The Mediancentre Rule for Many Alternatives: CT Scholars Summer Research Final Report

Daniel Gnoutcheff (Class of 2011, Union College)
Prof. William Zwicker (adviser)

September 20, 2009

Abstract

We continue research into the mediancentre voting rule, the recently introduced variant of the Borda count. In particular, we extend the investigation of this rule to beyond 3 alternatives. We have determined that at up to 4 alternatives, the mediancentre rule seems to outperform the mean rule in terms of manipulability, as expected. We have also found that it seems to have a higher Condorcet efficiency. Finally, we have continued work on finding reliable ways to find ties, and we have found two classes of profiles that we believe describe all 2-way ties for 3 alternatives.

Contents

1 Overview

My research falls within VOTING THEORY, the study of VOTING RULES that use the preferences expressed by members of a group of people to determine the preferences of that group as a whole. This is much more interesting than it sounds, as there are many voting rules to choose among and selecting the right one is surprisingly difficult. Consider, for example, four properties that we'd want a voting rule to have:

Monotonic: Changing a vote to favor some candidate more shouldn't harm that candidate's chances of winning.

Inclusive: Every candidate should have some chance of winning.

Non-dictatorial: No single voter should be able to determine the outcome.

Non-manipulable: A voter should not be able to obtain a more favorable outcome by misrepresenting their preferences.[1]

These properties are formally stated in [9]. It is not a complete list, but many would argue that a "perfect" voting rule should have at least these properties. But we have already backed ourselves into a corner: the famous Gibbard-Satterthwaite theorem shows us that if there are at least 3 candidates and at least 2 voters, no voting rule will ever be able to satisfy all of the properties in this set of criteria. Clearly, the "perfect" voting rule does not exist; we must make trade-offs, and we need to understand the trade-offs we are making. This is why voting theory is valuable; it tells us about the properties of the voting rules available, giving us the information we need to make an informed choice as to what rule to use.

One rule that is popular among voting theorists is the BORDA COUNT, also known as the MEAN RULE. Each BALLOT (a vote that a voter can cast) lists all of the ALTERNATIVES (political candidates, referenda, etc.) in decreasing order of preference. Given a alternatives, there will be a total of $a!$ ballots, containing all the possible orderings (i.e. permutations) of the alternatives. The votes are collected into a PROFILE, a function that indicates how many voters cast a given ballot; given a profile P, $P(\beta)$ denotes the number of voters who cast ballot β.[2]

With every vote, each alternative is awarded a score according to a set of SCORING WEIGHTS, which determines what award an alternative should receive given their position on the preference ranking. For the Borda count, these weights are such that the most-preferred alternative has the highest score and the difference between two consecutive weights is always the same. For example, with 3 alternatives we may give 2 points to the favorite alternative, 0 points to the second favorite alternative, and -2 points to the least-favorite alternative. Given a profile, we can find the total score each alternative received, and the winning alternative is the one with the highest score. Sometimes we will have several alternatives tied for highest score, so we may have several winners.

While this is the most popular definition of the Borda count, it is also useful to consider the alternative definition suggested in [11], which is equivalent to the traditional definition but is geometric in nature. Let a denote the number of alternatives, and consider the Cartesian coordinate space $\mathbb{R}^a$. We associate each coordinate axis with an alternative (e.g. x-axis with alternative P, y-axis with Q, z-axis with R), and we represent each ballot as a location in $\mathbb{R}^a$ where each component is given by the award that the Borda count would have given to the alternative associated with that component's axis. These locations form the vertexes of a polyhedron named the a-permutohedron, and these vertexes represent all the possible permutations of the scoring weights. Figure 1 on page 5 shows a few examples of what these permutohedra look like. At each vertex, we place one point for each voter who cast the ballot that corresponds to that vertex; note that this means we often end up with several points at the same location. Once we represent the votes in this way, we find the MEAN of these points.

Definition: Let $a \geq 2$ be an arbitrary natural number. Consider a finite multiset P of elements in $\mathbb{R}^a$. Then the MEAN of P is

$$\frac{\sum_{p \in P} \vec{p}}{n}$$

This is a natural way to extend the familiar concept of the mean to apply to a set of points; just as we would sum a set of numbers and divide by the number of numbers to find the mean of those numbers, we find the mean

---

[1]It could be argued that manipulation is actually desirable. One might say that the voters who would manipulate an election in this way are the voters who are most involved in and passionate about the voting process, and thus are probably the people with the best-informed opinions. Perhaps such voters *should* have a more-than-equal influence on the outcome. However, this line of thought is not popular among voting theorists.

[2]This is more properly called an ANONYMOUS PROFILE, since all voters are considered equal. We don't care who voted for a particular ballot; we only care about how many.

of a set of points by "adding them up", component-by-component, and dividing each component by the number of points. Note that the mean is also the location where the sum of squared distances to all the points is the lowest.

For the mean rule (a.k.a. Borda count), this mean represents the outcome of the election. To interpret it, we consider the WINNING BALLOTS, the set containing the ballot or ballots whose corresponding locations are closest to the mean. Then we can find the WINNING ALTERNATIVES, the set containing the alternative or alternatives who appear at the top of the preference ordering of at least one of the winning ballots. Section 2.3 describes this interpretation process in more detail.

More abstractly, it could be said that the Borda count attempts to find the "average" of a profile and uses this average to select the winners. In the case of the Borda count, the mean is used as the average. However, the mean is not the only measure of average; other measures exist, such as the median. As discussed in [11], this suggests a whole class of AVERAGE-BASED VOTING RULES that all use an "average" but use different ways to find that average.

Averages based on the median concept are particularly interesting to us. In general, the median is often seen as an alternative to the mean that is more resistant to outliers. In the case of voting, this might suggest that a voting rule based on the median would be more resistant to manipulation than the Borda count would be. Of course, this requires extending the median so as to apply to a set of points rather than just a set of numbers. There are several such extensions; the one we use is the MEDIANCENTRE.

Definitions: Let $a \geq 2$ be an arbitrary natural number. Consider a finite multiset P of elements in $\mathbb{R}^a$. The SUM-OF-DISTANCES is a function $F_P : \mathbb{R}^a \to \mathbb{R}$ given by $F_P(\vec{c}) = \sum_{p \in P} \|\vec{p} - \vec{c}\|$.

The set of SUM-OF-DISTANCES MINIMIZERS is given by the function $M : \mathbb{R}^a \to \mathbb{R}^a$ defined as $M = F_P^{-1}[\{l\}]$, where l is the minimum value of F_P.

That is, $M(P)$ is the set of locations that minimize the sum-of-distances from all the points in P.

Definition: If $M(P)$ contains a single location $\vec{m}$, i.e. if $M(P) = \{\vec{m}\}$, then $\vec{m}$ is the MEDIANCENTRE of the points in P.

In other words, the mediancentre is the unique location that minimizes the sum of linear distances to all the given points. This differs from the mean, which minimizes the sum of *squared* distances. Note that there are some cases in which the mediancentre is not defined. Specifically, when (1) there are an even number of points, (2) all the points are collinear, and (3) the middle two points are distinct, then *all* the points on the line segment connecting the middle two points minimize the sum of distances. Thus, the sum-of-distances minimizers form a line segment rather than a single point. We have not determined how to interpret an "average" that is a line segment, so for now we consider the mediancentre to be undefined in this case. We describe the mediancentre in more detail in section 2.2.

Using the mediancentre, we create the MEDIANCENTRE RULE, the recently-introduced average-based voting rule that uses the mediancentre as its measure of average. In other words, it is a version of the Borda count that replaces the mean with the mediancentre. This new rule is the focus of our research. We have seen that it does indeed behave differently from the Borda count; there are several election profiles for which the mean and mediancentre rules produce different winners. To help understand how this new rule behaves, we have written a set of computer programs that probe at its properties so as to guide further research into it.

This work has been the focus of a long line of student research. Grant Lanterman investigated the the mediancentre rule with respect to manipulation in the case of 3 alternatives. Nikhil Srivastava performed some initial research on determining when mediancentre rule produces ties. In the summer of 2007, Ari Morse implemented another mediancentre-rule election solver (for 3 alternatives only) and investigated the rule's decisiveness. Later, over the 2007-2008 academic year, Andy Mackenzie investigated the rule's monotonicity.[3]

This summer, I significantly extended and modified the mediancentre-finder so as to work with any number of alternatives (sections 2.2 and 2.3), continued work on various ways to visualize the rule (section 2.1), and found ways to detect ties between 2 of 3 alternatives (section 2.4). I also extended the investigation of the Mediancentre rule's manipulability to 4 alternatives, suggesting surprisingly high rates of manipulability but nonetheless maintaining the new rule's lead over the mean rule (section 2.5). Also investigated were Condorcet efficiencies for 3 and 4 alternatives, which revealed significant differences in behavior between profiles with even and odd numbers of voters (section 2.6). Discussed also are numerous ideas for future research (section 3) and an overview of the programs written (section 4).

