SuperHare: an algorithm for proportional representation

Fred Richman
New Mexico State University

Abstract. The traditional Hare proportional system has a number of flaws when used with a small electorate. For certain values of the parameters the algorithm doesn't even work. In addition, the problem of ties becomes acute, so the role of chance in determining the outcome is increased. We present here a system that deals with these flaws while still staying very much in the spirit of the original system. The main innovations are floating quotas, fractional quotas, and breaking ties by complete enumeration of possible scenarios.

The Hare proportional system addresses the problem of electing a representative body. The motivating idea is that any substantial faction of the voters should be represented proportionally among the candidates elected. The mechanism used is the **single transferable vote**: each voter gets a single vote-for one candidate-but it is transferable to another candidate if the voter's (first) choice is either eliminated from contention, or already has sufficiently many votes to be elected.

In this paper we examine various ways to fill in the details of this general scheme. In particular, we note that the traditional Hare system cannot be applied in certain cases where the representative body is not a very small fraction of the electorate, a situation that arises when a smallish group chooses a committee from itself. We suggest an algorithm , called **SuperHare**, that (a) always applies, (b) minimizes the element of chance, and (c) easily accommodates incomplete ballots.

SuperHare requires quite a bit of calculation if done by hand, but only a modest amount from the point of view of a computer. Not only is there the tedium of dealing with fractional votes, there is the complexity introduced by the fact that SuperHare breaks ties by exploring the consequences of every possible choice. SuperHare was devised with computer implementation in mind, and various forms of it have been running on a personal computer for several years.

1 Single transferable vote systems

Thomas Wright Hill is credited with working out the single transferable vote system, in 1821, in conjunction with the election of a committee of boys at his son's school (see Lakeman, 1982, page 46). In these elections, the candidates would stand in the front of the room, and the voters would go stand by the candidate they favored. There are three main issues to resolve in a single transferable vote system:

- What is the quota? (the number of votes it takes to elect)
- What do we do with excess votes? (above the quota)
- How do we eliminate losing candidates? (if no one achieves the quota)

Suppose there are six candidates and $N = 15$ voters for $k = 3$ slots. If the votes come in 5 5 5 0 0 0, the solution is clear. What if the votes come in 6 3 3 1 1 1? The quota Q must be 4 or 5; indeed the quota must satisfy

$$\frac{N}{k+1} < Q \leq \frac{N}{k}$$

to preclude electing k+1 candidates, on the one hand, and to allow electing k candidates on the other hand. So the first candidate has exceeded his quota. In practice, some of the boys would move from the first candidate

to one of the others so as not to waste their votes. The idea of not wasting votes plays a major role in arguments in favor of single transferable vote systems.

Suppose the quota *Q* is 4, and after the 2 excess votes for the first candidate are transferred, the situation is * 3 3 2 2 1. Now the boy voting for number six may see that his candidate has no chance, whereas he can influence the election by voting for some else; so he moves to his second choice. This migration continues until two more candidates achieve the quota, barring a student who unsportingly sticks with a clear loser, or a problematic choice of who is to drop out between two candidates with the same smallest number of votes.

It was this system, in essence, that became well known with the publication of Thomas Hare's *Treatise on the Election of Representatives, Parliamentary and Municipal* in 1859. To effect a more orderly process, each voter rank orders the list of candidates, so that his preference between any two candidates will be known in case he acquires the right to choose between candidates other than his favorite. If a candidate gets at least *Q* first-place votes, he is elected, *Q* of his ballots are removed from the pool, and his name is struck from each of the ballots left in the pool. If no candidate gets *Q* first-place votes, then some candidate is eliminated, and his name is struck from each of the ballots in the pool. This process is then repeated until *k* candidates are elected. Thus each ballot is used to elect exactly one candidate.

2 The Droop quota

Traditionally the quota *Q* is chosen to be the least integer greater than $N/(k+1)$, the smallest possible value consistent with filling at most *k* positions. This is called the **Droop quota**. There are a number of reasons why it is preferred. For one thing, any candidate who achieves the Droop quota must eventually be elected, because no other *k* candidates can get as many votes (unless he is arbitrarily eliminated in favor of a candidate who has fewer votes). We get a tidier process if we elect him immediately. For example, if $k = 1$, this simply says that we elect anyone who gets a majority of the votes, without going through the formalities of eliminating the other candidates first. Also, because anyone who gets the Droop quota must be elected, any votes in excess of the Droop quota may be construed as wasted, and nobody wants to cast a wasted vote.

