Democratic Elections in Faulty Distributed Systems

Himanshu Chauhan and Vijay K. Garg

Parallel and Distributed Systems Lab,
Department of Electrical and Computer Engineering,

THE UNIVERSITY OF TEXAS AT AUSTIN

Motivation – Leader Election

Conventional Problem

Node with the highest id should be the leader. All the nodes in the system should agree on the leader.

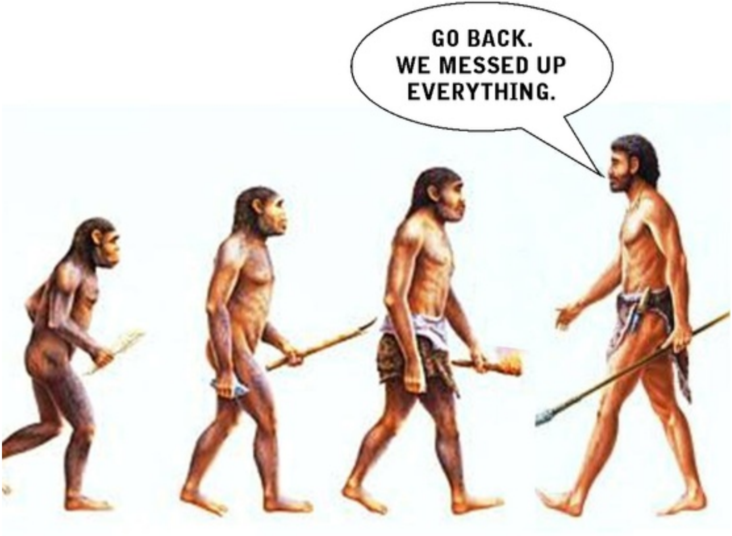
Motivation – Leader Election

Conventional Problem

Node with the highest id should be the leader. All the nodes in the system should agree on the leader.

- Philosophers of Ancient Athens would protest!

Motivation – Leader Election



Democratic Leader Election

- *Elect* a leader
  - Each node has individual preferences
  - Conduct an election where every node votes

Democratic Leader Election

- *Elect* a leader
  - Each node has individual preferences
  - Conduct an election where every node votes

Use Case:

- Job processing system

- Leader distributes work in the system

Democratic Leader Election

- *Elect* a leader
  - Each node has individual preferences
  - Conduct an election where every node votes

Use Case:

- Job processing system
- Leader distributes work in the system
- Worker nodes vote, based upon:
  - Latency of communication with *prospective* leader
  - Individual work load

Democratic Leader Election

- *Elect* a leader
  - Each node has individual preferences
  - Conduct an election where every node votes

Use Case:

- Job processing system
- Leader distributes work in the system
- Worker nodes vote, based upon:
  - Latency of communication with *prospective* leader
  - Individual work load
- Enter 'Byzantine' Voters!

Why Not Use Exisiting Approaches?

'Multivalued Byzantine Agreement', Turpin and Coan 1984,
'$k-$set Consensus', Prisco et al. 1999

- Every voter sends her *top* choice
- Run Byzantine Agreement
  - Agree on the choice with most votes

Why Not Use Exisiting Approaches?

'Multivalued Byzantine Agreement', Turpin and Coan 1984,
'$k-$set Consensus', Prisco et al. 1999

- Every voter sends her *top* choice
- Run Byzantine Agreement
  - Agree on the choice with most votes

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

Why Not Use Exisiting Approaches?

'Multivalued Byzantine Agreement', Turpin and Coan 1984,
'$k-$set Consensus', Prisco et al. 1999

- Every voter sends her *top* choice
- Run Byzantine Agreement
  - Agree on the choice with most votes

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

Elect choice with most votes (at top) : c or b

Why Not Use Exisiting Approaches?

'Multivalued Byzantine Agreement', Turpin and Coan 1984,
'k−set Consensus', Prisco et al. 1999

- Every voter sends her *top* choice
- Run Byzantine Agreement
  - Agree on the choice with most votes

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

Elect choice with most votes (at top) : c or b But . . .

Why Not Use Exisiting Approaches?

'Multivalued Byzantine Agreement', Turpin and Coan 1984,
'k−set Consensus', Prisco et al. 1999

- Every voter sends her *top* choice
- Run Byzantine Agreement
  - Agree on the choice with most votes

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b				a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice				b	b	b	

Elect choice with most votes (at top) : c or b But ...