---

[3]Unfortunately, many of the results generated by Ari Morse and Andy Mackenzie were probably inaccurate, due to the issues discussed in section 2.4.

2 Notable Work and Results

2.1 Visualizing Profiles

A long-standing side project involves finding ways to visualize various profiles and how the mediancentre rule behaves under them. Ari Morse generated figures that displayed all the of mediancentres of a number of 3-alternative profiles. Prof. Davide Cervone wrote an excellent JavaScript-based mean/mediancentre visualizer. Andy Mackenzie started work on visualizing the mediancentre rule for 4 alternatives. We have continued the work that Mackenzie started.

Ultimately, the mean, the mediancentre, and the voting rules based on them are geometrical, so visualizing them is natural. All of the ballots are represented as vertexes of a permutohedron, and since these vertexes represent all the possible permutations of the scoring weights, every vertex lies on the hyperplane $x+y+z+\cdots = s$, where s is the sum of the scoring weights used. If a is the number of alternatives, then this plane represents an $a-1$ dimensional cross-section of this space. Thus, the permutohedron used for the 3 alternative case can be conveniently represented in 2-space, and the permutohedron used for the 4 alternative case can be represented in 3-space.

To create these visualizations, we need some way to project this a-dimensional hyperplane into $a-1$ dimensional space. One way to do this is to rotate this plane so as to align its normal vector with one of the coordinate axes of $\mathbb{R}^a$. Once we do this, all points on the plane will have the same value for the component associated with that axis. Thus, we can discard this component and plot the permutohedron using the remaining components, and the shape of the permutohedron will be preserved.

The easiest way to perform rotation is to use a transformation matrix. We find a vector normal to the plane in question; typically, we will use a vector of the form $\langle 1, 1, 1, \cdots \rangle$. Then, we find $a-1$ vectors that are perpendicular with each other and with the plane's normal vector. This will give us a set of a mutually perpendicular vectors. Essentially, we want to perform a rotation such that this set of vectors becomes aligned with the coordinate axes. One way to do this is to normalize these vectors and place them as the rows of a matrix, like so:

$$
\begin{bmatrix}
x_1 & y_1 & \cdots \\
x_2 & y_2 & \cdots \\
\vdots & \vdots & \cdots \\
x_{a-1} & y_{a-1} & \cdots \\
1/\sqrt{a} & 1/\sqrt{a} & \cdots
\end{bmatrix}
$$

Note that $\langle 1/\sqrt{a}, 1/\sqrt{a}, \cdots \rangle$ is the normalized form of $\langle 1, 1, 1, \cdots \rangle$. Since the rows of this matrix are mutually perpendicular unit vectors, this is an orthogonal matrix. Thus, if we use this matrix as a transformation matrix, we will perform either a rotation or a rotation with an inversion. Further, we know that this matrix has one of two values for its determinant; it is either 1 (in which case the matrix only performs a rotation) or -1 (in which case it also performs an inversion). See [7, 10, 4] for more details. Since the permutohedra we are working with are symmetric with respect to inversion, we could simply ignore the possibility of an inversion, but it is easy to avoid. If the determinant of our transformation matrix is -1, we can fix it by negating one of the vectors. The resulting matrix will still be orthogonal, and it will have a determinant of 1. Once we have a suitable matrix, we can apply it to the points of the permutohedron; once we do that, all of the points will have the same value for the last component. This component can then be discarded when graphing.

An example will help illustrate the process. For 3 alternatives, the permutohedron is a regular hexagon, and its vertexes lie on the plane $x + y + z = s$, which is normal to $\langle 1, 1, 1 \rangle$. Trial and error produces $\langle -1, 2, -1 \rangle$ and $\langle 1, 0, -1 \rangle$ as two mutually perpendicular vectors that are parallel to $x + y + z = s$. Normalizing these vectors gives $\langle 1/\sqrt{3}, 1/\sqrt{3}, 1/\sqrt{3} \rangle$, $\langle -1/\sqrt{6}, 2/\sqrt{6}, -1/\sqrt{6} \rangle$, and $\langle -1/\sqrt{2}, 0, 1/\sqrt{2} \rangle$, and putting them into a matrix gives

$$
\begin{bmatrix}
-1/\sqrt{6} & 2/\sqrt{6} & -1/\sqrt{6} \\
1/\sqrt{2} & 0 & -1/\sqrt{2} \\
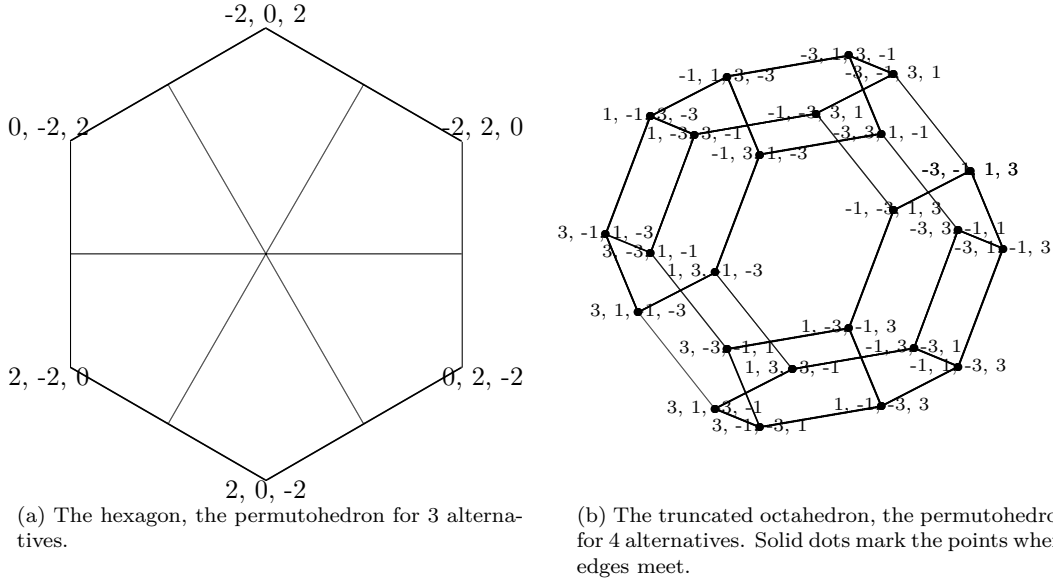1/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3}
\end{bmatrix}
$$

The determinant of this matrix is -1, so to avoid inversion, we pick a row and negate it. Here, we negate the second row:

$$
\begin{bmatrix}
-1/\sqrt{6} & 2/\sqrt{6} & -1/\sqrt{6} \\
-1/\sqrt{2} & 0 & 1/\sqrt{2} \\
1/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3}
\end{bmatrix}
$$

(a) The hexagon, the permutohedron for 3 alternatives.

(b) The truncated octahedron, the permutohedron for 4 alternatives. Solid dots mark the points where edges meet.

Figure 1: Permutohedra for 3 and 4 alternatives. The labels reflect the corresponding vertex's location in $\mathbb{R}^3$ and $\mathbb{R}^4$, respectively.

Given a point P at $\langle P_x, P_y, P_z \rangle$, we can find its new location under the transformation by evaluating:

$$
\begin{bmatrix}
-1/\sqrt{6} & 2/\sqrt{6} & -1/\sqrt{6} \\
-1/\sqrt{2} & 0 & 1/\sqrt{2} \\
1/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3}
\end{bmatrix}
\times
\begin{bmatrix}
P_x \\
P_y \\
P_z
\end{bmatrix}
=
\begin{bmatrix}
-P_x/\sqrt{6} + 2P_y/\sqrt{6} - P_z/\sqrt{6} \\
-P_x/\sqrt{2} + P_z/\sqrt{2} \\
P_x/\sqrt{3} + P_y/\sqrt{3} + P_z/\sqrt{3}
\end{bmatrix}
$$

Note that if P is on the plane $x + y + z = s$, then after this rotation, the last component is a constant given by $P_x/\sqrt{3} + P_y/\sqrt{3} + P_z/\sqrt{3} = (P_x + P_y + P_z)/\sqrt{3} = s/\sqrt{3}$. This is the component that we can discard. Graphing the points using the remaining components gives figure 1a.

