Promoting Fairness and Priority in k-Winners Selection Using IRV

Md Mouinul Islam
mi257@njit.edu
CS, NJIT
Newark, NJ, USA

Soroush Vahidi
sv96@njit.edu
CS, NJIT
Newark, NJ, USA

Baruch Schieber
sbar@njit.edu
CS, NJIT
Newark, NJ, USA

Senjuti Basu Roy
senjutib@njit.edu
CS, NJIT
Newark, NJ, USA

ABSTRACT

We investigate the problem of finding winner(s) given a large number of users' (voters') preferences casted as *ballots*, one from each of the m users, where each ballot is a ranked order of preference of up to ℓ out of n items (candidates). Given a group protected attribute with k different values and a priority that imposes a selection order among these groups, the goal is to satisfy the priority order and select a winner per group that is most representative. It is imperative that at times the original users' preferences may require further manipulation to meet these fairness and priority requirement. We consider manipulation by modifications and formalize *the margin finding problem under modification* problem. We study the suitability of Instant Run-off Voting (IRV) as a preference aggregation method and demonstrate its advantages over positional methods. We present a suite of technical results on the hardness of the problem, design algorithms with theoretical guarantees and further investigate efficiency opportunities. We present exhaustive experimental evaluations using multiple applications and large-scale datasets to demonstrate the effectiveness of IRV, and efficacy of our designed solutions qualitatively and scalability-wise.

CCS CONCEPTS

• **Information systems → Top-k retrieval in databases**.

KEYWORDS

Fairness; k-Winners Selection; Instant Runoff Voting

ACM Reference Format:
Md Mouinul Islam, Soroush Vahidi, Baruch Schieber, and Senjuti Basu Roy. 2024. Promoting Fairness and Priority in k-Winners Selection Using IRV. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671735

1 INTRODUCTION

The task of finding the winner, i.e., the most favorable item or candidate from a given set of m users' (voters') preferences over n items (candidates), has found a wide variety of applications such as in hiring candidate(s) for a job, selecting member(s) of a committee, finding winning candidate(s) in a competition, in electoral voting, or even in recommender systems. IRV (Instant Run-off Voting) is a ranked choice voting mechanism that has been gaining popularity lately as an electoral system in Australia, Ireland, and the U.S. [13, 15, 20, 25, 29, 31, 35]. In this paper, we study the applicability and computational implications of adapting IRV to preference data to enable group fairness while satisfying a priority order.

Preference data considering faculty hiring. Table 1 represents ranked order of up to top-5 preferences over 7 candidates who have applied to a faculty position. Preferences are provided by 10 committee members (voters). Each of these ranked orders of preferences constitutes a ballot.

Committee member	1st choice	2nd choice	3rd choice	4th choice	5th choice
Jack	Zoey	Mira			
Emma	Laura	Gina	Molly	Kim	Zoey
Monica	Zoey	Molly	Kim	Gina	Sara
Daniel	Zoey	Molly	Sara	Gina	
Max	Mira	Molly	Sara	Kim	Zoey
John	Sara	Gina	Kim	Zoey	
Amy	Gina	Sara	Kim	Mira	Zoey
Alice	Sara	Gina	Kim	Molly	Zoey
Bob	Kim	Gina	Sara	Molly	Zoey
Steve	Kim	Gina	Sara		

Table 1: Preferences over 7(n) candidates by 10 committee members(m) upto 5-th position (ℓ)

Fairness and priority order. Group fairness is studied considering the protected attribute of the candidates to ensure equal representation of each group [22, 33]. The example assumes that research area is one such protected attribute with $k = 3$ different values (it is not hard to extend this to race, gender, ethnicity, or any other protected attribute). The value of this attribute for each of the candidates and the priority among these values is given in Table 2.

Priority Order	Protected attribute (area)	Candidates
First	DM	Molly, Laura
Second	ML	Gina, Kim, Sara
Third	AI	Zoey, Mira

Table 2: Fairness and Priority Orders

Goal. The goal is to return one candidate per protected attribute group that is most representative of the committee members' preferences while obeying the priority order of the groups. In our example, we need to first select a Data Mining (DM) candidate, then a Machine Learning (ML) candidate, and finally an Artificial Intelligence (AI) candidate. Selections are made in the priority order, and if at

KDD '24, August 25–29, 2024, Barcelona, Spain
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0490-1/24/08.
https://doi.org/10.1145/3637528.3671735

any point there are no more available positions, no more selections are made. For instance, if there are 2 positions available and if both the selected DM and ML candidates accepted their offers, then an AI candidate will not be selected.

The IRV process. The IRV process [24, 30] is a multi-stage process [7] that simulates $n-1$ run-off rounds, where in each such round one item is eliminated. The single item that survived the eliminations after all rounds is the winner. More specifically, given the original preferences of the users (voters), an initial tally of the first choice votes of every candidate is performed in round 1. The item that has the lowest number of first choice votes is eliminated. Ties are broken arbitrarily. After the elimination, all the ranked orders that include the eliminated item are updated, and the items following this eliminated item in the ranked order are advanced one place up. This concludes round 1. This is iterated $n-1$ times, namely, the tally is recomputed, and the item that has the lowest number of first choice votes is eliminated, where ties are broken arbitrarily.

Using the running example, as shown in the left of Table 4, the IRV process eliminates *Molly* in round 1, *Mira* in round 2 (and the respective vote gets transferred to *Sara*), and *Gina* in round 3. This process continues further making *Sara* the winner after 6 rounds.

Motivation. The resurgence of IRV is motivated by a range of expected benefits, including, ensuring majority support for the winner, reducing conflict within the electorate, reducing strategic voting, and increasing diversity of the winners [30]. IRV is amenable to incomplete ranked order, making the process further suitable for applications where users are not obligated to provide full order. Multiple recent works [16], [10] have demonstrated the superiority of IRV over plurality voting [26], as well as positional voting mechanisms (such as Borda [19]) to promote proportional representation of solid coalition and anti-plurality. IRV is used in elections in Australia, Ireland, and several U.S. states [20, 31], demonstrating its practical utility and effectiveness. Additionally, IRV can enhance the diversity and fairness of recommendations in AI systems. Table 3 summarizes some of the advantages of IRV compared to other selection methods. Refer to Appendices A.1, A.2 for further details.

Method	Anti-plurality	Proportional representation	Suitable to incomplete order
Scoring based	×	×	×
Plurality	×	×	×
Positional	×	×	✓
IRV	✓	✓	✓

Table 3: Comparison of aggregation methods

IRV Margin computation. Recall that in our example the IRV process chooses *Sara* as the winner of the ballots. Clearly, *Sara* does not satisfy the priority order of selecting a DM candidate first. Hence, some ballot modifications are needed. If Jack's ballot in Table 1 is changed by replacing *Zoey* with *Molly*, a series of 6 run-off rounds are simulated after that, as listed in the right of Table 4, which makes *Molly* the winner. If instead *Laura* is to be made the winner, this will require at least 3 ballot modifications, for example, by replacing the top choice of Alice, Monica, and John with *Laura*. Intuitively too, *Molly* is a better choice because it is liked *as the second choice by* 3 *out of* 7 *original committee members*. A similar process must also be carried out for ML and AI independently. Our goal is to find the minimum ballot modification that results in an outcome that satisfies the k priority orders. We refer to this problem as IRV *margin computation* [26] to satisfy k priority orders (denoted by **MqKIRV** for $k \geq 1$ and **MqIRV** for $k = 1$). To the best of our knowledge, we are the first to initiate a principled study on this.

Candidate	Tally	R1	R2	R3	R4	R5	R6	New Tally	R1	R2	R3	R4	R5	R6
Zoey	3	3	3	3	3	4		2	2	2	2			
Sara	2	2	3	4	4	6	8	2	2	3	3	3	5	
Kim	2	2	2	2	3			2	2	2	2	2		
Laura	1	1	1	1				1	1	1				
Gina	1	1	1					1	1					
Mira	1	1						1						
Molly	0							1	2	2	3	5	5	7

Table 4: IRV rounds after ballot modification (left): *Sara* winner IRV rounds after ballot modification (right): *Molly* winner

Why ballot modification? An alternative to ballot modification could be the following – filter out candidates that do not satisfy the priority order (e.g., delete all ML and AI candidates for finding the top DM candidate) and run preference aggregation over the remaining ones. We note that such a filtering process could lead to undesired results. Imagine that there are 99 voters and n candidates, denoted $A, B, C_1, \ldots, C_{n-2}$, where candidates A and B belong to the group with the top priority and candidates $C_1, \ldots, C_{n-2}$ belong to another group. The preferences of 50 voters is $C_1 > \cdots > C_{n-2} > A$ and the preferences of the other 49 voters is $B > C_1 > \cdots > C_{n-2}$. An aggregation mechanism based on filtering will choose A as the winner for the top priority group and C_1 for the other group. On the other hand, modifying a single vote from $C_1 > \cdots > C_{n-2} > A$ to $B > C_1 > \cdots > C_{n-2}$ will result in choosing B and C_1, which seems to be a much better reflection of the voters' preferences.