$\#(a > b) = 4, \quad \#(b > a) = 3$

Why Not Use Exisiting Approaches?

'Multivalued Byzantine Agreement', Turpin and Coan 1984,
'$k-$set Consensus', Prisco et al. 1999

- Every voter sends her *top* choice
- Run Byzantine Agreement
  - Agree on the choice with most votes

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice				c	c	c	a
2^{nd} choice	a	a	a	a	a	a	
3^{rd} choice	c	c	c				c

Elect choice with most votes (at top) : c or b But ...

$\#(a > b) = 4, \quad \#(b > a) = 3$ and $\#(a > c) = 4, \quad \#(c > a) = 3$

Model & Constructs

System

- n processes (voters)
- f Byzantine processes (voters) : *bad*
- Non-faulty processes (voters) : *good*
- $f < n/3$

Model & Constructs

System

- n processes (voters)
- f Byzantine processes (voters) : *bad*
- Non-faulty processes (voters) : *good*
- $f < n/3$

Jargon

$\mathcal{A}$: Set of candidates
Ranking: Total order over the set of candidates.
Vote: A voter's preference ranking over candidates.
Ballot : Collection of all votes.
Scheme : Mechanism that takes a ballot as input and outputs a winner.

Conducting Distributed Democratic Elections

- Use Interactive Consistency
  - Agree on everyone's vote[1]
  - Agree on the ballot
- Use a *scheme* to decide the winner

[1] We use Gradecast based Byzantine Agreement by Ben-Or et al.

Byzantine Social Choice

Social Choice

Given a ballot, declare a candidate as the winner of the election.

Arrow 1950-51, Buchanan 1954, Graaff 1957

Byzantine Social Choice

Given a set of n processes of which at most f are faulty, and a set $\mathcal{A}$ of k choices, design a protocol elects one candidate as the social choice, while meeting the 'protocol requirements'.

Byzantine Social Welfare

Social Welfare

Given a ballot, produce a *total order* over the set of candidate.

Arrow 1950-51, Buchanan 1954, Graaff 1957, Farquharson 1969

Byzantine Social Welfare

Given a set of n processes of which at most f are faulty, and a set $\mathcal{A}$ of k choices, design a protocol that produces a *total order* over $\mathcal{A}$, while meeting the 'protocol requirements'.

Protocol Requirements

1. *Agreement*: All good processes decide on the same choice/ranking.

Protocol Requirements

1. *Agreement*: All good processes decide on the same choice/ranking.
2. *Termination*: The protocol terminates in a finite number of rounds.

Validity Condition

Validity: Requirement on the choice/ranking decided, based upon the votes of good processes.

Validity Condition

Validity: Requirement on the choice/ranking decided, based upon the votes of good processes.

- S: If v is the top choice of all good voters, then v must be the winner.
- S': If v is the last choice of all good voters, then v must **not** be the winner.
- M': If v is last choice of majority of good voters, then v must **not** be the winner.

Validity Conditions

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

Table: Ballot of 7 votes (P_6, P_7 Byzantine)

Validity Conditions

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

Table: Ballot of 7 votes (P_6, P_7 Byzantine)

M (Elect majority of *good* voters) : elect b

Validity Conditions

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

Table: Ballot of 7 votes (P_6, P_7 Byzantine)

M (Elect majority of *good* voters) : elect b

P (Do not elect a candidate that is not the *top* choice of any *good* voters) : *do not* elect a

$BSC(k, V)$

Byzantine Social Choice problem with k candidates, and validity condition/requirement V.

$BSC(2, M)$:

$BSC(k, V)$

Byzantine Social Choice problem with k candidates, and validity condition/requirement V.

$BSC(2, M)$:

- M: elect top choice of majority of *good* votes

Byzantine Social Choice – Impossibilities

$BSC(k, V)$

Byzantine Social Choice problem with k candidates, and validity condition/requirement V.

$BSC(2, M)$:

- M: elect top choice of majority of *good* votes
- Impossible to solve for $f \geq n/4$

Byzantine Social Choice – Impossibilities

$BSC(k, V)$

Byzantine Social Choice problem with k candidates, and validity condition/requirement V.