A similar process can be used for 4 alternatives. The permutohedron is a truncated octahedron lying on $x + y + z + w = s$. The normal vector is $\langle 1, 1, 1, 1 \rangle$, and we can use perpendicular vectors $\langle -1, 1, -1, 1 \rangle$, $\langle -1, -1, 1, 1 \rangle$, and $\langle -1, 1, 1, -1 \rangle$. Normalizing these vectors and putting them as the rows of a matrix gives:

$$
\begin{bmatrix}
-1/2 & 1/2 & -1/2 & 1/2 \\
-1/2 & -1/2 & 1/2 & 1/2 \\
-1/2 & 1/2 & 1/2 & -1/2 \\
1/2 & 1/2 & 1/2 & 1/2
\end{bmatrix}
$$

The determinant is 1, so no changes are needed. This transformation matrix, along with a plotting program capable of generating 3D graphs, was used to generate figure 1b.

Once we have these figures, we can represent actual profiles in various ways. One way to is to show "dots" at each vertex whose size corresponds to the number of voters who cast the ballot corresponding to that vertex. We can also show the mediancentre on these figures, which further helps visualize the mediancentre rule. This is already implemented for 3 alternatives by Prof. Davide Cervone's visualization program, shown in figure 2 on the following page. Note that for every edge of the hexagon, we draw the line segment connecting that edge's midpoint with the origin. This forms six "wedges", each corresponding to a ballot. Whenever the mediancentre is within a vertex's "wedge", the ballot associated with that vertex is the sole winning ballot, as it will coincide with the vertex closest to the mediancentre.

2.2   Estimating the Mediancentre

Unlike the mean, the mediancentre is very difficult to compute exactly, so we must design and implement a suitable estimator. Ari Morse wrote an estimator that was limited to 3 alternatives. As part of my work, I have
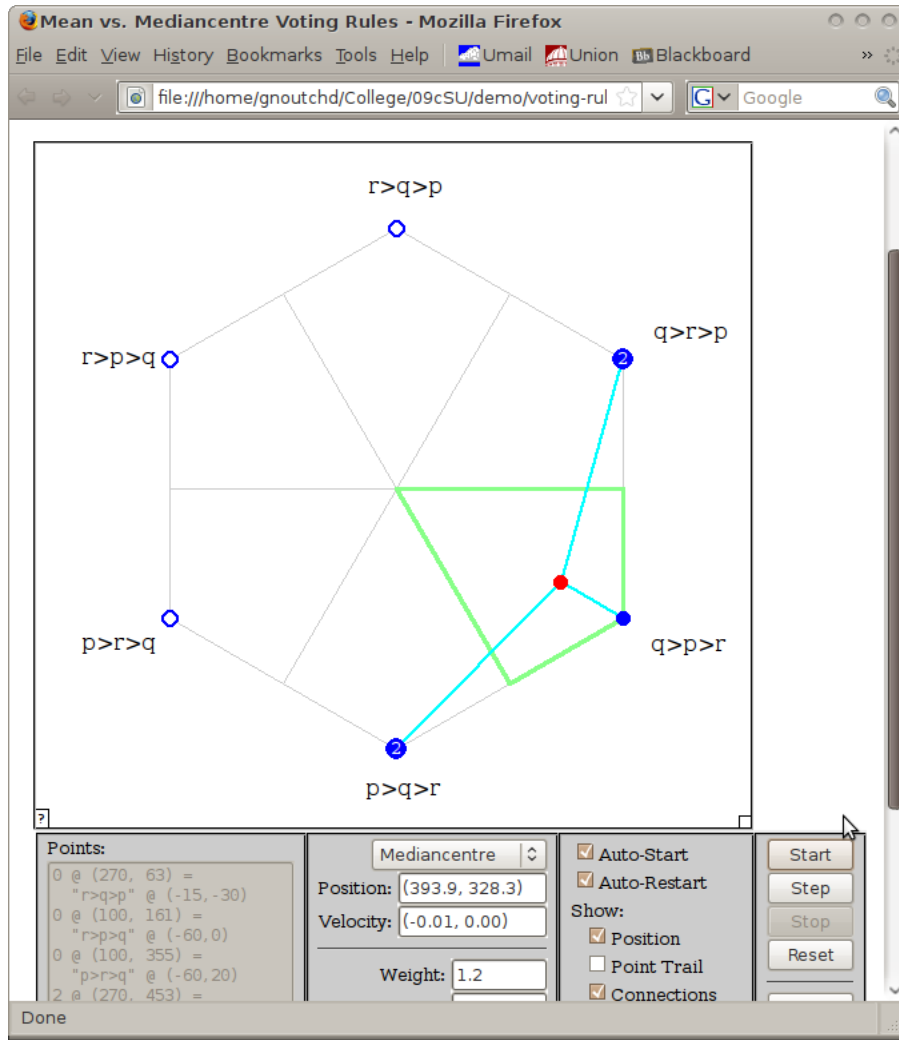
Figure 2: Prof. Davide Cervone's mediancentre visualization program, showing the mediancentre of a profile with 5 voters.
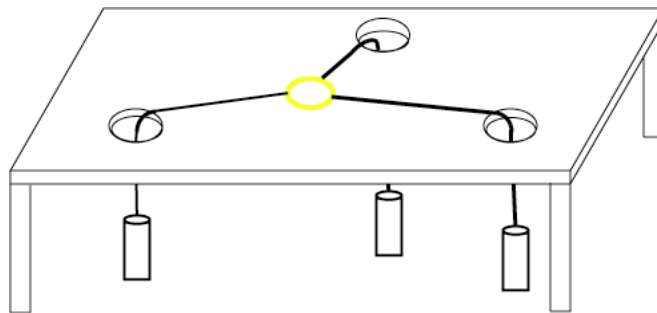

Figure 3: The pulley model of the mediancentre. (Figure credit: Ari Morse.) Each point is represented as a weight on a string that pulls on a ring. The ring stops moving once it reaches the mediancentre.

rewritten it to remove this restriction; now it can handle any number of alternatives. For the most part, however, the method used remains unchanged from Morse's implementation.

To find the mediancentre of a set of points, we use a physical model, the "pulley model". As shown in figure 3 on the previous page, each point is represented as a weighted string pulling on a ring. The mediancentre of this set of points is the location where the ring will settle, i.e. the place there the net force applied to the ring is zero. The mediancentre can be estimated by modeling the behavior of this ring. To do this, we make some starting guess at the mediancentre's location, and we place the ring there[4]. To improve our estimate, we find the net force applied to the ring and move the ring some distance in the direction of that force. After moving the ring, we recompute the net force and move again, repeating until we are satisfied with our estimate (usually when the net force is at or near zero). This algorithm is known as the method of descent; see [5] for a full justification of it.

Many discussions of the descent algorithm are not formulated in terms of net force; most speak in terms of the negative gradient of the sum-of-distances function. In almost all cases, these two formulations are equivalent; the sum-of-distances function is analogous to the potential energy of a set of weighted strings, and the usage of the negative gradient is consistent with the idea that net force is given by the negative gradient of potential energy. The only case where these versions differ is when one of the points that we are finding the mediancentre of is used as an initial mediancentre estimate (i.e. if we place the ring right at one of the holes in figure 3). In such a case, the gradient is not defined, but we can still describe the net force that the ring would experience. We would consider the destabilizing force created by the combined action of the weights at the other locations as opposed to the restoring force created by the weights at the current location. If the destabilizing force exceeds the restoring force, the ring will experience a net force and move away from the point. Otherwise, the ring will experience zero net force and will remain in place. Since net force is defined everywhere, a descent algorithm based on net force works regardless of what initial estimate we use. Ari Morse used the negative gradient, but we have elected to use net force.

Another notable feature of the new implementation is the mechanism that determines *how far* to move the ring in each iteration. A simple-minded approach is to move a distance equal to the magnitude of the net force, but past researchers found that this is too much. In general, we use the magnitude of the net force with some multiplier. In the past, researchers found a suitable multiplier by trial and error. Ari Morse used a multiplier of 0.15 by default, along with an ad-hoc set of rules that reduced the multiplier in certain circumstances. The new implementation uses a more complex approach; it attempts to select the multiplier that causes the greatest reduction in the sum-of-distances function. It starts with a multiplier of 1 and computes the sum-of-distances that would result if we moved the ring according to that multiplier. Then, it halves the multiplier and determines if this results in a lower sum-of-distances. This continues until we find that halving the multiplier fails to improve the sum-of-distances, in which case we've already found a near-optimal multiplier. The sum-of-distances function is convex, which implies that if reducing the multiplier increases the sum-of-distances, further reducing the multiplier would further increase the sum-of-distances.

