Metric Distortion with Elicited Pairwise Comparisons

Soroush Ebadian[1] , **Daniel Halpern**[2] , **Evi Micha**[2]
[1]University of Toronto
[2]Harvard University
soroush@cs.toronto.edu, dhalpern@g.harvard.edu, emicha@cs.toronto.edu

Abstract

In many social choice applications, information about individuals' preferences can only be elicited using a limited number of pairwise comparisons. In these cases, the task is twofold: we must first choose the queries, and then second, we must aggregate the responses to choose an outcome. We study the problem of designing algorithms for this setting. To compare the effectiveness of different outcomes, we use the metric distortion framework. In addition, we consider various constraints on the query algorithms, namely, placing restrictions on how the choice of the next query may depend on previous answers. Our main contributions are nearly optimal algorithms for all settings considered.

1 Introduction

When aggregating a population's preferences to make a decision, we must contend with the fact that our access to preferences may be limited. Take as a motivating example *Reinforcement Learning with Human Feedback (RLHF)*, a process used to fine-tune large language models. At a high level, the goal is to choose model parameters that yield desirable outcomes according to the participants. We typically learn what is "desirable" by asking comparison queries, i.e., on prompt x, do you prefer output y or output z? However, notice that these responses only give us partial information about the individuals' true underlying preferences. It would be completely infeasible to learn, say, a ranking for each person over all possible prompt-output combinations. As another example, consider the opinion aggregation platform *Polis* [Small *et al.*, 2021]. Here, we would like to summarize a community's opinions on an issue (e.g., how should we regulate Uber?). Participants submit free-form comments ("I think Passenger Liability Insurance should be mandatory") and are shown comments of other users to vote on, either agree or disagree. However, similar to the fine-tuning example, there are simply too many submissions from other users to feasibly ask every individual about every comment. In both these examples, human input is a scarce resource that must be carefully allocated.

A unifying theme of the applications discussed is that there are two stages to the aggregation pipeline: first, we must decide which queries to ask each participant, and second, given the responses, we must choose an outcome. While the field of computational social choice [Brandt *et al.*, 2016] gives many tools to understand and design systems for the latter stage, the main focus of this work is the former. In particular, we consider the problem of *choosing a limited number of pairwise-comparison queries to effectively aggregate heterogeneous preferences*.

When selecting the queries, it is crucial to consider various constraints that a designer might need to satisfy. For example, on platforms like Polis, it is not guaranteed that we can interact with a user, then acquire more information from other users, and later return to the original user for a second round of queries. For this, we may want algorithms that are *single-pass*: the queries asked to earlier users do not depend on queries to ones arriving later. Another desirable property a practitioner might want to satisfy is a notion of fairness, say, the queries asked for each agent to be independent of the responses of other agents; this can help ensure that interaction with the system remains unaffected by the arrival time. The same need may also arise from temporal concerns: if agents all answer simultaneously, we simply do not have data from the other agents. Conversely, we may want to have queries be independent of an agent's *own* answers, ensuring an unbiased interaction with the system. Therefore, our main research questions are:

How can we design a limited number of pairwise comparison queries to achieve efficient outcomes? Furthermore, how well can we do so under various restrictions such as limited rounds of interactions and which answers a query can depend on?

To understand the effectiveness of various query choices, we need a way to measure the quality of the outcome. For this, we turn to the celebrated *distortion framework* [Procaccia and Rosenschein, 2006]. Here, it is assumed that the comparison feedback submitted by the agents is derived from more expressive underlying preferences, typically scalar values. The more expressive values give rise to concrete objective values for each of the outcomes. The goal is to design a procedure that minimizes *distortion*, the worst-case ratio between the best possible objective value and the objective value of the chosen outcome. Without additional assumptions, achieving bounded distortion becomes impossible, even with complete knowledge of each agent's ranking. In this work, we make the frequently-made natural assumption that agents have costs induced by distances in an underlying metric space [Anshelevich *et al.*, 2018]. Beyond just a theoretical necessity, this restriction seems especially aligned with the motivating examples discussed.[1] To summarize, our goal is to optimize metric distortion using a limited number of practical pairwise comparison queries.

1.1 Our approach and results

We assume there are m alternatives, and we can ask each agent up to t pairwise comparison queries. A pairwise comparison query (or simply a query) involves two alternatives that an agent is asked to rank. Note that if t is quite large relative to m (e.g., on the order of $\Omega(m \log m)$), we can learn the full ranking of an agent.[2] However, in applications like RLHF and Polis, t is much smaller.

We first show that a query algorithm making at most t queries to each agent, even if it can ask these in an arbitrary order with arbitrary dependencies, will incur distortion $\Omega(m/t)$ (Section 3.1).

Next, we consider algorithms that match this lower bound. We are especially interested in algorithms that require few rounds of interactions, as it may not be reasonable to assume that agents return later in the process after answering queries. Of particular importance is the extreme case: *single-pass* algorithms needing only one round of interaction. Strikingly, we show that there exists a single-pass algorithm that achieves optimal distortion of $O(m/t)$ (Section 3.2).

Finally, we consider other kinds of restrictions on the adaptivity of the algorithm, i.e., how the queries depend on previous answers. We focus on two types of adaptivity. The first, called *inter-agent adaptivity*, indicates whether the queries made to an agent can depend on responses from other agents. The second one, called *intra-agent adaptivity*, indicates whether the queries made to an agent can depend on previous responses from that agent. Here, we devise nearly tight (up to logarithmic factors) upper and lower bounds for the optimal distortion achievable under all combinations of these restrictions (Section 4). The results are summarized in Table 1. Note that all upper bounds are induced by single-pass algorithms, while the lower bounds hold for all algorithms, regardless of the number of passes.

[1]In Polis, as part of the aggregation process, users and comments are mapped to a common low-dimensional feature space representing "opinion" space summarizing the various views. Within opinion space, voters are more likely to approve comments near them. Hence, Polis is implicitly assuming it is possible to map users/comments to a metric space. In RLHF, the possible models generating input/output prompts live in a feature space. An agent is assumed to give feedback according to a reward function of these features. If the reward function is concave, it is natural to assume that an agent has an ideal point maximizing their reward function and prefers inputs closer to this point.

[2]This is equivalent to using comparison-based queries to sort a list. Hence, this can be done via any sorting algorithm, e.g., MergeSort. [Cormen *et al.*, 2001]

Inter-agent	Intra-agent	
	Yes	No
Yes	$\Theta\left(\frac{m}{t}\right)$	$O\left(\frac{m}{t} + \log m\right)$, $\Omega\left(\frac{m}{t}\right)$
No	$O\left(\frac{m^2 \log^2 t}{t^2}\right)$, $\Omega\left(\frac{m^2}{t^2}\right)$	$\Theta\left(\frac{m^2}{t}\right)$

Table 1: Summary of results for different inter- and intra-agent adaptivity combinations. All upper bounds are achieved with single-pass rules. All lower bounds hold regardless of the number of passes.

1.2 Related Work

The most closely related work is that of Anagnostides *et al.* [2022], who also consider the metric distortion problem while eliciting preferences. However, the preference elicitation method differs from ours; they assume that an algorithm can query two candidates $\{a, b\}$ and, in response, learns which of the two candidates a majority of voters prefer. While this could be implemented in our model, note that this would require multiple rounds of interaction and also makes additional restrictions, e.g., all voters will be asked the same queries, and only the majority winner is given, rather than the magnitude of this majority. Hence, we are interested in similar questions under a different query model, which we view as more suited for our motivating applications.

