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
himanshu@utexas.edu, garg@ece.utexas.edu

Abstract. In this paper, we show that for elections in distributed systems the conversion from non-binary choices to binary choices does not always provide optimal results when the preferences of nodes are not identical. With this observation, we study the problem of conducting democratic elections in distributed systems in the form of social choice and social welfare functions with three or more candidates. We present some impossibility and possibility results for distributed democratic elections in presence of Byzantine behavior. We also discuss some existing election schemes, and present a new approach that attempts to mitigate the effects of Byzantine votes. We analyze the performance of these schemes through simulations to compare their efficacy in producing the most desirable social welfare rankings.

1 Introduction

Many problems in distributed systems require *election* for processes to carry out globally consistent actions. For example, the problem of binary consensus can be viewed as an election between two possible choices. The value decided by the protocol can be considered the winner elected by the system. The *leader election* problem requires that all the processes in the system agree on a leader. The agreed upon leader may then perform certain privileged tasks on assuming this role. Most protocols for leader election select processes with the lowest or the highest identifier value as the leader. It can be argued that such a selection on the basis of identifiers does not constitute an 'election' in true sense as the results are not based on the choices of the involved nodes in the system, assuming the nodes can indicate their preferences. Given that one of the fundamental problems in the area of distributed systems, the Byzantine Agreement problem, assumes malicious intent as well as collusion, it seems natural that the problem of fair democratic elections be also studied in this context.

Democratic elections have been studied extensively in the fields of economics and game theory. A large set of interesting problems for elections with three or more candidates have already been explored [1,2]. Arrow's theorem, an important result on this topic, shows impossibility of elections under some specific requirements [3]. Yet, the confluence of democratic elections (with more than two candidates) and distributed protocols has not been explored to the best of our knowledge. Involvement of Byzantine

* Supported by NSF NSF CNS-1115808, CNS-0718990, and the Cullen Trust for Higher Education Endowed Professorship.

D. Frey et al. (Eds.): ICDCN 2013, LNCS 7730, pp. 176–191, 2013.

processes in the system presents some additional challenges for this task. The notion of *strategy-proofness* [4] does not readily apply to Byzantine processes as they can be considered unaffected by individual losses. In this paper, we introduce the notion of democratic elections in distributed settings by addressing the following sub-problems:

Sub-optimality of Standard Protocol: With the background setting of elections, the idea of deciding on a winner based on every node's most preferred choice seems appealing. With this approach the eventual winner would be the node receiving highest number of votes (ties broken arbitrarily). This scheme is called 'plurality' scheme in economics and game-theory literature. However this approach does not always lead to outcomes that maximize the overall gains for the system. The term 'gains' may be attributed to any property that is relevant to global observations of the system, such as the overall latency of a message broadcast or the average load on each node in the system in some distributed computing protocol. For example, let us consider a system with seven nodes $\{P_1, P_2, \ldots, P_7\}$ that run a distributed protocol in presence of a leader node that controls the distribution of work. Based on the resulting latency or load values of their individual communications with three possible candidate nodes, the seven nodes want to elect a leader. Let a,b, and c denote the three possible outcomes of such an election among three candidates. For one such instance of voting assume that Table 1 represents the votes of each node in the system. This tabular representation means that P_1 prefers the outcome b the most, and then prefers a over c; the preferences of all other processes can also be inferred in this manner. 'Plurality' method on this vote profile, with coin-toss based tie breaking, elects b or c with equal probability, and never elects a . However, it is easy to verify that a beats both b and c on individual pairwise comparisons. Additionally, if a positional vote counting mechanism[1], such as Borda-Count Method (see Section 5) [5] is applied, then a's score is strictly higher than those of both b and c. Hence, even though election of a seems the most optimal outcome for the overall system, the standard approach never elects a, and by electing either b or c reduces the social welfare[2] of the system.

Table 1. Votes by Processes P_1 to P_7

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

Strategic Voting by Byzantine Processes: The Byzantine processes can exhibit any kind of malicious behavior. One such malicious act is to cast strategic votes so that the overall social welfare is not maximized. For example, in Table 1, if P_7 is Byzantine, it may broadcast its vote (to all the processes) with c as its first choice and a as its

[1] Considers the positions of each candidate in all the votes, assigning fixed points to each position and then computing aggregate points of every candidate.

[2] Standard term from economics literature; defined in Section 2. For detailed explanations see [6].

last. With the changed vote profile, even the pairwise comparison, or positional voting schemes would not elect a. Thus, this fault would result in decrease of overall social welfare of the system. For binary choices, [7] studies the similar problem when one of the two available choices is more desirable, and it is beneficial for the system to agree on that choice despite the efforts of Byzantine processes.

In short the contributions of this paper are following:

- We introduce the problem of democratic elections in distributed systems by studying: Social Choice and Social Welfare in distributed settings with Byzantine faults.
- We present the impossibility and possibility results for some specific requirements for the problems.
- We propose a social welfare function called Pruned-Kemeny, and by means of simulations show that our scheme significantly outperforms other popular schemes for Byzantine Social Welfare problem.