Finally, the new implementation adds a small error check - we refuse to move the ring if this would increase the sum-of-distances. In some cases, especially when we get *very* close to the true mediancentre, the old mediancentre estimator could end up worsening the estimate (as judged by the sum-of-distances). The new implementation ensures this never happens.

We hope that these changes make the new implementation faster and more effective at finding the mediancentre. Basic testing suggests that the new implementation is at least as good as the original in terms of the accuracy of the estimates, but more research needs to be done.

There are two special cases that must be checked for specifically, as the method of descent does not handle them well. One is the case where the mediancentre is undefined. Recall that the mediancentre is undefined whenever (1) there are an even number of points, (2) all the points are collinear, and (3) the middle two points are distinct. In this case, the set of distance minimizers forms a line segment connecting the middle two points. The method of descent will not detect this; the "ring" will settle on some point on that line and it will appear as if a unique distance minimizer has been found. So we must test for this case beforehand. Since we are working with points on the vertexes of a permutohedron, a set of points is collinear if and only if they all lie on two vertexes. Thus, for our purposes, the mediancentre is undefined if and only if all voters are evenly split over two ballots. This is easy to test for.

The second special case we must test for is the case where the mediancentre is actually at one of the permutohedron's vertexes. This can happen when, for example, all votes are on a single ballot. While the descent

---

[4]We often use the mean for this estimate, as it often is fairly close to the mediancentre and is easy to compute.

algorithm can handle this case correctly, it is often quite slow. Thus, we manually check the vertexes first before invoking the descent algorithm. This can be done by "placing" the ring at each of the vertexes and computing the net force it experiences there; if the force is zero, the mediancentre has been found. Note that the same routine used to compute net force for the descent algorithm can also be used for this test. This simplifies the implementation, which is another reason to use a net-force based descent algorithm.

2.3 Finding the Outcome

Once we find the average of a set of votes (i.e. the mediancentre, the mean, or something else), we must convert it into an actual election outcome. Recall that the WINNING BALLOTS are those ballots that are closest to the average, and the WINNING ALTERNATIVES is the set of alternatives that are placed at the top of the preference ordering of at least one of the wining ballots. If there is only one winning ballot, we clearly have one winning alternative, the alternative with the highest rank on the sole winning ballot. If there are several winning ballots, we may still have only one winning alternative if all of the winning ballots have the same alternative at the top of their preference orderings. Otherwise, we will end up with a tie among several alternatives.

To find the winning ballots, we could use the brute-force approach of computing each ballot's distance from the average and searching for the shortest distances, but there is a much faster and more elegant way. We can consider the preference ordering that is INDUCED by the average. Specifically, we know that given alternatives P and Q, if the component of the average associated with P is greater than the component associated with Q, then all of the winning ballots must rank P over Q[5]. Similarly, Q must be ranked above P if Q is associated with the larger component, and if the two components are equal, then P and Q are tied for the same position and may appear in any order on the winning ballots. This is enough to completely determine the set of winning ballots.

To analyze an outcome (and to find the winning ballots), it's useful to create an ordered partition of the alternatives. Each alternative is placed into a subset along with the other alternatives it's tied with, and the subsets are ordered according to the preference ranking induced by the average. This representation is used extensively in our outcome analysis program, and it is also the representation used when printing election outcomes for the user to read. For example, we may have "P>Q>R", a single winning ballot, corresponding to a partition where each alternative is placed in its own subset; "P>[QR]" represents a tie between ballots P>Q>R and P>R>Q, corresponding to a partition where P is in the top-ranked subset and Q and R are put together in the lower-ranked subset; and "[PQR]", a tie between all six ballots, corresponding to a partition where all three alternatives are in the same subset.

This ordered partition is very useful for further analysis of the outcome. To find the winning alternatives, we need only look at the highest-ranked subset in this partition[6]. To find the winning ballots, we find all the permutations of the alternatives within each subset. Then, we can "build" one of the winning ballots by stepping along each subset in descending order of rank, selecting one permutation of that subset, and adding the alternatives in the subset to the end of the new ballot's preference ranking in the order specified by the selected permutation. If we proceed to "build" more ballots using all the possible combinations of permutations, we will generate all the winning ballots.

2.4 Mediancentre Ties

As discussed in section 2.3, an average-based voting rule only gives ties when the average has at least one pair of equal coordinates. Unfortunately, this equality is *very* difficult to test for when using the mediancentre as the average. We compute the mediancentre with floating-point arithmetic, and it is well known that equality is difficult to establish in floating-point. Because of very subtle rounding errors, two operations that are mathematically equivalent still might produce slightly different results when done by computer. Furthermore, recall that we are using mediancentre *estimates*, so true equality is unlikely to occur even if we ignore the traditional problems of floating-point. Thus, if we attempt find ties in the mediancentre rule by using the methods described in section 2.3 literally, we are virtually guaranteed to miss nearly all tied outcomes and incorrectly declare them as non-ties. Since we are very interested in the effects of ties, this is a major problem.

---

[5]This is proved in section A on page 18.

[6]For performance reasons, the code used to find the winning alternatives does not depend on the full ordered partition; since we only need the top-ranked subset, generating *all* the subsets is a waste of time. Rather, the code generates only the top-ranked subset, which is done by looking at the average itself and finding the alternatives associated with the components with the highest value.

The kludge we have used is to have the outcome analyzer consider two components to be "equal" if the difference between then is below an arbitrary "error" threshold. However, we have found that this does not work well; indeed, we believe that for this reason, much of the data generated by Ari Morse and Andy Mackenzie was inaccurate. We have no good method for selecting the error threshold; we must find it by trial and error, checking the outcome analyzer's results on a set of profiles known to generate ties. If the threshold is too high, we end up declaring ties when we should not (false ties), and if it is too low, we fail to declare ties when we should (false non-ties). Worse, it appears that there does not exist a value that eliminates all incorrect results. In the case of 3 alternatives, once we set the threshold high enough (i.e. to about 10^{-5}) to correctly declare ties for all those profiles that are known to give ties, we start getting what appear to be false ties for 15 voters or more. Clearly, this kludge is not reliable; we need a better way to detect ties.

Fortunately, useful patterns have been found among the profiles that cause ties. For 3 alternatives and odd numbers of voters, Nikhil Srivastava found some patterns among the profiles that result in a tie between all six ballots (and thus all three alternatives). To describe the patterns he found, a few definitions are necessary.

Firstly, recall that we can treat a profile as a function. We can also speak of adding and multiplying these functions:

Definitions:

Given profiles A and B, both for a alternatives, $A + B$ is the profile where, for every ballot β, $(A+B)(\beta) = A(\beta) + B(\beta)$.

Given a profile A and a non-negative integer n, nA is the profile where $(nP)(\beta) = n \times P(\beta)$ for every ballot β.

We will call a profile N a COMBINATION of a finite set of profiles $\{A, B, C, \ldots\}$ if it can be expressed in the form $N = aA + bB + cC + \ldots$, where $\{a, b, c, ...\}$ is a multiset of non-negative integers.

The idea of combining profiles in this way is sometimes quite useful. If we know something about the mediancentre and/or the sum-of-distance minimizers of the component profiles, we can often say something about the mediancentre of the combined profile. This is because the mediancentre is based on the sum-of-distances: the sum-of-distances function treats every point in the profile independently, which means that we can add sum-of-distances functions in the same way we add profiles:

Proposition: Let F_C denote the sum-of-distances function of a profile C, and let A and B denote profiles for some number of alternatives a. For all locations l in R^a, $F_{A+B}(l) = F_A(l) + F_B(l)$.

This implies something quite interesting: if l is a sum-of-distances minimizer for both F_A and F_B, then l is also a location that minimizes $F_A + F_B$ and thus F_{A+B}. In other words, any location that minimizes both F_A and F_B also minimizes F_{A+B}. Stated formally:

Proposition: Let $M(A)$ denote the sum-of-distances minimizers under profile A. Let A and B be profiles for some number of alternatives a. We have that $M(A) \cap M(B) \subseteq M(A + B)$.

Corollary: Let $\mathbb{P}$ be a finite set of profiles. Let $G = \bigcap_{P \in \mathbb{P}} M(P)$. If C is a combination of the profiles in $\mathbb{P}$, then $G \subseteq M(C)$.

Stated informally, this means that if l is a sum-of-distances minimizer for all profiles in a set $\mathbb{P}$, then l is also a sum-of-distances minimizer of *any* profile that is a combination of the profiles in $\mathbb{P}$. This is a very useful property, and Srivastava was able to exploit it to help find ties.

