Simulating the Effects of Misperception on the Manipulability of Voting Rules

Johann Mitlöhner, Daniel Eckert, and Christian Klamler

Abstract

The fact that rank aggregation rules are susceptible to manipulation by varying degrees has long been known. In this work we study the effect of noise on manipulation i.e. we assume that individuals are not able to perceive the preferences of others without distortion. To study the frequency of various outcomes we simulate a large number of rank aggregations and manipulations on random profiles with the help of a software package developed by the authors in the Python language and discuss some preliminary results.

1 Motivation

The extent to which various aggregation rules are susceptible to manipulation has been studied in a number of investigations, including simulation studies [6]. Manipulation here means to strategically misrepresent one's true preferences in order to change the election outcome to a personally more favorable one [4]. This strategic misrepresentation is based on knowledge of the other voters' preferences. In this work we relax the assumption of perfect information. We study situations where individuals manipulate while perceiving a noisy version of the other voters' preferences. Such manipulations with noisy information have the interesting property that while they change the outcome to a more favourable one given the noisy information they may fail to do so given the true preferences of the other voters, leading to situations where the manipulator is worse off than without manipulation. This can be interpreted as a form of punishment for lying; the extent to which various aggregation rules produce this effect is the subject of this paper.

For our simulation study we have implemented a set of well-known voting rules [3] in the programming language Python.[1] These rules are applied to a large number of random profiles in a setting with misperception which we describe in the next section. The details of the implementation are outlined after that, and the paper concludes with the discussion of the simulation results.

[1] At the website http://prefrule.sourceforge.net the complete package is available for download, and http://balrog.wu-wien.ac.at/∼mitloehn/prefrule provides an interactive web interface to the system.

2 Manipulation and Misperception

We assume that each voter has complete and strict preferences over the set of candidates. A profile p is a set of n strict orders over the set of candidates C, e.g. $p = ((a \succ b \succ c \succ d), (b \succ c \succ a \succ d), (c \succ a \succ b \succ d))$ denotes a profile with $n = 3$ voters and the set of $m = 4$ candidates $C = \{a, b, c, d\}$. A rank aggregation rule R applied to a profile p derives an aggregate ranking $p_* = R(p)$ which is either a strict or a weak order.

The distance d of the aggregate ranking p_* and some order p_i is measured by taking the positional difference of the winner of p_* in p_i, e.g. with $p_i = (a \succ b \succ c)$ and $p_* = (b \succ a \succ c)$ the distance is $d(p_i, p_*) = 1$.[2] A manipulation is successful if voter i is able to decrease the distance d by stating manipulated preferences p'_i, e.g. if with $p'_i = (a \succ c \succ b)$ the aggregate ranking becomes $p'_* = ((a = b) \succ c)$ then $d(p'_i, p'_*) = 0.5$.

The assumption that any voter i has perfect knowledge of the remaining profile p_{-i} i.e. the preferences of all other voters is somewhat unrealistic. In this work we explore a setting where the manipulating voter is mistaken in the perception of the remaining profile p_{-i} by a certain amount of error i.e. instead of the true p_{-i} the noisy p^e_{-i} is perceived.[3] We define the error e as the number of pairwise exchanges in adjacent pairs of candidates in some ranking(s) p_j where $j > 1$, e.g. with $p_{-i} = ((b \succ c \succ a \succ d), (a \succ c \succ b \succ d))$ and $p^e_{-i} = ((b \succ c \succ a \succ d), (c \succ a \succ b \succ d))$ we have an error of $e = 1$ since there is one switch of a and c in the last voter.