2 Preliminaries

In economics and game theory, elections have been studied primarily in two forms – social choice functions, and social welfare functions [6]. For both of these forms, the voters are required to cast their vote indicating their preferences over all the candidates. As the result of voting, social choice functions elect one candidate as the winner; whereas social welfare functions produce an overall ranking of the candidates. Formally, these functions are defined as follows:

Let $\mathcal{A}$ be a set of choices/candidates and $\{P_1, ..., P_n\}$ be the set of n voters. Let $\mathcal{L}$ denote the set of linear orders on $\mathcal{A}$ ($\mathcal{L}$ is isomorphic to the set of permutations on $\mathcal{A}$). The preferences of each voter P_i are given by $\prec_i \in \mathcal{L}$, where $a \prec_i b$ means that P_i prefers choice b to choice a. A social welfare function $\mathcal{W}$ is a function of the form $\mathcal{W} : \mathcal{L}^n \rightarrow \mathcal{L}$; and a social choice function C takes the form $C : \mathcal{L}^n \rightarrow \mathcal{A}$.

The preferences of a voter are *strict* if the voter is not indifferent between any two candidates. Throughout the paper, we limit our focus to *strict* preferences. Construction of a social choice function from a social welfare function, and vice-versa is trivial [8]. Given a social welfare function $\mathcal{W}$, one could construct a social choice function by simply declaring the top-most ranked candidate in the result obtained by $\mathcal{W}$ as the social choice. Conversely, to construct a welfare function $\mathcal{W}$ from a given choice function C one could apply C over k candidates and place the winning candidate on top of the result of $\mathcal{W}$, and repeat this processes $k - 1$ times (at each iteration, removing the candidates already placed in the result).

A *ranking* is a total order over a fixed set of candidates. A *vote* is an individual voter's preference ranking over the set of candidates. Based on the above notation, $\prec_i$ is the vote of voter P_i. A *ballot* is a collection of the votes. The size of the ballot is the number of votes the ballot contains. A *scheme* is a mechanism that takes a ballot as input and produces a ranking or a winner as output. Given a ballot, the ranking/winner produced by any scheme is called the result of the scheme on that particular ballot instance.

Condorcet Candidate: If a candidate is preferred by all the voters over each of the other candidates in a head-to-head comparisons, then such a candidate is called *Condorcet Candidate*. It is not necessary that such a candidate always exists.

3 Model

We assume a synchronous distributed system consisting of n processes. In our model any two nodes in the system can communicate privately with each other, thus the induced communication graph is complete. Of the n nodes in the system, at most f can be Byzantine. For the synchronous model of communication, [9] showed that agreement can only be guaranteed when $f < n/3$. Throughout this paper, we assume that this bound of $f < n/3$ holds. All non-faulty processes in the system are called *good* processes, and the faulty processes are called *bad* processes. The terms *voters*, *processes*, and *nodes* represent the same entities, and are used interchangeably. The set of choices, $\mathcal{A}$, is known to all the nodes in the system and each node votes with its *strict* preferences as a total order over the elements of $\mathcal{A}$.

Byzantine Social Choice Problem: 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 elects one candidate as the social choice (while providing the guarantees 1 to 3 listed below).

Byzantine Social Welfare Problem: 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 ranking of the choices (while providing the guarantees 1 to 3 listed below).

Protocol Guarantees

1. *Agreement*: All good processes decide on the same choice/ranking.
2. *Termination*: The protocol terminates in a finite number of rounds.
3. *Validity:* This condition imposes requirement on the choice/ranking decided based upon the preferences provided by the good processes.

If V is the validity condition selected for the election, then $\boldsymbol{BSC}(k, V)$ denotes the Byzantine Social Choice problem over k choices that satisfies the validity condition V; similarly $\boldsymbol{BSW}(k, V)$ denotes the Byzantine Social Welfare problem that is defined with the constraints of V for the available k choices. Some examples of validity conditions are listed in Table 2 in the context of BSC problem.

In the standard Byzantine agreement problem [9], all the good processes must agree on a common value $v \in \mathcal{A}$. The only requirement on the decided value is that if all good processes propose the same value v, then the value decided must also be v. If all good processes do not propose the same value, then there is no requirement on the value that is decided. In Byzantine Social Choice/Welfare problem the value decided by the protocol is important, as some of the choices/rankings may be more desirable than others.

4 Byzantine Social Choice (BSC)

For the Byzantine Social Choice (BSC) problem, we always require agreement, and termination conditions but may want to impose different validity conditions. In the

Table 2. Various Validity Conditions for Byzantine Social Choice

Condition	Description
S	If v is the top choice of all good voters, then v must be the winner.
M	If v is top choice of majority of 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.
P	If v is not the top choice of any good process, then v must *not* be the winner.

standard Byzantine Agreement literature, the problem of deciding from more than two choices is considered equivalent to that of choosing from a set of two choices because a solution for either one of the problems can be used to solve the other [10]. However, as we show in this paper (Sec. 1), this is not the case for the BSC problem. $\boldsymbol{BSC}(k, V)$ denotes a BSC problem over k choices that satisfies the validity condition V. Thus, $BSC(2, S)$ is the standard binary Byzantine Agreement. Note that when k equals two, S, P and S' are equivalent to the standard validity requirements for binary Byzantine Agreement protocol [11]. Similarly, M and M' are equivalent when there are only two choices.

