before τ', then a ranking algorithm outputs τ no later than τ'.

Aequitas [9] outputs transactions in batches, and "no later than" means "in the same batch." Transactions within a batch are ordered arbitrarily. Themis [8] outputs a total ordering, with the assertion that this total ordering could be divided into contiguous (disjoint) batches satisfying this property.

There are two problems with this definition. First, prior work relies on a fixed choice of parameter γ, yet neither higher nor lower γ gives stronger fairness properties. Second, Definition 3.1 is vacuously satisfied on all output orderings, for every γ, and thus provides no meaningful ordering guarantees (every ordering can be divided into contiguous, disjoint batches satisfying this property, by putting every transaction into one batch). Instead, we construct the following, stronger definition.

Definition 3.2 ((γ, δ)-minimal-batch-order-fairness). An ordering is (γ, δ)-minimally-batch-order-fair if, for any transaction pair (τ, τ') that is received in that order by at least γn replicas but output by the protocol in the reverse ordering, then there is a sequence of transactions $\tau' = \tau_1, \dots, \tau_k = \tau$ where at least $(\gamma - 2\delta)n$ replicas receive $\tau_i \prec \tau_{i+1}$ and τ_i is output before τ_{i+1}.

Definition 3.2 captures the notion that a protocol cannot distinguish between a δ-fraction of faulty replicas misreporting a transaction ordering. Given this indistinguishable δ fraction, the protocol outputs minimally-sized batches. A stronger notion of minimality, in the face of faulty replicas, is not achievable.

This definition does not discuss "batches" or "minimality," but approximately minimal batches (the strongly connected components of Lemma 3.4) can be recovered from any ordering satisfying it. The second condition of Lemma 3.4 limits the size of output batches.

Definition 3.3 (Ordering Graph). Suppose that replicas report ordering votes $(\sigma_1, \dots, \sigma_n)$ on a set of transactions V. The *Ordering Graph* $G(\sigma_1, \dots \sigma_n)$ is a complete, weighted directed graph with vertex set V and, for each transaction pair (τ, τ'), an edge of weight $w(\tau, \tau') = \alpha$ if αn replicas report receiving τ before τ'.

LEMMA 3.4. *Suppose that an output ordering satisfies (γ, δ)-minimal-batch-order-fairness. Compute the ordering graph $\hat{G}$, drop all edges with weight below $\gamma - \delta$, and compute the strongly connected components of the remainder.*

- *If γn replicas received $\tau \prec \tau'$, then either τ and τ' are in the same strongly connected component, or all transactions in the component containing τ are output before any transactions in the component containing τ'.*
- *If γn replicas receive $\tau \prec \tau'$ and there is no sequence of transactions $\tau' = \tau_1, \dots, \tau_k = \tau$ where at least $(\gamma - 2\delta)n$ replicas receive $\tau_i \prec \tau_{i+1}$, then all transactions in the component containing τ are output before any transactions in the component containing τ'.*

4 STREAMING RANKED PAIRS

We now turn to the streaming setting that is our focus. At minimum, a streaming algorithm must be *monotonic* and *asymptotically live.* If replicas extend their ordering votes, the algorithm, when run again on the extensions, must only extend its prior output, and must eventually output each transaction.

Definition 4.1 (Monotonicity). A ranking algorithm $\mathcal{A}(\cdot, \ldots, \cdot)$ is *monotonic* if, given two sets of ordering votes $(\sigma_1, \ldots, \sigma_n)$ and $(\sigma'_1, \ldots, \sigma'_n)$, such that σ'_i extends σ_i for all $i \in [n]$, $\mathcal{A}(\sigma'_1, \ldots, \sigma'_n)$ extends $\mathcal{A}(\sigma_1, \ldots, \sigma_n)$.

Definition 4.2 (Asymptotic Liveness). A ranking algorithm $\mathcal{A}(\cdot, \ldots, \cdot)$ is *asymptotically live* if, given any set of countably infinite ordering votes $(\hat{\sigma_1}, \ldots, \hat{\sigma_n})$ and any transaction τ in those votes, there exists an N such that when each $\hat{\sigma_i}$ is trunctated to the first N elements of the ordering to produce σ_i, τ is included in $\mathcal{A}(\sigma_1, \ldots, \sigma_n)$.

