IRV margin

January 23, 2023

1 IRV DistanceToPi computation given an elimination sequence (addition only)

Given an elimination sequence π and a set of ballots B, the algorithm **DistanceToPi (addition)** finds the minimum number of ballot additions required to ensure the elimination sequence π. When we consider only ballot additions, a single added ballot can contribute to multiple candidates' tally in different rounds. For example, consider an elimination order c_5, c_4, c_3, c_2, c_1 where c_1 is the winner. A single ballot (c_3, c_1) can contribute one vote to candidate c_3 at round 3. Once c_3 is eliminated, the same ballot can contribute one vote to candidate c_1. When candidates are eliminated according to π in each round, the number of votes of the non-eliminated candidates should be at least equal to the number of votes of the eliminated candidate in that round. Following the previous example, c_3 is the eliminated candidate in round 3. Candidates c_2, c_1 should have at least the same votes as c_3 in their tally to avoid elimination. In each round of the algorithm, a dictionary called $addTally[c]$ is maintained, which keeps track of the number of votes needed to add to each candidate to avoid elimination. $addTally[c]$ contributes to c's tally in all the rounds of the election until c has been eliminated. Once c is eliminated, $addTally[c]$ is distributed to other non-eliminated candidates. The required number of ballots that needs to be added to a candidate's tally in each round can come from two source: (i) either as a newly added ballots (ii) or from previously added ballots of eliminated candidates. The subroutine **distanceSingleRound** calculates the total number of ballots ($TotalAdd$) needed to add in a round, at the same time, it also updates $addTally[c]$ for each candidate. The total number of ballots ($TotalAdd$) required to add in a round can either come from the added ballots of already eliminated candidates. In that case, $addTally[c]$ of eliminated candidates are set to zero as they are transferred to other candidates who are not eliminated. Otherwise, if $TotalAdd$ is greater than the sum of $addTally$ of previously eliminated candidates, then new ballots are added after transferring all of $addTally[c]$. Finally, the sum of $addTally$ is returned as a result of the algorithm.

The algorithm **DistanceToPi (addition)** and subroutine **distanceSingleRound** are presented in algorithm 1 and 2 respectively. Line 1,2 of **DistanceToPi (addition)** initialize $addTally[c]$ to zero and π' to π. At first round, the $prevElm$ is an empty set, in later rounds it holds the previously eliminated candidates (Line 4). In line 5 and 6, the projected ballots B' and partial elimination sequence π' is calculated. In line 7, subroutine **distanceSingleRound** is called which returns the total number of ballots ($totalAdd$) needed to add in that round and updates the dictionary $addTally[c]$ for each candidate. Line 8 to 15 runs a while loop until the size of $prevElm$ and $totalAdd$ are both greater than zero. In line 9, a candidate e is popped from $prevElem$. If $TotalAdd \geq addTally[e]$ then ballots are transferred from $addTally[e]$ to $TotalAdd$ by setting $addTally[e]$ to zero and reducing $TotalAdd$ by $addTally[e]$. Otherwise, $addTally[e] = addTally[e] - TotalAdd$ and $TotalAdd = 0$. Finally, $sum(addTally)$ is returned at line 20.

The subroutine **distanceSingleRound** is called in each round of **distanceSingleRound** and takes projected ballots B', partial elimination sequence π' and $addTally$ as input. Line 1 to 4 of **distanceSingleRound** calculates the $Tally[c]$ of each candidate in that round. At line 5, $c_1 = \pi[1]$ is set to be the eliminated candidate. Line 6 iterates all candidates $c \in \pi \setminus c_1$. Line 7 checks if $Tally[c_1] + addTally[c_1] > Tally[c] + addTally[c]$ and line 8 calculates the number of ballots that needs to be added (set as $diff$) to ensure c is not eliminated. At line 9 and 10, $diff$ is added to $TotalAdd$ and $addTally[c]$. Finally, $TotalAdd$ and $addTally$ are returned.

