The study of mutation tolerance of voting mechanisms

Akshat Kumar[1], Peng Peng[2], and Jie Peng[3]

[1] University of Waterloo 20548735 a77kumar@uwaterloo.ca
[2] University of Waterloo 20560954 p8peng@uwaterloo.ca
[3] University of Waterloo 20564547 j29peng@uwaterloo.ca

Abstract. As the world becomes an increasingly connected place, it is becoming apparent many decision problems can translated into social choice problems, due to this we believe that evaluation of voting mechanisms deserves more attention. By this study, we wish to examine the noise-tolerant of several popular voting mechanisms using noise models. We devised five different mutations that we found to have practical context. From the results, we found that the stability of voting mechanisms is not just related on the degree of the mutation but is also related to the kind of mutation.

1 Introduction

Group decision problems are becoming more and more prevalent thanks to grater connectivity from social networking and media. Voting is a prevalent medium of implementing social choice or aggregating the choice of a group, where the concerned group is large in size and also is there multiple candidates. The popularity and the large number of social choice problems has led to the development of numerous social choice functions, some of these are supposed to be better than the others. The paper[2] talks about the existence of a True Ranking that is the correct solution to the social choice problem, it also proposes that the preference can be viewed as a noisy interpretation of the true ranking.This study compares some popular social functions based on the empirical data gathered by experiments that test the stability of the results of the social function when the preferences are altered randomly. By this study we aim to compare popular voting mechanisms based on the deviation experienced after preference mutation.

1.1 Our Model and Result

The preference dataset that we used in our study is from Preflib.org, a website providing a reference library of preference data. We choose a voting dataset for Sushi which contains thousands of complete strict rank orders of 10 different kinds of sushi. All of this data is processed as input by four prevalent voting mechanisms and then produces corresponding winners. The noise models that are induced to the original ranking are generated based on consideration of actual voting. Due to the randomicity of the noise, we iterate the experiment a thousand times to investigate the tendency of variation instead of only studying limited numbers of instances. While comparing the deviation between the original ranking and the mutated one, we take advantage of Kendall Tau Distance as the measurement to compare the entire rankings instead of only comparing the winners. The results show notable difference resilience to noise among different voting mechanisms. Moreover, the degree of the noise also affects the performance of voting mechanisms.

1.2 Related Work

Many researchers showed that the votes or rankings of choices can be interpreted as a noisy perception of this correct outcome. They think there is a true rankings behind the votes. They believe the absolute truth that some candidates are better than others and this is independent on the voters preference. The preferences are only the noisy estimation of this absolute truth. So they proposed methods to compute the maximum likelihood estimate of the correct rankings. Vincent and Tuomas[1] required that the votes are drawn independently given the correct outcome and they thought without this restriction, all voting rules have the property of be seen as noise of the true rankings. They studied the cases both for where outcomes are winners and for where outcomes are rankings. Their results showed that only some of the common voting rules have this property.

While in [5], Matthew and Lirong tried to solve whats the optimal preference function? thus the result is the MLE of the true ranking. They defined simple ranking scoring functions (SRSFs) and show SRSFs satisfies this kind of property. The authors also defined composite ranking scoring functions (CRSFs) and show it coincides with SRSFs. The key properties they chose are consistency and continuity. Finally, they drew the conclusion that any PF that is consistent, continuous, and neutral is an SRSF. Another conclusion is that any PF that is consistent and neutral is a CRSF.

Although many of the researchers had the view that the preferences of the candidates are noisy estimation of the true ranking, none of them tried to explore the magnitude of the noise and which kind of the noise the preferences have. There is no model of the noise and showing how the noise influences the ranking. So our work is tried to explore the magnitude of the noise and which kind of noise the preferences have.

2 Voting Mechanisms

For our study, we are going to use popular mechanisms used in large elections involving a large number of voters and sufficiently larger number of candidates as well. We want to use popular mechanisms as we want to make the study relevant to real world scenarios.

We plan to consider the following mechanism for our study:

1. Single Transferable Vote (STV)
2. Instant-runoff voting (IRV)
3. Simple Plurality
4. Plurality at Large

2.1 STV

The single transferable vote (STV) is a voting system designed to achieve proportional representation through ranked voting. Implemented in large settings in Parliamentary elections in Ireland, Australia, Upper house of Parliament elections and many more.

2.2 IRV

Instant-runoff voting is an electoral system used to elect a single winner from a field of more than two candidates. It is a preferential voting system in which voters rank the candidates in order of preference rather than voting for a single candidate.

Ballots are initially distributed based on each elector's first preference. If a candidate secures more than half of votes cast, that candidate wins. Otherwise, the candidate with the fewest votes is eliminated. Ballots assigned to the eliminated candidate are recounted and assigned to those of the remaining candidates who rank next in order of preference on each ballot. This process continues until one candidate wins by obtaining more than half the votes.

Instant-runoff voting is used to elect members of the Australian House of Representatives and most Australian State Governments, the President of India, members of legislative councils in India, the President of Ireland, and the parliament in Papua New Guinea. It is also used in Northern Ireland by-elections and for electing hereditary peers for the British House of Lords.

2.3 Simple Plurality

In simple plurality voting each elector has a single vote, and this is cast, in single member constituencies, for one candidate. The winner will be the candidate who has more votes than any other individual candidate.

The most common system, used in Canada, the lower house (Lok Sabha) in India, the United Kingdom, and most elections in the United States, is simple plurality.

Approval voting was used for papal conclaves, in Venice in the 13th through 18th centuries and the selection of the Secretary-General of the United Nations has involved rounds of approval polling

2.4 Plurality at Large

By treating each candidate as a separate question, "Do you approve of this person for the job?" approval voting lets each voter indicate support for one, some, or all candidates. All votes count equally, and everyone gets the same number of votes: one vote per candidate, either for or against. Final tallies show how many voters support each candidate, and the winner is the candidate whom the most voters support.

3 Noise Models

3.1 General Noise Models

Voting, as the most common approach of aggregating preferences from multiple agents, has been studying for decades and is still attracting scholars all over the world. However, from another perspective, voting can be viewed as a maximum likelihood estimation problem, since most voting has its true result, which is based on the absolute goodness of the candidates rather than the votes that voters provide. In this case, voters votes become some noises, compared with the ideal result. Vincent Conitzer and Tuomas Sandholm first proposed the study on how common voting rules can be interpreted as maximum likelihood estimation problems in 2012[2]. Actually, investigation of noise models dates back to hundreds years ago when Marquis de Condorcet [6] presented the earliest noise model in which voters are required to vote for a pair of candidates. The probability of voting correctly is p >1/2 and voting incorrectly is 1 - p. Afterwards, scholars constantly improve the system of noise models[3][4]. One of the noteworthy studies which probably can be used to our project is Critchlow et al.s work in which they introduced four types of noise models[5].

Nevertheless, the aforementioned noise models are inherent defects of some voting mechanisms, which is not the goal of our study. Thus, we propose new noise models based on consideration of actual context in voting, and mathematical analysis. The random mutation noise is designed for simulating the scenarios where voters misunderstand the voting rule, e.g., reverse the orders, and the mutation based on mathematical equations like normal distribution and Poisson distribution simulates the scenarios where voters hardly make choice from candidates with similar goodness. Normal noise model simulates the situation in which the candidates are similarly good thus the voters will hesitate to vote. For the destructive mutation, the voters try to construct all the rankings from the least good one to the best good one, namely every time choosing the worst one. For the constructive mutation, the voters try to construct all the rankings from the best one to the least good one, just in the opposite direction of the destructive mutation.

Details of our own noise models are mentioned below.