$BSC(2, M)$:

- M: elect top choice of majority of *good* votes
- Impossible to solve for $f \geq n/4$

Reason:
$f \geq n/4 \Rightarrow$ can not differentiate b/w *good* and *bad* votes

$BSC(2, M')$:

- M': **do not** elect the last choice of majority of *good* votes
- Impossible to solve for $f \geq n/4$

Byzantine Social Choice – Possibilities

$BSC(k, S \wedge M')$:

- S: if v is first choice of all *good* voters, elect v
- M': if v' is last choice of majority of *good* voters, **do not** elect v'

Byzantine Social Choice – Possibilities

$BSC(k, S \wedge M')$:

- S: if v is first choice of all *good* voters, elect v
- M': if v' is last choice of majority of *good* voters, **do not** elect v'
- Solvable for $k \geq 3$

Byzantine Social Choice – Possibilities

$BSC(k, S \wedge M')$:

- S: if v is first choice of all *good* voters, elect v
- M': if v' is last choice of majority of *good* voters, **do not** elect v'
- Solvable for $k \geq 3$

Approach:

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

Byzantine Social Choice – Possibilities

$BSC(k, S \wedge M')$:

- S: if v is first choice of all *good* voters, elect v
- M': if v' is last choice of majority of *good* voters, **do not** elect v'
- Solvable for $k \geq 3$

Approach:

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

- Round 1 : Agree on *last* choices of all voters

Byzantine Social Choice – Possibilities

$BSC(k, S \wedge M')$:

- S: if v is first choice of all *good* voters, elect v
- M': if v' is last choice of majority of *good* voters, **do not** elect v'
- Solvable for $k \geq 3$

Approach:

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

$$n = 7, \qquad f = 2, \qquad \lfloor (n-f)/2 + 1 \rfloor = 3$$

- Round 1 : Agree on *last* choices of all voters
- Remove any candidates that appears $\lfloor (n-f)/2 + 1 \rfloor$ times or more

Byzantine Social Choice – Possibilities

$BSC(k, S \wedge M')$:

- S: if v is first choice of all *good* voters, elect v
- M': if v' is last choice of majority of *good* voters, **do not** elect v'
- Solvable for $k \geq 3$

Approach:

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

$$n = 7, \qquad f = 2, \qquad \lfloor (n-f)/2 + 1 \rfloor = 3$$

- Round 1 : Agree on *last* choices of all voters
- Remove any candidates that appears $\lfloor (n-f)/2 + 1 \rfloor$ times or more
- $f < n/3 \wedge k \geq 3 \Rightarrow$ at least one candidate that would not be removed

Byzantine Social Choice – Possibilities

$BSC(k, S \wedge M')$:

- S: if v is first choice of all *good* voters, elect v
- M': if v' is last choice of majority of *good* voters, **do not** elect v'
- Solvable for $k \geq 3$

Approach:

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

$$n = 7, \qquad f = 2, \qquad \lfloor (n - f)/2 + 1 \rfloor = 3$$

- Round 1 : Agree on *last* choices of all voters
- Remove any candidates that appears $\lfloor (n - f)/2 + 1 \rfloor$ times or more
- $f < n/3 \wedge k \geq 3 \Rightarrow$ at least one candidate that would not be removed
- Round 2 : Use *top* choices from remaining candidates, agree and decide

$BSC(k, V)$ Results – Summarized

Requirement	Unsolvable	Solvable
S	-	$k \geq 2$
S'	-	$k \geq 2$
M	$f \geq n/4 \wedge k \geq 2$	-
M'	$f \geq n/4 \wedge k = 2$	$k \geq 3$
P	$f \geq 1 \wedge k \geq n$	$f < min(n/k, n/3)$ $\wedge\ 2 \leq k < n$

Table: Impossibilities & Possibilities for $BSC(k, V)$

Byzantine Social Welfare – Schemes

Given a ballot, produce a total order over the set of candidates

Byzantine Social Welfare – Schemes

Given a ballot, produce a total order over the set of candidates

Place-Plurality Scheme:

Byzantine Social Welfare – Schemes

Given a ballot, produce a total order over the set of candidates

Place-Plurality Scheme:
k candidates

Given a ballot, produce a total order over the set of candidates

Place-Plurality Scheme:
k candidates

for $1 \le i \le k$
    c_i = candidate with most votes at position i in ballot
    $result[i] = c_i$
done