When searching for ties among all 6 ballots, there are two types of profiles of interest:

Definitions:

A TRIANGULAR PROFILE is a profile for 3 alternatives with 3 votes such that ballots cast correspond to a set of vertexes that form an equilateral triangle.

Given a ballot β, $R(\beta)$ denotes the ballot that is formed by completely reversing the preference ordering given by β. An OPPOSITE PAIR PROFILE is a profile for 3 alternatives with 2 votes such that the ballots cast are of the form β and $R(\beta)$.

Examples of triangular and opposite pair profiles are given in figure 4 on the next page. What's significant about these profiles is that they both include, as a sum-of-distances minimizer, the location that induces a tie between all 6 ballots. Let us restate this formally.
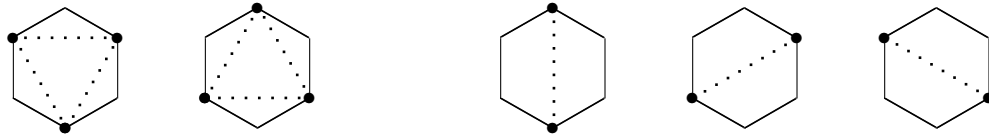
(a) Triangular profiles

(b) Opposite-pair profiles
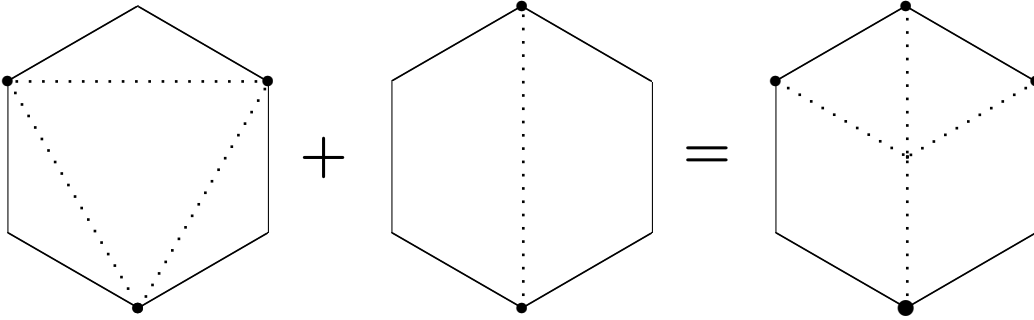
(c) Example of combining a triangular profile and an opposite-pair profile.

Figure 4: Basic profiles used to find 6-ballot ties and an example of combining these profiles. Any profile that is a combination of the basic profiles will give a 6-ballot tie.

Definition: The STRADDLE-POINT is the location that is equidistant from all the locations corresponding to ballots.

If the average (e.g. mediancentre) is at the straddle-point, we will have a tie among all of the alternatives. The word "straddle" reflects the fact that the mediancentre rule is completely indecisive in this situation, i.e. it "straddles the fence".

Proposition: The sum-of-distances minimizers under any triangular or opposite-pair profile includes the straddle-point.

Corollary: Any profile that can be expressed as a linear combination of triangular and opposite-pair profiles will include the straddle-point as a sum-of-distances minimizer.

We now have a way to generate a large number of profiles that include the straddle-point as a sum-of-distances minimizer. Recall that for any profile, the set sum-of-distances minimizers may be one of two kinds; it will be either of a line segment or of a single location, and as section 2.2 shows, it's easy to determine which one it will be. If the profile in question is a combination of triangular and opposite-pair profiles, and if we know that the sum-of-distances minimizers must consist of only a single location, we now know that this single location must be the straddle-point. And, of course, if we have a single sum-of-distances minimizer, that location is the mediancentre. Thus we have that:

Corollary: Under the mediancentre rule, any profile that can be expressed as a combination of triangular and opposite pair profiles will give an outcome that is either undefined or a 6-ballot tie.

In both our work and that of Srivastava's, it has been found that every profile that can be expressed as a combination of these basic profiles does indeed give a 6-ballot tie. We have also found that the converse appears to be true; for the profiles we tested, every profile that appears to be a 6-ballot tie with the error threshold kludge is indeed a combination of triangular and opposite-pair profiles. This leads us to:

Conjecture: Under the mediancentre rule, every profile that gives a 6-ballot tie is a combination of triangular and/or opposite-pair profiles.
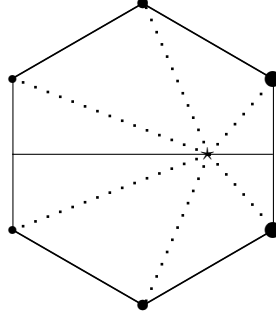
Figure 5: A line-symmetric profile.

In an extension of Srivastava's work, we investigated ties that occur when there are an even number of voters. Srivastava's tests for 6-ballot ties still work, but we now must also consider ties between only 2 ballots. We have found that unlike for 6-ballot ties, we are not able to combine profiles to generate 2-way ties, but we have nonetheless found two types of profiles that always give 2-way ties.

The first and simplest type is a LINE SYMMETRIC profile:

Definition: Consider the hexagon formed by the locations representing the ballots for 3 alternatives. Consider a line that bisects this hexagon and intersects it at two of its edges at their midpoints. If the multiset of points that represents a profile P is symmetric about this line, then we say that the profile is LINE SYMMETRIC.

Proposition: Under the mediancentre rule, a line symmetric profile will give an outcome that is either undefined, a 2-ballot tie, or a 6-ballot tie.

An example of a line symmetric profile is given in figure 5. If all the votes are evenly split over two adjacent vertexes, then the mediancentre is undefined. Otherwise, the mediancentre will be somewhere on the bisecting line. Note that this line intersects the straddle-point, and of course we have a 6-ballot tie if the mediancentre is at this point; this happens when the votes are evenly split among all the ballots. However, such a profile is also expressible as a combination of triangular and/or opposite-pair profiles, so it will be caught and identified as a 6-ballot tie if we apply Srivastava's tests before checking for line symmetry. The final case is a actual 2-ballot tie, an example of which is shown in figure 5.

The second kind of 2-way tie profile, a CROSS PROFILE, is much more complex. We define it like so:

Definitions:

A CHORD PROFILE is a 3-alternative profile with a number of votes greater than zero where all the votes are evenly split over two ballots such that the preference orderings on the two ballots are not reversals of each other but vertexes corresponding to the ballots are not adjacent. We will call the line segment connecting these two vertexes the CHORD of this profile.

If a profile can be expressed as a sum of two chord profiles such that the chords intersect, then we call it a CROSS PROFILE.

Figure 6a on the next page shows some example chord profiles, and 6b contains an example of two chord profiles being combined to form a cross profile. We can apply what we know about combining profiles and sum-of-distances functions to find the mediancentre of a such a profile. Note that the chord of a chord profile is also the the set of sum-of-distances minimizers for that profile. This means that when two chord profiles are combined to form a cross profile, the intersection point of the chords must be a sum-of-distances minimizer of the new cross profile. Thus we have:

Proposition: Let A and B be chord profiles such that the chords of A and B intersect. The mediancentre of the cross profile $A + B$ is at the intersection of the two chords.

11
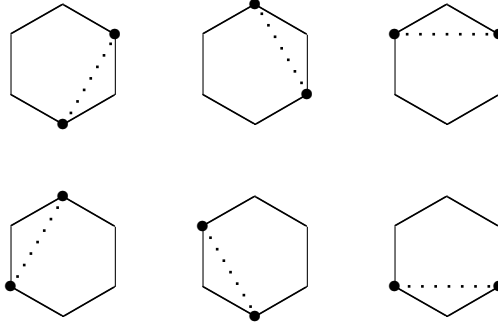
(a) Examples of chord profiles. Chord profiles are used to help define and analyze cross profiles.
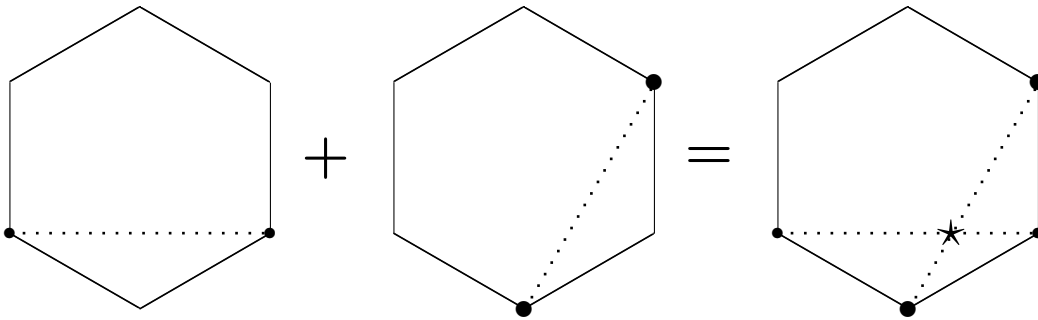
(b) An example of a cross profile, created by adding two chord profiles.