3.2 Proposed Noise Models

3.2.1 Random Noise

Random noise simply means mutating the original rankings randomly with different degrees. We divided this noise models up into two types, which are mutation on both extremes and mutation in the middle of rankings. Particularly, as the number of candidates of each ranking is 10, we define the first two and the last two ranks as the ranks on extremes, hence the remaining six are ranks in the middle. The Figure 1 shows the distribution of original ranking, where horizontal axis denotes the index of rankings and vertical axis denotes the number of votes at this rank.

The four ranks on extremes are extracted and shuffled. Then we insert them back to the original rankings. The Figure 2 shows the distribution of rankings with mutation on both extremes.

The six ranks in the middle are extracted and shuffled. Then we insert them back to the original rankings. The Figure 3 shows the distribution of rankings with mutation in the middle.

3.2.2 Normal Noise Model

The normal noise model is based on the normal distribution. The normal distribution is remarkably useful because it shows that averages of random variables independently drawn

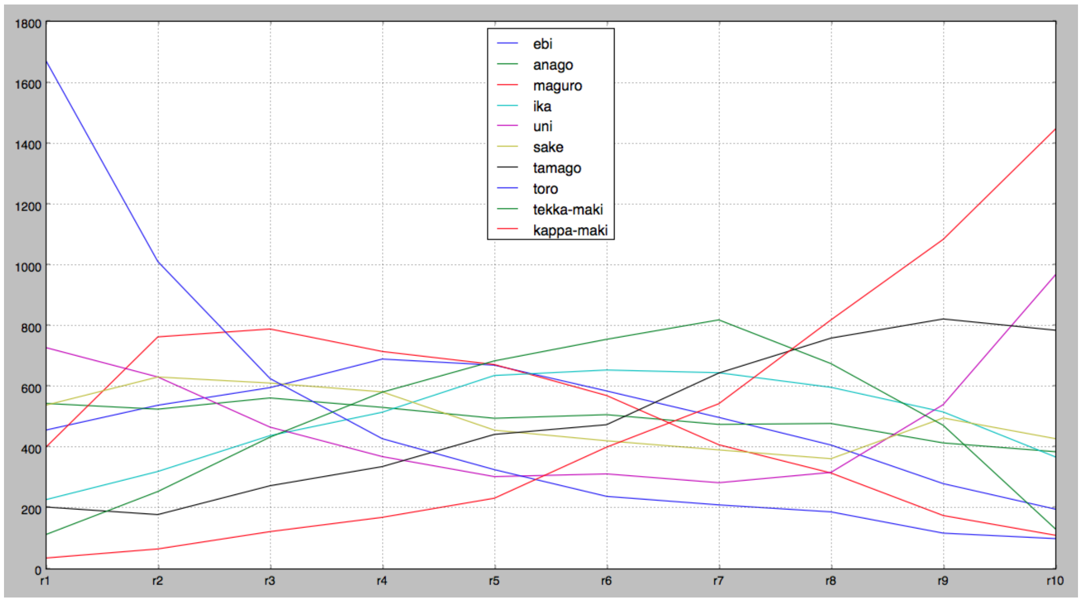

Fig. 1. Original Ranking.

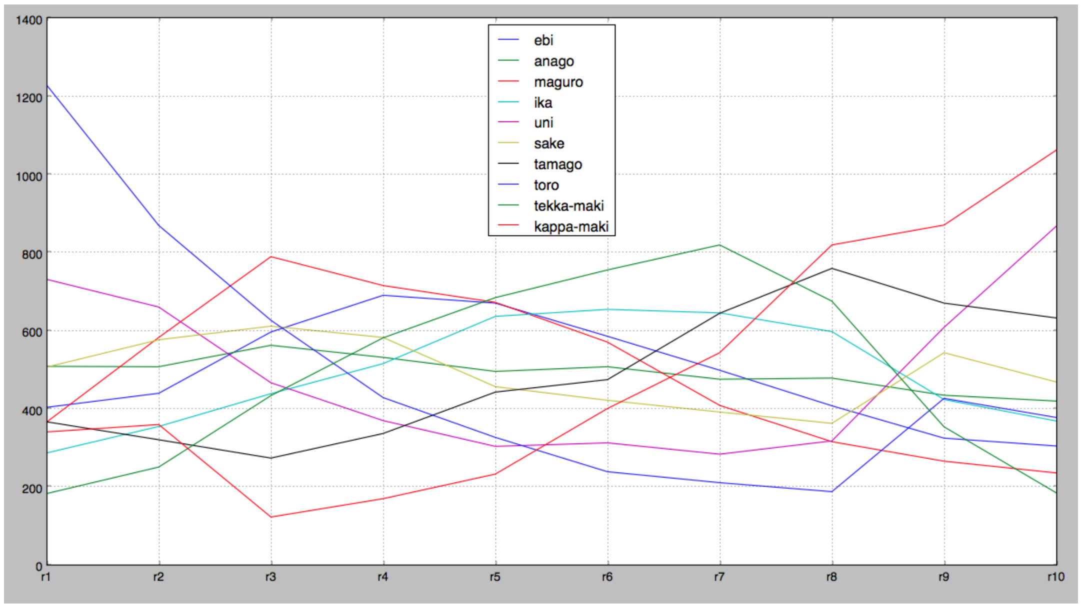

Fig. 2. Mutation on extremes.

from independent distributions are normally distributed. For example, if you flip a coin for

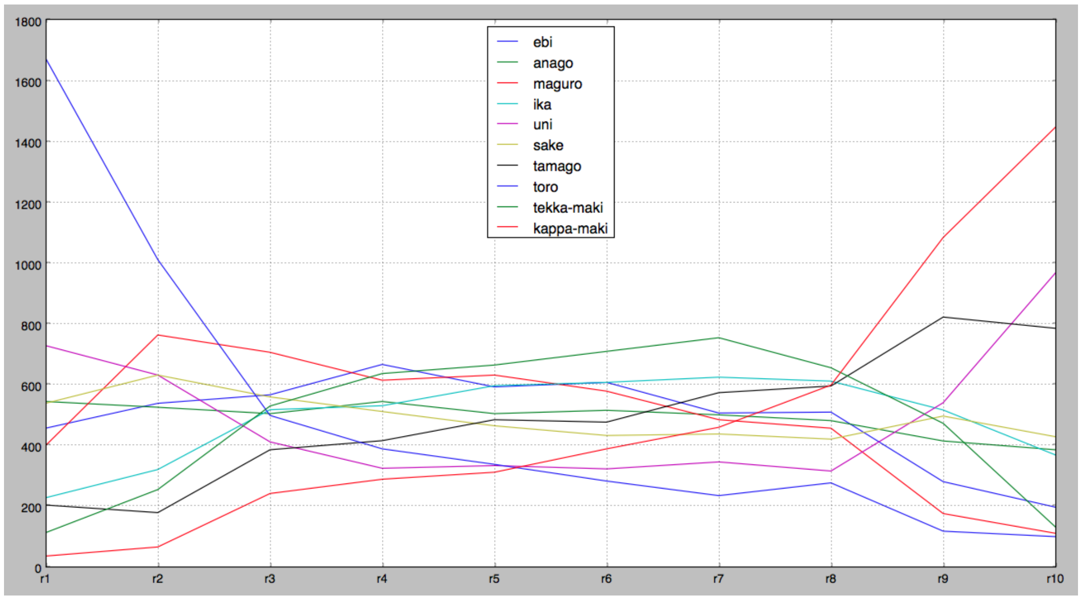

Fig. 3. Mutation in the middle.

100 times, then the number of times the head is the front side obey normal distribution. So the normal distribution is very useful when you what to model the sum of independent, identically distributed distributions.

As we think the voters votes are independent, identically distributed distributions, the result of the ranking has some kinds of noise which applies to the normal noise distribution. Especially when the candidates can be ranked easily, the voters know which candidate is good or bad, but the candidates in the middle cannot be determined easily. In our design, we assume the preference before mutation is the true preference of the voters and we want to exert the normal noise to the preference thus to test the normal-noise-resilience of every voting mechanism. For more details of the implementation, please refer to algorithm 1.