$BSC(3, M)$ is the Byzantine social choice problem on three choices with agreement, termination, and the majority validity condition. We show in Section 4.1 that this problem is impossible to solve in a distributed system. However, somewhat surprisingly $BSC(3, M')$ is possible to solve. As an example of $BSC(3, M')$ consider the problem of leader election in a distributed system with Byzantine processes. Suppose that processes need to choose a leader among three choices. It is known that one of the three choices may be Byzantine and the good processes would want to avoid its election. Although there is no initial agreement on which of these choices is Byzantine, it is a reasonable assumption that majority of good processes will identify the Byzantine choice correctly. In Section 4.2, we give a protocol for solving $BSC(3, M')$.

Observe that $BSC(k, S)$ is simply the standard Byzantine agreement problem in which every process proposes its first choice. Hence $BSC(k, S)$ is solvable for any k so long as $f < n/3$. It is also possible to solve $BSC(k, S \wedge S')$. We give such a protocol in Section 4.2.

4.1 Impossibilities

Arrow [3] showed that for elections with three or more alternatives, no voting system that provides two basic properties: *Pareto Efficiency* and *Independence of Irrelevant Alternatives (IIA)*, can guarantee non-dictatorial elections. In this section, we show impossibilities for elections in distributed systems with Byzantine faults. We focus on instances of $BSC(k, V)$ problems which are impossible to solve for specified values of k and V. Let us first consider the case when k equals two. For this case, the conditions S, P and S' are equivalent. Standard Byzantine agreement protocols can be used to solve $BSC(2, S)$.

Lemma 1. *There is no protocol to solve* BSC$(2, M)$ *when* $f \geq n/4$.

Proof: If $f \geq n/4$, then good processes are at most $3n/4$. Suppose that the set of choices is $\{a, b\}$. Assume that just the bare majority of good processes propose value a. Thus, the total number of processes proposing a is at most $3n/8 + 1$. The number of processes proposing b is at least $5n/8 - 1$. Then for $n \geq 4$, we have that more processes are proposing b. Since processes do not know which processes are good, this problem is indistinguishable from the instance where $5n/8 - 1$ good processes propose b and remaining $3n/8 + 1$ processes propose a. In the second instance, the protocol must choose b, and therefore it will also choose b in the first instance. ■

Lemma 2. *There is no protocol to solve* BSC$(2, M')$ *when* $f \geq n/4$.

Proof: For binary choices, $k = 2$, it can be easily observed that the problem $BSC(2, M')$ is equivalent to $BSC(2, M)$. Thus, based on the result of previous lemma, $BSC(2, M')$ is also unsolvable when $f \geq n/4$. ■

Lemma 3. *There is no protocol to solve* BSC(k, P) *for any* $k \geq n$ *when* $f \geq 1$.

Proof: Given that $k \geq n$, consider the case when each process proposes a different value. Since each value appears exactly once, there is no way to distinguish the value proposed by a bad process from that proposed by a good process. ■

Theorem 1. *There is no protocol to solve* BSC(k, M) *for any* $k \geq 2$ *when* $f \geq n/4$.

Proof: Suppose that there is a protocol that solves $BSC(k, M)$ for any $k \geq 3$. We will use this protocol to solve $BSC(2, M)$. Given an instance of $BSC(2, M)$ problem, all the processes construct an instance of $BSC(k, M)$ by first constructing $k-2$ *artificial* choices. However, none of the good processes use these choices as their first two choices. Now they run the protocol for $BSC(k, M)$ which must choose a value that has been proposed by the majority (ties broken in favor of lower value) of good processes as the first choice. All good processes return this as the decided value for the given BSC $(2, M)$ problem. But by Lemma 1, $BSC(2, M)$ is unsolvable. ■

4.2 Possibilities

As $BSC(k, S)$ is solvable by standard Byzantine agreement [10] and $BSC(k, M)$ is unsolvable, it is natural to seek some validity conditions that admit solution. Consider the following validity condition:

M_o *(Overwhelming Majority)*: If there is a choice that is the first choice of at least $3/4^{th}$ good processes, then all the good processes elect that choice.
It can be observed that any protocol that ensures M also ensures M_o. Similarly M_o is a stronger requirement than S, and thus any protocol providing guarantee on M_o also guarantees S.

Lemma 4. *Protocol* α *given by Algorithm 1, solves* BSC(k, M_o) *when for any* $k \geq 2$ *when* $f < n/3$.