We recognize the ethical concerns related to modifying votes to promote fairness. In our work, we argue that modifying inputs (votes) is more morally responsible than altering outputs (results). This is because changing outputs inherently modifies the original preferences. Our approach minimizes changes to the original preferences, making it a more responsible method. This approach aligns with the concept of preprocessing versus postprocessing in fair classification, which is widely accepted in the field. Prior work in this line has also been conducted in ranking and recommendation systems. References such as [10, 16, 26] support the ethical and practical advantages of our method. By adopting this approach, we ensure that the modifications are transparent, justified, and aimed at promoting equitable representation.

Technical Contributions (Sections 3 and 4). We make multiple technical contributions in terms of analyzing the studied problems as well as designing solutions for them. We prove that **MqIRV** is NP-hard, even when the ballot size is at most $\ell = 2$ by reducing an instance of the known NP-complete problem Exact Cover by 3-Sets (X3C) to an instance of **MqIRV**. Inspired by [13, 29] we then design an algorithmic framework AlgExact that gives an exact solution and considers all possible permutations of the candidates that end in a candidate that satisfies the priority order. Solving AlgExact thus requires repeatedly solving a subproblem DistTo, which, given a permutation, finds the smallest number of ballot modifications needed to ensure that the order of the candidates eliminated in $n-1$

run-off rounds of IRV follow this order. Unfortunately, we prove that the decision version of DistTo is NP-hard, even when $\ell = 3$, by reducing an instance of X3C to DistTo.

We further study efficiency opportunities of AlgExact by enabling early terminations via branch and bound. The idea is to avoid making expensive DistTo calls by computing a lower bound on the margin for every possible suffix of every permutation, and eliminating a permutation in its entirety if the lower bound on its margin is not smaller than the current upper bound on the margin of the **MqIRV** instance. To that end, we design a lower bound computation algorithm DistToLB and an upper bound computation algorithm **MqIRVUB** that are highly effective and computationally lightweight. We also study the DistTo problem under different preference manipulation models – for example, we study *how to only add the smallest number of ballots* to the existing set of ballots, such that the priority orders are satisfied. We refer to this as the DistToAdd problem. We present an efficient exact solution for the DistToAdd problem. We also present an integer programming formulation for **MqIRV** which is non-trivial. We finally design a highly scalable heuristics that is shown to work well in practice.

Experimental Evaluations (Section 5). Our final contribution is experimental – we use four real world large scale datasets motivated by different electoral voting and recommender systems applications, as well as one synthetically generated very large datasets. Our experimental evaluations have the following findings: **(a)** We empirically show that **MqIRV**results in a significantly smaller anti-plurality index [16] (i.e., it does not select candidates that are disliked by the majority of voters) compared to alternative approaches such as plurality voting [26] or Borda [19]. **(b)** We present an in-depth case study demonstrating that ballot modification results in a lower anti-plurality index compared to alternative approaches such as filtering. **(c)** We demonstrate that AlgExact is optimal, yet more scalable than existing solutions that could be adapted to our problem [29], [13]. **(d)** We empirically demonstrate the optimality of DistToAddAlg and its scalability, as well as the quality and scalability of our designed approximate solution by varying several pertinent parameters and comparing with appropriate additional baseline algorithms.

We present the discussion of related work in Section 6 and conclude in Section 7.

2 DATA MODEL & PROBLEM

In this section, we describe the data model, following which we formally define the problem and prove its hardness.

2.1 Data model

Ballot/preference. Preference of a user is elicited using a ballot b containing a ranking up to position at most ℓ, where c_i is the i-th preferred candidate. Using the running example, c_1 and c_5 are *Gina*, and *Zoey*, respectively of user *Amy's* ballot.

Ballot profiles. The data contains the preferences/ballots $\mathcal{B}$ of m users/voters over a set C of n items/candidates. Using the running example, $m = 10$, $n = 7$. The columns in Table 1 show $\mathcal{B}$.

Preference aggregation. A preference aggregation method $\mathcal{F}$ takes $\mathcal{B}$ as input and selects a winner from the candidates/items. Given fairness criteria and priority order, the goal is to make use of $\mathcal{B}$ and $\mathcal{F}$ multiple (k) times to select k different winners in the priority order. Table 2 shows $k = 3$ such constraints for recommending top DM, ML, and AI candidates. We use IRV as $\mathcal{F}$, as discussed in more detail in Section A.2.

Preference manipulation models. We consider two different preference manipulation models, where only the first one satisfies the number of ballot invariance property (i.e., the total number of votes remains unchanged) and is our primary focus in this work.

(1) **Manipulation by modification.** Given a ballot b with ranking up to position j ($j \leq \ell$) positions, update any number of entries in b considering candidates from C. As an example, Jack's ballot (see Table 1) is changed from *Zoey, Mira* to *Molly, Mira.* Note that changing Daniel's ballot from *Zoey, Molly, Sara, Gina* to *Mira, Kim* also constitutes to a single ballot modification.
(2) **Manipulation by addition.** Add a new ballot b with ranking of up to ℓ candidates from C.

Handling ties in IRV Recall that according to our definition ties during the IRV process are broken arbitrarily. It is not difficult to see that the way these ties are broken may impact the value of the margin. Indeed, in our example of ballot modification candidate *Molly* is the winner after just a single modification only in case the ties are broken in a very specific way. We postulate that any consistent choice would be effective in our case, since we use the margin to distinguish among choices and are not interested in the actual value of the margin.

2.2 Problem Definitions & Hardness

Problem Definition 1. MqIRV *(IRV Margin satisfying a single query constraint). Given a set of ballots $\mathcal{B}$ eliciting m voters ranked preferences of up to ℓ positions over a given set C of n candidates, and a query constraint that specifies a subset of the candidates, find the minimum number of ballots that need to be modified in order to ensure that the winner of the IRV election belongs to the subset specified in the query constraint.*

Running Example. Referring to Table 2, if the query constraint specifies selecting a DB candidate, then the minimum number of ballot modifications required to ensure that is 1, where *Zoey* in Jack's ballot is swapped by *Molly.* If instead *Laura* is to be made the winner, this will require 3 ballot modifications. Hence, the margin to satisfy the query constraint is 1.

Theorem 2.1. **MqIRV** *is NP-Complete, even when $\ell = 2$.*

Proof is given in Appendix A.3.

Problem Definition 2. MqKIRV *(IRV Margin satisfying k query constraints.) Given a set of ballots $\mathcal{B}$ eliciting m voters ranked preferences of up to ℓ positions over a given set C of n candidates, and a query with k constraints, each specifies a subset of the candidates, find the minimum number of ballots that need to be modified in order to ensure that the winners of k independent invocations of the IRV election (each starting from the original ballots) belong to the respective subsets specified k query constraints.*

Theorem 2.2. **MqKIRV** *is NP-Complete, even when $\ell = 2$.*

Proof. Follows trivially from Theorem 2.1. □

Running Example. Considering the running example again (Table 2), $k = 3$ and the ballots are shown in Table 1. The winner for DB is *Molly* (margin = 1), for ML it is *Sara* (margin = 0), for AI it is *Zoey* (margin = 1). The minimum number of ballot modifications (margin) required to ensure all three constraints is 1+0+1 = 2.

3 ALGORITHMS FOR MqIRV AND MqKIRV

In this section, we focus on designing exact solutions for **MqIRV** and **MqKIRV**. In Section 3.2 we discuss AlgExact, a branch-and-bound algorithm for **MqIRV** that is capable of effective pruning of the search space. In Section 3.3 we present a non-trivial integer programming formulation of **MqIRV**. These exact algorithms apply also exact algorithms for for **MqKIRV** as follows from the following simple theorem.

Theorem 3.1. *An optimal solution for* **MqKIRV** *corresponds to solving* **MqIRV** *optimally k times.*

3.1 Required Definitions

We first give some definitions that will be useful when discussing our algorithms.

Signature. Let $\mathcal{S}$ be the set of all possible partial or total rankings over C (including those that do not appear in $\mathcal{B}$). A signature $s \in \mathcal{S}$ is one such partial or total ranking. The total number of possible signatures is $|\mathcal{S}| = \sum_{x=1}^{\ell} \binom{n}{x} \cdot x!$. For example, both {*Molly, Sara*} and {*Zoey, Molly, Sara, Gina*} are valid signatures even though the former is not present in Table 1.

Tally $t_r(c)$ or first choice votes. The tally or first choice votes of a candidate c at round r, denoted as $t_r(c)$, is defined as the number of ballots in round r in which c is the first choice candidate. Using the running example, tally of *Sara*, *Zoey*, and *Kim* at the beginning of round 5 are: $t_5(Sara) = 4$, $t_5(Zoey) = 3$, and $t_5(Kim) = 3$.