3.2.3 Destructive and Constructive Manipulation

In probability theory and statistics, the Poisson distribution is a very famous discrete probability distribution. It expresses the probability of certain number of events occurring in a fixed interval of time or in a certain space. The rate of the event should be certain average value, which is the definition of λ. The occurrence of this event should be independently of the time since the last event. Then when can model the occurrence of this event using Possion Distribution. A discrete random variable X is said to have a Poisson distribution with parameter λ >0, if, for K is a natural number, the probability mass function of X is given by(1)

$$f\left(k;\lambda\right) = Pr\left(X = k\right) = \frac{\lambda^{k}\epsilon^{-\lambda}}{k!} \tag{1}$$

Algorithm 1 Normal Distribution

```
procedure NORMALNOISEMUT ((raw_pref))
    $new\_pref = raw\_pref$
    for $rowinnew\_pref$ do
        for index from 0 to 9 do
            if $random(0,1) < degree * gaussianFunc(index)$ then
                $tem = random(0,9)$
                $origin\_vote = row[index]$
                $row[index] = tem$
                $origin\_index = row.index(tem)$
                $row[origin\_index] = origin\_vote$
            end if
        end for
    end for
    return $new\_pref$
end procedure
```

In our model, we assume if the voter knows clearly which candidate he hates most but does not have an idea which one he likes best, he may rank the candidates from his least-like one to his most-like one. Every time he votes a candidate, he may make a mistake which deviates from the true rank. As these mistakes occur with a known average rate which is the degree of our algorithm and independently of the time since the last event, they satisfy the definition of Poisson distribution. Therefore, we think in this situation it will exert a noise obeying the Poisson distribution. So the voters favorite candidate is the one he votes last, thus it has the most probability to deviate from the true one. So we will exert a noise obeying the Poisson distribution to the original preference, the first several rankings will be mutated most. This is called destructive manipulation, as the most important rankings will be destroyed and reconstructed. The algorithm of destructive mutation is shown in algorithm 2.

Algorithm 2 Destructive Mutation

```
procedure DESTRUCTIVEMUT ((raw_pref))
    $new\_pref = raw\_pref$
    for $rowinnew\_pref$ do
        for index from 0 to 9 do
            if $random(0,1) < poissonFunc(index, degree)$ then
                $tem = random(0,9)$
                $origin\_vote = row[index]$
                $row[index] = tem$
                $origin\_index = row.index(tem)$
                $row[origin\_index] = origin\_vote$
            end if
        end for
    end for
    return $new\_pref$
end procedure
```

As for the constructive manipulation, the situation is almost the same as the destructive manipulation, but this time the voter know exactly which candidates he prefers most and do not have a clear idea of which one he likes most. This situation always happens when all the candidates are very excellent and cannot be easily voted. So the voter will vote from his most favorite to his least favorite candidate. The algorithm of constructive mutation is shown in algorithm 3.

Algorithm 3 Constructive Mutation

```
procedure CONSTRUCTIVEMUT ((raw_pref))
    $new\_pref = raw\_pref$
    for $rowinnew\_pref$ do
        for index from 0 to 9 do
            if $random(0,1) < poissonFunc(10 - index, degree)$ then
                $tem = random(0,9)$
                $origin\_vote = row[index]$
                $row[index] = tem$
                $origin\_index = row.index(tem)$
                $row[origin\_index] = origin\_vote$
            end if
        end for
    end for
    return $new\_pref$
end procedure
```

4 Experiment

4.1 Experiment Procedure

At the beginning of our project, we were trying to conduct two separate experiments, for the first experiment we will use a fixed mutation model to compare mutation-resistance of different voting mechanisms, Figure 4, for the second experiment we attempt to measure the mutation-resistance of a voting mechanism by incrementally increasing the degree of mutation in the preferences, Figure 5. However, as the project progresses, we find its better to combine the two experiments together and removed the overlapped steps. Therefore, we induce the noise to the original preferences and use the mutated preferences to test the resilience to noise of different voting mechanisms. Meanwhile, we changed the degree of mutation to each voting mechanism.

4.2 Measurement

The measurement that we used to examine the difference between results generated from original preferences and mutated preferences is Kendall Tau Distance. Since calculating the

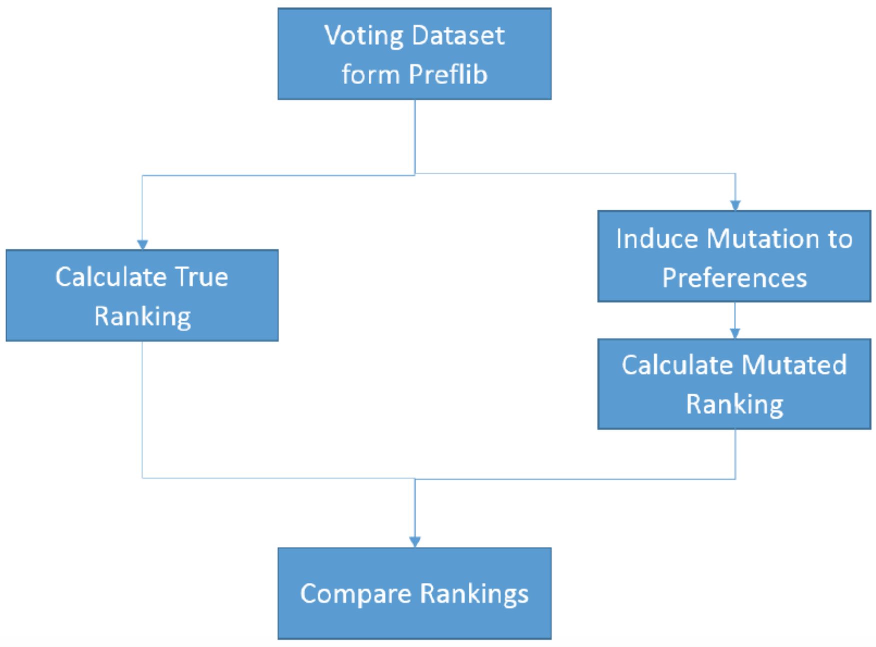
Voting Dataset form Preflib
Calculate True Ranking
Induce Mutation to Preferences
Calculate Mutated Ranking
Compare Rankings

Fig. 4. First Experiment.

Kendall Tau Distance requires a pair of complete rankings, whereas voting mechanisms directly produce the winners of the voting, we need to derive the ranking from the result by ourselves.

4.2.1 Kendall Tau Rank Correlation

As the equation shown, Kendall Tau Distance basically calculates the percentage of discordant pairs between two lists, which is desirable for measuring rankings.

$$\tau = \frac{P - Q}{\sqrt{(P + Q + T) \times (P + Q + U)}} \tag{2}$$

where P is the number of concordant pairs, Q the number of discordant pairs, T the number of ties only in the first ranking, and U the number of ties only in the second ranking. Since for our experiment we do not have ties, both T and U have a value of zero.

The Kendall Tau score can range between -1 and +1. A score of -1 means that all the ranks are different between the two rankings. A score of +1 means that all the ranks are concordant between the two rankings. A score of zero means that there are equal number of concordant and discordant ranks in between the two rankings.