Proof: Let v be the value proposed by at least $3/4^{th}$ fraction of processes. It is easy to see that $3/4 * (n - f) > 1/4 * (n - f) + f$ for all values of $f < n/3$. Hence, all processes decide on v. ■

Algorithm 1. Protocol α at P_i to ensure $BSC(k, M_o)$ and therefore also $BSC(k, S)$

```
T: array[1..n] (container to store all the votes)          /* Proposals */
vote: array[1..k] (ranking of k candidates)                /* My vote */
/* Every process proposes its first choice */
T[i] = vote[1]                                             /* index starts from 1 */
/* Step 1: Exchange first choice with all */
for j = 1 to n do
   send T[i] to P_j
   receive T[j] from P_j
   /* if no value received from P_j set T[j] = 0 */
end for
/* Step 2: Agree on T vector : the ballot of all votes */
for j = 1 to n do
   run Standard Byzantine Agreement on T[j];
end for
/* Step 3: Choose the value with the highest tally, breaking ties in favor of the smaller value */
return the least value from 1..k that has the highest frequency in T.
```

We showed in Section 4.1 that P is impossible to achieve when $k \geq n$. However, if choices are limited, then P can be guaranteed as follows.

Lemma 5. *Protocol α solves* BSC(k, P) *for* $2 \leq k < n$ *when* $f < min(n/k, n/3)$.

Proof: It is sufficient to show that the largest tally would be of a value proposed by a correct process. Suppose, if possible, the largest tally is for the value v which is not proposed by any good process. The tally for v can be at most f. There are $n - f$ proposals by good processes. None of these proposals is for v, and therefore all these proposals are for remaining $k - 1$ values. Since none of these values had tally more than v, we get that the total number of proposals possible are $(k - 1) * f$. From $f < n/k$, we obtain that $(k - 1) * f < n - f$ which is a contradiction because all correct processes make at least one proposal. ■

However, if we were to require $(M' \wedge P)$ and use the steps in the protocol α with suitable adjustments (not picking a social choice that would violate M') to handle the validity requirements – it would be evident that the modified protocol α would not satisfy $(M' \wedge P)$. It is not possible for a protocol to deterministically know which nodes are *good* and which are *bad* in all the instances. Thus to provide M' the only option any deterministic protocol would have to discard a choice that appears $\lfloor (n - f)/2 + 1 \rfloor$ or more times as the bottom choice in the ballot. Consider the example ballot presented in Table 3, with P_6 and P_7 as Byzantine voters. In this example, a simple majority over the first choices would result in choosing c as the winner which violates M'. The modified protocol α (that attempts to provide M') will elect a as the winner. However, an overwhelming

Table 3. A ballot with P_6 and P_7 as Byzantine Voters

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

majority of *good* processes, 4 out of 5, prefer b over a. Note that the choice a is not the first choice for any process, leave alone being the first choice of a *good* process. In this example, with $n = 7$ and $f = 2$, $\lfloor (n - f)/2 + 1 \rfloor$ is 3 and thus it is clear that any protocol that guarantees M' can only choose a as winner (because both b and c are last choices for at least three processes).

We now show the surprising result that $BSC(k, M' \wedge S)$ is solvable for $k \geq 3$ when $f < n/3$. Protocol β, shown in Algorithm 2, is based on the idea of processes proposing their last choice. Since Byzantine processes may send conflicting values to different processes, Protocol β first agrees on the vector T of last choices. Each process then discards the values that appear as the last choice at least $\lfloor (n - f)/2 + 1 \rfloor$ times. It should be noted that since $f < n/3$, the size of *discard* set in protocol β is at most two. Now all the processes run Byzantine Agreement with their top choice from the remaining set.

Algorithm 2. The $BSC(k, M' \wedge S)$ Protocol β at P_i

```
T: array[1..n] (container to store all the votes)          /* Proposals */
vote: array[1..k] (ranking of k candidates)                /* My vote */
/* Every process proposes its last choice */
T[i] = vote[k]
/* Step 1: Exchange last choice with all */
for j = 1 to n do
   send T[i] to P_j
   receive T[j] from P_j
   /* if no value received from P_j set T[j] = 0 */
end for
/* Step 2: Agree on T vector, ballot of last choice votes */
for j = 1 to n do
   run Standard Byzantine Agreement on T[j];
end for
/* Step 3: Eliminate unqualified choices */
discard = set of choices to discard; initially {φ}
for j = 1 to k do
   /* count returns the frequency of any value in T */
   if (count(vote[j]) >= ⌊(n − f)/2 + 1⌋) then
      add vote[j] to discard
   end if
end for
/* Step 4: Now use the remaining choices for selecting top choices of processes */
run Byzantine Agreement on top choice ∉ discard
```

Lemma 6. *Protocol β, given by Algorithm 2, solves* BSC$(k, M' \wedge S)$ *for* $k \geq 3$ *when* $f < n/3$.

Proof: We first note that if P_i is good then $T[i]$ at P_j will be same as the value proposed by P_i. This means that if there is any value v that is considered the last choice by a majority of good processes then it appears at least $\lfloor (n-f)/2 + 1 \rfloor$ times in T vector; all such values are discarded. Since $k \geq 3$ and $|\ discard\ | \leq 2$, there is at least one value which is not discarded by any good process. Hence, the agreement phase in step 4 leads to selection of a choice proposed by some good process.