With faulty perceptions manipulations can result in a distance increase instead of a decrease. This can be viewed as a punishment for lying. The situation is described in eqs. 1 and 2.

$$d(p_i, R(p'_i, p^e_{-i})) < d(p_i, R(p_i, p^e_{-i})) \tag{1}$$

$$d(p_i, R(p'_i, p_{-i})) > d(p_i, R(p_i, p_{-i})) \tag{2}$$

Voter $i = 1$ perceives the noisy profile p^e_{-i} and based on this observation choses manipulated preferences p'_i that decrease the distance d as shown in eq. 1. However, when the aggregation rule R is applied to the manipulated preferences p'_i and the *true* remaining profile p_{-i} the result is shown in eq. 2: the distance is increased i.e. voter i has not gained but lost by manipulating.

This type of punishment would be an attractive quality of rank aggregation rules since if it occurs frequently enough it incites voters to state their true preferences and refrain from manipulation. The question remains whether a situation of this type is a rare exceptional case that has little meaning for the evaluation and comparison of aggregation rules in this respect, or a phenomenon common enough for quantitative analysis.

[2] In the case of more than one winner the average distance is used.

[3] In our simulations the manipulator is always voter $i = 1$.

In order to answer this question we define the expected benefit from manipulation $E(\Delta d)$ as the weighted sum of distance changes for the fractions of successful and failed manipulations:

$$E(\Delta d) = \frac{|S|}{|S|+|F|} \sum_{p \in S} (d^m - d^0) + \frac{|F|}{|S|+|F|} \sum_{p \in F} (d^m - d^0) \tag{3}$$

Here d^0 denote the distance without manipulation, and d^m is the distance achieved with manipulation, both as calculated when the aggregation rule is applied to the manipulated preferences p'_i of voter $i = 1$ and the true preferences p_{-i} of the remaining voters. S is the set of profiles where voter $i = 1$ successfully manipulated i.e. where $d^m < d^0$, while F is the set of profiles where manipulation failed i.e. it resulted in punishment.

3 Simulations

In order to study the frequency of the punishment effect we have implemented a number of well-known voting rules in a software package developed in the Python programming language. The simulation generates a stated number of random profiles for n voters and m candidates where rankings are independent i.e. anonymous culture. The set of rules to explore is another parameter, since some rules are computationally much more expensive than others and for that reason may be excluded in some simulation runs. The rules implemented are: Borda (BO), Copeland (CO), Kemeny (KE), Plurality (PL), Antiplurality (AP), Transitive Closure (TC), Maximin (MM), Slater (SL), Nanson (NA), Young (YO), and Dodgson (DO).

Since voters' preferences are assumed to be complete and strict a profile is implemented as a nested list with integers for the candidates, e.g. [[0,1,2],[2,0,1],[0,2,1]] for $((a \succ b \succ c), (c \succ a \succ b), (a \succ c \succ b))$. Aggregate relations are encoded as binary matrices denoting weak preference i.e. if $r_{i,j} = 1$ and $r_{j,i} = 0$ then $c_i \succ c_j$; if $r_{i,j} = r_{j,i} = 1$ than $c_i = c_j$. Therefore, the nested list [[1,1,1],[1,1,1],[0,0,1]] denotes the aggregate ranking $((a = b) \succ c)$. For printing the rankings and aggregate relations are transformed into more readable versions using plain text symbols, such as $a > b > c$. Table 1 shows a sample random profile generated by the system and the corresponding aggregate rankings resulting from various voting rules. This profile was selected for variety of results; in a typical sample the aggregate relations are much more similar [2].

The Python code is about ten to twenty times slower than a comparable version of a subset of the code written in the C programming language. However, in contrast to low level languages like C and Java used in earlier work [1] the Python language provides more clarity and elegance of syntax. Therefore it is less error-prone and saves programmer time instead of execution time. The