Figure 6: Chord and cross profiles.

Since we are working with a regular hexagon, the angles formed between the chord and the hexagon's edges are the same for every chord, so the triangle formed by the mediancentre and the two closest vertexes is isosceles. Therefore, the two vertexes closest to the mediancentre are equidistant from it, so we have a 2-ballot tie.

Proposition: Under the mediancentre rule, a cross profile gives a 2-ballot tie.

As with 6-ballot ties, we have found the 2-way ties declared with the error-threshold kludge are all quite consistent with these patterns.[7] When the threshold is set to 10^{-5}, and when there are less than 15 voters, all profiles declared as 2-way ties with the error threshold are also declared as ties by programs that check for these patterns. Once we reach 15 voters, there are a few profiles declared as 2-way ties that don't match these patterns, but they look suspiciously like false-ties, especially since that we suspect that 2-way ties are impossible with odd numbers of voters. These results encourage us to make another conjecture:

Conjecture: Under the mediancentre rule, all profiles that generate 2-ballot ties are either line symmetric profiles or cross profiles.

With our two conjectures, we are optimistic that we do indeed have a reliable way to detect ties. However, we have not formally proven any of this, nor have we made any attempt to extend this to 4 alternatives or more. Currently, the error threshold kludge is the only way to detect ties among 4 alternatives, and we have made no attempt determine how well this method performs in this case. More research needs to be done to improve this situation.

---

[7]It should be noted that the error threshold approach often struggles to detect ties on cross profiles; when the threshold is lowered, the first false non-ties are often cross profiles.

2.5 Manipulability

Manipulability is one of the properties we have investigated. We wish find when the mediancentre rule is prone to manipulation, how often it is prone, and how it compares to the mean rule in this regard. As hinted at in section 1, manipulability relates to whether some voter can obtain a more favorable outcome by misrepresenting their preferences. Stated more formally:

Definitions:

Let P be a profile and β be a ballot for which $P(\beta) \geq 1$. Let γ be a ballot that is distinct from β. Consider the VOTE SWITCH $Q = S(P, \beta \to \gamma)$, the profile that is formed by moving one vote from β to γ; that is, $Q(\beta) = P(\beta) - 1$, $Q(\gamma) = P(\gamma) + 1$, and for all ballots σ such that $\beta \neq \sigma \neq \gamma$, $P(\sigma) = Q(\sigma)$. Let p and q be the sole winners of a voting rule V under profiles P and Q, respectively. If:

1. p and q do indeed exist,
2. they are distinct, and
3. q is ranked above p on β,

then the vote-switch Q is a MANIPULATION of P under voting rule V.

If there exist manipulations of P under under V, then P is said to be MANIPULABLE under V.

The idea behind this definition is that, assuming ballot β represents some voter's true preferences, that voter should not be able to obtain a more favorable outcome by misrepresenting their preferences and casting γ instead.[8] This is a very simplistic way to think about manipulability, but it is a reasonable way to start.

For these investigations, we use an extended version of the mediancentre rule, the mediancentre rule with a FIXED TIE-BREAKING AGENDA. The tie-breaking agenda is a predefined ordering of the alternatives used to to break ties between alternatives. If a voting rule gives several winning alternatives, we select the one that is ranked highest in the agenda ordering. In all other cases, including those cases where the voting rule is undefined, the result is left unchanged. Tie-breaking agendas have often been used to avoid the issue of ties when analyzing a voting rule; it was first proposed in [6], and it was first applied to studies of manipulability in [1].

Currently, we find manipulable profiles by brute force. Given a base profile, we find all the profiles that can be generated from the base profile by changing one vote, and we check each profile generated to determine if it represents a manipulation of the base profile. To find what fraction of a set of profiles are manipulable, we repeat this process for every profile in that set. We make no attempt to reuse the results from the mediancentre rule; for every base profile and potential manipulation considered, we re-run the mediancentre-solver described in section 2.2, even if the same profile was considered earlier in program execution. This is usable for low numbers of voters, but it is grossly inefficient. We expect that these programs can be made much faster with further research.

Figure 7 on the following page shows the results we have so far. For each number of alternatives and each number of voters, we have computed the ratio of the number of profiles manipulable under the mediancentre rule to the number of profiles for which the mediancentre is defined. We have considered profiles with 3 and 4 alternatives and with 3 to 6 voters. For comparison, we have done the same for the mean rule. Note that the concerns raised in section 2.4 apply; we do not know if we are finding ties correctly in the 4 alternative case, so these results are subject to revision. Generally, the mediancentre appears to be more resistant to manipulation than the mean rule, which we expected. However, we are surprised by how many profiles seem to be manipulable in the 4 alternative case. Further research is needed to verify these results.

Note that the tie-breaking agenda is not the only way to handle ties; there are other methods worth researching. For example, one can discuss IRRESOLUTE MANIPULABILITY, which considers ties directly. For example, one may have a situation where the base profile has a single winner P, and a voter who prefers S over P may be able to manipulate the election into producing a 2-way tie between P and S; this would be considered a manipulation when considering irresolute manipulability, but it is missed when using a tie-breaking agenda that ranks P over S. This is discussed in more detail in [8, 9].

---

[8]One issue with this definition is that it assumes that each voter's true preferences are perfectly represented by some ballot. However, problems arise when there is a tie between alternatives within a single voter's preferences. Things become even more complicated one one considers issues of psychology, which may suggest that a voter may not even know what their true preferences are. At present, however, we are not prepared to deal with such complexities.
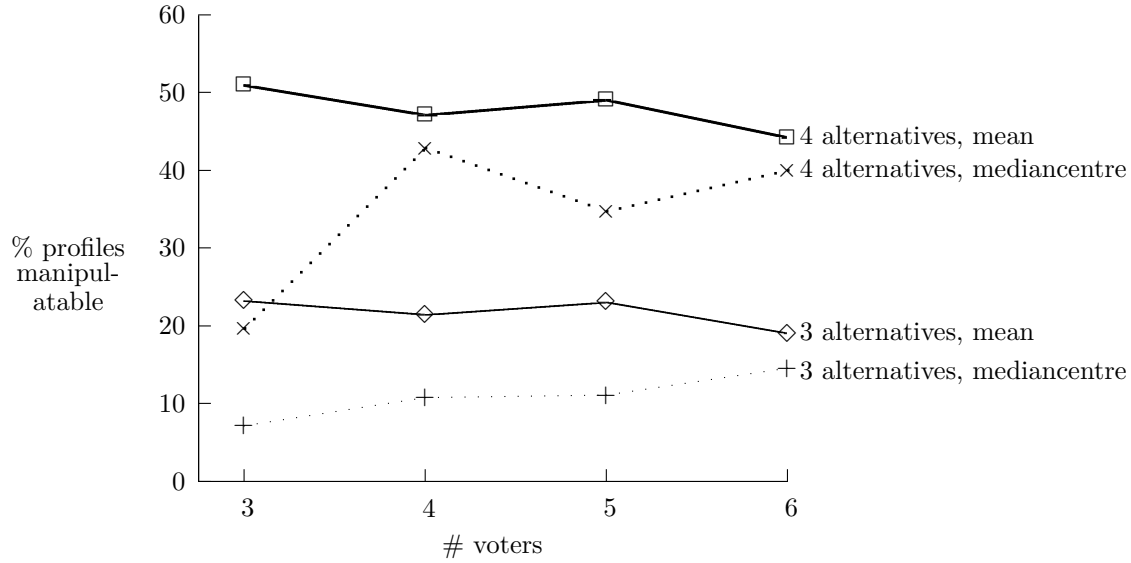
Figure 7: Percent profiles manipulable under the mean and mediancentre rules. Ties are resolved using a tie-breaking agenda. This graph shows the ratio to the number of manipulable profiles to the total number of profiles for which the rule is defined. For the mean, which is defined for all profiles, this total is equal to the number of profiles given. For the median, it is somewhat lower. For these low numbers of voters, the median rule is generally more resistant to manipulation, as expected. However, we are surprised by how high some of these ratios are.