Slightly further afield is considering metric distortion, where only partial information about each agent's ranking is given. For example, the same work [Anagnostides *et al.*, 2022] builds on work by Kempe [2020b] where voters reveal their top-k candidates for a fixed value $k \leq m$. Kempe [2020b] also considers the more general problem of b bits of information being communicated about a ranking using a communication scheme fixed upfront.

Metric distortion itself (using complete rankings) was introduced by Anshelevich *et al.* [2015]. They show that no social choice rule has distortion better than 3, while the well-known Copeland rule achieves distortion 5, among bounds for other well-known rules. Followup work proved bounds for even more rules [Skowron and Elkind, 2017; Goel *et al.*, 2017], but none beat this barrier of 5. Significant progress was made by Munagala and Wang [2019], who introduced a rule achieving distortion 4.236. The gap was finally closed by Gkatzelis *et al.* [2020], who gave a rule achieving distortion 3. Anshelevich *et al.* [2021] provide a survey of these results, as well as progress in non-metric distortion more broadly.

In addition to improvements in lowering distortion, other progress has been made. Kempe [2020a] gives an LP-duality framework for analyzing the distortion of existing rules. Kizilkaya and Kempe [2022] and Kizilkaya and Kempe [2023] both provide new rules achieving optimal metric distortion requiring much simpler analyses. We build on techniques from Kempe [2020a] as well as the rule of Kizilkaya and Kempe [2022]; a more in-depth description can be found when we use them.

Preference elicitation, more broadly, has a rich history in computational social choice with several variations. A more general summary can be found in Brandt *et al.* [2016]; however, we focus on the most closely related works here. Lu and

Boutilier [2011] has the most similar style pairwise queries to our model; however, their objective is quite different from metric distortion. Amanatidis *et al.* [2021] study the tradeoff of distortion and query complexity; however, their queries are quite dissimilar, directly eliciting cardinal values and relative magnitudes rather than information about the ranking. Preference elicitation questions are also often studied through the lens of communication complexity, i.e., the number of bits that need to be revealed rather than the number of queries of a specific type. For example, Conitzer and Sandholm [2005] give the communication complexity of determining the winner of several common voting rules (where each agent privately knows their ranking). This has also been applied to non-metric distortion, where voters can reveal a certain number of bits about their utilities rather than just the rankings [Mandal *et al.*, 2019, 2020].

2 Model

For $s \in \mathbb{N}$, define $[s] = \{1, 2, \ldots, s\}$.

Metric Voting. We consider the setting of single-winner elections where we have a set $C = [m]$ of m candidates and a set $V = [n]$ of n voters. We assume there is a (pseudo)metric[3] over $C \cup V$ characterized by a distance function $d : (C \cup V) \times (C \cup V) \to \mathbb{R}_{\geq 0}$. That is, for all $i, j, k \in C \cup V$, we have that $d(i, i) = 0$, $d(i, j) = d(j, i)$, and $d(i, j) \leq d(i, k) + d(k, j)$ (the triangle inequality). Furthermore, each voter $i \in V$ has a *preference ranking* $\sigma_i = \sigma_i(1) \succ_i \cdots \succ_i \sigma_i(m)$ such that $a \succ_i b$ implies that $d(i, a) \leq d(i, b)$ — we allow ties to be broken arbitrarily. The collection of all voters' preference rankings form a *preference profile* $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$. An *instance* is a tuple $(d, \boldsymbol{\sigma})$.

Utilitarianism. Given an instance $(d, \boldsymbol{\sigma})$, the *social cost* (or simply cost) of a candidate $c \in C$ is defined as $\text{cost}_d(c) := \sum_{i \in V} d(i, c)$. When d is clear from context, we may drop it from the notation by simply writing $\text{cost}(c)$. Under this measure, the optimal candidate is the one minimizing social cost. We will use the notation $\text{opt}(d, \boldsymbol{\sigma}) := \min_{c \in C} \text{cost}_d(c)$ for the minimum cost.

Algorithms with Pairwise Comparison Queries. We consider algorithms that do not have direct access to the underlying metric nor the preference profile; rather, they are only aware of C and V and can learn of voters' preferences via a sequence of *pairwise comparison queries*. That is, the algorithm can choose a voter i and two candidates $\{a, b\}$ and learn which candidate voter i prefers, $a \succ_i b$ or $b \succ_i a$. After receiving the responses, the algorithm (i.e., a voting rule) returns a candidate c as the winner. More formally, by a slight abuse of notation, we let $\mathcal{A}(d, \boldsymbol{\sigma})$ denote the candidate returned by an algorithm $\mathcal{A}$ on instance $(d, \boldsymbol{\sigma})$.

We are interested in algorithms that do not make too many queries to each voter. If an algorithm makes at most t queries to each voter i, we call it a *t-query* algorithm.

[3]A pseudometric is a generalization of a metric that allows two distinct points to be distance 0 from each other. In other words, we allow voters and candidates to be at the same location.

Distortion. The quantitative measure of efficiency we investigate in this work is *distortion*. For a given instance $(d, \boldsymbol{\sigma})$, the distortion of candidate c is defined as

$$\text{dist}(c \mid (d, \boldsymbol{\sigma})) = \frac{\text{cost}_d(c)}{\text{opt}(d, \boldsymbol{\sigma})},$$

i.e., the ratio of the social cost of c to the optimal social cost. This value is always at least one, and the lower the ratio, the more efficient the candidate. The distortion of an algorithm $\mathcal{A}$ is defined as

$$\text{dist}_m(\mathcal{A}) = \sup_{(d, \boldsymbol{\sigma}), |C| = m} \text{dist}(\mathcal{A}(d, \boldsymbol{\sigma}) \mid (d, \boldsymbol{\sigma})),$$

where the supremum is over all instances with m candidates and any number of voters. In other words, the distortion of an algorithm $\mathcal{A}$ is its worst-case guarantee in terms of the multiplicative approximation to the optimal social cost.

2.1 Algorithm Restrictions

In the setup described so far, an algorithm can make queries to voters in an arbitrary order, with the option to revisit voters it has already queried or choose each subsequent query based on arbitrary prior responses. This flexibility may be costly or even impractical in certain scenarios. Hence, we are interested in the optimal distortion of algorithms subject to various constraints. We describe the restrictions of interest below.

Number of passes. We will say an algorithm is *p-pass* if the sequence of queries can be partitioned into at most p contiguous sequences where agent indices are non-decreasing. We will call it *single-pass* if it is 1-pass.

Inter-agent adaptivity. An algorithm is *inter-agent nonadaptive* if the queries it makes to voter i do not depend on responses from other voters $i' \neq i$. In other words, the choice of queries can be decomposed into n subprocedures $A_1, \ldots, A_n$ such that A_i only makes queries and learns the responses of voter i (the aggregation given these responses can be arbitrary). Note that an inter-agent nonadaptive algorithm can always be implemented as a single-pass algorithm because we can run these procedures sequentially (first A_1, then A_2, and so on).

Intra-agent adaptivity. An algorithm is *intra-agent nonadaptive* if the queries asked to voter i do not depend on responses to previous queries to that agent.