3.2 AlgExact for MqIRV

We propose an algorithmic framework AlgExact that is an exact solution to the **MqIRV** problem. The algorithmic solution is developed by creating a branch and bound tree, akin to two prior works [13, 29].

For a given winner w, the solution considers all possible permutations of candidates that need to be eliminated (i.e., $(n-1)!$), where each permutation represents an elimination order simulating $n-1$ run-off rounds of IRV. The height of the tree is at most n. Each node of the tree contains two values: (a) an elimination order π, (b) a score that represents the number of ballot modifications to realize π (we formalize that as DistTo below). Each edge of the tree represents the next candidate to be eliminated. An artificial root node connects the branches of the subtree, where each subtree represents a $w \in W$ as the winner, where W is the constrained winner set specified by the query. Except for the fake root node, the relationship between any parent and child nodes in the tree is as follows: (i) At any parent node with elimination order π, the child node has elimination order $\pi' = c + \pi$, for some $c \in C - \pi$, and (ii) DistTo(π) $\leq$ DistTo(π') [29]. The leaf nodes end with a full permutation, where the last candidate is from W. The maximum number of possible leaf nodes is $= |W| \times (n-1)!$. AlgExact solves the sub-problem DistTo formalized below, repeatedly, at each node of the branch and bound tree.

Algorithm 1 AlgExact

```
Input: Ballot profile $\mathcal{B}$, set of Candidates $C$, set of preferred candidates $W$.
Output: MqIRV
1:  $ub = \infty$
2:  $lb = 0$
3:  initialize priority $queue$ with tuples $(w, 0)$ where $w \in W$
4:  while $queue$.notEmpty() do
5:      $\pi, lb = queue$.pop()
6:      for $c \in C \setminus \pi$ do
7:          $\pi' = c + \pi$
8:          $lb =$ DistToLB($\mathcal{B}, C, \pi'$)
9:          if $lb > ub$ then
10:             prune $\pi'$
11:         else
12:             $queue$.add($\pi', lb$)
13:         end if
14:         if $|\pi'| == n$ then
15:             $ub = \min(ub,$ DistTo($\mathcal{B}, C, \pi'$))
16:         end if
17:     end for
18: end while
19: MqIRV = $ub$
20: Return MqIRV
```

Problem Definition 3. DistTo. *Given an elimination order over the candidates π (complete or partial order, $|\pi| \leq |C|$) and a database of ballot profiles $\mathcal{B}$, find the minimum number of ballots that must be modified to achieve π.*

Theorem 3.2. DistTo *is NP-hard, even when $\ell = 3$.*

Proof omitted due to space constraints.

AlgExact explores the tree level by level (refer to Figure 9 in Appendix A.4) and makes an attempt to prune part of the tree based on the lower bound of a branch (which corresponds to an elimination order) and an upper bound of the value the **MqIRV** instance.

Definition 3.3. **Upper bound of an instance MqIRVUB.** Given an **MqIRV** instance, **MqIRVUB** is defined as an upper bound of the number of ballots that must be modified to satisfy the query constraint.

Definition 3.4. **Lower bound (**DistToLB**) of** DistTo(π)**.** Given an **MqIRV** instance and an elimination order π, DistToLB is a lower bound on the number of ballots that must be modified to achieve π, namely, DistToLB(π) $\leq$DistTo(π).

Running Example. Figure 9 shows one such partially constructed tree for our running example.The candidates are represented by their unique ids, and any red node and the sub-tree under them are fully pruned. Each such red node has DistToLB(π) that is not smaller than the **MqIRVUB** of the **MqIRV** instance (e.g., DistToLB($[1, 3, 5]$) = 4 is larger than **MqIRVUB** = 2). Compared to prior works [13, 29], we propose both effective as well as computationally efficient **MqIRVUB** and DistToLB solutions, as we discuss in Section 4.

3.3 IP for MqIRV

MqIRV can be formulated as an integer linear program (IP). The objective of the IP is to minimize the number of ballot modifications required to ensure that the winner is the preferred candidate. Next, we describe how to formulate this IP.

$$\min \sum_{s\in\mathcal{S}} a_s \ \text{ subject to}$$

$$
\begin{array}{lll}
m_s + a_s - d_s = y_s & \forall\, s \in \mathcal{S} & (1)\\
m \geq y_s \geq 0 & \forall\, s \in \mathcal{S} & (2)\\
m_s \geq d_s \geq 0 & \forall\, s \in \mathcal{S} & (3)\\
m - m_s \geq a_s \geq 0 & \forall\, s \in \mathcal{S} & (4)\\
\sum_{s\in\mathcal{S}} a_s = \sum_{s\in\mathcal{S}} d_s & & (5)\\
u_{c_i,c_j} + u_{c_j,c_i} = 1 & \forall\, \{c_i, c_j\} \subseteq C & (6)\\
u_{c_i,c_j} + u_{c_j,c_r} + u_{c_r,c_i} \geq 1 & \forall\, \{c_i, c_j, c_r\} \subseteq C & (7)\\
v_{s,c_i,\tilde{c}} = u_{c_i,\tilde{c}} \cdot \Pi_{x=1}^{i-1} u_{\tilde{c},c_x} & \forall\, s \in \mathcal{S}\ \forall\, \tilde{c} \in C & \\
 & \forall\, i \in \{1, \ldots, |s|\} & (8)\\
\sum_s (y_s \cdot v_{s,\hat{c},\hat{c}}) \geq u_{\hat{c},\tilde{c}} \cdot \sum_s (y_s \cdot v_{s,\tilde{c},\tilde{c}}) & \forall\, \{\hat{c}, \tilde{c}\} \subseteq C & (9)
\end{array}
$$

The IP for MqIRV

For each ballot signature $s \in \mathcal{S}$, let m_s denote the number of ballots with signature s in the original ballot profile. Define $m = \sum_{s\in\mathcal{S}} m_s$, so that m counts the total number of ballots in the original election profile. Note that the values of m_s and m are determined by the original election profile. Let a_s denote the number of ballots that are modified to s from a different ballot signature, d_s denote the number of ballots that are modified from s to another ballot signature, and y_s denote the total number of ballots with signature s after the modifications.

Constraint 1 requires that the number of ballots with a new signature s be equal to the number of ballots that originally had the signature s, plus the number that changed from something else to s, minus the number that changed from s to something else. Constraint 2 states that the number of ballots that end with signature s cannot be more than the total number of ballots that were cast in the election. Constraints 3 and 4 require that one cannot change more ballots of signature s than the number of ballots that originally had the signature s, and that the number of ballots that are modified to signature s must be nonnegative and no more than the number of ballots that had a signature different than s originally.

Constraint 5 implies that the total number of ballots changed *from* any signature is equal to the total number of ballots changed *to* any signature.

Constraints 6 and 7 correspond to the elimination order. Assume C is the set of all candidates. For every pair $\{c_i, c_j\} \subseteq C$, define u_{c_i,c_j} as a binary variable that is 1 iff candidate c_j is eliminated before candidate c_i. For completeness also define $u_{c_i,c_i} = 1$, for every $c_i \in C$. To guarantee that the variables u_{c_i,c_j} define an order, Constraint 6 requires it to be antisymmetric and Constraint 7 requires it to satisfy the triangle inequality.

For a signature s of an original ballot and candidates $\hat{c}$ and $\tilde{c}$ (which may be equal), define the binary variable $v_{s,\hat{c},\tilde{c}}$ to be 1 iff when candidate $\tilde{c}$ is eliminated $\hat{c}$ is the top candidate in the signature that had originally been signature s. Bit $v_{s,\hat{c},\tilde{c}}$ is trivially 0 if $\hat{c}$ does not appear in s. Let signature $s = c_1, c_2, \ldots, c_\ell$, where c_x is the x-th candidate on the ballot, c_1 is the top choice while c_ℓ is the bottom. Assume from now on that $\hat{c} = c_i$. For candidate c_j in s, where $j < i$, bit v_{s,c_i,c_j} is 0 since c_j is ranked higher than c_i is s. Assume from now on that $\tilde{c} \neq c_j$, for $j \in \{1, \ldots, i-1\}$. Constraint 8 on $v_{s,c_i,\tilde{c}}$ ensures that all the candidates $c_1, c_2, \ldots, c_{i-1}$ are eliminated before $\tilde{c}$ is eliminated, and in case $c_i \neq \tilde{c}$, candidate c_i is eliminated after $\tilde{c}$ is eliminated. Thus signature s contributes to c_i's tally when $\tilde{c}$ is eliminated. Note that since by definition $u_{c_1,c_1} = 1$, we get that $v_{s,c_1,c_1} = 1$, which holds trivially. The constraint in its current format is not linear since it is a product of bits. Later, we show how to convert it to linear constraints.