Another reason for choosing the Droop quota is that it guarantees that a majority party will elect a majority of the candidates. An example, taken from Lakeman (1982) shows how that can fail. Suppose $N = 100$ and $k = 5$, and we take the quota to be 20. There are three parties running, so the top choices on each ballot are taken, in order, from one of three disjoint lists of candidates. The votes come in 52, 35, 13. The first party gets its top 2 choices, and the second party its top choice. The ballots remaining are divided 12, 15, 13, so the only possible result is that the majority party will elect no more candidates, and the other two parties will elect one each. Had we used the Droop quota of 17, the majority party would elect 3, and the second party 2.

The traditional Hare system can work only if the Droop quota does not exceed N/k, that is, if there is an integer *Q* such that

$$\frac{N}{k+1} < Q \leq \frac{N}{k}.$$

Note that, for example, there is no integer *Q* such that $8/4 < Q \leq 8/3$. This condition holds if and only if *k* is a quotient of *N* in the sense that *N* can be written as *k* times some integer *Q*, plus an integer smaller than *Q*. It holds for most practical values of *k* and *N*; in particular it holds whenever $k^2 \leq N$. It also holds whenever *N* is a multiple of *k*. On the other hand, it fails if $k^2 = N+1$, for example, if $k = 3$ and $N = 8$, or if $k = 10$ and $N = 99$. Thus we cannot have a universally applicable system with a constant quota *Q*.

3 The local Droop quota

One way to overcome the quota problem is to recompute the Droop quota *Q* after each batch of *w* winners is found, to take into consideration that *k* has decreased by *w*, and *N* has decreased by *wQ*. We call this recomputed quota the **local Droop quota**. Thus we start afresh after each batch of winners is found, approaching each decision in the same fashion, which allows a recursive description of the algorithm. In this

way we adhere to the principle established in the traditional Hare system that the quota be the smallest number consistent with filling at most *k* positions.

As an example, suppose we have 3 slots and 20 votes, so the quota *Q* is 6. After a winner is found we are down to 2 slots and 14 votes. The Droop quota is still 6; the local Droop quota is now 5. It is easy to see that a candidate with 5 votes in this scenario will eventually be declared a winner, even under the traditional Hare system, because no candidate who achieves a *local* Droop quota can be eliminated.

In a traditional Hare, where *Q* is the Droop quota, the local Droop quota can almost halve. If $N = k^2$, then the local Droop quota starts out as *k* and ends up as the greatest integer not exceeding (*k*+3)/2. If we take *Q* to be the local Droop quota, then the local Droop quota can decrease at most by one. This happens only when *N* is divisible by *k*+1, hence *N* = (*k*+1)(*Q*-1). It is easy to verify that if this happens, then the local Droop quota goes from *Q* to *Q*-1 and stays there from then on (under the assumption that only one winner is selected at a time: see below).

Because *Q* may greatly exceed the local Droop quota in a traditional Hare process, we can witness the same phenomenon there that partially motivated the choice of the Droop quota in the first place: we may be required to go through the motions of eliminating candidates before we can declare a clear winner to be a winner.

My original motivation for using the local Droop quota was a programmer's preference for a clean recursive procedure. I was hesitant to adopt it because it differed from the traditional Hare system. However, when I discovered that the traditional system didn't even work in all cases, I decided that the modified system was better.

Even if we use the local Droop quota, there is trouble if we declare winners in batches. Suppose *k* = 5 and *N* = 12, so *Q* = 3. If the votes come in 3 3 3 3 0 0, and we elect the first four candidates, then we are up the creek. We can get around this by electing the candidates one by one. First, for example, we elect candidate one; then the votes are * 3 3 3 0 0, and the local Droop quota stays at 3. Then we elect candidate two; the votes are * * 3 3 0 0, and the local Droop quota is still 3. Electing candidate three, the votes are * * * 3 0 0, and the local Droop quota is 2, so when we elect candidate four, we will still have a ballot left to elect a fifth winner. This seems like an awkward fix, however, and it involves choosing which of the winning candidates is elected first. We will see how to handle such choices when we consider how to eliminate losing candidates.