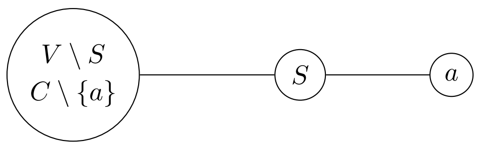
Full nonadaptivity. If an algorithm is inter- and intra-agent nonadaptive, we will call it *fully nonadaptive*. Such an algorithm can be implemented such that all queries are fixed upfront without depending on any of the responses.

3 t-Query Algorithms

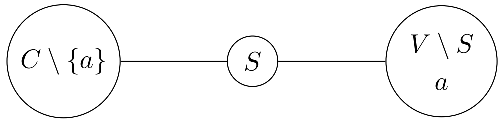
We begin by considering optimal distortion bounds for t-query algorithms. First, we establish a lower bound for any algorithm. Then, we present an algorithm that not only matches this lower bound but is also single-pass.

3.1 Lower Bound

We start by establishing a lower bound on distortion when each agent can answer up to t queries without any other restrictions on the algorithm.



(a) Metric for the case where the algorithm does not return a.



(b) Metric for the case where the algorithm returns a.

Figure 1: Metrics constructed for the proof of Theorem 1.

Theorem 1. *All t-query algorithms incur a distortion of at least* $\Omega(m/t)$.

In fact, we show an even stronger result. This holds for any algorithm that makes at most tn queries in total *regardless* of how they are distributed among the agents. Any algorithm making t queries per voter trivially makes at most tn queries.

Proof. Let $R = \{a_1 \succ_{i_1} b_1, a_2 \succ_{i_2} b_2, \ldots\}$ with $|R| \leq tn$ be an arbitrary sequence of responses the algorithm has queried and learned, and let c be the candidate returned given these responses. Since there are at most $2|R| \leq 2tn$ candidate appearances in $|R|$, there exists a candidate $a \in C$ that appears in less than $2tn/m$ many queries. Let $S \subseteq V$ be the set of voters that were ever queried on a. We have that $|S| \leq 2tn/m$.

We do a case analysis based on $c = a$ or $c \neq a$. The metrics are constructed in Figure 1, where the distance function is defined by the path length. First, we show that there are profiles for both metrics that are consistent with the responses in R. In both metrics, voters in S are equidistant from all candidates, and voters in $V \setminus S$ are equidistant from all candidates except a. We construct the profile as follows. We choose rankings for voters in S in an arbitrary way consistent with R. For the voters in $V \setminus S$, depending on the case, we either place a at the top or bottom of their ranking and complete the rest of the ranking arbitrarily in a way consistent with R. Since no voter in $V \setminus S$ was queried on a, this is always possible, and by the equal distances, these profiles are consistent with the metrics.

Case $c = a$. Consider the metric in Figure 1a. Note that $\text{cost}(a) = 2(|V| - |S|) + |S| = 2n - |S|$. For any candidate $c' \in C \setminus \{a\}$, $\text{cost}(c') = |S|$. Therefore,

$$\text{dist}(c) \geq \frac{\text{cost}(c)}{\text{cost}(c')} = \frac{2n - |S|}{|S|} \geq \frac{m}{t} - 1.$$

Case $c \neq a$. Consider the metric in Figure 1b. Note that $\text{cost}(a) = |S|$ and, for $c \neq a$, $\text{cost}(c) = |S| + 2(|V| - |S|)$. As in the previous case, we get a lower bound of $m/t - 1$. □

3.2 A Matching Upper Bound

Next, we consider designing algorithms to match this lower bound. Recall that using $O(m \log m)$ queries, it is possible to learn an agent's entire ranking. Hence, if t is sufficiently large ($\Omega(m \log m)$), we can simply learn all of the agents' complete rankings and then run any constant-distortion rule

Algorithm 1 Single-Pass Plurality Veto

```
1: Initialize $\text{plu}(c) = 0$
2: for each voter $i$ among the first $\lceil |V|/2 \rceil$ voters do
3:     $c_i \leftarrow \text{top}(\sigma_i, C_{\text{active}})$
4:     $\text{plu}(c_i) \leftarrow \text{plu}(c_i) + 1$
5: for each voter $i$ among the next $\lceil |V|/2 \rceil - 1$ voters do
6:     $C_{\text{active}} \leftarrow \{c \in C \mid \text{plu}(c) \geq 1\}$
7:     $c_i' \leftarrow \text{bottom}(\sigma_i, C_{\text{active}})$
8:     $\text{plu}(c_i') \leftarrow \text{plu}(c_i') - 1$
   return the unique candidate $a$ with $\text{plu}(c) \geq 1$.
```

Algorithm 2 t-Query Single-Pass Adaptive

```
1: $R \leftarrow \lceil (m-1)/t \rceil$;  $C_{\text{active}} \leftarrow C$
2: for each round $r = 1$ to $R$ do
3:     $C_r \leftarrow \min\{t+1, |C_{\text{active}}|\}$ candidates from $C_{\text{active}}$
4:     $V_r \leftarrow$ the next $\lfloor n/R \rfloor$ voters
5:     $c_r^* \leftarrow$ Algorithm 1($C \leftarrow C_r, V \leftarrow V_r$)
6:     $C_{\text{active}} \leftarrow (C_{\text{active}} \setminus C_r) \cup \{c_r^*\}$
   return the unique candidate $a \in C_{\text{active}}$
```

such as *Copeland* [Anshelevich *et al.*, 2015] or *Plurality Veto* [Kizilkaya and Kempe, 2022].

In fact, Plurality Veto can be implemented using $O(m)$ queries. As this will be useful to our later analysis, we describe the rule and query implementation in more detail. Plurality Veto runs in two phases. In the first, it determines the *plurality score* of each candidate, i.e., the number of voters that ranks that candidate first. The second begins with each candidate having a number of points equal to their plurality score. It sequentially goes through the voters (in an arbitrary order), and for each, asks the voter their *least favorite* candidate who still has a nonzero number of points; it then decrements that candidate by one point. The winning candidate is the last one remaining when the final voter is asked to decrement.[4]

Note that both determining a favorite or least favorite candidate among a set S requires $|S|-1$ pairwise queries. Hence, this can be implemented by asking each voter at most $2m - 2$ pairwise queries. Therefore, when $t \geq 2m-2$, Plurality Veto has optimal distortion.

However, there are still two drawbacks to this result. First, even though Plurality Veto requires only $O(m)$ queries, it needs to do this in two phases. Hence, its naïve implementation requires two passes. We may wonder if the same can be done with a single-pass algorithm. Second, and more important, this gives us no insight in how to handle $t < 2m - 2$, a crucial case for many applications. Our next result handles both of these matters, devising a single-pass algorithm that yields asymptotically optimal distortion for all t.

Theorem 2. *For $n \geq m/t$ and $t \in [m-1]$, Algorithm 2 is a single-pass, t-query algorithm achieving a distortion of $O(m/t)$.*

[4]Note that the total number of points given out is n, and each voter decreases by one. Hence, when the final voter is asked, exactly one candidate will remain with a single point.

The proof of the theorem appears later in this section. To better illustrate the ideas, we first present Algorithm 1 for the special case of $t = m - 1$. This is essentially a variant of Plurality Veto, which runs in a single pass. Next, by building on these ideas and using Algorithm 1 as a subroutine, we solve the problem for all $t \in [m]$.