Given a ballot, produce a total order over the set of candidates

Place-Plurality Scheme:
k candidates

for $1 \leq i \leq k$
  $c_i =$ candidate with most votes at position i in ballot
  $result[i] = c_i$
done

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

Result : $b \succ a \succ c$

Median of a Ballot

Distance (d) between rankings: # of pair-orderings on which rankings differ

Pairwise Comparison, Condorcet, circa 1785

Median of a Ballot

Distance (d) between rankings: # of pair-orderings on which rankings differ

Pairwise Comparison, Condorcet, circa 1785

r	r'	d
a	b	1
b	a	– differ on
c	c	(a, b)

Median of a Ballot

Distance (d) between rankings: # of pair-orderings on which rankings differ

Pairwise Comparison, Condorcet, circa 1785

r	r'	d
a	c	2
b	b	– differ on
c	a	(a, b) and (b, c)

Median of a Ballot

Distance (d) between rankings: # of pair-orderings on which rankings differ

Pairwise Comparison, Condorcet, circa 1785

r	r'	d
a	c	2
b	b	– differ on
c	a	(a, b) and (b, c)

Median (m) of ballot: Ranking that has least distance from overall pair-wise comparisons in the ballot

Kemeny-Young Scheme

(1) J. Kemeny, 1959, (2) H. Young, 1995

Goal: Get as close to the median as possible.

Kemeny-Young Scheme

(1) J. Kemeny, 1959, (2) H. Young, 1995

Goal: Get as close to the median as possible.

For ranking r, let $P_r :=$ ordered pairs from r.

Kemeny-Young Scheme

(1) J. Kemeny, 1959, (2) H. Young, 1995

Goal: Get as close to the median as possible.

For ranking r, let $P_r :=$ ordered pairs from r.
Example: $r = a \succ b \succ c$ then, $P_r = \{(a, b) \quad (b, c) \quad (a, c)\}$

Kemeny-Young Scheme

(1) J. Kemeny, 1959, (2) H. Young, 1995

Goal: Get as close to the median as possible.

For ranking r, let $P_r :=$ ordered pairs from r.
Example: $r = a \succ b \succ c$ then, $P_r = \{(a, b) \quad (b, c) \quad (a, c)\}$

For a given ballot B:

$$score(r, B) = \Sigma \text{ (frequency of } p \text{ in } B)$$

S_k: set of all permutations of k candidates ($k!$ permutations)

foreach ranking $r \in S_k$ **do**
compute $score_r = score(r, B)$
done

Kemeny-Young Scheme

(1) J. Kemeny, 1959, (2) H. Young, 1995

Goal: Get as close to the median as possible.

For ranking r, let $P_r :=$ ordered pairs from r.
Example: $r = a \succ b \succ c$ then, $P_r = \{(a, b) \quad (b, c) \quad (a, c)\}$

For a given ballot B:

$$score(r, B) = \Sigma \text{ (frequency of } p \text{ in } B)$$

S_k: set of all permutations of k candidates ($k!$ permutations)

foreach ranking $r \in S_k$ **do**
compute $score_r = score(r, B)$
done
select ranking with maximum $score_r$ value as the outcome

Kemeny-Young Scheme – Example

Candidates: $\{a,b,c\}$

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

$$\#(a \succ b) = 4, \qquad \#(b \succ a) = 3, \qquad \#(a \succ c) = 4,$$
$$\#(c \succ a) = 3, \qquad \#(b \succ c) = 4, \qquad \#(c \succ b) = 3$$

Kemeny-Young Scheme – Example

Candidates: $\{a, b, c\}$

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

$$\#(a \succ b) = 4, \quad \#(b \succ a) = 3, \quad \#(a \succ c) = 4,$$
$$\#(c \succ a) = 3, \quad \#(b \succ c) = 4, \quad \#(c \succ b) = 3$$

Permutations:

a	a	b	b	c	c
b	c	a	c	a	b
c	b	c	a	b	a

Kemeny-Young Scheme – Example

Candidates: $\{a,b,c\}$

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

$$\#(a \succ b) = 4, \quad \#(b \succ a) = 3, \quad \#(a \succ c) = 4,$$
$$\#(c \succ a) = 3, \quad \#(b \succ c) = 4, \quad \#(c \succ b) = 3$$