It is also easy to verify that the protocol satisfies S. If all good processes have v as their first choice, then it cannot appear $\lfloor (n-f)/2 + 1 \rfloor$ times as the last choice. Hence no good process will discard this choice and will propose it in step 4. ■

Lemma 7. *Protocol β does not guarantee M_o condition.*

Proof: Consider the vote ballot presented in Table 4 in which 4 out of 5 *good* processes have b as their first choice. Since it can not differentiate between *good* and *bad* processes based on the ballot, protocol β would be forced into electing a as the social choice. ■

Table 4. A ballot with P_6 and P_7 as Byzantine Voters

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

5 Byzantine Social Welfare (BSW)

The problem of Byzantine Social Welfare can be seen as an extension to the BSC problem. In the Byzantine Social Welfare (BSW) problem, the goal is to produce a *ranking*, a total order over k candidates, of choices as the result of elections. Multiple such schemes exist in the literature of economics and game theory. We now discuss some of these as social welfare functions, and propose a new scheme called Pruned-Kemeny specially tailored towards handling Byzantine votes. We focus only on the schemes that require a single round of voting. After exchanging their votes with all the other processes in the system, the processes participate in $O(f)$ rounds of agreement to ensure that all the good processes agree on the same ballot.

From here on, for notational convenience we use a short form representation of rankings such that *abc* represents ranking $a \succ b \succ c$.

PlacePlurality: For each position in the result ranking, the scheme finds the candidate with most votes for that position in the ballot, and places this candidate at that position in the result. Only the candidates that are not already placed in the result ranking are considered. Plurality based schemes satisfies S and S' criteria. Revisiting the example ballot of Table 1 from Section 1, one can verify that the rankings *cab* and *bac* are the two possible outcomes of a social welfare function that applies PlacePlurality.

Pairwise Comparison: This scheme uses the Condorcet Criterion and compares the pairwise preferences over the ballot for all $\binom{k}{2}$ pairs. Detailed description of the scheme is presented in [12]. Using this scheme, the output for the welfare function on the ballot of Table 1 is *abc*.

Borda Count: This scheme applies the positional voting approach to calculate the points scored by each candidate. Each position in a vote is assigned unique points - top position given the highest, and the bottom position given the lowest points. The cumulative score of a candidate is the sum of all the points it accumulates over the complete ballot. The resulting social welfare ranking is the list of candidates sorted in non-increasing order of their overall scores (ties broken arbitrarily). For the ballot of Table 1, the social welfare result using Borda Count scheme is *abc*.

Young in [13] shows with convincing arguments that most of the simple schemes, including the three presented above, do not necessarily produce the outcomes that are optimal in terms of overall representation of the voter preferences. The discussion in [13] points out that schemes that use the *mean* as the representative outcome of the voting process tend to generate 'inferior' results as opposed to the schemes that try to compute an outcome that is close to the *median*. The following two schemes, use the median as the basis for the result computation, and we show by means of simulation that they do produce *better* social rankings.

Kemeny-Young Scheme: This approach, proposed by J. Kemeny and H. Young in [14,13], uses a metric to identify a ranking that is closest to the median of the ballot. The metric used in this scheme is the *distance* between rankings, where *distance* between any two rankings is defined as the number of pairs on which the rankings differ. For example, taking $r = abc$ and $r' = bac$ the *distance* between these two rankings is 1, whereas if $r' = cba$, then the *distance* between r and r' is 3.

Algorithm 3 presents the steps involved in the scheme. The scheme iterates over each of the possible $k!$ permutations of k candidates and considers each ranking (permutation). The goal is to identify a ranking that maximizes the agreement on pairwise comparisons with the overall ballot. For a detailed analysis of the scheme, we refer the reader to [13]. Applying this scheme on the ballot in Table 1 gives a result ranking of *abc*.

Pruned-Kemeny: We propose a scheme called Pruned-Kemeny that is aimed towards mitigating the damaging effects of *bad* voters. The key motivation for this scheme is that good voters, while indicating their individual preferred choices would in addition be also inclined towards the final outcomes that are beneficial to the overall system. Where as the *bad* voters would not only send conflicting information to the *good* voters, but also focus on manipulating their vote preferences in order to reduce the overall welfare of the system. The steps of the scheme are presented in Algorithm 4. Similar to the Kemeny-Young scheme, our approach also iterates through all the possible permutations of candidates but we restrict the ballot in consideration for each iterated permutation. The restriction on the ballot is attained in the following manner:

Let $\mathcal{P}$ denote the set of all permutations of k candidates, and $\mathcal{B}$ be the initial ballot of n voters. For each ranking $r \in \mathcal{P}$ compute a pruned ballot $\mathcal{B}'$ by setting $\mathcal{B}' = \mathcal{B} \setminus \mathcal{F}$, where $\mathcal{F}$ is the collection of f most *distant* rankings in $\mathcal{B}$ from r. Hence, size of the