Warm-up Case: $t = m - 1$

Algorithm Description. Algorithm 1 uses two series of adaptive queries:

- $\mathrm{top}(\sigma_i, C')$: A series of adaptive queries that finds the most preferred candidate of voter i among a subset of candidates $C' \subseteq C$. This can be done using $|C'| - 1$ queries (compare the first two candidates, then compare the preferred one with the third candidate, and so on).
- $\mathrm{bottom}(\sigma_i, C')$: Similar to above but for the least preferred alternative of voter i among $C' \subseteq C$.

Algorithm 1 accumulates plurality scores of the first $\lceil |V|/2 \rceil$ voters in $\mathrm{plu}(c)$ (lines 2–4). Then, for each of the second $\lceil |V|/2 \rceil - 1$ voters, among the remaining candidates with positive plurality score C_{active}, the voter indicates her least preferred candidate c'_i by decreasing the candidate's plurality score by one (lines 5-8). Since the number of increments is exactly one more than the number of decrements, only one candidate remains with a positive score, which subsequently is returned.

Distortion Analysis. To prove the distortion bound, we need the following technical definition and lemmas.

Definition 1 (**(a, b)-path**). *Fix two candidates $a, b \in C$. A sequence of distinct voters $v_1, \ldots, v_k$ is called a* voter path from a to b *(or an (a, b)-path for short) if there exists a sequence of $k + 1$ not necessarily distinct candidates beginning with a and ending with b, $a = c_0, c_1, \ldots, c_{k-1}, c_k = b$ such that, for each $j \in [k]$, $c_{j-1} \succ_{i_j} c_j$. Slightly abusing notation, we will refer to a set of voters $V' \subseteq V$ as an (a, b)-path if some order of V' is an (a, b)-path. We will say two (a, b)-paths are disjoint if they do not share any voters.*

The following is essential to the distortion analysis of Algorithms 1 and 2. The proof appears in Appendix A.

Lemma 1. *Fix $a, b \in C$. Suppose there exist ℓ disjoint (a, b)-paths. Then, $\frac{cost(a)}{cost(b)} \leq \frac{2n}{\ell} + 1$.*

The definition of (a, b)-paths and the result of Lemma 1 are similar to the definition of chains of preferences and Corollary 5.3 of Kempe [2020a]. The key difference is that we require the voters along each path to be distinct, which allows us to prove stronger properties.

Next, using Lemma 1, we show that there are $\lceil n/2 \rceil$ disjoint voter paths from the candidate returned by Algorithm 1 to each other candidate.

Lemma 2. *There exist $\lceil |V|/2 \rceil$ disjoint voter paths from the candidate selected by Algorithm 1 to any other candidate.*

Proof. Let a be the candidate returned by Algorithm 1 and $b \in C \setminus \{a\}$ be an arbitrary candidate. Let V_1 be the first $\lceil n/2 \rceil$ voters initializing the plurality scores and V_2 be the second set of voters decreasing the scores. Note that $|V_1| - 1 = |V_2|$. Create a one-to-one function $g : V_2 \to V_1$ as follows. As per the algorithm, for $i \in V_1$, c_i is the most favorite candidate for i; and, for $i \in V_2$, c'_i is the candidate indicated by i as her least favorite among the still active candidates (line 7). Map i to a corresponding voter $g(i)$ increasing the plurality score of c'_i. This way, g is injective and $c'_i = c_{g(i)}$.

Next, we show there exist $\lceil n/2 \rceil$ disjoint voter paths from a to b. For all $i \in V_2$, either $c'_i = a$ or $a \succ_i c'_i$. This holds because, since a remains active until the end, both a and c'_i had positive score, and i indicates c'_i as her least preferred alternative among the active set. The following constitutes $\lceil n/2 \rceil$ voter paths from a to b.

- If $c'_i = b$, then $a \succ_i b$ and there is voter path from a to b on $\{i\}$.
- If $c'_i \neq b$, we have $a \succ_i c'_i$ and $c'_i = c_{g(i)} \succ_{g(i)} b$. The latter holds because $c_{g(i)}$ is $g(i)$'s most preferred alternative. Therefore, there is a voter path from a to b on $\{i, g(i)\}$.
- Lastly, take the single voter $i^* \in V_1 \setminus \{g(i') \mid i' \in V_2\}$ who does not appear in the range of f. Her top vote must have been a as a is the only candidate with positive score in the end. Thus, $a \succ_{i^*} b$ and there is a voter path from a to b on $\{i^*\}$.

The above constructs $|V_2| + |\{i^*\}| = \lceil |V|/2 \rceil$ disjoint voter paths from a to b. ☐

As an aside, note that if we directly apply Lemma 1 to Lemma 2, this shows that Algorithm 1 achieves distortion 5.

General Case: $t \in [m - 1]$

Now, we build on Algorithm 1 and design a single-pass t-query algorithm that achieves an optimal distortion of $O(m/t)$.

Algorithm Description. Algorithm 2 runs in $R = \lceil (m - 1)/t \rceil$ rounds. In each round $r \in [R]$, it takes up to $t + 1$ candidates from the active candidate set C_{active}, denoted C_r. It then runs Algorithm 1 with a new set of $\lfloor n/R \rfloor$ voters, denoted V_r, along with the candidates in C_r, which return a candidate $c^*_r \in C_r$. Then, Algorithm 2 removes all candidates in C_r from the active candidate set C_{active} except for c^*_r. Because of this removal procedure, in each of the first $R - 1$ rounds, it eliminates t candidates from the active set. In the final round, there are at most $m - (R - 1)t \leq 1 + Rt - (R - 1)t \leq t + 1$ candidates remaining, from which a single winning candidate is kept in the active set and subsequently returned.

Distortion Analysis. To utilize Lemma 1 similar to that in Lemma 2, we find a sufficient number of disjoint voter paths from the candidate returned by Algorithm 2 to each of the other candidates.

Proof of Theorem 2. Let a be the candidate returned by Algorithm 2 and $b \in C \setminus \{a\}$ be an arbitrary candidate.

Consider a directed graph G over candidates described as follows. At each round r, draw a directed edge from the winning candidate c^*_r to all candidates $C_r \setminus \{c^*_r\}$. Note that each candidate $c \neq a$ has exactly one incoming edge, i.e., if c is eliminated at round r, the directed edge $c^*_r \to c$

is in G. Thus, G forms a directed rooted tree at a. Let $a = c_0 \to \ldots \to c_k = b$ be the unique directed path in G from a to b. For all $j \in [k]$, the edge $c_{j-1} \to c_j$ is created in a different round r_j. By Lemma 2, there are $\ell := \lceil \lfloor n/R \rfloor / 2 \rceil$ voter paths from c_{j-1} to c_j on a disjoint set of voters V_{r_j}. Note that V_{r_j}'s are disjoint as the algorithm selects a new set of voters at each round. Thus, we can extend the ℓ paths from c_0 to c_1 with the ℓ paths from c_1 to c_2 by arbitrarily matching those two sets of paths and get ℓ disjoint paths from c_0 to c_2. Then, we extend the paths from $c_0 = a$ to c_3 and so on to $c_k = b$. Therefore, by Lemma 1 and that this holds for all $b \in C \setminus \{c\}$ we have

$$\begin{aligned}\mathrm{dist}(a) &\le \max_{b\in C} \frac{\mathrm{cost}(a)}{\mathrm{cost}(b)} \le \frac{2n}{\lceil \ell/2 \rceil} + 1 \le \frac{2n}{\lfloor n/R \rfloor / 2} + 1 \\ &\le \frac{4n}{n/2R} + 1 = 8 \left\lceil \frac{m-1}{t} \right\rceil + 1,\end{aligned}$$

where in the last inequality we used $n \ge R = \lceil \frac{m-1}{t} \rceil$ and that $\lfloor x \rfloor \ge \frac{x}{2}$ for $x \ge 1$. ☐

4 Algorithms with Adaptivity Constraints

We now turn to varying restrictions on algorithm adaptivity. We begin with a tight analysis of the fully nonadaptive setting (Section 4.1). Then, we relax the constraints by requiring only one of inter- and intra-agent nonadaptivity at a time (Sections 4.2 and 4.3).