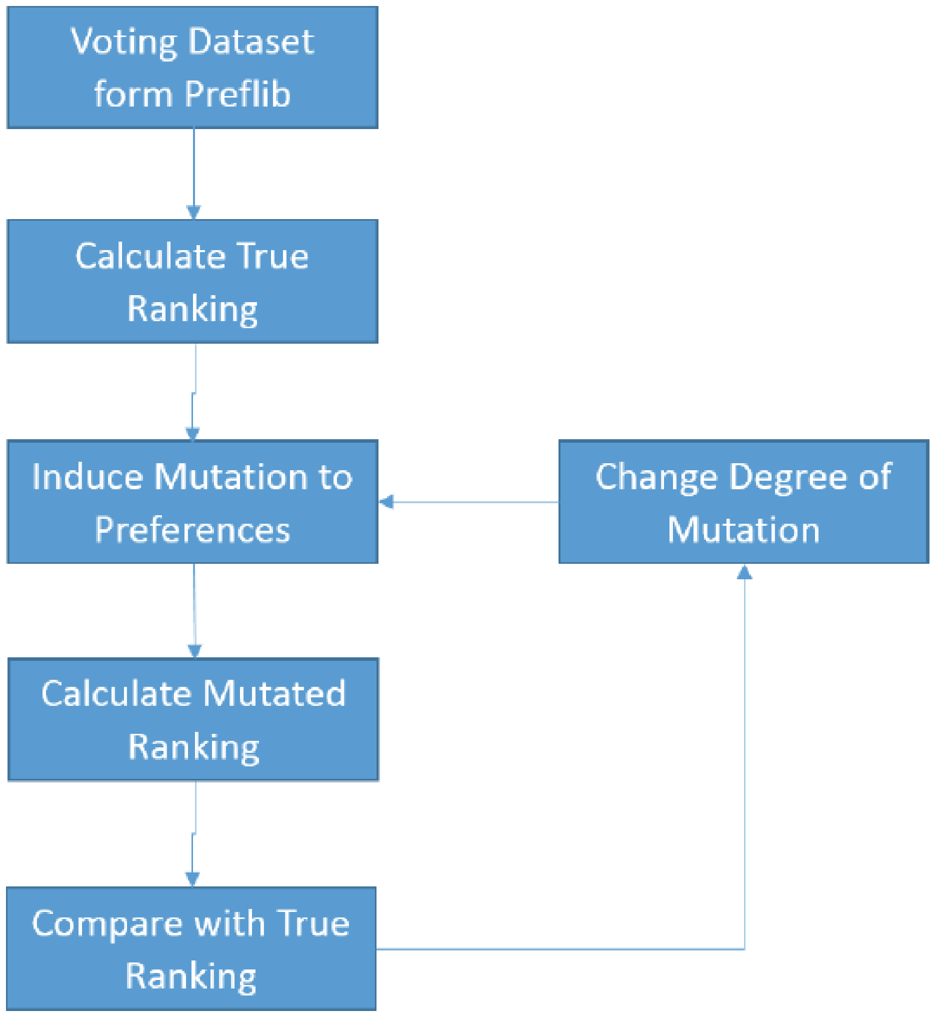
Voting Dataset form Preflib
Calculate True Ranking
Induce Mutation to Preferences
Change Degree of Mutation
Calculate Mutated Ranking
Compare with True Ranking

Fig. 5. Second Experiment.

4.2.2 Kernel density estimation

In statistics the Kernel density estimation is used to estimate the density of random values based on a finite sample size. It is used for data smoothing where inferences are made about a large dataset based on a relatively smaller sample size. The smoothness of the density curve is based on smoothness bandwidth, we have used Scotts rules for bandwidth selection for optimal and data-based histograms.

4.3 Design

The preference dataset that we used contains 10 candidates, which are 10 different types of Sushi, and 4925 complete votes. We induce five kinds of noise to the original preferences and iterate the execution one thousand times, as we want to trail the overall tendency.

4.4 Implementation

4.4.1 Syncluster

The experiments demands a large amount of computation, we have had to make some unforeseen additions to the code base in order to facilitate correct evaluation. For the evaluation we made use of the University of Waterloo SYN cluster.

4.4.2 Multi-Process Multi-Threaded Design

The experiment has been designed to make maximum use of concurrency to enable swift and parallel execution of non-related tasks. The most computational complexity of the experiment resides in the preference mutation, so the experiment uses the multiple threads to execute each of the mutations in parallel.

We have made use of multiple child processes for parallel experiment execution without blocking, as we generate the results of the experiment by executing it multiple times, we make use of a pool of child processes to execute multiple instances of the experiment in parallel.

5 Results

In this section, the result is shown in five different sub-sections according to five noise models that we argued before. For each noise model, we test three different degrees of mutation and the experiment is iterated more than one thousand times, as we want to study the tendency of variation of resilience to noise for different voting mechanisms. Density graphs and scatter graphs are used to demonstrate the result. Generally, for random mutation, it can be found that the increase of mutation degree leads to more changes to the final result for all voting mechanisms, and mutation happens on extremes plays a more important role than that happens in the middle. For normal distribution noise, IRV shows a stable performance than other three. When added destructive noise, IRV and Plurality at Large are better. Mutated by constructive noise, Plurality at Large and IRV seems to be more stable than the other two.

5.1 Random Mutation on Extremes

From Figures 6,7,8,9, we find that when mutation happens on the extremes of rankings, Plurality at Large and Plurality outperform other two voting mechanisms, and Plurality at Large even shows a relatively strong resilience to the increase of mutation degree. STV and IRV are both sensitive to the noise; however, STV performs better than IRV overall.

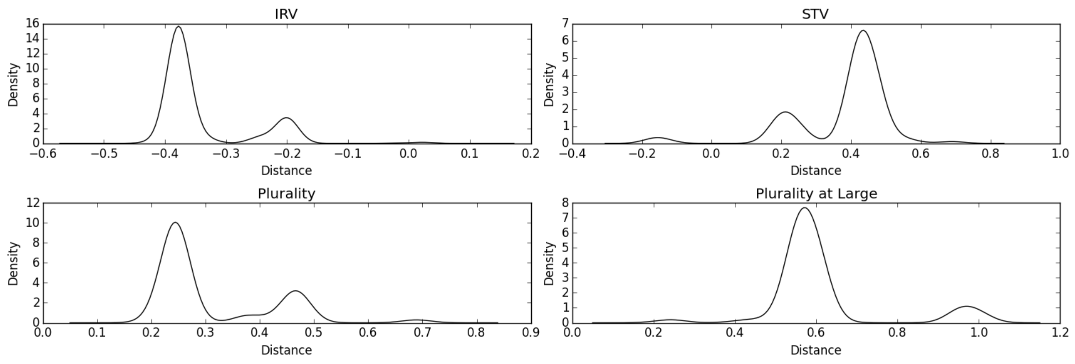

Fig. 6. KDE graph at mutation degree 0.3 on extremes.

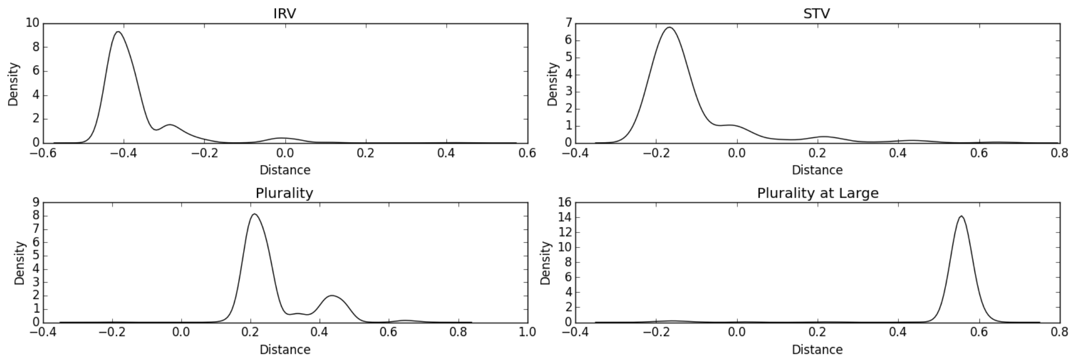

Fig. 7. KDE graph at mutation degree 0.4 on extremes.