Algorithm 1 DistanceToPi (Addition only)

Input: Set of ballots B, an elimination order $\pi = \{c_1, \ldots, c_r, \ldots, c_n\}$
Output: $distance$
1: $addTally[c] = 0, \forall c \in C$
2: $\pi' = \pi$ and $r = 1$
3: **while** $|\pi'| \geq 1$ **do**
4: $prevElm \leftarrow \{c_1, \ldots, c_{r-1}\}$ if $r > 1$ otherwise $prevElm = \{\}$
5: $\pi' \leftarrow \pi \setminus prevElm$
6: B' = reduced profile of B considering only candidates in π'
7: $TotalAdd, addTally$ = **distanceSingleRound**$(B', \pi', addTally)$
8: **while** $|prevElm| > 0$ and $TotalAdd > 0$ **do**
9: $e = prevElm.pop()$
10: **if** $TotalAdd \geq addTally[e]$ **then**
11: $TotalAdd = TotalAdd - addTally[e]$
12: $addTally[e] = 0$
13: **else**
14: $addTally[e] = addTally[e] - TotalAdd$
15: $TotalAdd = 0$
16: **end if**
17: **end while**
18: $r = r + 1$
19: **end while**
20: Return $sum(addTally)$

Algorithm 2 distanceSingleRound

Input: Set of ballots B, π, $addTally$.
Output: $TotalAdd, addTally$
$Tally[c] = 0, \forall c \in \pi', TotalAdd = 0$
for $c \in \pi$ **do**
  $Tally[c] = |\{b : c = first(b), \forall b \in B\}|$
end for
$c_1 \leftarrow \pi[1]$
for $c \in \pi \setminus c_1$ **do**
  if $Tally[c_1] + addTally[c_1] > Tally[c] + addTally[c]$ **then**
    $diff = (Tally[c_1] + addTally[c_1]) - (Tally[c] - addTally[c])$
    $TotalAdd = TotalAdd + diff$
    $addTally[c] = addTally[c] + diff$
  end if
end for
Return $TotalAdd, addTally$

1.0.1 Proof of optimality

Lemma 1. *Addition of two ballots: (i) one ballot (c_r) added to avoid elimination of candidate c_r at round r where candidate c_r had enough ballots in his tally to avoid elimination until round r without the added ballot (c_r) and (ii) another ballot (c_x) added to avoid elimination of candidate c_x at a round x earlier than r ($x < r$), and candidate c_x is eliminated before candidate c_r according to the elimination order π. The addition of these two ballots ((c_r) and (c_x)) of size one can be replaced with a single ballot (c_x, c_r) of size two without changing the resulting elimination order.*

Proof 1. *Ballot (c_x, c_r) will increase the tally of candidate c_x by 1 vote till round x which is similar to addition of ballot (c_x). Once c_x is eliminated at round x, the projected ballot will be $\rho_{\pi \setminus c_x}(c_x, c_r) = c_r$. As a result, from round $x+1$ the ballot (c_x, c_r) will contribute 1 vote to candidate c_r's tally. As c_r does not require this additional 1 vote to avoid elimination up until round r ($x < r$), the resulting elimination order will not alter if (c_x, c_r) is added instead of ballots (c_r) and (c_x).*

Corollary 1. *Addition of two ballots: (i) one ballot (c_r) added to avoid elimination of candidate c_r at round r where candidate c_r had enough ballots in his tally to avoid elimination until round r without the added ballot (c_r) and (ii) another ballot $(c_{x_1}, c_{x_2}, \ldots)$ added to avoid elimination of candidates $c_{x_1}, c_{x_2}, \ldots$ at a round $x_1, x_2, \ldots$ respectively earlier than r ($x_1 < r$ and so on), and candidates $c_{x_1}, c_{x_2}, \ldots$ are eliminated before candidate c_r according to the elimination order π. The addition of these two ballots ((c_r) and $(c_{x_1}, c_{x_2}, \ldots)$) of any size can be replaced with a single ballot $(c_{x_1}, c_{x_2}, \ldots, c_r)$ without changing the resulting elimination order.*

Corollary 2. *Addition of 4 ballots: (i) two ballots (c_{r_1}) and (c_{r_2}) added to avoid elimination of can-*