Algorithm 3. Kemeny-Young Scheme

```
P = Set of all permutations of k candidates,
B = agreed upon ballot of n votes
maxScore= 0, maxRank = nil
for each ranking r in P do
   score = Kemeny-YoungScore(r, B)
   if score > maxScore then
      maxScore= score
      maxRank = r
   end if
end for
return maxRank

Kemeny-YoungScore(ranking, ballot):
  score = 0
  for each pair (a>b) in ranking do
    score = score + # of a>b in ballot
  end for
return score
```

Algorithm 4. Pruned-Kemeny Scheme

```
P = Set of all permutations of k candidates,
B = agreed upon ballot of n votes
maxScore= 0, maxRank = nil
for each ranking r in P do
   F = f most distant rankings from r in B
   B' = B \ F
   score = Kemeny-YoungScore(r,B')
   if score > maxScore then
      maxScore = score
      maxRank = r
   end if
end for
return maxRank
```

restricted ballot $\mathcal{B}'$ is $n - f$. The score for ranking r is its Kemeny-Young score on $\mathcal{B}'$. The result of the scheme is the ranking with highest score (ties broken arbitrarily). For instance, when applied to the ballot of Table 1, this scheme produces *bac* as the result.

We show shortly that the problem of finding a solution to the election problem using either Kemeny-Young or Pruned-Kemeny scheme is NP-Hard. In the context of distributed systems, the round and message complexities for agreement on the the ballots, performed before application of the schemes, are essentially the complexities of the protocols used reach agreement. We use the Gradecast based Byzantine agreement protocol presented in [15], mainly because this protocol provides the early termination property. Based upon this, the agreement requires $O(f)$ rounds, and has the message complexity of $O(fn^3)$. For proofs and detailed discussions on these bounds we refer the reader to [15].

Lemma 8. *The problem of finding the result of a ballot using Pruned-Kemeny scheme is NP-Hard.*

Proof: Consider any instance of the problem of finding optimal rankings with Kemeny-Young scheme. Each such instance can be converted to an instance of the problem of finding the result with Pruned-Kemeny with f set to zero. Hence, the Pruned-Kemeny based optimization problem is at least as hard as the Kemeny-Young based problem, which is already known to be NP-Hard [16]. ■

Theorem 2. *Pruned-Kemeny satisfies S and S' requirements.*

Proof: First, we prove that Pruned-Kemeny satisfies S. Let us assume that Pruned-Kemeny violates S, and thus its output is ranking r that does not put v on top when all the good processes put v as their top choice. Hence, there is at least one candidate u that

is immediately above v in r. Construct a new ranking r' by swapping the places of u and v in r. We now show that r' would have a higher Kemeny-Young score than r, which would be a contradiction. Since r' puts v above u, it agrees with all the good processes on at least one more pairwise comparison. It may disagree with the votes of all the bad processes. Also, in the worst case scenario r' discards f good votes during the protocol run. Thus in the worst case the overall Kemeny-Young score of r' increases by

$$(n - f) - f - f = n - 3f$$

in comparison to the Kemeny-Young score of r. The first term of $(n - f)$ is due to the increment (by at least one point) in score for each *good* vote, however if we assume that it is possible to discard f *good* votes in the worst case, the second term indicates that adjustment. Also, all the *bad* processes might provide exact opposite rankings in their votes, hence a further decrease of f (third term) in points is possible in the worst case. Since $n \geq 3f + 1$, the score of r' is strictly greater than that of r, which means r being selected as the final outcome of Pruned-Kemeny is a contradiction. S' can be shown similarly by placing v at the bottom of each good vote. ■

Similar to the Kemeny-Young scheme, Pruned-Kemeny also performs exponential computations by iterating over all the $k!$ permutations. However, for small values of k and large values of n, the performance of the scheme is acceptable.

Lemma 9. *Kemeny-Young scheme satisfies S and S' requirements.*

Proof: As Kemeny-Young is a special case of Pruned-Kemeny scheme with f set to zero; the proof immediately follows from Theorem 2. ■

6 Simulation Results

It is possible to have scenarios in which the *bad* voters need not just send conflicting information, but may as well have much more malignant intentions. Consider the case when the *good* voters want to reach a consensus on a ranking that is beneficial to the system as a whole, and thus have similar if not exactly the same preferences. On the other hand, the *bad* voters may want to minimize the benefit that the system may attain by the resulting welfare ranking (that is the outcome of the election). Schemes that do not assume that a small section of voters might behave in this manner, may thus produce rankings which are prone to manipulation by the *bad* voters. Given the knowledge that at most f voters can be *bad*, our scheme Pruned-Kemeny tries to produce best possible social welfare outcomes in presence of such hostile voting by the *bad* voters.