Constraint 9 is for every ordered pair of candidates $\hat{c} \neq \tilde{c}$. It guarantees that if $u_{\hat{c},\tilde{c}} = 1$, namely $\hat{c}$ is eliminated after $\tilde{c}$, then in the round in which $\tilde{c}$ is eliminated the number of ballots in which $\hat{c}$ is the top candidate is at least the number of ballots in which $\tilde{c}$ is the top candidate. The constraint is written as a product of bits and an integer (later, we show how to convert it to linear constraints).

If we want to force candidate $\hat{c}$ to be the winner we need to add the constraints $u_{\hat{c},\tilde{c}} = 1$, for every $\hat{c} \neq \tilde{c}$. Alternatively, if we want to force candidate $\hat{c}$ *not* to be the winner we need to add the constraint $\sum_{\tilde{c}\neq\hat{c}} u_{\tilde{c},\hat{c}} \geq 1$. In addition, we can change the objective function to count only additions or only deletions or any linear combination of additions, deletions, and modifications. For our case we set the objective function to be: minimize $\sum a_s$, which is the number of ballots modifications.

In the last two constraints, we used (i) product of bits, and more generally (ii) product of a nonnegative number and bits. We show how to linearize a product of a nonnegative number and bits as long as we have an upper bound on the number. Let $u_1, \ldots, u_x$ be x bits, and A be a non-negative number. Assume that m is an upper bound on A. (As in our case, since m is the total number of signatures.) The constraints that replace $z = A \cdot \Pi_{i=1}^{x} u_i$ are as follows.

$$
\begin{array}{ll}
z \leq u_i \cdot m & \text{for } i \in \{1, \ldots, x\}\\
z \leq A & \\
z \geq A + \left(\sum_{i=1}^{x} u_i - x\right) \cdot m & \\
z \geq 0 &
\end{array}
$$

4 EFFICIENT ALGORITHMS

This section is dedicated to further investigation of computational efficiency. In Section 4.1, we describe an improved algorithm for computing DistToLB. Due to space constraints we had to omit the description of two additional algorithms: (1) an improved **MqIRVUB** algorithm that is computationally efficient and can be applied as an efficient heuristic for the **MqIRV** problem, and (2) an efficient (polynomial time) algorithm for DistTo in case only ballot additions are allowed. Thus, demonstrating that DistTo becomes a computationally tractable in this special case.

4.1 An Improved DistToLB Algorithm

In this section, we discuss an improved lower bound calculation algorithm for DistTo(π). The intuition is the following: given π and two candidates c and c', if c needs to be eliminated before c' in round i, where $t_i(c)$ and $t_i(c')$ are the number of first choice votes of c and c' in round i, respectively, then at least $\left\lceil \frac{t_i(c)-t_i(c')}{2} \right\rceil$ number of first choice votes from c needs to go to c'. That is, lb, the lower bound of round i is calculated as the half of the difference of tally between c and c'. Finally, the maximum over all of these is returned as the output of the algorithm. Algorithm 2 has the pseudocode.

Algorithm	Efficient AlgExact	Blom
Number of IP calls	AI: 1 ML: 1 DM: 2	AI: 143 ML: 108 DM: 107
Runtime (s)	0.057	0.626

Table 5: Efficiency improvement using MqIRVUB and DistToLB for the running example

Running example. Assume, $\pi = [Gina, Molly, Zoey] = [4, 6, 0]$ where 4 is eliminated first. Initially, $t_1(Gina) = 6$, $t_1(Zoey) = 3$, $t_1(Molly) = 1$. To ensure *Gina* is eliminated, at least $\max\{\lceil \frac{6-1}{2} \rceil, \lceil \frac{6-3}{2} \rceil\} = 3$ ballot modifications are required. After *Gina* is eliminated, $t_2(Zoey) = 5$, $t_2(Molly) = 4$. Required modifications of the ballot to ensure that *Zoey* wins $= \lceil \frac{5-6}{2} \rceil = 0$. Therefore, $lb = \max(3, 0) = 3$.

Using the running example, Algorithm 2 reduces a significant number of DistTo (which is solved using IP) calls. For example, $lb = \text{DistToLB}([4, 6, 0]) = 3 \leq \text{DistTo}([4, 6, 0])$. Hence AlgExact prunes the branch $[4, 6, 0]$ without having to make an expensive DistTo call (this is because lb for this branch $> ub$). Table 5 shows efficiency improvement using DistToLB and **MqIRVUB** inside AlgExact over prior work [13].

Algorithm 2 Algorithm for DistToLB

```
    Input: Set of ballots $\mathcal{B}$, an elimination order $\pi$
    Output: DistToLB(DistTo($\pi$))
1:  $lb = 0$
2:  while $|\pi| > 1$ do
3:      $c = \pi.pop\_front()$
4:      for $c' \in \pi \setminus e$ do
5:          $lb = \max(lb, \lceil \frac{t_i(c)-t_i(c')}{2} \rceil)$
6:      end for
7:  end while
8:  Return $lb$
```

Theorem 4.1. *Algorithm 2 returns a valid lower bound on* DistTo*(π).*

Lemma 1. *The running time of Algorithm 2 is $O(n^2 + m\ell)$.*

5 EXPERIMENTAL EVALUATIONS

We conducted experiments to analyze our algorithms, implemented in Python 3.8 on a Windows 11, i7, 16GB RAM setup. Results are averages from 10 runs. The code and data could be found in the github [3].

5.1 Experiment Design

We have three goals. (a) Assess the effectiveness of **MqKIRV** (Section 5.2). (b) Evaluate the quality of our designed algorithms for **MqIRV** and **MqKIRV** problems (Section 5.3). (c) Evaluate their scalability (Section 5.4). We analyzed four real-world and one synthetic dataset, with comprehensive details provided in Table 6.

Dataset Name	m	n	Description
NSW Senate Election data	533	1,520k	Candidates from five parties.
San Francisco Election data	18	193k	Board of supervisors, district attorney, and mayoral results.
MovieLens	100k	100k	User movie ratings.
Adressa News	100k	100k	News articles with user ratings.
Synthetic	1,000k	1,000k	Random preference rankings.

Table 6: Real world and synthetic datasets
(m denotes number of candidates and n number of voters)

5.1.1 Baseline Algorithms. The following algorithms are implemented.

1. *Filtering-Borda* [32]. We implement a baseline where candidates who do not satisfy the query constraints are first filtered out. Then, considering the remaining candidates, the preferences of the voters are aggregated using the "positional" scoring mechanism Borda [32] that assigns a score to each candidate corresponding to the positions in which a candidate appears within each voter's ballot. This baseline is implemented to evaluate two aspects: 1. Why a ballot modification is necessary, and 2. effectiveness of a different positional aggregation mechanism and its effectiveness over IRV.

2. *Plurality voting* [21, 23]. The winner is the candidate who represents a plurality of voters' first choice or, in other words, receives more first choice votes than any other candidate. That makes plurality voting among the simplest of all electoral systems. This baseline is implemented to evaluate effectiveness of a non positional aggregation mechanism and its effectiveness over IRV.

3. *Blom et al.* [13]. Magrino et al. [29] propose a simple lower bound based on the DistTo of any π of length n. Given two elimination orders, if one is the suffix of another, then, the DistTo of the suffix could be used as the DistToLB of DistTo for the longer elimination order. Blom et al. [13] propose an improved lower bound over [29] based on the last round margin $l(c', c)$ between any pair of candidates c and c' (to ensure c' is eliminated before c), where $l(c', c)$ is half of the difference in the tallies of c' and c (first choice votes). This idea is generalized to generate lower bound of margin to ensure an elimination order ending in π, which is $\max\{l(c', c)\}$, where $c' \in C - \pi$, $c \in \pi$.

4. *Random.* We implement an algorithm that runs iteratively. In the first iteration, it randomly selects a ballot and modifies it. In the next iteration, it doubles the number of selected ballots to be modified (and selects those ballots randomly) and repeats the process until the query constraints are satisfied.

5. *IP for DistToAdd.* We implement an integer programming based solution for the DistToAdd problem.

These algorithms are compared against our proposed DistToLB and **MqIRVUB** solutions inside AlgExact. We also compare AlgApprx against these solutions and the implemented IP for **MqIRV**. Finally, we compare our designed solution DistToAddAlg with its corresponding IP implementation.

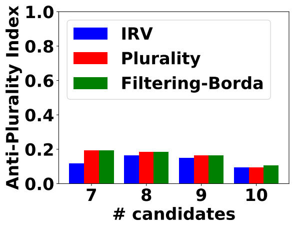


Figure 1: Anti-plurality index using NSW election dataset

5.1.2 *Measures.* To evaluate anti-plurality, we measure the anti-plurality index that is proposed in a related work [16]. Anti-plurality index of a preference aggregation method is computed by looking at each winner candidate i that the method produces and then calculating the percentage of voters who prefer i the least (i.e., it is the last choice on their ballots). The average anti-plurality index is then calculated by taking the average over multiple queries. To evaluate the quality of our designed algorithms, we compare approximation factors of margins produced by different algorithms (margin produced by the proposed algorithm/ exact margin), as well as compare the exact margin values. Finally, we compare the effectiveness of the proposed algorithms based on the number of expensive DistTo calls they make (smaller is better). To evaluate scalability, we evaluate the pruning effectiveness of the algorithms and the overall running time.