Note that when algorithms are intra-agent nonadaptive (Sections 4.1 and 4.3), we can no longer learn each agent's full ranking with $O(m \log m)$ queries. To do so, we instead have to ask about all $\binom{m}{2} = \Theta(m^2)$ possible pairwise comparisons. Hence, even for $t = \omega(m \log m)$, we may not be able to achieve constant distortion. Furthermore, recall that any inter-agent nonadaptive algorithms can trivially be implemented to be single-pass.

4.1 Fully Nonadaptive Algorithms

We show that the optimal distortion value for t-query fully nonadaptive algorithms is $\Theta(m^2/t)$. This is a factor of m worse than the unrestricted setting.

Theorem 3. *All fully nonadaptive t-query algorithms incur a distortion of at least $\Omega(m^2/t)$. Furthermore, for $n \ge \binom{m}{2}/t$ and $t \le \binom{m}{2}$, there exists a fully nonadaptive t-query algorithm matching this bound with a distortion of $O(m^2/t)$.*

We defer the $\Omega(m^2/t)$ lower bound to Appendix B. Below, we outline the algorithm and then analyze its distortion.

Algorithm Description. There are $\binom{m}{2}$ pairs of candidates. The algorithm distributes queries among voters as equally is possible in a way that each pair of candidates $(c, c') \in \binom{C}{2}$ is given to at least $\lfloor nt/\binom{m}{2} \rfloor$ voters. This can be done in various ways. For instance, concatenate $\lceil nt/\binom{m}{2} \rceil$ copies of the list of all $\binom{m}{2}$ pairs and assign the ith voter the pairs in range $[(i-1)t+1, (i+1)t]$. Such a distribution can be determined beforehand and, thus, is fully nonadaptive.

The algorithm aggregates the results as follows. For each pair $c, c' \in C$, draw a directed edge from c to c' if a majority of voters queried on $\{c, c'\}$ preferred c to c' (in the case of a tie, pick an arbitrary direction). Note that this is now a *tournament* graph, where between every two nodes, there is one directed edge. Finally, from this graph, select a *king* vertex a. A vertex is a king if it reaches all other vertices by at most two edges. In other words, a is a king if for all other candidates $b \in C \setminus \{a\}$, either a has an edge to b or there exists another candidate c such that a has an edge to c and c has an edge to b. It is well-known that king vertices exist in tournament graphs (e.g., West [2001]).

Distortion Analysis. To prove Theorem 3, we utilize the following lemma derived in Kempe [2020a] (a special case of Corollary 5.3).

Lemma 3 ([Kempe, 2020a])**.** *If at least ℓ voters prefer a to b, then $\frac{cost(a)}{cost(b)} \le \frac{2n}{\ell} - 1$. Furthermore, if there is a candidate $c \in C \setminus \{a, b\}$ such that f voters prefer a to c, and also f voters prefer b to c, then $\frac{cost(a)}{cost(b)} \le \frac{2n}{\ell} + 1$.*

Proof of Theorem 3. Let a be the king vertex in the pairwise-majority graph described above. Fix a candidate $b \in C \setminus \{a\}$. Since a is a king vertex, there is a path of length at most 2 from a to b. Each edge indicates the existence of $\lceil \frac{1}{2} \lfloor nt/\binom{m}{2} \rfloor \rceil$ voters preferring the candidate on the tail of the edge to the candidate on the head. By Lemma 3 and that $n \ge \binom{m}{2}$, we have $\mathrm{dist}(a) \le 4m^2/t$. ☐

4.2 Inter-Agent Nonadaptive Algorithms

Moving away from the most restricted case of full nonadaptivity, we find that by allowing intra-agent adaptivity, we can design algorithms that achieve better distortion by a factor of roughly $\widetilde{O}(t)$ (up to logarithmic factors). Only proof sketches are given. The complete proofs of Theorems 4 and 5 can be found in Appendices C and D, respectively. Our upper bound builds on the ideas from the fully nonadaptive setting.

Theorem 4. *For $n \ge m^2 \log^2 / t^2$, there exists an inter-agent nonadpative t-query algorithm achieving a distortion of $O(m^2 \log^2 t / t^2)$.*

Proof Sketch. The key idea of the algorithm from the fully nonadaptive case was to have a sufficient number of pairwise comparisons between all pairs. Exploiting intra-agent adaptivity, for each voter, we can sort ℓ candidates based on her preferences using $O(\ell \log \ell)$ queries. From a sorted list of ℓ candidates, we can infer $\binom{\ell}{2}$ pairwise comparisons — candidates appearing earlier are preferred to the ones appearing later. This is significantly stronger than the one learned comparison per query in the fully nonadaptive setting. Using this observation, we query voters to sort different sets of ℓ candidates. Similar to Theorem 3, we select a king vertex from the majority pairwise comparison graph. ☐

We complement the above theorem by showing its optimality up to logarithmic factors with a surprisingly-intricate lower bound.

Theorem 5. *All inter-agent nonadaptive algorithms have distortion $\Omega(m^2/t^2)$.*

Proof Sketch. Recall query choices of an intra-agent nonadaptive algorithm $\mathcal{A}$ can be decomposed into n subroutines $A_1, \ldots, A_n$, one for each agent, that chooses the next query based on previous responses of that agent. The proof incorporates ideas from showing lower bounds on comparison-based sorting algorithms, namely, representing these algorithms as binary decision trees. Each node corresponds to a query, and the two branches correspond to the two possible responses. With this representation, we are able to show that, for many pairs of candidates $\{a, b\}$, each algorithm A_i will find two rankings of the form $\sigma^1 = a \succ b \succ \cdots$ and $\sigma^2 = b \succ a \succ \cdots$ indistinguishable, i.e., they will both be asked the same queries and receive the same responses.

From this, we can find a single pair $\{a, b\}$ such that this property holds for a large fraction of the agents. The algorithm will be unable to distinguish the case of a large fraction of agents having a as their first choice and b as their second vs b as their first choice and a as their second. Since it must make the same choice in these two instances, one of these must be suboptimal. The final piece is to construct a simple metric in which choosing the "wrong" candidate on one of these profiles leads to large distortion. $\square$

4.3 Intra-Agent Nonadaptive Algorithms

We now turn to designing t-query algorithms that are *intra-agent* nonadaptive, i.e., each voter is asked up to t pairwise comparison queries all at once and returns her answers to all the t queries together. Its distortion guarantee is as follows.