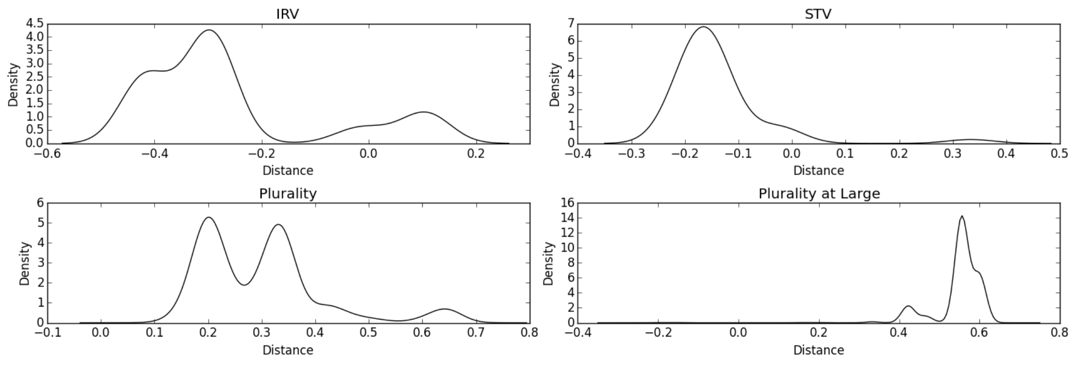

Fig. 8. KDE graph at mutation degree 0.5 on extremes.

5.2 Random Mutation in the middle

From Figures 10,11,12,13, it can be noted that for all of the four voting mechanisms, the influence from mutation happens in the middle of rankings is much slighter than that happens

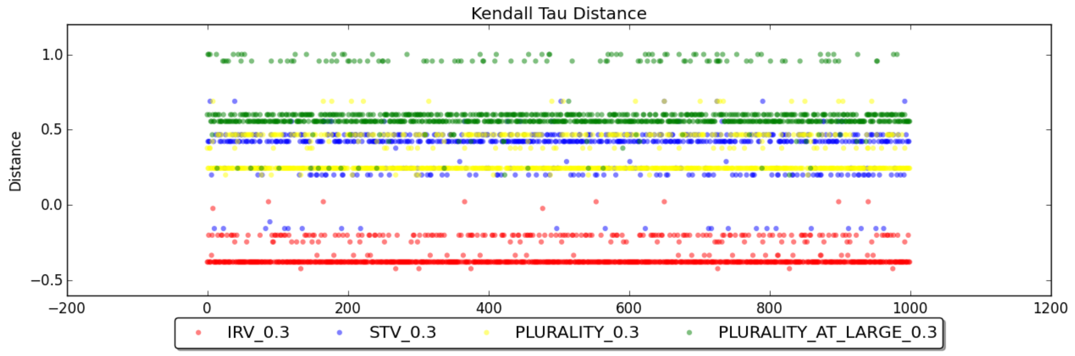

Fig. 9. Scatter graph at Mutation Degree 0.3 on extremes.

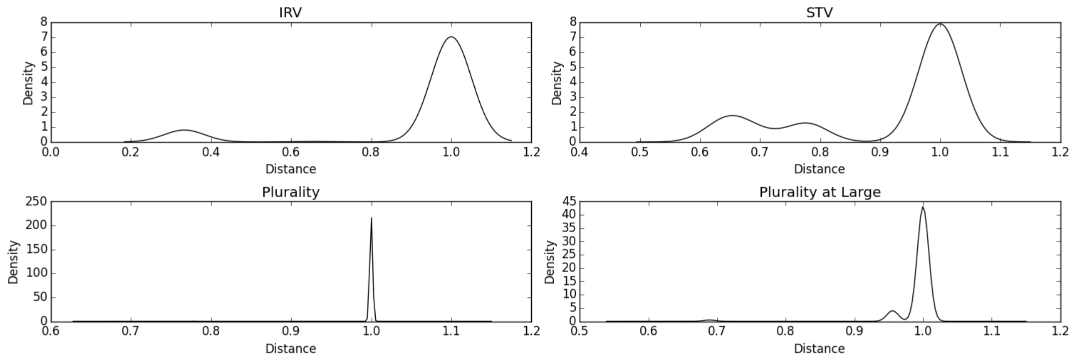

Fig. 10. KDE graph at mutation degree 0.3 in the middle.

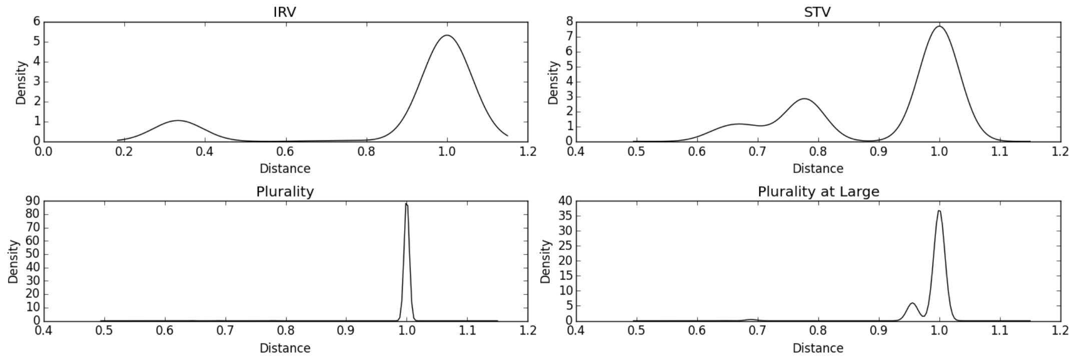

Fig. 11. KDE graph at mutation degree 0.4 in the middle.

on extremes. Plurality at Large and Plurality still display their preponderance than STV and

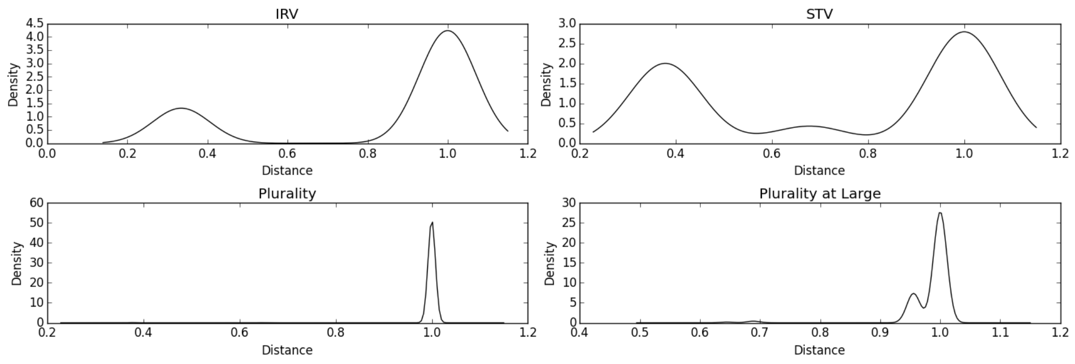

Fig. 12. KDE graph at mutation degree 0.5 in the middle.

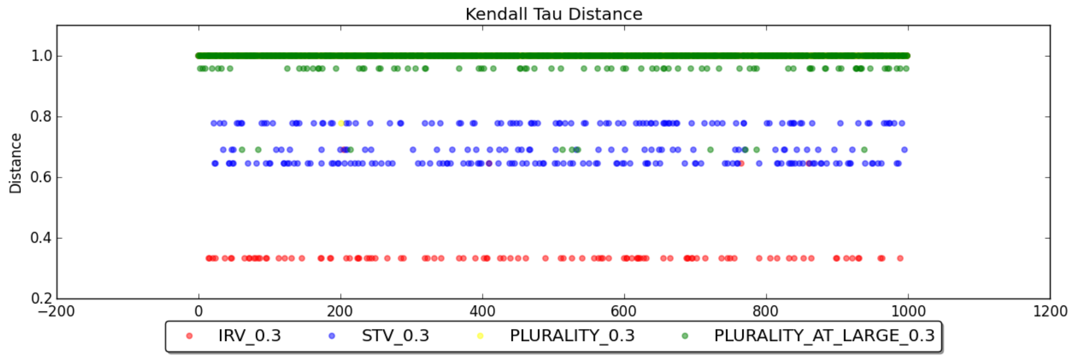

Fig. 13. Scatter graph at Mutation Degree 0.3 in the middle.