2.6   Condorcet Efficiency

A STRONG CONDORCET WINNER for a given profile is the alternative who can defeat every other alternative in a 1-on-1 race. A Condorcet winner does not exist for every profile, but if there is one, we generally like to have our voting rules select it as the winner. The CONDORCET EFFICIENCY of a given voting rule and a given set of profiles is the ratio of the number of profiles for which the rule selects the Condorcet winner to the total number of profiles for which the Condorcet winner exists.

We implemented a program to find Condorcet efficiencies by solving and checking every profile given. Figure 8 on the next page contains the results. Note that we see very different behavior for even and odd numbers of voters, so it seems best to consider these cases separately. As with our manipulability investigations, these results are subject to revision as we find out how to verify ties for 4 alternatives.

3   Ideas for Further Research

Visualizing 4-alternative profiles. We may wish to develop the equivalent of Davide Cervone's 3-alternative mean/mediancentre program for 4 alternatives.

Finding ties for 4-alternatives and more. It is important for us to have a reliable way to detect ties, as discussed in section 2.4.

Caching mediancentres. Every time we run a new program to check some new property of the mediancentre rule, we end up re-running the mediancentre finder for every profile. Sometimes, we end up solving the same profile many times within a single instance of a program (e.g. within the manipulability checks described in section 2.5). There is no need for all this repetition; it can all be avoided by computing the mediancentres of each profile and storing the results to disk. These results could then be loaded to main memory as needed.
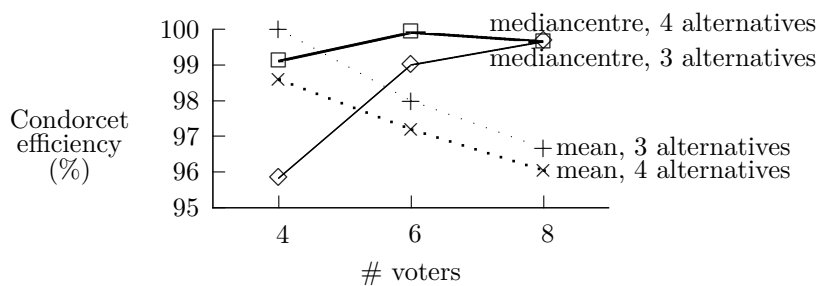
Considering root profiles. The permutohedron has many symmetries; it can be transformed in numerous ways to produce another permutohedron of the same shape. We can apply the same kind of transformations on a profile, and if we have the original profile's mediancentre, we can apply the same transformations on it to
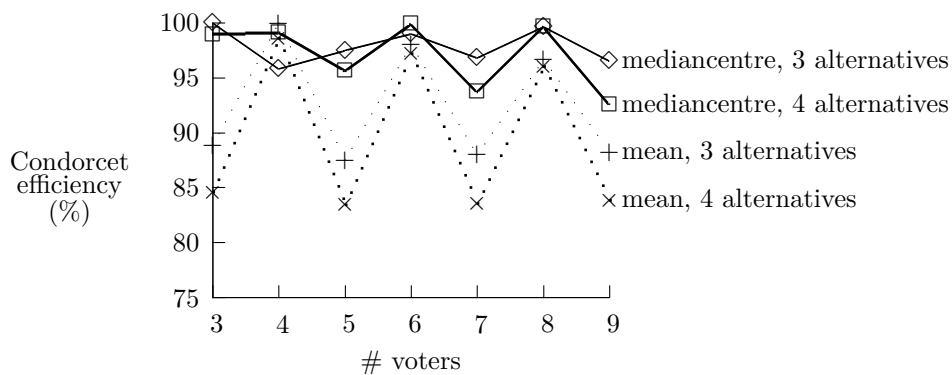
(a) All numbers of voters
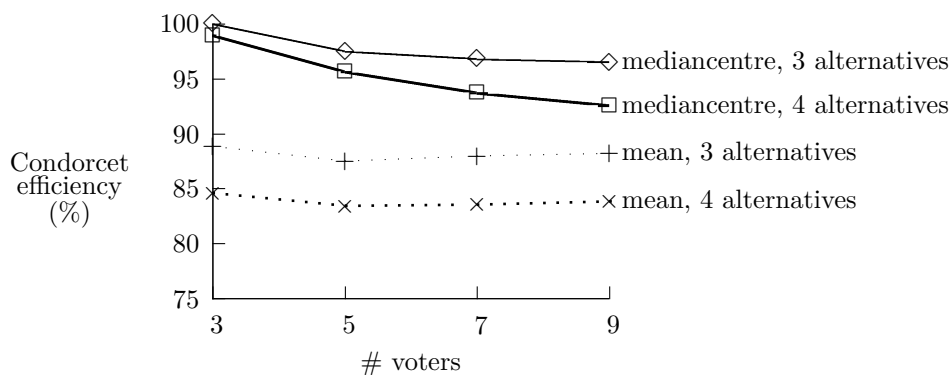
(b) Odd numbers of voters

(c) Even numbers of voters

Figure 8: Condorcet efficiencies

15

get the new profile's mediancentre. Thus, once we compute the mediancentre of one profile, we can easily find the mediancentre of all the profiles that have the same structure as does the first. Other researchers have looked into these symmetries (e.g. in [3]), using the name ROOT PROFILES to describe these structures. We should see if we can exploit the methods they developed so as to reduce the number of profiles we have to compute and store the mediancentre of.

Optimizing the mediancentre finder. As discussed in section 2.2, there were some tweaks made to the mediancentre finder made with the hope of improving accuracy and speed. While the new implementation is fairly accurate, it has not been determined if it is any faster. There are a few profiles where the new solver takes a very long time to complete; these profiles should be checked for potential performance improvements.

Replacing the method of descent. [2] claims to present a faster way to find the mediancentre. It may be worth implementing the algorithm it describes.

Manipulability and ties. As discussed in section 2.5, there are several different ways to handle ties when considering manipulability. They are worth investigating.

Weak Condorcet winners. A weak Condorcet winner is an alternative who beats *or ties* every other alternative in a 1-on-1 race. Unlike with strong Condorcet winners, a given profile may have several weak Condorcet winners (who are all tied with each other). Whether there are an even or odd number of voters has a large influence on ties, so it would be interesting to see how weak Condorcet winners would affect the disparity between the even and odd case with respect to Condorcet efficiency.

Decisiveness and one-way monotonicity. Andy Mackenzie and Ari Morse investigated these properties for the mediancentre rule for 3 alternatives. It would be interesting to extend this to 4 alternatives or more.

4 Computer Programs: Overview

Our research involves writing and running a set of Java utilities that compute the mediancentre and analyze the properties of the mediancentre rule. Here is a high-level overview of how these programs are structured and used. Additional documentation is available in the form of comments in the source files.

Data Structures

The VECTOR class is used to represent vectors in Cartesian coordinate space. It should not be confused with the standard java.util.Vector class. The name (and the associated name-space collision) remains for historical reasons; it should be renamed. Note that a Vector instance may represent an "undefined vector" - this is used (for example) to represent an undefined mediancentre. This is analogous to "not a number" values for floating point numbers. See the source code documentation of Vector.is_nav() and Vector.NaV for details.

Internally, profiles are represented as arrays of $a!$ integers, where a is the number of alternatives. To determine what ballot corresponds to some element in these arrays, we reference the BALLOTS class. The number of alternatives is set when the Ballots class is instantiated. Then, the Ballots.get() function accepts an index for the profile array and returns a position VECTOR that contains the position of the vertex that corresponds to the ballot for that profile entry. Ballots also provides a few methods for interpreting and describing these ballots, including methods that indicate what scoring weights are used, the names of the alternatives associated with each component index, and the string representations of various ballots.

Profile Enumeration

The PROFILENUMERATION class is responsible for finding and listing all profiles with a alternatives and n voters. The list is stored in a text file named "a_n.txt". Each profile is stored as a line of space-separated integers, suitable for parsing and inputting into the array of integers that forms the internal profile representation.

These text files are generated purely for historical reasons; there is no real reason for storing these lists. When the programs are modified to cache mediancentre solutions, these lists should be replaced with files that contain mediancentre estimates along with profiles.

Profile Analysis

The PROFILEANALYZER and OUTCOMEANALYZER classes form the core of our set of programs. The first class is instantiated with a profile array and a Ballots instance with which to interpret it. A ProfileAnalyzer instance encapsulates a profile and a contains a running estimate of the location of the mediancentre under that profile. The ProfileAnalyzer.optimize_median_estimate() method uses the method described in section 2.2 to generate a very good estimate of the mediancentre; then the get_median_estimate() method can be used to get the estimate. ProfileAnalyzer also exposes other methods that allow one to set and manipulate the mediancentre estimate; this is exploited by some programs (e.g. the Solve program) to gain more control over and information about the estimation process. ProfileAnalyzer also has a method for finding the mean, which is used by the mediancentre estimator and is also useful by itself for analyzing the mean rule.