Permutations:

a	a	b	b	c	c
b	c	a	c	a	b
c	b	c	a	b	a

pairs: $\{(a,b) \quad (b,c) \quad (a,c)\}$

Kemeny-Young Scheme – Example

Candidates: $\{a,b,c\}$

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

$\#(a \succ b) = 4$, $\#(b \succ a) = 3$, $\#(a \succ c) = 4$,
$\#(c \succ a) = 3$, $\#(b \succ c) = 4$, $\#(c \succ b) = 3$

Permutations:

a	a	b	b	c	c
b	c	a	c	a	b
c	b	c	a	b	a
12					

pairs: $\{(a,b) \quad (b,c) \quad (a,c)\}$

Kemeny-Young Scheme – Example

Candidates: $\{a,b,c\}$

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

$$\#(a \succ b) = 4, \quad \#(b \succ a) = 3, \quad \#(a \succ c) = 4,$$
$$\#(c \succ a) = 3, \quad \#(b \succ c) = 4, \quad \#(c \succ b) = 3$$

Permutations:

a	a	b	b	c	c
b	c	a	c	a	b
c	b	c	a	b	a
12	11	11	10	10	9

Kemeny-Young Scheme – Example

Candidates: $\{a,b,c\}$

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

$$\#(a \succ b) = 4, \quad \#(b \succ a) = 3, \quad \#(a \succ c) = 4,$$
$$\#(c \succ a) = 3, \quad \#(b \succ c) = 4, \quad \#(c \succ b) = 3$$

Permutations:

a	a	b	b	c	c
b	c	a	c	a	b
c	b	c	a	b	a
12	11	11	10	10	9

Kemeny-Young Scheme Result: $a \succ b \succ c$

Pruned-Kemeny-Young Scheme (this paper)

Objective: Minimize the influence of *bad* voters on the outcome

Pruned-Kemeny-Young Scheme (this paper)

Objective: Minimize the influence of *bad* voters on the outcome

f *bad* voters ($f < n/3$)

B: Agreed upon ballot; S_k: set of all permutations of k candidates

Pruned-Kemeny-Young Scheme (this paper)

Objective: Minimize the influence of *bad* voters on the outcome

f *bad* voters ($f < n/3$)

B: Agreed upon ballot; S_k: set of all permutations of k candidates

foreach ranking $r \in S_k$ **do**
  $F = f$ most distant rankings from r in B
  define $B' = B \backslash F$
  compute $score_r = score(r, B')$
done

Pruned-Kemeny-Young Scheme (this paper)

Objective: Minimize the influence of *bad* voters on the outcome

f *bad* voters ($f < n/3$)

B: Agreed upon ballot; S_k: set of all permutations of k candidates

foreach ranking $r \in S_k$ **do**

$F = f$ most distant rankings from r in B

define $B' = B \backslash F$

compute $score_r = score(r, B')$

done

select ranking with maximum $score_r$ value as the outcome

Pruned-Kemeny-Young – Example

$n = 7, \qquad f = 2$

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

Pruned-Kemeny-Young – Example

$n = 7, \qquad f = 2$

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

a	a	b	b	c	c
b	c	a	c	a	b
c	b	c	a	b	a

Pruned-Kemeny-Young – Example

$n = 7, \qquad f = 2$

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

a	a	b	b	c	c
b	c	a	c	a	b
c	b	c	a	b	a

Pruned-Kemeny-Young – Example

$n = 7, \qquad f = 2$

	P_1	P_2	P_3	P_6	P_7
1^{st} choice	b	b	b	c	a
2^{nd} choice	a	a	a	a	b
3^{rd} choice	c	c	c	b	c

a	a	b	b	c	c
b	c	a	c	a	b
c	b	c	a	b	a
		11			

Pruned-Kemeny-Young – Example

$n = 7, \qquad f = 2$

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

a	a	b	b	c	c
b	c	a	c	a	b
c	b	c	a	b	a
9	8	11	6	10	6

Pruned-Kemeny-Young – Example

$n = 7, \qquad f = 2$

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
1^{st} choice	b	b	b	c	c	c	a
2^{nd} choice	a	a	a	a	a	a	b
3^{rd} choice	c	c	c	b	b	b	c

a	a	b	b	c	c
b	c	a	c	a	b
c	b	c	a	b	a
9	8	11	6	10	6

Pruned-Kemeny Scheme Result: $b \succ a \succ c$

Evaluating Scheme Efficacy

Suppose ω is an *ideal* ranking over k candidates

- ω as the election outcome $\Rightarrow$ maximum social welfare

Evaluating Scheme Efficacy

Suppose ω is an *ideal* ranking over k candidates

- ω as the election outcome $\Rightarrow$ maximum social welfare

- All *good* voters in the system favor ω
  - *goodProb*: probability of a *good* voter putting $a \succ b$ in her vote if $a \succ_\omega b$