IRV, whereas Plurality performs more stably than Plurality at Large this time. IRV still does not perform as well as other three voting mechanisms.

5.3 Normal Noise Mutation

From Figure 14,15,16, we can see under the mutation of normal noise the IRV mechanism is always the most noise-tolerant one and most stable one when compared with other three mechanisms. Although, when the degree is small, plurality at large is also good, it changed a lot along with the increase of the degree. So plurality at large is not ideally noise-tolerant and stable. From the scatter point graph 17, it also illustrates that plurality at large and IRV are the most stable one when degree is 0.5. Plurality mechanism is relatively stable along with increase of the degree when compared with plurality at large and STV, so when there is a normal-distributed noise inherently in the preference, we cannot easily get the true ranking by removing the noise. The worst preformed mechanism is STV as its deviation from the true ranking is the biggest, which means it cannot easily show the true preference of voters when most of the voters hesitate about the vote.

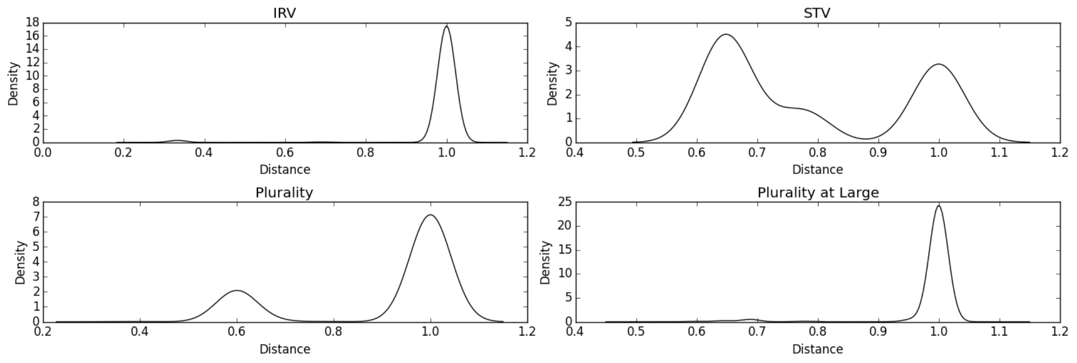

Fig. 14. KDE graph of Normal noise mutation at degree 0.25.

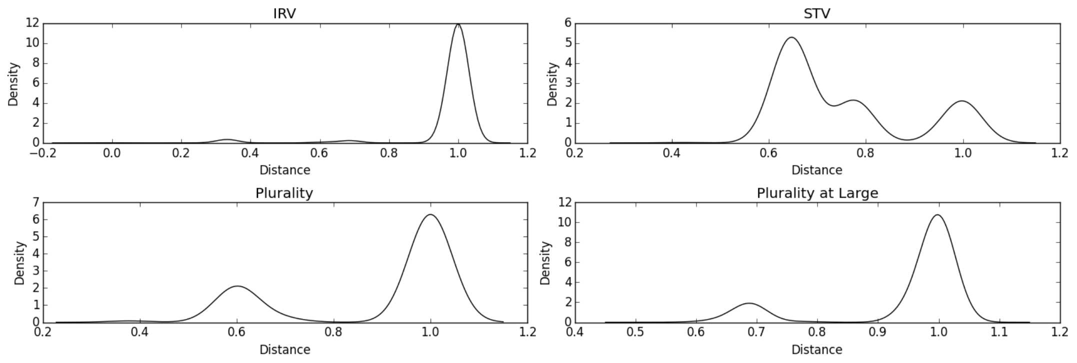

Fig. 15. KDE graph of Normal noise mutation at degree 0.5.

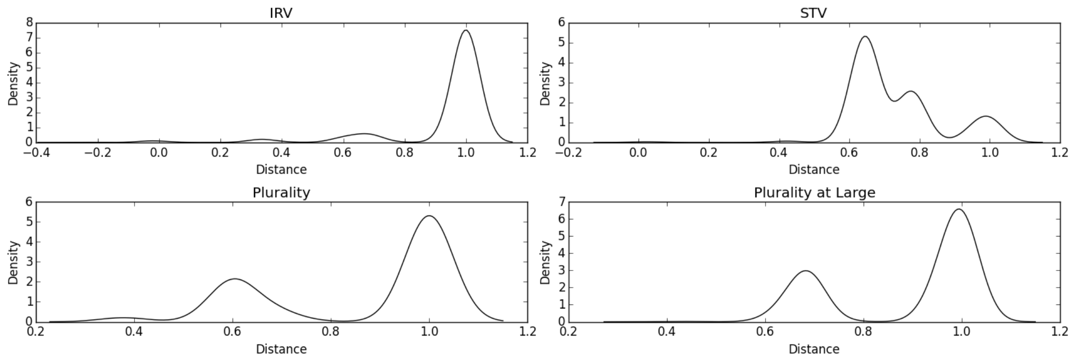

Fig. 16. KDE graph of Normal noise mutation at degree 0.75.

5.4 Destructive Manipulation

After we mutated the preference by destructive manipulation, we can see the IRV mechanism is also the most stable one. From the probability density graphs 18,19,20, we can see the

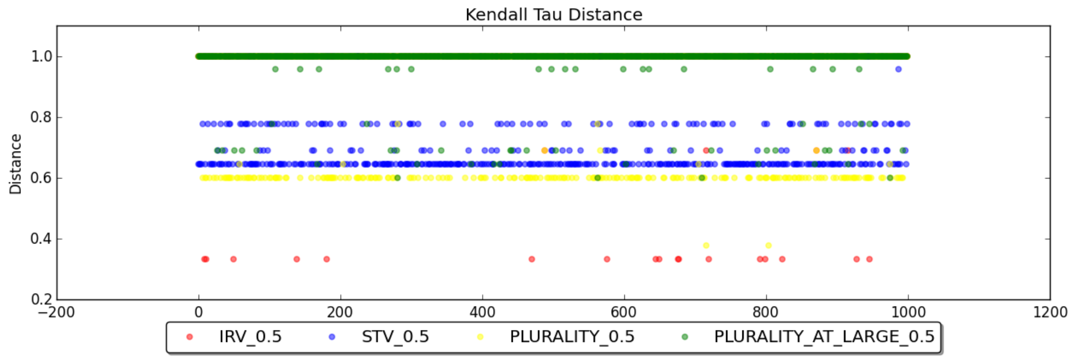

Fig. 17. Scatter Graph of normal noise mutation at degree 0.25.

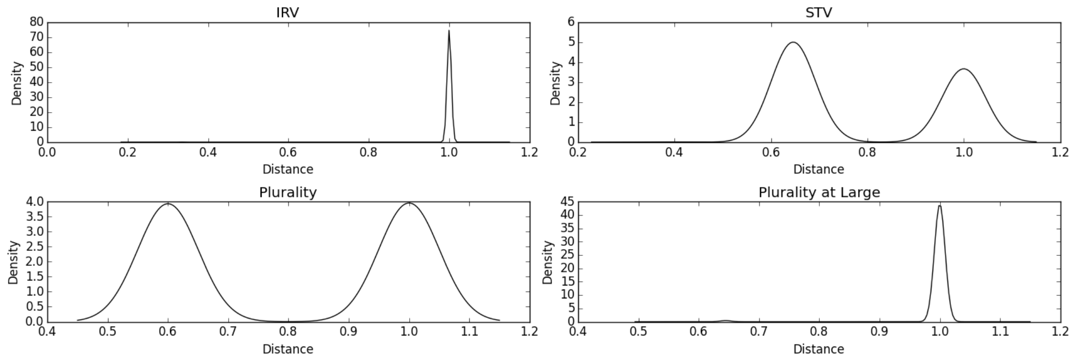

Fig. 18. KDE graph of distructive mutation at degree 1.