Table 1: Sample random profile and aggregate rankings.

```
pr: abcd,cadb,cabd,bdca,bcda
BO: [[1,0,0,1],[1,1,0,1],[1,1,1,1],[0,0,0,1]]    c>b>a>d
CO: [[1,1,1,1],[1,1,1,1],[1,1,1,1],[0,0,0,1]]    a=b=c>d
TC: [[1,1,1,1],[1,1,1,1],[1,1,1,1],[0,0,0,1]]    a=b=c>d
NA: [[1,0,0,1],[1,1,1,1],[1,0,1,1],[0,0,0,1]]    b>c>a>d
MM: [[1,0,0,1],[1,1,1,1],[1,1,1,1],[1,0,0,1]]    b=c>a=d
KE: [[1,0,0,1],[1,1,1,1],[1,0,1,1],[0,0,0,1]]    b>c>a>d
SL: [[1,1,1,1],[0,1,1,1],[0,0,1,1],[0,0,0,1]]    a>b>c>d
YO: [[1,0,0,1],[1,1,1,1],[1,1,1,1],[1,0,0,1]]    b=c>a=d
DO: [[1,0,0,1],[1,1,1,1],[1,1,1,1],[0,0,0,1]]    b=c>a>d
PL: [[1,0,0,1],[1,1,1,1],[1,1,1,1],[0,0,0,1]]    b=c>a>d
AP: [[1,0,0,1],[1,1,0,1],[1,1,1,1],[1,0,0,1]]    c>b>a=d
```

Python language has been termed "executable pseudo-code"; fig. 1 shows an example.

Table 2 shows timings taken on a Dual Core 3.2 GHz Intel Pentium D running Debian Linux. The data show that where execution time t as a function of m is concerned the positional rules Borda, Plurality, and Antiplurality, together with Copeland, Maximin, and Nanson form the most efficient group which allows them to be applied to a wide range of parameter values. The Transitive Closure rule and the Young rule form an intermediate group, while the Kemeny, Slater, and Dodgson rules show significant increases with m in execution time even in the small parameter range tabulated: the Kemeny and Slater rules with an $O(m!)$ term for trying all permutations of candidates; and the Dodgson rule with $O((nm)!)$ for trying pairwise exchanges of adjacent candidates.

4 Results

Using the software package described the manipulation with misperception has been simulated with $n = 5$ voters and $m = 4$ candidates. The error level was $e = 1$ i.e. a single misperception modelled as a pairwise exchange of adjacent candidates in the the remaining profile p_{-i} as perceived by voter $i = 1$. Table 3 shows the results.

These results are preliminary due to their limited parameter range; as such they indicate that Copeland shows the punishment effect to a much higher degree than Borda and Kemeny. Success and punishment, if they materialize at all, are pronounced most strongly in Kemeny and Slater, the only rules amoung the set investigated that always produce strict aggregate preferences. The lowest expected change in distance occurs with Transitive Closure, the rule that tends to produce a high number of indifferences in the aggregate relations.

Figure 1: This function takes a vector of scores and constructs the corresponding binary relation. The function call after the >>> prompt shows how the code can be tested in the interactive environment provided by the Python interpreter, a feature that is very useful in program development.

```
def scorel(sc):
  m=len(sc)
  r=mat(m,m)
  for i in range(m):
    for j in range(m):
      if sc[i]>=sc[j]: r[i][j]=1
  return r

>>> scorel([5,9,8,2])
[[1, 0, 0, 1], [1, 1, 1, 1], [1, 0, 1, 1], [0, 0, 0, 1]]
```

Table 2: Execution times in seconds for 1000 random profiles with $n = 9$ voters and $m = 4, 5, 6, 7, 8$ candidates.