4 Irish Senate rules

We turn to the question of excess votes. If a candidate gets *x* – *Q* first-place votes, we must remove *Q* of his ballots. Which *Q* of them do we remove? In the usual Hare system this is either done at random or, somewhat similarly, by removing the *first Q* ballots that list the candidate first. A more equitable, deterministic way is to leave all *x* ballots in the pool, but weight them by the factor (*x*-*Q*)/*x*; thus if a candidate receives 7 first-place votes when *Q* = 6, then 6/7 of each of those 7 ballots is considered to have been used up electing the candidate, and 1/7 of each ballot is returned to the pool. This method is used in electing the Irish Senate (see Bogdanor, 1984, page 86). To avoid fractions we return all of the ballots to the pool and weight each of the unused ballots by a factor of 7. The general procedure is to let *d* be the least common denominator of the fractional weights (*x*-*Q*)/*x* in lowest terms, weight each winning ballot by *d*(*x*-*Q*)/*x*, and weight all the other ballots by *d*. The new number of ballots is the sum of these weights. We then recalculate *Q*.

For medium-sized problems, the repeated taking of greatest common denominators can easily result in integers that are so large as to exceed the precision of the computer you are working on. Choosing one winner at a time doesn't avoid the problem either. In practice we must occasionally rescale everything by, say, dividing each of the weights by 1000 and rounding the results to the nearest integer. That's a bit inelegant; if we are going to do that, why not use fractional ballots in the first place, held to the precision of the computer, that is, as real numbers? The problem there is that we don't have a natural choice for the Droop quota. We will return to this point later.

5 Eliminating losers

If there are no winners, then there are at least k+1 candidates with first-place votes. The names of the candidates who have received no first-place votes are then removed from the ballots. Then the name of the candidate who received the least number of first-place votes is also removed from the ballots. If $t > 1$ candidates receive the least number of first-place votes, then we eliminate each of these candidates with probability 1/t. By this I mean that we will see what happens when we eliminate each of the t candidates, and average the results. The same technique can be used to break ties among winners to determine who is declared winner first.

While this procedure eliminates the necessity of choice at this stage, we might not end up with a clear set of winners. Instead we might get a list of possible slates of k winners, with each slate assigned a probability. The recommended procedure now is to choose that slate with the highest probability, if there is a unique such. If there is a tie, we will just have to break it somehow; the choice problem cannot be completely avoided, but we have kept it to a minimum.

As an example, consider the extreme case of tying where $N = 4$ and $k = 2$ and the four ballots rank candidates *A*, *B*, *C* and *D* as follows: *ABCD*, *BCDA*, *CDAB*, and *DABC*. The result is an assignment of a nonzero probability to each of the six possible slates.

AB	*AC*	*AD*	*BC*	*BD*	*CD*
1/8	1/4	1/8	1/8	1/4	1/8

Why do slates *AC* and *BD* do better than the others? Well, *A* is the choice of voter number 1, so if *A* is chosen then it is less likely that *B*, the second choice of voter number 1, will be chosen (because the ballot of voter number 1 will be used to elect *A*). Similarly if *D*, the choice of voter number 4, is chosen, it is less likely that *A* will be chosen. Hence *AC* is more likely than either *AB* or *AD*. Another way of thinking about it is that slates *AC* and *BD* give everybody either his first or second choice; the other slates don't.

6 Fractional quotas

If we allow fractional weights on ballots, then a candidate need not receive a whole number of votes, so the reason for choosing Q to be an integer disappears. But there is no smallest fraction that exceeds $N/(k+1)$, so what is the Droop quota in this case? I believe that the answer is this: to be elected, you must have *strictly more* than $N/(k+1)$ votes (so that no more than k candidates are elected), but *all* votes over $N/(k+1)$ are considered as excess (even though you need some in order to be elected).