A monotonic ranking algorithm implies a well-formed definition for aggregating a set of orderings on countably infinite sets of transactions. τ comes before τ' in the infinite case if there exists a finite subset of the input such that the algorithm outputs $\tau \prec \tau'$.

The Ranked Pairs algorithm [12] computes the ordering graph (Defn. 3.3), iterates through edges in order of weight, and greedily adds to edges to a set, so long as it does not create a cycle.

THEOREM 4.3. *Given a ordering vote (on every transaction in a finite set) from every replica, Ranked Pairs Voting simultaneously satisfies $(\gamma, \frac{f}{n})$-minimal-batch-order-fairness for every γ, and does not depend on any fixed bound on f.*

However, a streaming version of Ranked Pairs cannot always know whether the non-streaming equivalent would accept an edge. These edges are marked "*indeterminate*." First, we construct an ordering graph that overapproximates the unknown information.

Definition 4.4 (Streamed Ordering Graph).
Suppose that each replica submits an ordering vote $(\sigma_1, ..., \sigma_n)$. Let V be the set of all transactions that appear in each vote, and let $\hat{v}$ be the "future" vertex. The *Streamed Ordering Graph* is a weighted, directed, complete graph $\hat{G} = (V', E)$ on vertex set $V \cup \{\hat{v}\}$.

For each τ and τ', set weights $w(\tau, \tau')$ and $w(\tau', \tau)$ as in Definition 3.3. Set $w(\tau, \hat{v}) = 1$ for all τ. If there exists τ' that appears in the votes of some but not all replicas and which preceeds τ in at least one replica's vote, set $w(\hat{v}, \tau) = 1$, and otherwise $w(\hat{v}, \tau) = 0$.

Two details require mention. First, rather than recompute the entire ordering from scratch on every repeated invocation of the algorithm, Algorithm 1 caches the decisions made in a prior invocation, so the marginal runtime of the algorithm depends only on the number of new edges and transactions that arrive since the previous invocation. Second, the algorithm lazily (implicitly) constructs a tiebreaking rule between edges of equal weight; this is crucial for asymptotic liveness.

LEMMA 4.5. *Consider a counterfactual scenario where clients stop sending transactions, all replicas eventually receive every transaction, and every replica includes every transaction in its output vote.*

Whenever Algorithm 1 includes a determinate edge in H (resp. excludes an edge), that edge is included (resp. excluded) in the output of the non-streaming Ranked Pairs on the counterfactual input.

THEOREM 4.6. *Algorithm 1 is monotonic, and its output matches an initial segment of the output of the non-streaming Ranked Pairs (on the counterfactual of Lemma 4.5).*

ALGORITHM 1: Streaming Ranked Pairs

```
Input: An ordering vote from each replica $\{\sigma_i\}_{i \in [n]}$
Input: $\hat{H}$, the graph computed in the preceeding invocation
        of Algorithm 1
$\hat{G} = (V, E, w) \leftarrow \hat{G}(\sigma_1, ..., \sigma_n)$
$H \leftarrow (V, \emptyset)$
foreach edge $(\tau, \hat{v})$ and $(\hat{v}, \tau)$ with $w(e) = 1$ do
    Add $e$ to $H$, and mark it as indeterminate
end
Add all determinate edges of $\hat{H}$ to $H$, marked determinate
foreach $\gamma$ in $(1, \frac{n-1}{n}, \frac{n-2}{n}, \ldots, 0)$ do
    $E_\gamma \leftarrow \{e \in E \mid w(e) = \gamma\}$, ordered arbitrarily
    repeat
        $e = (\tau_i, \tau_j) \leftarrow$ first element of $E_\gamma$
        $U_{\tau_i,\tau_j} \leftarrow (V \setminus (R_{\tau_i} \cup P_{\tau_j})) \cup \{\hat{v}\}$
        If there is a directed path in $H \cap U_{\tau_i,\tau_j}$ from $\tau_j$ to $\tau_i$
          where every edge is determinate, then do not
          include the edge in $H$. Remove the edge from $E_\gamma$.
        Else if there is no directed path in $H \cap U_{\tau_i,\tau_j}$ from $\tau_j$
          to $\tau_i$ where every edge is either determinate or
          indeterminate, add $(\tau_i, \tau_j)$ to $H$ and mark it as
          determinate. Remove the edge from $E_\gamma$.
        Otherwise, defer $e$ to the end of the ordering on $E_\gamma$.
    until $E_\gamma = \emptyset$ or no progress is made in a full loop over $E_\gamma$;
    Mark all remaining edges in $E_\gamma$ as indeterminate
end
Output: The topological sort of $H$, up to (not including) the
        first transaction bordering an indeterminate edge
```