Rule	$m = 4$	$m = 5$	$m = 6$	$m = 7$	$m = 8$
BO	0.07	0.07	0.08	0.09	0.10
CO	0.09	0.09	0.10	0.11	0.13
PL	0.07	0.07	0.08	0.09	0.10
AP	0.07	0.07	0.08	0.09	0.10
MM	0.08	0.10	0.11	0.13	0.14
NA	0.08	0.09	0.10	0.11	0.13
TC	0.15	0.25	0.43	0.72	1.21
YO	1.21	1.93	2.97	3.71	5.15
KE	0.14	0.52	3.58	31.61	318.46
SL	0.15	0.47	3.03	26.20	253.14
DO	2.32	12.31	51.98	160.56	464.17

Table 3: Results of 100000 random profiles with $n = 5$ voters and $m = 4$ candidates with error level $e = 1$. Explanation of headings: M: number of profiles with noise manipulable by some voter; M_1: number of profiles with noise manipulable by voter one; S: number of profiles with successful manipulation without noise i.e. where voter one succeeded in decreasing the distance; Δd_S: average distance decrease for successful manipulation; F: number of profiles where manipulation failed i.e. it resulted in punishment; Δd_F: average distance increase for failure i.e. punishment; $E(\Delta d)$: expected change in distance resulting from manipulation as defined in eq. 3.

Rule	M	M_1	S	Δd_S	F	Δd_F	$E(\Delta d)$
BO	59731	28551	13960	-0.715	2704	0.714	-0.484
CO	32358	8942	4185	-0.545	1430	0.635	-0.244
KE	27171	7727	3260	-1.327	1073	1.232	-0.693
PL	57534	16430	12052	-0.718	1884	0.835	-0.508
AP	52331	25976	21194	-0.675	1694	0.715	-0.572
TC	22909	7356	3853	-0.422	881	0.995	-0.158
NA	25710	9469	3621	-0.938	1349	1.068	-0.393
MM	28052	9101	4727	-0.445	828	0.805	-0.259
SL	27321	7731	3601	-1.334	1057	1.266	-0.744
YO	28052	9151	4731	-0.446	787	0.805	-0.267
DO	33698	10437	4721	-0.579	1576	0.625	-0.278

Apart from TC the Copeland rule shows the strongest punishment effect.

The data also show that the punishment effect for manipulation with misperception is not a rare exceptional case. It occurs frequently enough to provide an additional dimension for the evaluation and comparison of voting rules.

5 Conclusions

This paper described the prefrule software package for preference aggregation and reported the results of simulations of various voting rules on a large number of random profiles. Specifically, the manipulability and the effect of misperception of preferences of other voters was investigated. It has been shown that manipulators can lose rather than gain from manipulation in a setting with misperception. The susceptibility of various rank aggregation rules to these effects has been explored in simulation runs. Future work will test the validity of these results for a wider range of parameters and expand the range of applications of the software package developed for the simulations.

References

[1] Daniel Eckert, Christian Klamler, and Johann Mitlöhner. Condorcet efficiency, information costs, and the performance of scoring rules. *Central European Journal of Operations Research*, 2005:1.

[2] Daniel Eckert, Christian Klamler, Johann Mitlöhner, and Christian Schlötterer. A distance-based comparison of basic voting rules. *Proc. Joint Workshop on Decision Support Systems, Experimental Economics and e-Participation*, Graz, Austria, 2005.

[3] Peter C. Fishburn. Condorcet Social Choice Functions, *SIAM Journal of Applied Mathematics*, 33:469–489, 1977.

[4] Donald Saari. Susceptibility to manipulation. *Public Choice*, 64:21–41, 1990.

[5] Donald Saari. *Decisions and Elections - Explaining the Unexpected.* Cambridge University Press, 2001

[6] David Smith. Manipulability Measures of Common Social Choice Functions. *Social Choice and Welfare*, 16:639–661, 1999.

Johann Mitlöhner
Vienna University of Economics and Business Administration
Augasse 2–6, A-1090 Vienna, Austria
Email: `mitloehn@wu-wien.ac.at`

Daniel Eckert
University of Graz
Universitätsstr. 15/E4, A-8010 Graz, Austria
Email: `daniel.eckert@uni-graz.at`

Christian Klamler
University of Graz
Universitätsstr. 15/E4, A-8010 Graz, Austria
Email: `christian.klamler@uni-graz.at`