Theorem 6. *For $n \geq m/t + \lceil \log_2 t \rceil$ and $t \in [m-1]$, Algorithm 3 is a single-pass, intra-agent nonadaptive, t-query algorithm that achieves a distortion of $O(m/t + \log m)$.*

Algorithm Description. Algorithm 3 runs in R rounds (for R that will be determined later) querying an equal number of $\lfloor n/R \rfloor$ new voters in each. For round $r \in [R]$, the corresponding group of voters are queried on ℓ_r disjoint pairs of candidates, arbitrarily selected from the active candidate set C_{active}. For each of the ℓ_r pairs, the candidate losing the majority of the $\lfloor n/R \rfloor$ votes is eliminated from C_{active}. This process goes on until one candidate remains, which is subsequently returned.

Since we want at most t queries made to each voter, in each round, we can ask about at most t pairs (i.e., $\ell_r \leq t$). While $|C_{\text{active}}| \geq 2t$, there are enough candidates such that the algorithm can, in fact, make all t queries (i.e., $\ell_r = t$). However, after $\lfloor m/t \rfloor - 1$ rounds, fewer than $2t$ candidates will remain. Then, the algorithm pairs up as many candidates as possible in C_{active} (one candidate will be unpaired when $|C_{\text{active}}|$ is odd), and $\ell_r = \lfloor |C_{\text{active}}|/2 \rfloor$ candidates are eliminated. This can go on for at most $\lceil \log_2 2t \rceil \leq \lceil \log_2 t \rceil + 1$ many rounds until a single candidate remains. Hence, there will be at most $\lfloor m/t \rfloor + \lceil \log_2 t \rceil$ rounds.

To prove the distortion bound, we again utilize Lemma 1 by finding a sufficient number of disjoint voter paths from the chosen candidate to the other candidates. The proof can be found in Appendix E.

While the lower bound of Theorem 1 shows the optimality of the above result for $t = O(m/\log m)$, it is unclear what the best achievable distortion is for $t = \omega(m/\log m)$.

Algorithm 3 t-Query Single-Pass Intra-agent Nonadaptive

```
1: $R \leftarrow \lfloor m/t \rfloor + \lceil \log_2 t \rceil$; $C_{\text{active}} \leftarrow C$
2: **for** each round $r = 1$ to $R$ **do**
3:     $\ell_r \leftarrow \min\{t, \lfloor |C_{\text{active}}|/2 \rfloor\}$
4:     Let $C_r \leftarrow \{(a_{r,1}, b_{r,1}), (a_{r,2}, b_{r,2}), \ldots, (a_{r,\ell_r}, b_{r,\ell_r})\}$ be $\ell_r$ pairs of distinct candidates from $C_{\text{active}}$
5:     **for** each voter $i$ among the next $\lfloor n/R \rfloor$ voters **do**
6:         Present the $\ell_r$ pairs $(a_{r,1}, b_{r,1}), \ldots, (a_{r,\ell_r}, b_{r,\ell_r})$
7:     **for** each pair in $\{(a_j, b_j) \mid j \in [\ell_r]\}$ **do**
8:         Let $c_{r,j}$ be the candidate losing the majority of comparisons (ties broken arbitrarily)
9:         $C_{\text{active}} \leftarrow C_{\text{active}} \setminus \{c_{r,j}\}$
   **return** the unique candidate $a \in C_{\text{active}}$
```

Theorem 6 continues to give a $O(\log m)$ upper bound, but the lower bound becomes $O(m/t)$, and only constant once $t = \Omega(m)$. For $t = \Omega(m^2)$, we know it is possible to achieve $O(1)$ distortion because, at this point, it is possible to learn all agents' complete rankings and run any of the well-known constant-distortion rules. However, we leave it as an interesting open question to close this gap for intermediate values of t, or find the minimum t with constant distortion.

5 Discussion

To summarize, we have studied the design of a limited number of pairwise queries to optimize metric distortion. We considered a variety of constraints on our algorithms regarding both the rounds of interactions and the dependency of queries on previous answers. In all cases, we provide either tight or nearly tight upper and lower bounds. In the case of inter- and intra-agent nonadaptivety, there remain logarithmic gaps.

As for follow-up work, the most direct would be to close these gaps. However, more broadly, while requiring few rounds of interaction and both inter- and intra-agent nonadaptivity capture an interesting set of constraints on the algorithms, there are a plethora of reasonable notions one could consider. For example, we may wish for the algorithm to be *anonymous*, demanding that each agent's queries depend only on their ranking, σ_i, rather than their identity. An even stricter property is to require that all agents receive the *exact* same queries, ensuring a strong notion of equality in how different agents interact with the system. We leave optimizing distortion in these settings and the development of new desirable properties as intriguing directions for future work.

Lastly, in certain applications, there can be such an enormous number of alternatives that our bounds become completely impractical. For instance, in variations of Polis, the domain of alternatives can be extended beyond just submitted comments to *any* statement from a natural language, a seemingly limitless domain. In such cases, when it is impossible to ensure that each candidate is queried by at least one voter, achieving bounded distortion is hopeless. We need to either (i) make additional assumptions about the domain of alternatives (say, they arise from a low-dimensional feature space), (ii) consider queries that are more powerful than pairwise comparisons, or (iii) use other measures of efficiency. We again leave these as exciting directions for further work.

References

G. Amanatidis, G. Birmpas, A. Filos-Ratsikas, and A. A. Voudouris. Peeking behind the ordinal curtain: Improving distortion via cardinal queries. *Artificial Intelligence*, 296:103488, 2021.

I. Anagnostides, D. Fotakis, and P. Patsilinakos. Metric-distortion bounds under limited information. *Journal of Artificial Intelligence Research*, 74:1449–1483, 2022.

E. Anshelevich, O. Bhardwaj, and J. Postl. Approximating optimal social choice under metric preferences. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, pages 777–783, 2015.

E. Anshelevich, O. Bhardwaj, E. Elkind, J. Postl, and P. Skowron. Approximating optimal social choice under metric preferences. *Artificial Intelligence*, 264:27–51, 2018.

E. Anshelevich, A. Filos-Ratsikas, N. Shah, and A. A. Voudouris. Distortion in social choice problems: The first 15 years and beyond. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4294–4301, 2021. Survey Track.

F. Brandt, V. Conitzer, U. Endriss, J. Lang, and A. D. Procaccia, editors. *Handbook of Computational Social Choice*. Cambridge University Press, 2016.

V. Conitzer and T. Sandholm. Communication complexity of common voting rules. In *Proceedings of the 6th ACM Conference on Economics and Computation (EC)*, pages 78–87, 2005.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.

V. Gkatzelis, D. Halpern, and N. Shah. Resolving the optimal metric distortion conjecture. In *Proceedings of the 61st Symposium on Foundations of Computer Science (FOCS)*, pages 1427–1438, 2020.

A. Goel, A. K. Krishnaswamy, and K. Munagala. Metric distortion of social choice rules: Lower bounds and fairness properties. In *Proceedings of the 18th ACM Conference on Economics and Computation (EC)*, pages 287–304, 2017.

D. Kempe. An analysis framework for metric voting based on LP duality. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, pages 2079–2086, 2020.

D. Kempe. Communication, distortion, and randomness in metric voting. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*, pages 2087–2094, 2020.

F. E. Kizilkaya and D. Kempe. Plurality veto: A simple voting rule achieving optimal metric distortion. In *Proceedings of the 31th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 349–355, 2022.

F. E. Kizilkaya and D. Kempe. Generalized veto core and a practical voting rule with optimal metric distortion. In *Proceedings of the 24th ACM Conference on Economics and Computation (EC)*, page 913–936, 2023.