5.1.3 *Query and Parameters.* Query constraints are generated randomly but by using party affiliation for NSW datasets, race of the candidates from the San Francisco Election dataset, and movie genre, and news type of the last two datasets, respectively. For evaluating **MqIRV**, we vary the size of the ballot (ℓ), number of users (m), and the number of candidates (n). We consider various combinations over these parameters to cover a wide range of recommendation settings. The default values are $n = 10$, $\ell = 4$ and $m = 1000$.

5.2 Goal 1: Analyzing Anti-plurality

For these experiments, NSW dataset is used. For each query, the set W is selected arbitrarily based on the 5 different party affiliations of the candidates – Labor Party or LAB, Christian Democratic Party or CDP, National Party or NLT, Liberal Party or LIB, The Greens or GRN. We compare average anti-plurality index of **MqIRV** margin computation based on plurality voting [21, 23] and margin computation based on `Filtering-Borda` in Figure 1 after running 133 queries. These results clearly indicate that **MqKIRV** results in significantly anti-plurality compared to the other baselines.

5.2.1 *A case study.* We present a case study to demonstrate efficacy of **MqIRV** to overcome anti-plurality. A smaller subset of NSW election data is used that contains 12 candidates and 33, 553 voters. A query is generated to select candidates that are either LIB or LAB. This makes $W = \{2, 5, 8, 10\}$ (these numbers are the unique ids of the candidates). **MqIRV** selects candidate 8 as the winner, while, Plurality voting and `Filtering-Borda` both select candidate 5. Upon further analysis, it appears that a total of 9884 voters like candidate 5 as their first choice, while a total of 5411 voters dislike candidate 5 (these voters place candidate 5 as their last choice on their ballots). For candidate 8, these two numbers are 9483 and 1863, respectively. In fact, about 25% of the voters put candidate 5 as one of their 3 least preferred candidates compared to only 2% voters that do so for candidate 8. This case study anecdotally demonstrates the efficacy of **MqIRV** to overcome anti-plurality. This case study also demonstrates why filtering based approach could skew the results, which **MqIRV** avoids by looking at the entire ballot and the order of all candidates.

5.3 Goal 2: Analyzing Quality

Approximation factor. In Table 7, we present the approximation factors of the **MqIRV** problems solved using different algorithms. The results are shown for 4 real datasets. Two of the exact solutions are compared against the IP formulation of **MqIRV** and exhibit approximation ratio of 1, as expected. AlgApprx has an approximation ratio between 1.91 and 3.15. On the other hand, *Random* has an approximation ratio between 3.61 and 4.21. As analyzed analytically, DistToAddAlg is an exact solution of DistToAdd and has an approximation ratio of 1.

Dataset	AlgExact	DistToAdd	AlgApprx	*Random*
NSW dataset	1	1	1.97	3.41
San Francisco Election	1	1	1.98	3.96
MovieLens	1	1	1.99	3.42
Adressa News	1	1	3.15	4.21

Table 7: Approximation factor of the algorithms

Margin. Figure 2 shows the box plot of difference in margin for AlgApprx and AlgExact varying n for all 4 real datasets over 10 different queries. These results corroborate that AlgApprx is an effective solution across all 4 datasets.

We also analyze the margin difference between AlgApprx and *Random* using one synthetic dataset and 3 real datasets varying n up to 1 million. For each run, we keep the number of ballots $m = n$. Figure 5 shows AlgApprx always returns smaller margin than *Random*. Using MovieLens data, *Random* margin is about 20 times larger than AlgApprx.

Number of DistTo **IP calls.** Finally, we show that AlgExact requires significantly less number of IP calls compared to *Blom* (Figure 6). On Adressa News dataset on $n = 10$, AlgExact invokes about 17 times less number of IP calls than what *Blom* does. These results demonstrate the effectiveness of our proposed DistToLB and **MqIRVUB** solutions, compared to the adapted version of [13].

5.4 Goal 3: Analyzing Scalability

Running time. In these experiments (Figure 3), we compare running time in seconds for AlgExact, AlgApprx, and *Blom* on 4 real world datasets by varying n, while keeping ℓ and m fixed. The exact algorithms show that the running time increases exponentially with increasing n. AlgApprx is almost 24333 times faster than Blom for $n = 12$ using MovieLens dataset. While AlgExact is 7.6 times faster than Blom for $n = 12$ using MovieLens dataset.

Figure 7 presents effect of varying ℓ and m on running time of AlgExact, AlgApprx, and Blom on 2 real world datasets. As expected, running time AlgExact does not significantly vary with increasing m and ℓ, as it is mostly driven by exponential 2^n cost of branch & bound tree exploration.

Running time in very large scale data. For these experiments, we compare running time of our efficient solution AlgApprx and compare that with *Random*. Figure 8 shows that the running time of

AlgApprx is significantly smaller than *Random*. Using the Adressa News dataset with $n = 100k$, $m = 100k$ and $l = 4$, the runtime for *Random* is about 100 times higher than AlgApprx.

Running time of DistToAddAlg. Figure 4 compares the running time between our exact solution DistToAddAlg for DistToAdd with IP based implementation (DistToIPADD). DistToIPAdd runtime increases exponentially with n as expected, whereas, DistToAddAlg runs in $O(n^2)$ time. For MovieLens dataset with $n = 10$ DistToAddAlg is 53 times faster than DistToIPAdd.

5.5 Summary of Results

Our first observation is that, **MqKIRV** significantly promotes lower anti-plurality, whereas, the other baselines do not. The case study demonstrates that ballot modification selects winner with lower anti-plurality index than plurality voting and a filtering based approach (`Filtering-Borda`) that could be myopic at times. Our second major observation is that our designed AlgExact enabled by effective lower bound DistToLB and upper bound **MqIRVUB** algorithm is highly effective as well as computationally efficient compared to their counterparts *Blom*. Third, AlgApprx exhibits empirical approximation factor around 2 (for 3 of the datasets) and runs significantly faster than the exact solutions (order of magnitude faster) and the *Random* baseline. Finally, consistent with our theoretical analysis, DistToAddAlg returns an exact solution for DistToAdd, runs in polynomial time, and is significantly faster (about 53 times for some datasets) than the IP based solution.

6 PRIOR WORK

We present related work covering three areas: (a) preference aggregation methods, (b) how to minimally update preferences so that the produced outputs satisfy additional criteria, and (c) multi-stage preference aggregation methods and their margin of victory computation.

We remark that it is evident from this prior work that we are the first to study an IRV based multi-stage preference aggregation procedures [7]. Also, our margin finding problem **MqKIRV** is different from previously known Margin of victory (MoV) problems, and our hardness results and algorithmic solutions to this problem extend the state of the art in this area.

Preference aggregation. Preference aggregation is closely studied in the context of group recommendation [1, 2, 5, 8, 9, 14, 28, 34], with the goal of selecting one or top-k items that are most suitable to the preference of all users in the group. These are also studied while promoting fairness in ranking and recommendation [33, 38]. In [16], the authors empirically demonstrate that multi-stage voting methods, such as STV and IRV offer benefits over positional preference aggregation methods (e.g., plurality voting, approval voting) in the recommendation contexts by handling hyperactive users in a more equitable and fair way.

Changing original preferences. The second line of related work exists in how to minimally update the original preferences of the users so that the produced outputs satisfy additional criteria. Some leading criteria include maximizing the satisfaction of some specific users considering rating based preference aggregation methods in top-k recommendation [9], changing the original winner, that is, computing margin, or producing Margin of victory (MoV), or satisfying fairness criteria [26, 36], to name a few. Among these, the most relevant to this work is the previous work on computing MoV. There are two types of MoV: constructive and destructive. In the constructive (destructive) version, the goal is to find the minimum number of changes to the ballots that is needed so that a special candidate is (not) elected. [37] has investigated the computational complexity and (in)approximability of computing MoV for various voting rules, including approval voting, all positional scoring rules, etc. [11] has introduced a sampling based probabilistic algorithm for finding the margin of victory, which can be used for many voting rules.

Multi-stage preference aggregation methods and their margin of victory computation. Multi-stage methods, such as STV and IRV, were introduced in the 19th century in electoral voting systems. [6] demonstrated that determining whether the MoV in an IRV election is at most 1 is NP-hard for both constructive and destructive versions. Moreover, there is no 2-approximation algorithm for it unless $P = NP$. In [17] the coalitional weighted manipulation is investigated. In [27] the authors present a branch and bound algorithm that calculates possible winners when only part (rather than all) of the ballots are accessible. The usage of [27] is to generate information on the result of an election and to announce it on election night, even when there are still some ballots that have not been counted. MoV of IRV [30] and STV [24] is studied in many related works [4, 12, 13, 29].