Evaluating Scheme Efficacy

Suppose ω is an *ideal* ranking over k candidates

- ω as the election outcome $\Rightarrow$ maximum social welfare

- All *good* voters in the system favor ω
  - *goodProb*: probability of a *good* voter putting $a \succ b$ in her vote if $a \succ_\omega b$

- All *bad* voters in the system act hostile
  - try to minimize social welfare by voting against ω
  - *badProb*: probability of a *bad* voter putting $b \succ a$ in her vote if $a \succ_\omega b$

Evaluating Scheme Efficacy

Suppose ω is an *ideal* ranking over k candidates

- ω as the election outcome $\Rightarrow$ maximum social welfare
- All *good* voters in the system favor ω
  - *goodProb*: probability of a *good* voter putting $a \succ b$ in her vote if $a \succ_\omega b$
- All *bad* voters in the system act hostile
  - try to minimize social welfare by voting against ω
  - *badProb*: probability of a *bad* voter putting $b \succ a$ in her vote if $a \succ_\omega b$
- Analyze outcomes generated by schemes

Evaluating Scheme Efficacy

Suppose ω is an *ideal* ranking over k candidates

- ω as the election outcome $\Rightarrow$ maximum social welfare

- All *good* voters in the system favor ω
  - *goodProb*: probability of a *good* voter putting $a \succ b$ in her vote if $a \succ_\omega b$

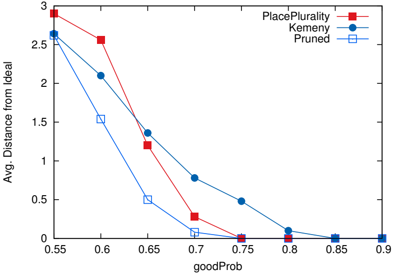
- All *bad* voters in the system act hostile
  - try to minimize social welfare by voting against ω
  - *badProb*: probability of a *bad* voter putting $b \succ a$ in her vote if $a \succ_\omega b$

- Analyze outcomes generated by schemes

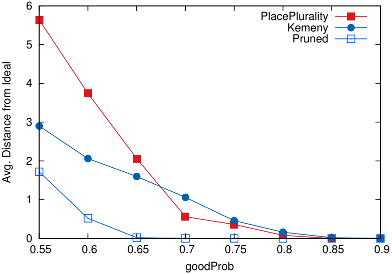
$$\#\text{ of voters} = 100,\ \#\text{ of } \textit{bad} \text{ voters} = 33,\ \textit{badProb} = 0.9$$

Simulation Results

Average (of 50 ballots) distances of produced outcomes from the ideal ranking

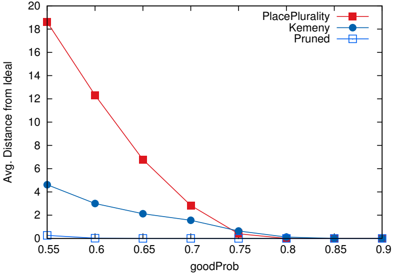


(a) # of Candidates = 3

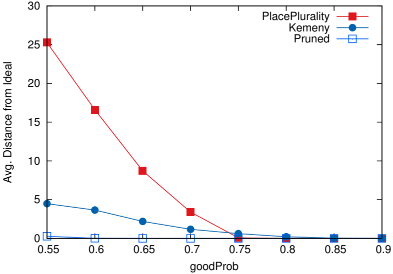


(b) # of Candidates = 4

Average (of 50 ballots) distances of produced outcomes from the ideal ranking



(c) # of Candidates = 7



(d) # of Candidates = 8

Conclusion

- Introduction of democratic election problem in distributed systems

Conclusion

- Introduction of democratic election problem in distributed systems

- Pruned-Kemeny-Young Scheme for Byzantine Social Welfare problem

Future Work

- Pruned-Kemeny-Young (and Kemeny-Young)
    - NP-Hard

Future Work

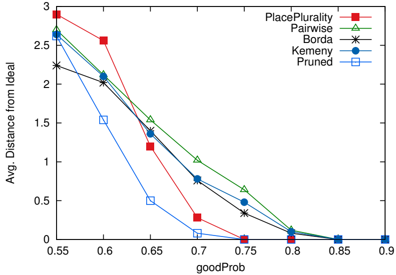
- Pruned-Kemeny-Young (and Kemeny-Young)
  - NP-Hard
  - Yet produce ‘better’ results