T. Lu and C. Boutilier. Robust approximation and incremental elicitation in voting protocols. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 287–293, 2011.

D. Mandal, A. D. Procaccia, N. Shah, and D. P. Woodruff. Efficient and thrifty voting by any means necessary. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 7178–7189, 2019.

D. Mandal, N. Shah, and D. P. Woodruff. Optimal communication-distortion tradeoff in voting. In *Proceedings of the 21st ACM Conference on Economics and Computation (EC)*, pages 795–813, 2020.

K. Munagala and K. Wang. Improved metric distortion for deterministic social choice rules. In *Proceedings of the 20th ACM Conference on Economics and Computation (EC)*, pages 245–262, 2019.

A. D. Procaccia and J. S. Rosenschein. The distortion of cardinal preferences in voting. In *Proceedings of the 10th International Workshop on Cooperative Information Agents (CIA)*, pages 317–331, 2006.

P. Skowron and E. Elkind. Social choice under metric preferences: scoring rules and STV. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, pages 706–712, 2017.

C. Small, M. Bjorkegren, T. Erkkilä, L. Shaw, and C. Megill. Polis: Scaling deliberation by mapping high dimensional opinion spaces. *Revista De Pensament I Anàlisi*, 26(2), 2021.

D. B. West. *Introduction to Graph Theory*, volume 2. Prentice Hall Upper Saddle River, 2001.

Appendix

A Proof of Lemma 1

Fix $a, b \in C$. We first show a useful property about (a, b)-paths. Namely, if V' is an (a, b)-path, then,

$$d(a,b) \leq 2 \cdot \sum_{i \in V'} d(i,b).$$

Indeed, let $k := |V'|$, $i_1, \dots, i_k$ be the order of agents in V', and $a = c_0, \dots, c_k = bbe$ be the corresponding candidate sequence. The inequality follows from the following:

$$\begin{aligned}
d(a,b) &= d(c_0, c_k) + \sum_{j=1}^{k} \left(d(c_j, c_k) - d(c_j, c_k)\right) \\
&= \sum_{j=1}^{k} \left(d(c_{j-1}, c_k) - d(c_j, c_k)\right) + d(c_k, c_k) \\
&= \sum_{j=1}^{k} \left(d(c_{j-1}, c_k) - d(c_j, c_k)\right) && (d(c_k, c_k) = 0) \\
&\leq \sum_{j=1}^{k} \left(d(c_{j-1}, i_j) + d(i_j, c_k) - d(c_j, c_k)\right) && \text{(Triangle inequality)} \\
&\leq \sum_{j=1}^{k} \left(d(c_j, i_j) + d(i_j, c_k) - d(c_j, c_k)\right) && (c_{j-1} \succ_{i_j} c_j) \\
&\leq \sum_{j=1}^{k} \left(d(i_j, c_k) + d(i_j, c_k)\right) && \text{(Triangle inequality)} \\
&= 2 \sum_{j=1}^{k} d(i_j, b).
\end{aligned}$$

With this, we are ready to prove the lemma. Let $V_1, \dots, V_\ell$ be the disjoint voter sets of the ℓ (a, b)-paths. By the disjointness of V_j's we have

$$\text{cost}(b) \geq \sum_{j=1}^{\ell} \sum_{i \in V_j} d(i,b) \geq \sum_{j=1}^{\ell} \frac{d(a,b)}{2} = \ell \cdot \frac{d(a,b)}{2},$$

where the second inequality follows from the derivation above. Using the triangle inequality, we have $\text{cost}(a) \leq \sum_{i \in V} d(i,b) + d(a,b) = \text{cost}(b) + n \cdot d(a,b)$. Hence,

$$\frac{\text{cost}(a)}{\text{cost}(b)} \leq \frac{\text{cost}(b) + n \cdot d(a,b)}{\text{cost}(b)} = 1 + n \cdot \frac{d(a,b)}{\text{cost}(b)} \leq 1 + n \cdot \frac{d(a,b)}{\ell \cdot d(a,b)/2} = \frac{2n}{\ell} + 1. \quad \square$$

B Proof of Theorem 3 (Lower Bound)

Fix a set of queries Q with $|Q| = nt$. Note that any nonadaptive queries must have a pair of candidates a, b that were queried together at most $nt/\binom{m}{2}$ times. Suppose that in response, whenever a is queried, it is preferred to all other candidates, and whenever b with $c \neq a$, it is preferred to c. Note that if the algorithm picks a candidate other than a, then the distortion is unbounded as this is consistent with all voters being distance 0 from a and distance 1 from all other candidates. Let S be the set of voters queried on both a and b. If a is chosen, then the metric space could be the one in Figure 1a, which results in distortion $\frac{2n}{|S|} - 1 = \Omega(m^2/t)$. $\square$

C Proof of Theorem 4

Take $\ell = \max\{\ell \mid (2\ell)\log_2(2\ell) \leq t\}$, so that we can sort a group of 2ℓ candidates using up to t queries. Partition the candidate set into $\lceil m/\ell \rceil$ groups of size $\leq \ell$ denoted by $\mathcal{C} = \{C_1, \ldots, C_{\lceil m/\ell \rceil}\}$. Divide the voters as equally as possible among pairs $(C_{j_1}, C_{j_2}) \in \binom{\mathcal{C}}{2}$, i.e., each pair of candidate groups is assigned at least $\lfloor n/\binom{\mathcal{C}}{2} \rfloor$ voters. The voters assigned to (C_{j_1}, C_{j_2}) will be queried to learn their full preference ranking over $C_{j_1} \cup C_{j_2}$. This way, all pairs of candidates $c, c' \in C$ are compared by at least $\lfloor n/\binom{|\mathcal{C}|}{2} \rfloor$ voters.

Similar to the algorithm for Theorem 3, we create the majority pairwise comparison graph and select a king node of this complete directed graph. Each edge of this graph indicates $\frac{1}{2} \cdot \lfloor n/\binom{|\mathcal{C}|}{2} \rfloor$ voters preferring one candidate to another. Thus, by Lemma 3, the king vertex of this graph achieves a distortion of at most

$$\begin{aligned}\frac{2n}{\frac{1}{2}\lfloor n/\binom{|\mathcal{C}|}{2} \rfloor} + 1 \leq 8\binom{|\mathcal{C}|}{2} + 1 \leq 8\left(\frac{m}{\ell}\right)^2 + 1 \\ \leq O\left(\frac{m^2 \max\{\log^2 t, 1\}}{t^2}\right).\end{aligned}$$

D Proof of Theorem 5

Fix an inter-agent nonadaptive, t-query algorithm $\mathcal{A}$. Without loss of generality, suppose that $\mathcal{A}$ always makes exactly t queries to each agent (if it does not, we can add arbitrary queries at the end and ignore the responses). Since the queries made by the algorithm to agent i can only depend on responses from i, we can decompose the query-making portion of $\mathcal{A}$ into n subroutines, $A_1, \ldots A_n$. The subroutine A_i decides which pair of candidates to query agent i on next, given their previous responses. We can represent each A_i as a binary decision tree with height $t+1$. Each internal node is labelled by a pair of candidates. Depending on the response of which candidate is preferred, either the left or right branch is followed.[5] Note that sometimes only one of the two branches can be induced by a ranking, e.g., if a certain response would lead to a cycle in the preferences. In such cases, a node has only one child.