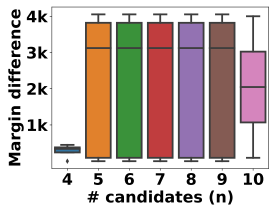
7 CONCLUSION

We study the suitability of Instant Run-off Voting (IRV) as a preference aggregation method to select k different winners to promote group fairness and priority. We formalize an optimization problem that aims at finding the margin, i.e., the smallest number of modifications of original users' preferences (ballots) so that the selected k winners satisfy all these query constraints. We present principled models and several non-trivial algorithmic and theoretical results. Our experimental analyses demonstrate suitability of IRV as a preference aggregation method over plurality voting and a filtering based approach, as well as corroborate our theoretical analysis.

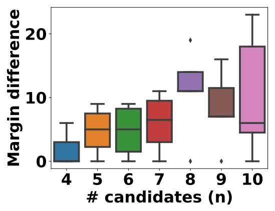
This work opens up many interesting directions – as an ongoing work, we are investigating how to design approximation algorithms with theoretical guarantees for **MqIRV**. We are also studying how our proposed solution AlgExact could be adapted to compute the margin for single transferable voting (STV) schemes.

ACKNOWLEDGMENTS

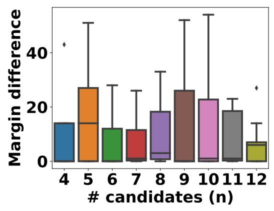
The work of Md Mouinul Islam, Soroush Vahidi, Baruch Schieber and Senjuti Basu Roy is supported by the following funding agencies: (1) National Science Foundation award number(s): 1942913, 2007935, 1814595 (2) Office of Naval Research award number(s): N000141812838, N000142112966



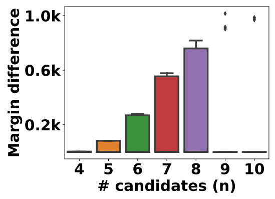
(a) NSW Election



(b) San Francisco Election

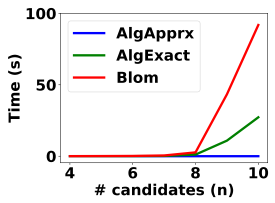


(c) MovieLens

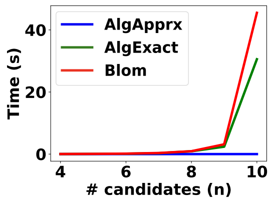


(d) Adressa News

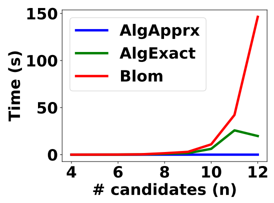
Figure 2: Margin difference between AlgApprx and AlgExact varying n



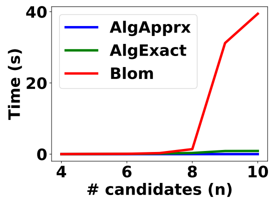
(a) NSW Election



(b) San Francisco Election

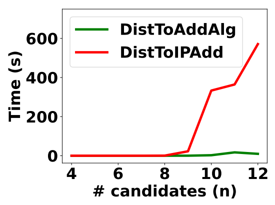


(c) MovieLens

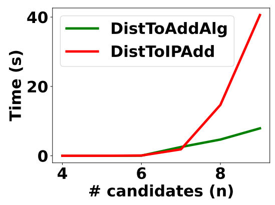


(d) Adressa News

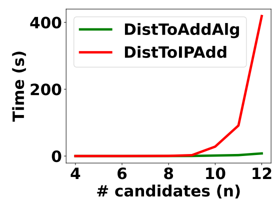
Figure 3: Runtime for AlgApprx, AlgExact, and *Blom* varying n



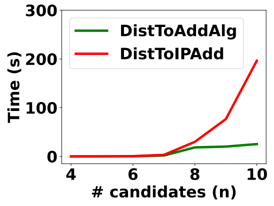
(a) NSW Election



(b) San Francisco Election

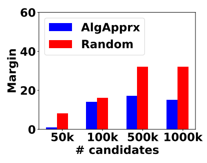


(c) MovieLens

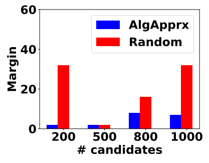


(d) Adressa News

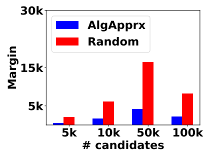
Figure 4: Runtime for DistToAddAlg and DistToIPAdd



(a) Synthetic

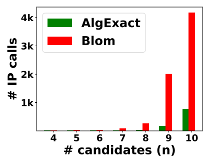


(b) MovieLens

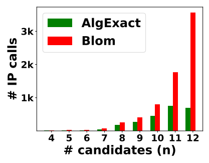


(c) Adressa News

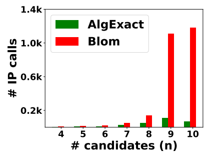
Figure 5: Margin for AlgApprx and Random



(a) NSW Election

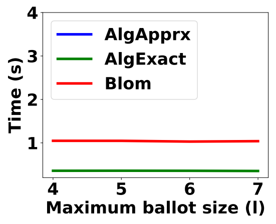


(b) MovieLens

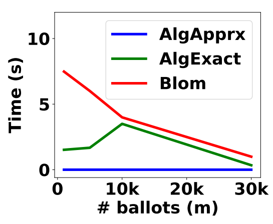


(c) Adressa News

Figure 6: #IP calls for AlgApprx and AlgExact varying n

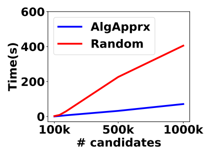


(a) Adressa News varying l

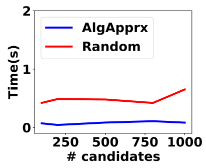


(b) Adressa News varying m

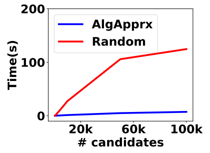
Figure 7: Runtime for AlgApprx, AlgExact, and *Blom* varying l, m



(a) Synthetic



(b) MovieLens



(c) Adressa News

Figure 8: Runtime for AlgApprx & Random

REFERENCES

[1] Sihem Amer-Yahia, Behrooz Omidvar-Tehrani, Senjuti Basu, and Nafiseh Shabib. 2015. Group recommendation with temporal affinities. In *International Conference on Extending Database Technology (EDBT)*.

[2] Sihem Amer-Yahia, Senjuti Basu Roy, Ashish Chawlat, Gautam Das, and Cong Yu. 2009. Group recommendation: Semantics and efficiency. *Proceedings of the VLDB Endowment* 2, 1 (2009), 754–765.

[3] Anonymous. 2023. Git link. https://anonymous.4open.science/r/selection_queries_using_irv-5AD0/README.md.

[4] Manel Ayadi, Nahla Ben Amor, Jérôme Lang, and Dominik Peters. 2019. Single Transferable Vote: Incomplete Knowledge and Communication Issues. In *18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 19)*. International Foundation for Autonomous Agents and Multiagent Systems, Montreal QC, Canada, 1288–1296. https://hal.science/hal-02307486

[5] Linas Baltrunas, Tadas Makcinskas, and Francesco Ricci. 2010. Group recommendations with rank aggregation and collaborative filtering. In *Proceedings of the 4th ACM Conference on Recommender Systems*. 119–126.

[6] John J. Bartholdi and James B. Orlin. 1991. Single transferable vote resists strategic voting. *Social Choice and Welfare* 8, 4 (1991), 341–354. http://www.jstor.org/stable/41105995

[7] Senjuti Basu Roy. 2022. Returning Top-K: Preference Aggregation or Sortition, or is there a Better Middle Ground? *SIGMOD Blog* (2022).

[8] Senjuti Basu Roy, Laks VS Lakshmanan, and Rui Liu. 2015. From group recommendations to group formation. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 1603–1616.

[9] Senjuti Basu Roy, Saravanan Thirumuruganathan, Sihem Amer-Yahia, Gautam Das, and Cong Yu. 2014. Exploiting group recommendation functions for flexible preferences. In *2014 IEEE 30th International Conference on Data Engineering*. IEEE, 412–423.

[10] Rachel Behar and Sara Cohen. 2022. Representative Query Results by Voting. In *Proceedings of the 2022 International Conference on Management of Data*. 1741–1754.