We now list the details of our experimental setup and the simulations performed to evaluate the utility of Pruned-Kemeny in computing 'near-optimal' welfare rankings in comparison to the other discussed schemes. Let ω represent an *ideal* ranking for the BSW problem, such that selection of ω as the result of the election maximizes the social welfare of the system. Let us assume that ω is not completely known to any good process, however each good process tends to favor the *ideal* ranking. The voting preferences of good and bad processes in presence of an *ideal* ranking are defined as follows:

Let *goodProb* denote the probability of a good voter ranking two candidates a and b in the same order as that in the ideal ranking ω, and *badProb* denote the probability of a *bad* voter ranking the candidates in the reverse order to that in ω. Hence, if ω ranks two candidates a and b with $a \succ b$ then *goodProb* is the probability that any *good* voter decides to put $a \succ b$ in its vote, and *badProb* is the probability that any *bad* voter puts $a \prec b$ in its vote. For our experimental setup we fix the following values:

$$n = 100, f = 33, badProb = 0.9$$

By setting f to its highest possible value, and *badProb* to a considerably high value in the possible range, we try to realize the assumption that *bad* voters would want to disrupt the election of ideal ranking, and would vote in opposite polarity of the *good* voters. The value of the number of candidates k is varied in the range [3,8]. For each value of k, the value of *goodProb* is varied from 0.55 to 0.90 in step increments of 0.05. For each such resulting configuration of $\langle k, n, f, goodProb, badProb \rangle$, 50 ballots (of $n = 100$ voters) are generated by fixing an *ideal* ranking and applying the probabilistic model on individual votes based on *goodProb* and *badProb*. We then apply the discussed schemes, and find the *distance* (defined in previous section) of their result rankings from the *ideal* ranking. We then compute the average distance over the 50 ballots for each configuration.

Figure 1 shows the variation in the average distance values. As evident from the plots, Pruned-Kemeny produces results that are much closer to the *ideal* ranking even for comparatively low values of *goodProb*. In addition, the plots also indicate that as the number of candidates increases, the results of Pruned-Kemeny consistently match the *ideal* ranking. Another interesting observation is that the distance of results for PlacePlurality from the *ideal* ranking increases significantly with increase in the number

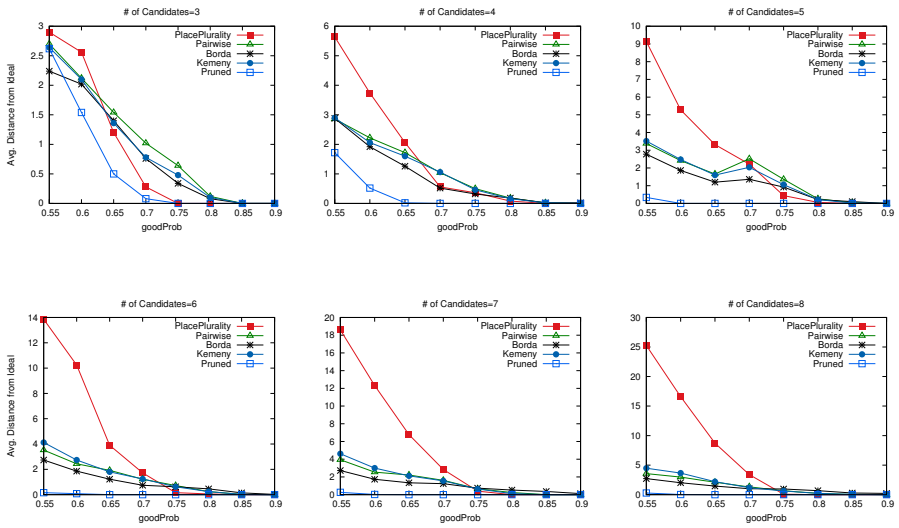

Fig. 1. Comparison of Average Distance of Results from Ideal

of candidates. This clearly indicates that PlacePlurality is not a good choice for a social welfare function. We argued in Section 1 that for more than three choices, the plurality based methods do not guarantee best results. The observations on the variation in result distances clearly validate our argument.

7 Discussion

Extensive literature is already present on the topic of leader election in distributed systems [11,17,18,19]. [11] presents various protocols and lower bounds for message complexity for the leader election problem in absence of Byzantine processes. Leader Election has also been studied in presence of Byzantine failures. [20] gives a randomized distributed protocol to elect a leader in the asynchronous full information model that tolerates $n/(6+\epsilon)$ cheaters with positive constant success probability in rounds that is polylogarithmic in n.

Our work studies the problem of democratic elections in a distributed system as social choice and social welfare ranking problems [6]. When number of choices is more than two, elections based on the top-preference-only model may not lead to optimal results, and hence we assume that processes in the system propose a ranking of candidates rather than a single leader. For agreement that is dependent only on the number of failures we use the deterministic early-stopping Byzantine agreement protocol from [15] to reach the agreement on every processes' vote within $min\{f+1, f_a+2\}$ rounds where f_a is the actual number of failures. We focus on the guarantees on the social choice or the social welfare ranking produced by the election, rather than on the message or bit complexity of election protocols.

Prisco et al. in [21] present some impossibility and possibility results for the *k-set* consensus problem in which each node starts with one value and the protocol must decide on a value so that at most total k values are decided by the correct processes. The *k-set* problem does not involve voting over multiple candidates. Under some specific boundary conditions there is a slight overlap between two impossibility results in [21] and those presented in this work.