Future Work

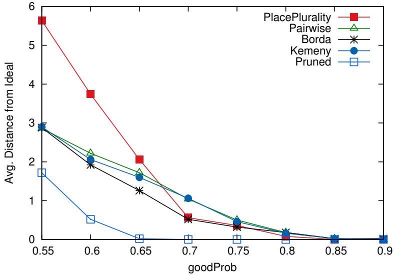
- Pruned-Kemeny-Young (and Kemeny-Young)
  - NP-Hard
  - Yet produce 'better' results
  - Explore techniques for finding 'better' outcomes in polynomial steps

Thanks!

Backup



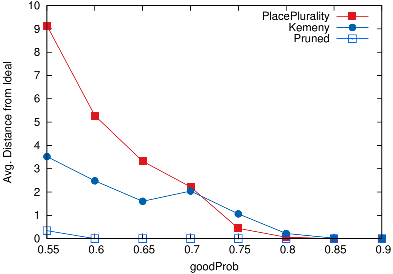
(e) # of Candidates = 3



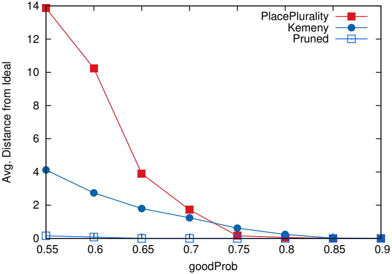
(f) # of Candidates = 4

Backup

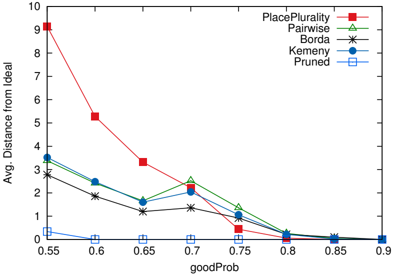
Average (of 50 ballots) distances of produced outcomes from the ideal ranking



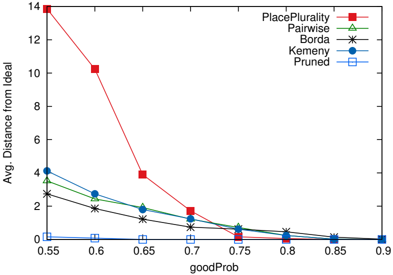
(g) # of Candidates = 5



(h) # of Candidates = 6

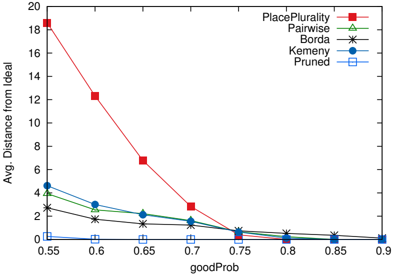


(i) # of Candidates = 5

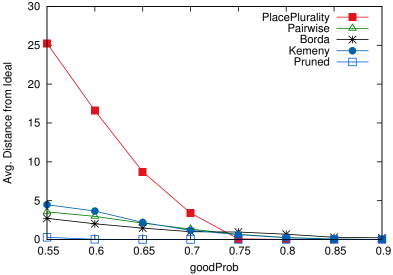


(j) # of Candidates = 6

Backup



(k) # of Candidates = 7



(l) # of Candidates = 8

Related Work

- Arrow's Impossibility Theorem, and his work on Social Choice and Welfare Theory
  - 1950, 1951
- Pairwise Comparison Schemes, Social Welfare Schemes, Theory of Voting, Welfare Economics
  - Condorcet circa 1785, Buchanan 1954, Graaff 1957, Kemeny 1959, Farquharson 1969, Ishikawa et al. 1979, Young 1988
- Multivalued Byzantine Agreement Schemes, Byzantine Leader Election, k-set Consensus
  - Turpin and Coan 1984, Ostrovsky et al. 1994, Russell et al. 1998, Kapron et al. 2008, Prisco et al. 1999