Once the mediancentre (or mean) is found, the election outcome that it corresponds to can be analyzed with OutcomeAnalyzer. When an OutcomeAnalyzer is instantiated, it is given the average to interpret and a Ballots instance to use for interpreting it. Then, the OutcomeAnalyzer supplies methods that are useful for interpreting the outcome; one can get the set of winning ballots, the list of the winning alternatives, or a useful string representation of the outcome. Note that the source code and documentation often uses the term "center" instead of "average"; early on, I tended to use the word "center" instead of "average" in an attempt to make it clear that the average we use is not necessarily the mean. The source code reflects this history.

Note that OutcomeAnalyzer uses the error threshold kludge described in section 2.4, and it uses a fixed, hard-coded threshold value. This is a design flaw; the error threshold should really be specified (or even automatically generated) in ProfileAnalyzer and passed to OutcomeAnalyzer. Currently, averages are represented as plain position vectors, using our Vector class; it may be desirable to replace this with a representation that specifies error thresholds.

Profile Input

Generally, we have written our programs to accept a list of profiles from standard input, in the form generated by ProfileNumeration. This gives us much flexibility with what profiles to process at a given time; we can process one or two profiles of interest by entering them directly into the terminal, or we can process a long list of profiles by using input redirection. Input redirection is supported by the command-line interfaces of all major operating systems. For example, the command

```
$ java Manipulability 4 <4_5.txt >4_5.manip.txt
```

runs the Manipulability class (see section 2.5) using the file '4_5.txt' as input (which was generated by ProfileNumeration, listing all profiles with 4 alternatives and 5 voters) and storing the results in a file named '4_5.manip.txt'.

There is a fair amount of busywork involved in reading these profiles. To avoid code duplication, this work has been centralized by the PROFILESOURCE class and PROFILESINK interface. ProfileSink is implemented by classes that accept profiles as input and output information about them in text form. ProfileSource is a place to put static methods that can feed profiles into and accept output from classes that implement the ProfileSink interface. Currently, it only contains ProfileSource.stdio(), which reads profiles from standard input and outputs results to standard input (along with some formatting to ease reading).

Applications

A number of programs have been written that either test the core classes or do something useful with them. Given below are their class names and brief descriptions.

CondorcetChecks For every profile given in standard input (which must match the number of alternatives given as a command line argument), find the Condorcet winner and, if it exists, determine if the mean and/or mediancentre rules select it.

LegacyCheck For every 3-alternatives profile given in standard output, find and compare the mediancentre and outcome generated by both the new ProfileAnalyzer/OutcomeAnalyzer system and by Ari Morse's original implementation. Used for testing. Not included in the public distribution of this code, as Ari's code has not been cleared for free distribution.

GarrisonedTester Lists the locations and descriptions of all the ballots for the given number of alternatives. Created for testing, but also useful for interpreting profiles by hand. The "garrisoned" in the name came from [5], which describes the process of associating a ballot with a location as "garrisoning".

Manipuability [sic] Taking the number of alternatives from the command line and reading profiles from standard input, finds and prints all manipulations of every given profile, under both the mean and mediancentre rules. (The name of this class is misspelled; this should be fixed.)

ProfileDump Simply finds and prints the mean and mediancentre of each profile given on standard input. The number of alternatives must be given as a command-line argument.

ProfileNumeration Described in the "Profile Enumeration" section above.

Solve Finds the mediancentre of the profile given in command line arguments, printing detailed information about the estimation process. After the profile, one can also specify "start-ballot <integer>" or "start <x coordinate> <y coordinate> ..." to specify a ballot vertex or some other location to use as the initial estimate for the method of descent.

TieChecks Reads 3-alternative profiles from standard input and determine if it has any of the patterns that are known to cause ties. It uses the SYMCHECKS class do to the actual symmetry checks; this class is not discussed in this document but the methods it uses are discussed in section 2.4. TieChecks also computes the mediancentre rule's outcome according to ProfileAnalyzer/OutcomeAnalyzer and determines if there is a disagreement about whether there is a tie or not.

Vis Reads a 3-alternative profile from command-line arguments and writes into the current working directory an HTML file named "vis.html" that visualizes the given profile using Davide Cervone's visualization program. This program hard-codes the filename of the visualizer, so it will need to be edited to work on your setup. As of this writing, Vis is not included in the public release of my work, as Prof. Cervone's program also has not been publicly released.

A Induced Ordering: Proof

Let A be a set of $n \geq 2$ distinct real numbers. Let P_A be the vertexes of the permutohedron of the numbers in A. Note that $P_A \subset \mathbb{R}^n$. Let $m \in \mathbb{R}^n$ be an arbitrary location in $\mathbb{R}^n$. Let W be the vertexes in P_A that minimize the distance to m; that is, for any $x, y \in \mathbb{R}^n$, $\|x - m\| < \|y - m\|$ iff $y \notin W$. Let $v \in W$ be arbitrary. Let a and b be distinct integers such that $0 < a \leq n$ and $0 < b \leq n$, and if $z \in \mathbb{R}^n$, then let z_a and z_b denote the ath and bth components of z, respectively. We claim that $m_a > m_b \implies v_a > v_b$.

Proof: Consider the case where we indeed have $m_a > m_b$. Assume, for contradiction, that $v_a \leq v_b$. Since a and b are distinct, v_a and v_b are distinct components of v, and since v is a vertex of the permutohedron of a set of distinct real numbers, $v_a \neq v_b$. Thus, we have $v_a < v_b$.

Now consider v', which is formed from v by swapping the ath and bth components of v. Since v' is just another permutation of the numbers in A, $v' \in P_A$. Since the ath and bth components are the only ones that have changed, we have that

$$\|v' - m\|^2 - \|v - m\|^2 = \left[(v'_a - m_a)^2 - (v_a - m_a)^2 \right] + \left[(v'_b - m_b)^2 - (v_b - m_b)^2 \right]$$

Substituting $v_a = v'_b$ and $v_b = v'_a$ gives

$$\left[(v_b - m_a)^2 - (v_a - m_a)^2 \right] + \left[(v_a - m_b)^2 - (v_b - m_b)^2 \right]$$

which simplifies to

$$2(v_a - v_b)(m_a - m_b)$$

Now, we have that $m_a > m_b$ and $v_a < v_b$, so $m_a - m_b$ is positive and $v_a - v_b$ is negative, so $2(v_a - v_b)(m_a - m_b) = \|v' - m\|^2 - \|v - m\|^2$ is negative. This gives $\|v' - m\|^2 < \|v - m\|^2$, or $\|v' - m\| < \|v - m\|$, so v' is closer to m than is v. But $v \in W$, so v was supposed to minimize the distance to m. So we have a contradiction, which proves that the premise $v_a \leq v_b$ must be false. So we have that $v_a > v_b$. End of proof.

B   Acknowledgments

C   References

[1] Fuad Aleskerov and Eldeniz Kurbanox. Degree of manipulability of social choice procedures. In Ahmet Alkan, Charalambos D. Aliprantis, and Nicholas C. Yannelis and, editors, *Current Trends in Economics: Theory and Applications*, pages 13–28. Springer, March 1999.

[2] F. K. Bedall and H. Zimmermann. Algorithm as 143: The mediancentre. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(3):325–328, 1979.

[3] Ömer Egecioglu. Uniform generation of anonymous and neutral preference profiles for social choice rules. Technical report, University of California, Santa Barbara, California, USA.

[4] Steven Richard Hollasch. Four-space visualization of 4d objects. Master's thesis, Arizona State University, August 1991.

[5] Andy Mackenzie. The mediancentre-borda rule and one-way monotonicity. November 2008.

[6] Herv Moulin. *Axioms of Cooperative Decision Making*. Cambridge University Press, Cambridge, 1988.

[7] Todd Rowland. Orthogonal matrix. In Eric W. Weisstein, editor, *MathWorld*.

[8] M. Remzi Sanver and William S. Zwicker. Monotonicity properties and their adaptation to irresolute social choice rules. 2009.

[9] Alan Taylor. *Social Choice and the Mathematics of Manipulation*. Cambridge University Press, Cambridge, 2005.

[10] Eric W. Weisstein. Rotation matrix. In Eric W. Weisstein, editor, *MathWorld*.

[11] William S. Zwicker. Consistency without neutrality in voting rules: When is a vote an average? *Mathematical and Computer Modelling*, 48(9-10):1357–1373, 2008. Mathematical Modeling of Voting Systems and Elections: Theory and Applications.