Consider our original example where $N = 15$ and $k = 3$, and the votes come in 6 3 3 1 1 1. The quota is 15/4 (instead of 4), so we elect the first candidate and have 9/4 excess votes (instead of 2). We now cut the weight of each of the six ballots that put candidate number one in first position from 1 to (9/4)/6 = 3/8, as we have used 5/8 of each of those ballots to enable the first candidate to reach the quota of 15/4. The new N is 45/4 and the new k is 2, so the new local Droop quota is (45/4)/3 = 15/4, the same as the old one! It is easy to see that this always happens; there is now no difference between the Droop quota and the local Droop quota.

Note that under this scheme it doesn't matter whether we elect the winners in batches or not. The troublesome example where $k = 5$ and $N = 12$, and the votes came in 3 3 3 3 0 0, is no problem because $N/(k+1)$ is 2, so there are four excess votes.

7 Super Hare

So what are the recommended modifications to the Hare proportional system? First of all, we calculate with fractional votes, at least after the first winner is declared, although there is no reason to disallow them from the start (and there are situations where we want to use them: see "missing data" below).

For a quota we take the local Droop quota, which, in this fractional setting, is the fraction $N/(k+1)$ itself, rather than the smallest integer greater than $N/(k+1)$. A candidate must *exceed* the quota in order to be elected, but all (fractional) votes over that quota are treated as excess.

Ties (for last) among the candidates are broken by eliminating each candidate in the tie in turn, and averaging the results. This may result in more than one slate of winners, each slate having a probability assigned to it. The slate with the largest probability is normally chosen; ties for first among the slates, if any, being broken by some random device.

To sum up, the input to SuperHare is a positive integer k and a list of linear orderings, called **ballots**, possibly with repetitions, of a set C containing at least k objects, called **candidates**. Each ballot is weighted by a nonnegative real number (presumably equal to 1 initially), and we let N be the sum of those weights. The output from SuperHare is a probability distribution on the set of k-element subsets of C. The algorithm works as follows.

For each c in C, let $v(c)$ be the total weight of the ballots in which c appears in the first position. Let W be the subset of C consisting of those candidates c such that $v(c) > N/(k+1)$, and let $\#W$ be the number of candidates in W. If $\#W = k$, then W is assigned probability 1 and we are done. If $0 < \#W < k$, multiply the weight of each ballot whose first element c is in W by

$$\frac{v(c) - N/(k+1)}{v(c)}.$$

Remove the candidates in W from C and from the ballots. Reduce k by $\#W$, and apply SuperHare (recursively) to the resulting data. The probability of a subset S of the original set C is zero unless S contains W, in which case it is the probability assigned to the subset $S\backslash W$ of $C\backslash W$ by the recursive call to SuperHare.

If $\#W = 0$, then all candidates that do not appear first in some ballot of nonzero weight are removed from C and from the ballots. Then we consider the set L of candidates c such that $v(c)$ is minimum. For each c in L we remove c from C and from the ballots, and call SuperHare; the probabilities on each of the returned slates are multiplied by $1/\#L$, and duplicate slates are combined.

8 Missing data

We have seen that, in theory, the local Droop quota does not change when we use SuperHare. More often than not, however, there will be some incomplete ballots from voters who rank their top two or three choices and then quit. These ballots must be removed when all candidates appearing on them have either been declared winners or have been eliminated, and this may happen when the ballots still have nonzero weight. The effect of this is that the total weight of the ballots decreases more rapidly than the theory predicts, so the local Droop quota can go down. This phenomenon increases the chances that a traditional Hare system will crash, unless we ignore ballots that are not completely filled out. In SuperHare we simply use the local Droop quota, that is, we recompute $N/(k+1)$ after each winner is chosen.

Another common form of missing data is for two candidates to be rated equally by a voter, despite instructions to the contrary. A tolerant judge can handle this by creating two ballots, each with weight 1/2, corresponding to the two different ways of ordering the indistinguishable candidates.

References

[]
Bogdanor, Vernon. 1984. *What is proportional representation*, Oxford: Martin Robertson & Co.

[]
Lakeman, Enid. 1982. *Power to elect*, London: William Heinemann Ltd.

File translated from TEX by TTH, version 2.27.
On 2 Dec 2001, 08:15.