Liveness follows from studying how marking an edge *indeterminate* propagates through the algorithm. Because of the way that the algorithm visits edges of equal weight, an edge is only marked *indeterminate* if there is an *indeterminate* edge of strictly higher weight that is "nearby."

LEMMA 4.7. *If Algorithm 1 marks an edge (τ_i, τ_j) with weight $\frac{k}{n}$ as indeterminate, there must be a path from τ_j to τ_i contained in U_{τ_i,τ_j} that contains an indeterminate edge of weight at least $\frac{k+1}{n}$.*

Asymptotic liveness follows by bounding the length of a chain of *indeterminate* edges (to at most n).

THEOREM 4.8. *Algorithm 1 is asymptotically live.*

Precise liveness depends on the network model. Our metric is the liveness delay induced by our ranking algorithm (which is added to that of an underlying communication protocol). Assumption 1 gives an example communication assumption.

ASSUMPTION 1 (SYNCHRONOUS NETWORK). *If a client sends a transaction τ at time t, all honest replicas include τ in their ordering votes before time $t + \Delta$.*

THEOREM 4.9. *A transaction τ is contained in the output of the algorithm of Algorithm 1 after at most $(n + 1)\Delta$ time.*

ACKNOWLEDGMENTS

This research is supported by the IOG Research Hub, and by the Stanford Future of Digital Currency Initiative.

REFERENCES

[1] Kenneth J Arrow. 1950. A difficulty in the concept of social welfare. *Journal of political economy* 58, 4 (1950), 328–346.

[2] Christian Cachin, Jovana Mićić, Nathalie Steinhauer, and Luca Zanolini. 2022. Quick order fairness. In *Financial Cryptography and Data Security: 26th International Conference, FC 2022, Grenada, May 2–6, 2022, Revised Selected Papers*. Springer, 316–333.

[3] Josep M Colomer. 2013. Ramon Llull: from 'Ars electionis' to social choice theory. *Social Choice and Welfare* 40, 2 (2013), 317–328.

[4] Marquis de Condorcet and Marquis de Caritat. 1785. An Essay on the Application of Analysis to the Probability of Decisions Rendered by a Plurality of Votes. *Classics of social choice* (1785), 91–112.

[5] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. 2020. Flash boys 2.0: Frontrunning in decentralized exchanges, miner extractable value, and consensus instability. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 910–927.

[6] Xavier Défago, André Schiper, and Péter Urbán. 2004. Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Comput. Surv.* 36, 4 (dec 2004), 372–421. https://doi.org/10.1145/1041680.1041682

[7] Günter Hägele and Friedrich Pukelsheim. 2001. Lulls' writings on electoral sytems. (2001).

[8] Mahimna Kelkar, Soubhik Deb, Sishan Long, Ari Juels, and Sreeram Kannan. 2021. Themis: Fast, strong order-fairness in byzantine consensus. *Cryptology ePrint Archive* (2021).

[9] Mahimna Kelkar, Fan Zhang, Steven Goldfeder, and Ari Juels. 2020. Order-fairness for byzantine consensus. In *Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part III 40*. Springer, 451–480.

[10] Ramon Llull and Jordi Gaya. 1978. *Ars notatoria.* Citema.

[11] Geoffrey Ramseyer and Ashish Goel. 2023. Fair ordering via streaming social choice theory. *arXiv preprint arXiv:2304.02730* (2023).

[12] T Nicolaus Tideman. 1987. Independence of clones as a criterion for voting rules. *Social Choice and Welfare* 4, 3 (1987), 185–206.

[13] Mohammad Amin Vafadar and Majid Khabbazian. 2023. Condorcet Attack Against Fair Transaction Ordering. In *5th Conference on Advances in Financial Technologies.*