didates c_{r_1} and c_{r_2} at round r_1 and r_2 respectively where candidates c_{r_1} and c_{r_2} had enough ballots in their tally to avoid elimination until round r_1,r_2 without the added ballot (c_{r_1}) , (c_{r_2}) and (ii) another two ballots $(c_{x_1}, c_{x_2}, \ldots)$ and $(c_{y_1}, c_{y_2}, \ldots)$ added to avoid elimination of candidates $c_{x_1}, c_{x_2}, \ldots$ and $c_{y_1}, c_{y_2}, \ldots$ at a round $x_1, x_2, \ldots$ and $y_1, y_2, \ldots$ respectively earlier than r_1, r_2 ($x_1 < r_1$, $x_1 < r_2$ and so on and $y_1 < r_1$, $y_1 < r_2$ and so on), and candidates $c_{x_1}, c_{x_2}, \ldots$ and $c_{y_1}, c_{y_2}, \ldots$ are eliminated before candidate c_{r_1} and c_{r_2} according to the elimination order π. Addition of there 4 ballots can be replaced with either (a) set of ballots $\{(c_{x_1}, c_{x_2}, \ldots, c_{r_1}), (c_{y_1}, c_{y_2}, \ldots, c_{r_2})\}$ or (b) set of ballots $\{(c_{y_1}, c_{y_2}, \ldots, c_{r_1}), (c_{x_1}, c_{x_2}, \ldots, c_{r_2})\}$ without changing the elimination order.

Lemma 2. *The total number of ballots of size one needed to add to the election profile B to realize elimination order $\pi = \{c_1, \ldots, c_r, \ldots, c_n\}$ is $d = \sum_{r=1}^{n-1} \left[\sum_{c \in S_r \setminus c_r} max(0, \{Tally(S_r, c_r) - Tally(S_r, c)\}) \right]$ where $S_r = \{c_r, \ldots c_n\}$ and $Tally(S, c) = |\{b \in B : first(\rho_S(b)) == c\}|$. This is an upper bound for **distanceTo** function for unbounded ballot size.*

Proof 2. *At round r, each non-eliminated candidate $c \in S_r \setminus c_r$ has to satisfy the constraint $Tally(S_r, c) \geq Tally(S_r, c_r)$ to avoid elimination where c_r is the eliminated candidate at round r. The total number of ballots of size one has to be added at round r is d[r] = $\sum_{c \in S_r \setminus c_r} max(0, \{Tally(S_r, c_r) - Tally(S_r, c)\})$. Hence, The total number of ballots of size one needs to be added to realize the elimination order π is $d = \sum_{r=1}^{n-1} d[r]$. As this is calculated considering ballots of size one, this distance will serve as an upper bound of **distanceTo** for unbounded ballot size.*

Lemma 3. *$\frac{d}{|C|}$ is a lower bound of **distanceTo** function for unbounded ballot size.*

Proof 3. *Each ballot of size one added in $|C|$ number of different rounds can be combined into a single ballot of maximum size $|C|$. Hence, $\frac{d}{|C|}$ is a lower bound of **distanceTo** function.*

Corollary 3. *The optimum number of ballot additions will occur when a maximum of the added ballots of size one can be replaced with a minimum number of added ballots of any size without changing the resulting elimination order.*

Lemma 4. *The algorithm **DistanceToPi (addition)** returns the optimal number of ballot additions required to realize the elimination order π.*

Proof 4. *According to corollary 1, a ballot (of any size) added to any eliminated candidate before round r can be combined with a ballot of size one added at round r or any later rounds without changing the elimination order. As a result, those two ballots can be combined, and $addTally$ of that eliminated candidate can be reduced by one. As long as $TotalAdd \geq addTally[e]$ for any round r then $addTally[e]$ (of previously eliminated candidate e) reduces to 0 and all of e's added ballots are combined with added* *ballots of this round ($TotalAdd$) or later rounds (we do not need to know which round, as long as $TotalAdd \geq addTally[e]$, we know $addTally[e]$ can be reduced to zero). From corollary 2, it does not matter which eliminated candidate's added ballots are combined with which non-eliminated candidate's added ballots. So, the order of iteration for reducing $addTally[e]$ also does not have any effect on the resulting elimination order. As a result, this process results in a maximum number of added ballots of size one being replaced with a minimum number of added ballots of any size without changing the resulting elimination order. Hence, the algorithm* ***DistanceToPi (addition)*** *returns the optimal number of ballot additions required to realize the elimination order π (corollary 2).*

Corollary 4. *For bounded ballot size (**ballot size** $< |C|$) the algorithm **DistanceToPi (addition)** returns a lower bound for the number of ballot additions required to realize the elimination order π.*

2 IRV lower bound computation given an elimination sequence

Given a ballot B and elimination sequence $\pi = \{c_1, \ldots, c_r, \ldots, c_n\}$, where c_n is the winner, and c_1 is the candidate eliminated in the first round. Our task is to find a lower bound for the minimum number of ballots that must be changed in order to achieve an election profile whose elimination order will be π starting from an election profile B. A single round of lower bound calculation is as follows:

1. Let c_r be the eliminated candidate at round r satisfying elimination order π. Here, $c_r = \pi(r)$
2. Say $\pi' = \{c_r, \ldots, c_n\}$, and B' is the reduced election profile which is the result of taking each ballot in B and removing any preferences involving candidates not in π'.
3. Candidate c_r is the first candidate to be eliminated to maintain π' as $c_r = \pi'(1)$. To satisfy elimination order π', any candidate $c_x \in \pi' \setminus \{c_r\}$ must have more primary votes than c_r considering reduced election profile B'
4. $lb_r = max_{c_x \in \pi' \setminus \{c_r\}} \frac{\text{primary votes of } c_r \text{ - primary votes of } c_x}{2}$. Here, primary votes are calculated for election profile B'

Considering all rounds, $LB = max(lb_r)$ for $r \in \pi$. Algorithm 3 describes lowerbound calculation.

Lemma 1. *For any round r, lower bound in step 4 is $lb_r \leq DistanceTo(B', \pi')$. Here, $\pi' = \{c_r, \ldots, c_n\}$, B' is the reduced election profile considering π', and $DistanceTo(B', \pi')$ returns the minimum number of ballots that must be changed in order to achieve an election profile whose elimination order will be π' starting from an election profile B'.*

Algorithm 3 LowerBound

Input: Set of ballots B, an elimination order $\pi = \{c_1, \ldots, c_r, \ldots, c_n\}$
Output: LB

```
LB = 0
π′ = π and r = 1
while |π′| ≥ 0 do
    π′ ← π \ {c_1, ..., c_{r−1}} if r > 1 otherwise π′ = π
    B′ = reduced profile of B considering only candidates in π′
    lb_r = LBSingleRound(B′, π′)
    LB = max(LB, lb_r)
    r = r + 1
end while
Return LB
```

Algorithm 4 LBSingleRound

Input: Set of ballots B, π
Output: lb

```
Tally[c] = 0, ∀c ∈ π′
lb = 0
for c ∈ π do
    Tally[c] = |{b : c = first(b), ∀b ∈ B}|
end for
c_1 ← π[1]
for c ∈ π \ c_1 do
    lb = max(lb, (Tally[c_1] − Tally[c]) / 2)
end for
Return lb
```

Proof. Candidate c_r has to be eliminated in the first round to ensure an elimination order π' as $c_r = \pi'(1)$. In any modified election profile (starting from B'), all other candidates $c_x \in \pi' \setminus \{c_r\}$ should have more primary votes than c_r. The lower bound lb_r for this round is calculated as: $\frac{1}{2}\times$ maximum difference of primary votes of c_r and any $c_x \in \pi' \setminus \{c_r\}$. To satisfy elimination order π', any modified election profile (starting from B') must ensure c_r is eliminated first, c_{r+1} is eliminated second and so on. To ensure c_r is eliminated first, at least lb_r number of ballot substitution is required. Hence, considering the definition of $DistanceTo(B', \pi')$, $lb_r \leq DistanceTo(B', \pi')$. □

Lemma 2. *Let $\pi = \{c_1, ..., c_n\}$ be a set of m candidates and $\pi' = \pi \setminus \{c_1\}$. Then, $DistanceTo(B', \pi') \leq DistanceTo(B, \pi)$, where B' is the reduced election profile for π'.*

Proof. see https://www.cs.cornell.edu/ tmagrino/papers/evt11.pdfMagrino section 3.4 Lemma 1. □

Corollary 5. *$LB = max_{r\in\pi}(lb_r)$ is less than equal to the minimum number of ballots that must be changed in order to achieve an election profile whose elimination order will be π starting from an election profile B.*

From Lemma 1 and Lemma 2, lower bound in any round r is $lb_r \leq DistanceTo(B', (c_r, ..., c_n)) \leq DistanceTo(B, (c1, \ldots, c_r, \ldots, c_n))$ where B' is the reduced profile from B considering candidates in $\{c_r, ..., c_n\}$. Hence, $LB = max_{r\in\pi}(lb_r) \leq DistanceTo(B, \pi)$ where $\pi = \{c1, \ldots, c_r, \ldots, c_n\}$. As a result, LB is less than equal to the minimum number of ballots that must be changed in order to achieve an election profile whose elimination order will be π starting from an election profile B.