[11] Arnab Bhattacharyya and Palash Dey. 2021. Predicting winner and estimating margin of victory in elections using sampling. *Artificial Intelligence* 296 (2021), 103476. https://doi.org/10.1016/j.artint.2021.103476

[12] Michelle Blom, Peter J. Stuckey, and Vanessa J. Teague. 2017. Towards Computing Victory Margins in STV Elections. arXiv:1703.03511 [cs.GT]

[13] Michelle Blom, Peter J. Stuckey, Vanessa J. Teague, and Ron Tidhar. 2015. Efficient Computation of Exact IRV Margins. arXiv:1508.04885 [cs.AI]

[14] Da Cao, Xiangnan He, Lianhai Miao, Yahui An, Chao Yang, and Richang Hong. 2018. Attentive group recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 645–654.

[15] David Cary. 2011. Estimating the Margin of Victory for Instant-Runoff Voting. In *Conference on Electronic voting technology/workshop on trustworthy elections*. USENIX Association, San Francisco, CA.

[16] Abhijnan Chakraborty, Gourab K Patro, Niloy Ganguly, Krishna P. Gummadi, and Patrick Loiseau. 2018. Equality of Voice: Towards Fair Representation in Crowdsourced Top-K Recommendations. arXiv:1811.08690 [cs.SI]

[17] Vincent Conitzer, Tuomas Sandholm, and Jérôme Lang. 2007. When Are Elections with Few Candidates Hard to Manipulate? *J. ACM* 54, 3 (Jun 2007), 14:1–14:33. https://doi.org/10.1145/1236457.1236461

[18] Palash Dey and Y. Narahari. 2015. Estimating the Margin of Victory of an Election using Sampling. arXiv:1505.00566 [cs.AI]

[19] Peter Emerson. 2013. The original Borda count and partial voting. *Social Choice and Welfare* 40 (2013), 353–358.

[20] FairVote. 2000–2024. Proportional RCV Information. https://fairvote.org/our-reforms/proportional-ranked-choice-voting-information/.

[21] Piotr Faliszewski, Piotr Skowron, Arkadii Slinko, and Nimrod Talmon. 2017. Multi-winner voting: A new challenge for social choice theory. *Trends in Computational Social Choice* 74, 2017 (2017), 27–47.

[22] David García-Soriano and Francesco Bonchi. 2021. Maxmin-fair ranking: individual fairness under group-fairness constraints. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 436–446.

[23] Bernard Grofman, Guillermo Owen, and Scott L Feld. 1983. Thirteen theorems in search of the truth. *Theory and Decision* 15, 3 (1983), 261–278.

[24] Wm. H. Hare. 1871. Application of Mr. Hare's System of Voting to the Nomination of Overseers of Harvard College. *Journal of Social Science: Containing the Transactions of the American Social Science Association* 3-4 (1871), 192–198. https://books.google.com/books?id=W7QRAAAAYAAJ

[25] Steven Hill and Robert Richie. 2005. Success for instant runoff voting in San Francisco. *National Civic Review* 94, 1 (2005), 65–69.

[26] Md Mouinul Islam, Dong Wei, Baruch Schieber, and Senjuti Basu Roy. 2022. Satisfying complex top-k fairness constraints by preference substitutions. *Proceedings of the VLDB Endowment* 16, 2 (2022), 317–329.

[27] Alborz Jelvani and Amelie Marian. 2022. Identifying Possible Winners in Ranked Choice Voting Elections with Outstanding Ballots. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing* 10, 1 (Oct. 2022), 114–123. https://doi.org/10.1609/hcomp.v10i1.21992

[28] Jae Kyeong Kim, Hyea Kyeong Kim, Hee Young Oh, and Young U Ryu. 2010. A group recommendation system for online communities. *International Journal of Information Management* 30, 3 (2010), 212–219.

[29] Thomas Magrino, Ronald Rivest, Emily Shen, and David Wagner. 2011. Computing the margin of victory in IRV elections. In *2011 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 11)*. USENIX Association, San Francisco, CA, 4–4. https://www.usenix.org/conference/evtwote-11/computing-margin-victory-irv-elections

[30] Eamon McGinn. 2020. Rating Rankings: Effect of Instant Run-off Voting on participation and civility. http://eamonmcginn.com.s3.amazonaws.com/papers/IRV_in_Minneapolis.pdf

[31] Robert A. Newland. 1972. Only half a democracy. *Representation, Journal of Representative Democracy* 12, 49 (1972), 38.

[32] Shmuel Nitzan and Ariel Rubinstein. 1981. A further characterization of Borda ranking method. *Public Choice* (1981), 153–158.

[33] Evaggelia Pitoura, Kostas Stefanidis, and Georgia Koutrika. 2022. Fairness in rankings and recommendations: an overview. *The VLDB Journal* (2022), 1–28.

[34] Abinash Pujahari and Dilip Singh Sisodia. 2020. Aggregation of preference relations to enhance the ranking quality of collaborative filtering based group recommender system. *Expert Systems with Applications* 156 (2020), 113476. https://doi.org/10.1016/j.eswa.2020.113476

[35] Anand D. Sarwate, Stephen Checkoway, and Hovav Shacham. 2012. Risk-Limiting Audits and the Margin of Victory in Nonplurality Elections. *Statistics, Politics and Policy* 3, 3 (December 2012), 29–64. https://doi.org/10.1515/spp-2012-0003

[36] Dong Wei, Md Mouinul Islam, Baruch Schieber, and Senjuti Basu Roy. 2022. Rank aggregation with proportionate fairness. In *Proceedings of the 2022 ACM SIGMOD International Conference on Management of Data*. 262–275.

[37] Lirong Xia. 2012. Computing the Margin of Victory for Various Voting Rules. In *Proceedings of the 13th ACM Conference on Electronic Commerce (EC '12)* (Valencia, Spain). Association for Computing Machinery, New York, NY, USA, 982–999. https://doi.org/10.1145/2229012.2229086

[38] Meike Zehlike, Ke Yang, and Julia Stoyanovich. 2021. Fairness in ranking: A survey. *arXiv preprint arXiv:2103.14000* (2021).

A APPENDIX

A.1 Different Aggregation Mechanisms

Plurality voting. In a plurality voting system, each voter is allowed to vote for one candidate, and the candidate who receives the most votes wins, regardless of whether they secure a majority of the votes. This system is straightforward and easy to understand but can result in a "winner-takes-all" outcome where the elected representative may not reflect the preference of the majority of voters. For instance, in our current example, *Zoey* wins the plurality vote with just 3 ballots in her favor. However, *Zoey* is also the least favored choice of 5 voters, underscoring the system's limitation in capturing the majority's true preference.

Scoring based. In scoring-based voting systems, voters score each candidate independently on a scale (e.g., 0 to 5 or 1 to 10). The scores for each candidate are then aggregated to determine the winner. This system allows voters to express not just a preference order but also the intensity of their preferences. Examples of scoring-based voting systems include Range Voting and Approval Voting. However, when the users provide preferences in a ranked order, there is no standard way to convert those preferences to scores.

Positional voting. Positional voting systems allow voters to rank candidates in order of preference. The most common form of positional voting is the *Borda Count*, where points are assigned to positions on the voters' preference lists. Using the running example, *Gina* emerged as the clear winner of *Borda Count* with a total of 17 points (3 points from Emma, 1 point from Monica, 0 point from Daniel, 2 points from John, 4 points from Amy, 3 points from Alice, 3 points from Bob, 1 points from Steve.). Gina's consistent ranking as a second top choices of many voters secured her victory.

A.2 IRV Properties

IRV is known to satisfy properties [16] that other preference aggregation measures are unable to accommodate.

IRV promotes proportional representation for solid coalitions. In social choice theory, a solid coalition for a set of candidates is defined as a set of voters who all rank every candidate in that set higher than any candidates outside that set. This criterion requires that if the number of such voters is at least half of the total number of voters, then one of those candidates from that set must win.

Consider a scenario in which two candidates with similar ideologies compete over the same pool of voters, resulting in divided votes and potentially allowing a third candidate with a different ideology that has fewer overall votes to win. IRV fulfills this criterion, whereas plurality voting [23] fails to do so. To demonstrate this property, notice that in our running example, there exists a solid coalition of voters who like ML (refer to Table 1 which shows 5 of the 10 users, John, Amy, Alice, Bob, and Steve rank the three ML candidates *Gina, Kim, Sara* higher than any other candidate). Clearly, if user preferences are aggregated using plurality voting, none of the ML candidates will be returned as the winner since *Zoey* has the highest number of first place votes, and will be selected as the winner. On the contrary, IRV will select *Sara* as the winner, and hence it is resistant to the ballot splitting problem.

IRV promotes anti-plurality. In social choice theory, the *majority loser criterion* was proposed to evaluate single-winner elections. It states that if a majority of voters prefer every other candidate over a given candidate, then that candidate must not win. IRV fulfills this criterion [16] (as there is a solid coalition for the rest of the candidates). Indeed, the candidate *Zoey* is the last choice of 6 out of the 10 users (Table 1), and thus IRV will not select it. Contrarily, plurality voting will select *Zoey* as the winner. In [16] this criterion is extended to define *anti-plurality* which requires that no candidate among the bottom $x\%$ of the ranked choices for the majority of the voters should be selected. Although not guaranteed, it is empirically shown in [16] that IRV fulfills this extended criterion anti-plurality frequently.