Each ranking σ that agent i could have corresponds to a path in this tree from the root to a leaf, where, at each internal node, the branch is taken based on the comparison query response according to σ. Note that if two rankings σ and σ' follow the same path down the tree, they are indistinguishable according to A_i, i.e., they lead to the exact same queries asked and responses received.

We will say a pair of candidates $\{a, b\} \subseteq C$ *deceive* A_i if there are two rankings σ and σ' such that $\sigma(1) = \sigma'(2) = a$, $\sigma(2) = \sigma'(1) = b$, and σ and σ' follow the same path in A_i. In other words, there is a run of the algorithm A_i such that it is plausible that a is ranked first and b is second or that b is first and a is second, and these two scenarios are indistinguishable. The property that $\{a, b\} \subseteq C$ *deceive* A_i is equivalent to there exists a path in A_i such that neither a nor b lose in a pairwise comparison. Indeed, taking any ranking corresponding to this path and then moving a to the top and b second or a second and b to the top will lead to the exact same responses and correspond to the same path.

Let $D_i = \{\{a, b\} \mid \{a, b\} \text{ deceive } A_i\}$ be the set of pairs that deceive A_i and let $E_i = \overline{D_i}$ be its complement, i.e., the remaining pairs that *do not* deceive A_i. We will show that $|E_i| \leq t^2$, that is, there are at most t^2 pairs that do not deceive A_i.

Consider an arbitrary path down the decision tree in A_i. Let T be the set of candidates that lost a pairwise comparison at least once on this path. Since the path is of length t and each node results in one candidate losing, $|T| \leq t$. By the above definition, if $\{a, b\} \in E_i$, we have that $\{a, b\} \cap T \neq \emptyset$. In other words, every pair in E_i *must* contain at least one member of T. Indeed, if $\{a, b\} \cap T = \emptyset$, then this is a path in which neither lost a nor b lost. Let $E_i^{a^*} = \{b \mid \{a^*, b\} \in E_i\}$ be the set of candidates who are paired with a^* in E_i. The above implies that $|E_i| \leq \sum_{a^* \in T} |E_i^{a^*}|$.

Fix some $a^* \in T$. We next show that $|E_i^{a^*}| \leq t$. Combined with the above argument, this implies that $|E_i| \leq |T| \cdot t \leq t^2$, as needed. Let σ be an arbitrary ranking with $\sigma(a^*) = 1$, and consider the path induced by σ. Let T^{a^*} be the set of candidates who lose at least once on this path. We have that $a^* \notin T^{a^*}$ because a^* is ranked first in σ. Furthermore, note that if $b \in E_i^{a^*}$, then $b \in T^{a^*}$. Indeed, if not, the above path would have neither a^* nor b losing, contradicting that $\{a^*, b\} \in E_i$. This means that $E_i^{a^*} \subseteq T^{a^*}$. Finally, notice that $|T^{a^*}| \leq t$ because the path is of length t. This implies the desired bound.

Since $|E_i| \leq t^2$, and $|D_i| + |E_i| = \binom{m}{22}$, this implies that $|D_i| \geq \binom{m}{2} - t^2$. By an averaging argument, there must be a pair $\{a, b\}$ that deceives at least $\frac{\binom{m}{2} - t^2}{\binom{m}{2}} \cdot n = (1 - \frac{2t^2}{m(m-1)}) \cdot n$ of the algorithms A_i.

Fix such a set $\{a, b\}$. Let $V' = \{i \mid \{a, b\} \in D_i\}$ be the set of agents whose algorithms are deceived. By the above, we have that $|V'| \geq (1 - \frac{2t^2}{m(m-1)}) \cdot n$. For each $i \in V'$, let σ_i^{ab} and σ_i^{ba} be a pair of ranings indistinguishable to A_i such that $\sigma_i^{ab}(1) = \sigma_i^{ba}(2) = a$ and $\sigma_i^{ab}(2) = \sigma_i^{ba}(1) = b$.

[5] This is very similar to representations of comparison-based sorting algorithms used to prove lower bounds on query complexity.

We can now construct a pair of profiles $\boldsymbol{\sigma}^{ab}$ and $\boldsymbol{\sigma}^{ba}$ where, in the first, each $i \in V'$ has ranking σ_i^{ab}, and in the second, each $i \in V'$ has ranking σ_i^{ba}. The remaining voters $V \setminus V'$ will have arbitrary rankings (but the same in both profiles). Note that $\boldsymbol{\sigma}^{ab}$ and $\boldsymbol{\sigma}^{ba}$ lead to the exact same queries and responses. Hence, the chosen candidate must be the same on both. Therefore, for at least one of a and b was *not* chosen to be winner. Without loss of generality, suppose it was a.

We will consider the profile $\boldsymbol{\sigma}^{ab}$ along with the following metric. Each $i \in V'$ has $d(i, a) = 0$. For all remaining pairs of voters and candidates (i', c), we will have $d(i', c) = 1$. Note that these distances are consistent with $\boldsymbol{\sigma}^{ab}$. Furthermore, the cost for a is at most $|V \setminus V'| \leq \frac{2t^2}{m(m-1)} \cdot n$, while the cost for any other candidate is n. Therefore, the distortion on this instance is at least

$$\frac{n}{\frac{2t^2}{m(m-1)} \cdot n} = \Omega(\frac{m^2}{t^2}). \quad \square$$

E Proof of Theorem 6

Let a be the candidate returned by Algorithm 3 and $b \in C \setminus \{a\}$ be an arbitrary candidate. Create the elimination directed graph G as follows. For each candidate $c \neq a$, there is a candidate $g(c)$ that won over c in a pairwise majority comparison by $\lfloor n/R \rfloor$ voters; draw a directed edge from $g(c)$ to c. Since each candidate has a single incoming edge, G forms a directed tree rooted at a.

There is a unique path from $a = c_0 \to c_1 \ldots \to c_k = b$. For all $j \in [k]$, let V_j be the set of voters V_j who made the (c_{j-1}, c_j) comparison resulting in elimination of c_j. The sets V_j's are disjoint since c_{j-1} must be eliminated after c_j is eliminated (in an earlier round). Let $V_j' \subseteq V_j$ be the voters preferring c_{j-1} to c_j. For each $i \in V_j'$, there is a voter path from c_{j-1} to c_j on $\{i\}$. Hence, there are $|V_j'| \geq \lceil \lfloor n/R \rfloor / 2 \rceil$ many disjoint voter paths from c_{j-1} to c_j. We can connect the voter paths from c_0 to c_1 and c_1 to c_2 by arbitrarily matching and extending them to voter paths c_0 to c_2. We can continue to make voter paths from $c_0 = a$ to $c_k = b$. All of these paths consist of unique sets of voters. By Lemma 1 and that the above holds for all $b \in C$, we have

$$\text{dist}(a) = \max_{b \in C} \frac{\text{cost}(a)}{\text{cost}(b)} \leq \frac{2n}{\lceil \lfloor n/R \rfloor / 2 \rceil} + 1 \leq \frac{4n}{\lfloor \frac{n}{R} \rfloor} + 1 = 8R + 1 = O(m/t + \log t),$$

where the last inequality holds by $n \geq R = \lfloor m/t \rfloor + \lceil \log_2 t \rceil$ and $\lfloor x \rfloor \geq x/2$ for $x \geq 1$. $\square$