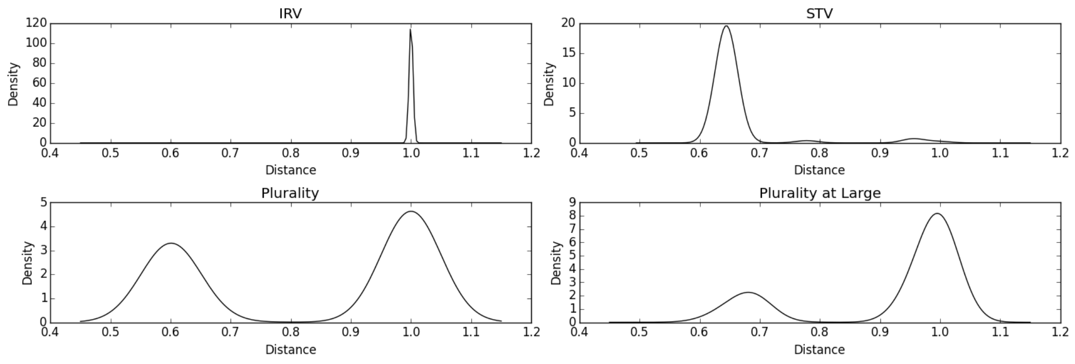

Fig. 19. KDE graph of distructive mutation at degree 3.

average of IRV keeps being 1 and only the variance increases along with the increase of the degree. As for the other three mechanisms, they are not stable at all. For example, plurality

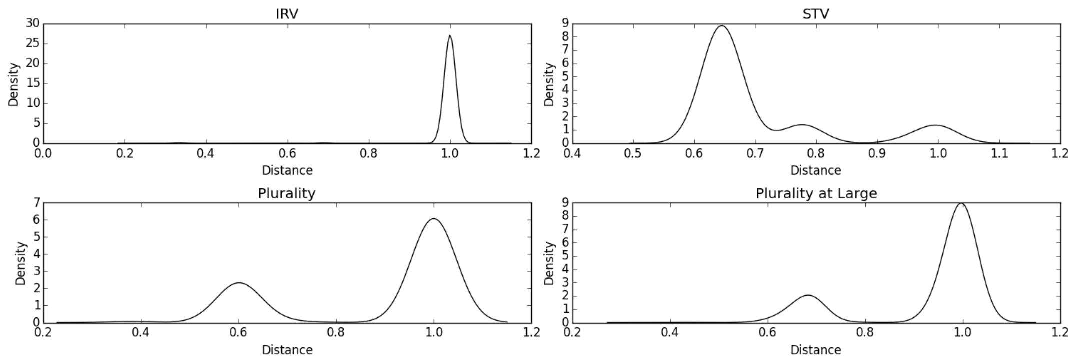

Fig. 20. KDE graph of distructive mutation at degree 5.

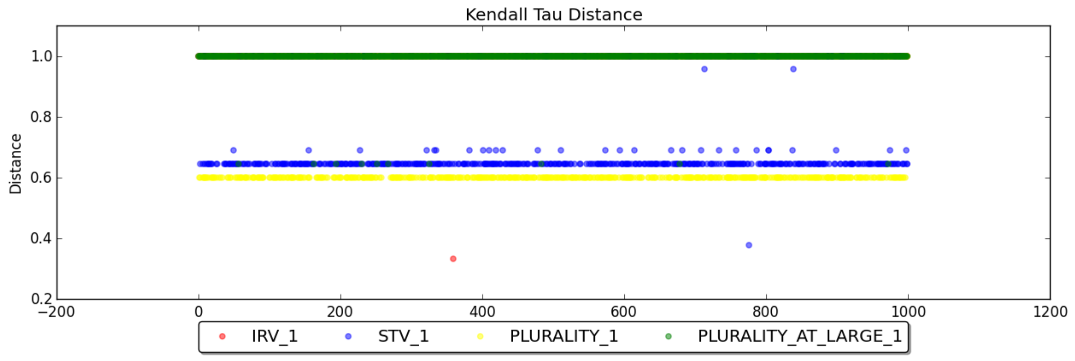

Fig. 21. Scatter graph of distructive mutation at degree 1.

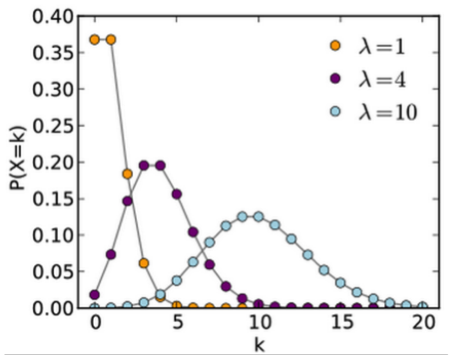

Fig. 22. Probability function of Poisson Distribution.

at large changed a lot though its good when the degree is small. We can see from the scatter point graph 21, IRV and plurality at large is steady when degree I s 1. One explanation is that the IRV processes the ranking from the last rankings by eliminate the last candidate at one time and reallocate the ranking. So when it processes the first several candidates, they will accumulate more votes, thus more tolerant of the mutation of the first several rankings.

As the degree is the average occurrence rate of the mistakes the voters make against the true ranking, so we can see an interesting phenomena that the plurality becomes closer to the true ranking along with the increase of the degree, though not too much. You can see from Figure 22, along with the increase of degree, the f(x) becomes smaller. So this time the definition of degree is different from the degree of other mutation.

5.5 Constructive Manipulation

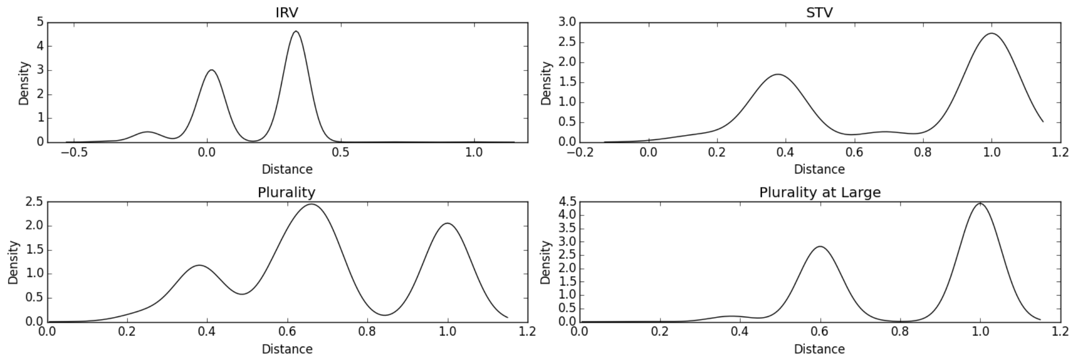

Fig. 23. KDE graph of constructive mutation at degree 1.

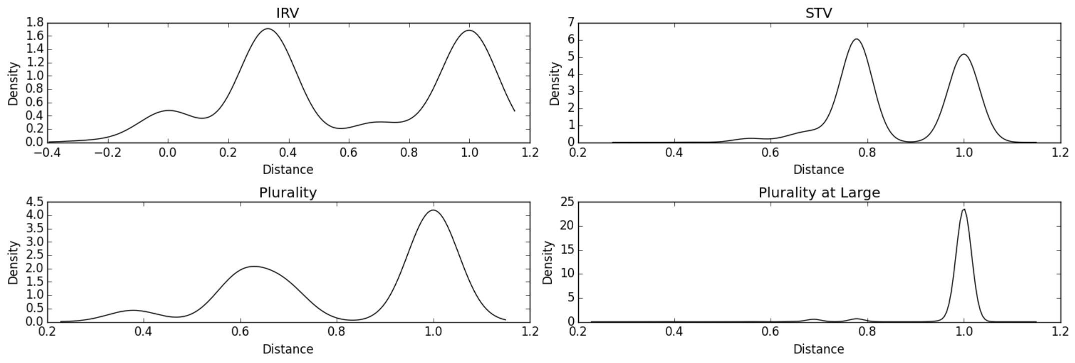

Fig. 24. KDE graph of constructive mutation at degree 3.