IRV vs. Plurality Voting. A popular voting mechanism is plurality voting, that selects that the winner that receives the highest number of top ranked votes. Using Table 1, note that plurality voting will choose *Laura* as the winner among the candidates in DM area, even though it is clear that between *Laura* and *Molly*, the latter is more preferred by the users. As we will demonstrate later our IRV based process will indeed choose *Molly*. Finally, it is known that finding the *margin* (the number of ballots that must be substituted in order to change the original winner [18, 26, 35, 37]) for IRV is NP-hard [13], making IRV less susceptible to manipulation.

IRV vs. Scoring based Voting. Scoring-based voting systems face challenges in terms of the proportionality of solid coalitions, anti-plurality, and having the complete preferences of voters in a way that accurately reflects voter intent. In scoring-based voting systems, the proportionality of solid coalitions as a strong preference for a particular candidate can be undermined if voters give nearly as high scores to other candidates, thus not providing a clear advantage to the coalition's preferred candidate. This aspect can also impact anti-plurality; since voters might give high scores to a candidate who is the last choice by the majority of the voters, this can lead to a winner who is disliked by most. In contrast, IRV upholds all three of these properties.

IRV vs. Positional Voting. Positional Voting, like the Borda Count, focuses on selecting the most preferred candidate rather than reflecting the depth of support among multiple choices. This may lead to a winner-takes-all outcome, often favoring the candidate with the highest first-choice or second-choice support, potentially disregarding the proportional strength of coalitions. For the same reason, Positional Voting can select a winner who is the last choice by the majority of voters. Let's consider an example where voters rank five candidates A, B, C, D, and E in order of preference. In this election, there are 20 votes of preference order {A, D, E, C}, indicating that A is ranked first, followed by D, E, and C; 10 votes of preference order {B, C, E, A}; and 11 votes of preference order {C, B, D, A} are cast. Applying the Borda Count, Candidate A accumulates 60 points (20 votes * 3 points), Candidate B, C, D, E accumulates 52, 53, 51 and 30 points respectively. Therefore, under the Borda Count, Candidate A emerges as the winner with the highest total points of 60. However, notice there is a solid coalition of B and C formed by 21 voters (they are preferred by 21 voters, more than half of voters, than any other candidates). IRV will select one of the candidates of group B or C. Again, A is the last preference by the majority of voters (21 voters), yet still wins the election, showing an anti-plurality scenario.

A.3 Hardness Results: MqIRV is NP-Complete

Theorem A.1. **MqIRV** *is NP-Complete, even when $\ell = 2$.*

Proof. The hardness proof is by reducing an instance of the known NP-hard problem 3-Exact Cover problem (3XC) to an instance of **MqIRV**. Our proof is inspired by the NP-Hardness proof of [6].

Consider an election in which m voters need to elect $k = 1$ candidate out of n candidates. In the election, each voter casts his/her ballot for two candidate in ranked order. The final candidate is determined using the IRV process. For a given instance of the election, we define the *margin* as the number of ballot changes required to ensure that a specific candidate wins.

We prove that computing the margin is NP-Complete. It is straightforward that the problem is in NP since the outcome of an IRV election can be computed in polynomial time. The NP-hardness is proved by reduction from the 3-Exact Cover problem (3XC). In this problem, we are given a universal set $\{e_1, \ldots, e_{3n}\}$, and $m \geq n$ subsets $S_1, \ldots, S_m$ of size 3 each. We need to determine whether there are n sets whose union is the universal set.

Suppose that we are given an instance of the 3XC problem. We show how to define a related IRV margin problem and then prove that the IRV has a margin n if and only if the answer to the respective 3XC problem is Yes.

The IRV problem has $2m+3n+2$ candidates $b_1, \ldots, b_m, c_1, \ldots, c_m, e_1, \ldots, e_{3n}$ and u, v. We must ensure that u wins the election. There are $6m + m^2 + m(m+5) + 3n(2m+10) + 2m + 11 + 2m + 13 = 2m^2 + 6mn + 15m + 30n + 24$ ballots as follows:

- For every $i \in [1..m]$, let $S_i = \{e_x, e_y, e_z\}$. There are 6 ballots that we call "cover ballots". These ballots are two of each $[b_i, e_x]$, $[b_i, e_y]$, and $[b_i, e_z]$
- For every $i \in [1..m]$ there are m ballots $[b_i, c_i]$
- For every $i \in [1..m]$ there are $m+5$ ballots $[c_i, b_i]$
- For every $i \in [1..3n]$ there are $2m+10$ ballots $[e_i, v]$
- There are $2m+11$ ballots $[v, u]$
- There are $2m+13$ ballots $[u, v]$

Suppose that the 3XC instance has an exact cover. Let the indices of the sets in the cover be $j_1 \ldots, j_n$. We change n ballots as follows. For every $i \in [1..n]$ change a ballot $[b_{j_i}, c_{j_i}]$ to $[c_{j_i}, b_{j_i}]$.

We successively eliminated all candidates who got the least number of votes, which is initially $m+5$. There are m candidates with this number of votes: $m-n$ candidates c_x, for $x \in [1..m] \setminus \{j_1 \ldots, j_n\}$, and n candidates b_x, for $x \in \{j_1 \ldots, j_n\}$. As a result of eliminating the $m-n$ candidates c_x, the number of votes of the candidates b_x, for $x \in [1..m] \setminus \{j_1 \ldots, j_n\}$ jumps to $2m+11$. As a result of eliminating the n candidates b_x, the number of votes of the candidates c_x, for $x \in \{j_1 \ldots, j_n\}$, jumps to $2m+5$. Also, since the union of the n sets S_x, $x \in \{j_1 \ldots, j_n\}$, is the universal set, the elimination of b_x in the $6n$ "cover ballots" causes the number of votes of *every* e_i to jump to $2m+12$.

Next, the n remaining candidates c_x, for $x \in \{j_1 \ldots, j_n\}$, with $2m+5$ votes are eliminated. This does not change the vote of any other candidate. Lastly, the $m-n$ candidates b_x, for $x \in [1..m] \setminus \{j_1 \ldots, j_n\}$, and v each with $2m+11$ votes are eliminated. None of the e_i is eliminated at this point because all of them have $2m+12$ votes. Then, all e_is will be deleted, each with $2m+12$ votes, and, finally, u wins with $2m+11+2m+13 = 4m+24$ votes.

We need to prove the other direction. Namely, if the margin is n then there is an exact cover. Suppose that the outcome of the elections can be changed to be u by at most n ballot changes. Since candidate v has one more vote than each of the $3n$ candidates $e_1, \ldots, e_{3n}$, we need to increase the votes of all the candidates $e_1, \ldots, e_{3n}$ by at least 2 so that none of the e_i is eliminated before v is eliminated. Because if any of e_is is eliminated before v is eliminated, then the second choice of e_i's ballot goes to v and the votes of v increase to $4m+21$. Then all e_i and u will be eliminated, and v wins the election, and u loses. The only way to ensure that none of e_is is eliminated before v is by eliminating some of the candidates b_j. This can be done by ballot changes that reduce the number of votes of some of the candidates b_j by 1 and increase the number of votes of the respective candidates c_j. This will cause some candidates b_j to be eliminated and thus increase the votes of the resulting elements e_i in the "cover ballots" corresponding to these candidates b_j. Since we can make only n ballot changes and since the cover ballots of any candidate b_j change the votes of only the 3 candidates from $\{e_1, \ldots, e_{3n}\}$ that correspond to the set S_j, the n candidates b_j eliminated first must correspond to an exact set. □

A.4 Figures

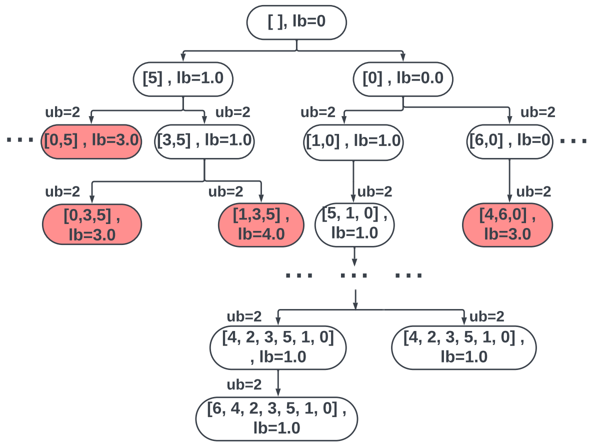


Figure 9: Partially explored tree for AlgExact , the candidates are represented with their ids, where red nodes and their subtrees are pruned