In the standard Byzantine agreement [9] the protocols only need to guarantee agreement on some value that is proposed by a good process. With this objective, the protocols do not need to guard against the possibility of Byzantine voters affecting the eventual outcome by strategic reporting of their values. However, as we saw in Section 3 it is important to design voting mechanisms that do not allow this advantage to Byzantine voters.

8 Future Work

If all the good processes lean towards some fixed *ideal* ranking, even with weak inclinations, the simulation results indicate that our proposed approach Pruned-Kemeny provides desired results with much higher accuracy in comparison to other schemes. However, determining the provable guarantees for optimal results under some specific conditions is an important open challenge for this work.

Another interesting problem is to differentiate between the ideal results, and the results that comply with the Condorcet Criterion. It should be noted that for some given ballot, it is possible to have a clear Condorcet candidate/ranking yet the ideal winner/ranking might differ from it. However, in terms of computational complexity both Kemeny-Young and Pruned-Kemeny schemes are NP-Hard, where as a Condorcet candidate/ranking can be found in polynomial time. With this observation, it would be beneficial to design a social welfare scheme that can strike a balance between these two approaches. Depending on the constraints of the computing environment, this balanced scheme could have the flexibility to employ either the PrunedKemeny or the Condorcet scheme so that the difference between the social welfare resulting from the two outcomes is either relatively small or bounded in some acceptable form.

9 Conclusion

In this paper, we introduced the problem of democratic elections in distributed systems. We showed that the standard approach of reducing three or more choices to binary choices does not guarantee optimal outcomes, and hence the standard assumption of always having binary choices is weak. We presented impossibility results under some specific validity requirements, as well as showed some surprising possibilities that result from availability of more than two choices.

For producing results that are close to an ideal ranking when there exists one, we proposed a new scheme called Pruned-Kemeny that aims to counter the votes of Byzantine processes. The results of our simulations show that for the purpose of finding ideal order, Pruned-Kemeny provides significantly improved results over existing voting systems.

References

1. Arrow, K.J.: Social Choice and Individual Values. Yale University Press (1951)
2. Farquharson, R.: A Theory of Voting. Yale University Press (1969)
3. Arrow, K.J.: A difficulty in the concept of social welfare. Journal of Political Economy 58, 328–346 (1950)
4. Ishikawa, S., Nakamura, K.: The strategy-proof social choice functions. Journal of Mathematical Economics 6, 283–295 (1979)
5. Saari, D.G.: Mathematical structure of voting paradoxes: Positional voting. Economic Theory 15, 55–102 (2000)
6. Graaff, J.V.: Theoretical Welfare Economics. Cambridge University Press (1957)
7. Garg, V.K., Bridgman, J., Balasubramanian, B.: Accurate Byzantine Agreement with Feedback. In: Fernàndez Anta, A., Lipari, G., Roy, M. (eds.) OPODIS 2011. LNCS, vol. 7109, pp. 465–480. Springer, Heidelberg (2011)
8. Buchanan, J.M.: Social choice, democracy, and free markets. Journal of Political Economy 62, 114–123 (1954)
9. Pease, M., Shostak, R., Lamport, L.: Reaching agreement in the presence of faults. Journal of ACM 27, 228–234 (1980)
10. Turpin, R., Coan, B.A.: Extending binary byzantine agreement to multivalued byzantine agreement. Inf. Process. Lett. 18, 73–76 (1984)

11. Lynch, N.: Distributed Algorithms. Morgan Kaufmann Publishers (1996)
12. Young, H.P.: Condorcet's theory of voting. American Political Science Review 82(4), 1231–1244 (1988)
13. Young, H.P.: Optimal voting rules. Journal of Economic Perspectives 9(1), 51–64 (1995)
14. Kemeny, J.G.: Mathematics without numbers. Daedalus, Quantity and Quality 88(4), 577–591 (1959)
15. Ben-Or, M., Dolev, D., Hoch, E.N.: Simple gradecast based algorithms. CoRR, vol. abs/1007.1049 v3 (2010)
16. Bartholdi, J., Tovey, C.A., Trick, M.A.: Voting schemes for which it can be difficult to tell who won the election. Social Choice and Welfare 6(2), 157–165 (1989)
17. Ostrovsky, R., Rajagopalan, S., Vazirani, U.: Simple and efficient leader election in the full information model. In: Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, STOC 1994, pp. 234–242 (1994)
18. Feige, U.: Noncryptographic selection protocols. In: Proceedings of the 40th Annual Symposium on Foundations of Computer Science (1999)
19. Russell, A., Zuckerman, D.: Perfect information leader election in log*n + o(1) rounds. In: Proceedings of the 39th Annual Symposium on Foundations of Computer Science, FOCS 1998, pp. 576–583 (1998)
20. Kapron, B., Kempe, D., King, V., Saia, J., Sanwalani, V.: Fast asynchronous byzantine agreement and leader election with full information. In: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008, pp. 1038–1047 (2008)
21. Prisco, R.D., Malkhi, D., Reiter, M.: On k-set consensus problems in asynchronous systems. In: Proceedings of the 18th ACM Symposium on Principles of Distributed Computing (PODC 1999), pp. 257–265 (1999)