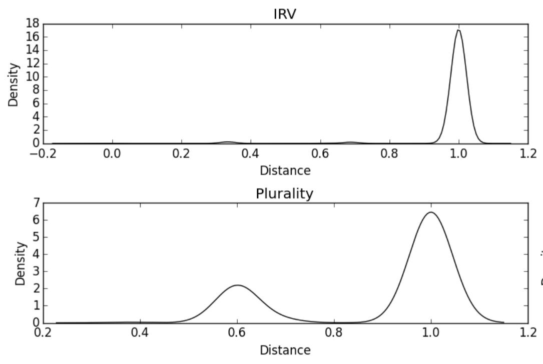

Fig. 25. KDE graph of constructive mutation at degree 5.

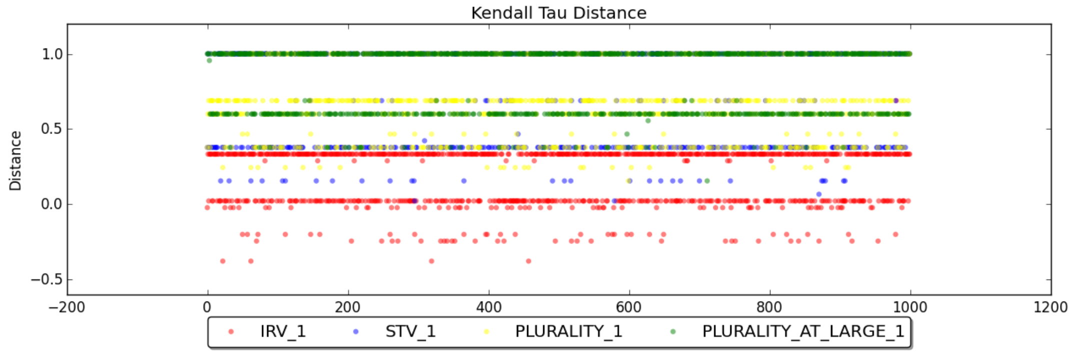

Fig. 26. Scatter graph of constructive mutation at degree 1.

From Figure 23,24,25,26,constructive mutation is very similar to the destructive manipulation, but constructive manipulation mainly changes the last several rankings. In contrast, destructive manipulation changes the first several rankings more. This time all of these three functions are not stable. Because constructive manipulation just exerts the noise of the destructive manipulation left and right reversed, they are symmetry. Along with the increase of the degree, IRV become closer to the true ranking. We think the reason is that the increase of the degree moves the mutation from the last rankings to the middle rankings, thus decreasing the influence. So we can say IRV is not tolerant with the noise exerted on the last rankings. One explanation is that IRV processes the ranking from back to front as it eliminates the last ranking at one time. This is an interesting conclusion. The proof can also be found from the graphs for STV.

6 Conclusion

By this project, we designed an experiment for measuring the noise-tolerance for four popular social functions and performed a comparative study based on the measurement. As far as we know, our study is the first empirical one on the topic. We find that different functions have unique noise-tolerant properties when exerted with different kinds of noise.

For the random noise models, the degree of mutation undoubtedly affects the performance of each voting mechanism, and the mutation happened on both extremes is more effective in influencing the rankings than that happened in the middle. Functions with non-transferable votes seem to be more resilient to noise than transferable votes.

When exerted with normal noise, IRV mechanism is always the most noise-tolerant one and most stable one compared with other three mechanisms. It also has a good performance when the preference is mutated by destructive manipulation. Although Plurality at Large performs well when the degree is small, it is not stable. When mutated by constructive manipulation, the conclusion is difficult to draw. In this condition, plurality at large outperforms other functions when degree is small. However, as the degree becomes bigger, IRV outperforms the other functions and becomes the most noise-tolerant one.

7 Future work

– The experiment can be extended to asses newer social choice mechanisms, variants of popular social functions. The experiment can be used as a test-bed of new mechanisms in mechanism design.
– Use larger preference set involving realistic voting scenarios. The experiment can used to do post analysis of entire of election process and asses if the vulnerability of the election result.
– In cases we are aware of the kind of noise that superimposes the preferences, we can use an extension of the experiment to induce negative noise to remove the noise and gain knowledge about the true preferences.
– Use more noise models in the experiment like mallows model [1]
– Use multiple rank co-relation metrics like the Spearman's ρ, Goodman and Kruskal's γ

References

[1] Conitzer, Vincent, and Tuomas Sandholm. "Common voting rules as maximum likelihood estimators." arXiv preprint arXiv:1207.1368 (2012).
[2] Egger, Peter. "On the role of distance for bilateral trade." The World Economy 31.5 (2008): 653-662.
[3] Berger, Adam L., Vincent J. Della Pietra, and Stephen A. Della Pietra. "A maximum entropy approach to natural language processing." Computational linguistics 22.1 (1996): 39-71.
[4] Critchlow, Douglas E., Michael A. Fligner, and Joseph S. Verducci. "Probability models on rankings." Journal of mathematical psychology 35.3 (1991): 294-318.
[5] Conitzer, Vincent, Matthew Rognlie, and Lirong Xia. "Preference Functions that Score Rankings and Maximum Likelihood Estimation." IJCAI. Vol. 9. 2009.
[6] de Condorcet, M. 1785. Essai sur lapplication de lanalyse a la probabilit ‘ e de d ecisions rendues a la pluralit ‘ e de voix. Imprimerie Royal. Facsimile published in 1972 by Chelsea Publishing Company, New York

Appendix

.1 Syncluster

The University of Waterloo SYN cluster is a multi-tenant compute facility which is used for network research. The SYN facility consists of 4 independent computing clusters named red, green, blue and yellow. Each cluster has 15 regular compute nodes and 1 large compute node. The compute nodes of the SYN cluster are stateless and the home directories are stored on a central admin node called as the white node. The white node is a gateway for accessing the other compute nodes in different clusters. The home directories are stored on a shared file system and can be accessed by all the compute nodes. The users wishing to use the SYN clusters need to pass a directory controlled authentication for accessing the white node, once this authentication is completed the user can update the directory for accessing other compute resources. Compute resources in the SYN cluster can be booked for use via the SYN Reservation System, which is a web-based graphical tool that clearly displays the currently available compute resources. Once a compute resource has been booked for a time slot, the user can access the particular compute node by first accessing the white node and then creating an ssh connection the white node and the compute node that is identified by a URL, we used putty for accessing the SYN cluster. A user has sudo-capability on the compute nodes they have reservations for, this is a very important capability that many shared compute environments used in the industry don not provide. We were able to setup our experiments with great pace thanks to the freedom enjoyed inside the SYN compute node.

.2 Experiment

The experiment used one large compute node of the SYN cluster which was used to host 15 different experiments. The experiment included 3 mutation levels each of 5 different noise models. Each individual experiment tested a noise model with one mutation level for 4 social functions (except normal mutation). As the mutations considered in this experiment were random it was impossible to deduce accurate correlations, so it we decided to aggregate the results based on the repeated execution of the experiments, so each of the 15 experiment actually represents a thousand repetitions of the mutation-social function procedure, so the all put together for this entire experiment we have performed around 1000 * 4 * 5 * 3 = 60000 elections using distinct preferences that were mutated randomly.