Stochastic Simulation of Sequential Game-Theory Voting Models

by Jeffrey S. Rosenthal, University of Toronto, July 2016, revised May 2017

Abstract. We discuss the use of stochastic simulation as a tool to learn about optimal behaviour and Nash equilibria of a sequential voting model proposed by Osborne (1996), related to Duverger's Law. We introduce a graphical Java applet which implements such simulations, and investigate its properties. We show that in an appropriate setup, the applet is guaranteed to eventually find behaviour which is within ϵ of being optimal.

Keywords: Voting model; Game theory; Nash equilibrium; Multiple parties; Stochastic simulation.

Mathematics 2010 Subject Classification:
Primary 91A06; Secondary 62M05, 62F10, 91F10.

1 Introduction

Game theory is often used to model strategies of political parties in elections (see e.g. Osborne, 2003, and the references therein). A common setup is that each of n different political parties chooses a political position $x_i \in \mathbf{R}$ (or chooses not to contest the election at all, by setting $x_i = \text{OUT}$). Voters are assumed to be distributed according to, say, the Uniform[0,1] distribution. Each voter will vote for whichever party's position is closest to their own. The game-theoretic payoff score for each political party is 0 if they do not contest the election, or 1 if they receive the most votes, or $1/k$ if they tie for most votes with a total of k different parties, or -1 if they contest the election and receive fewer votes than some other party. We focus here on the model in which the parties choose their positions *sequentially*, i.e. each party i is allowed to base their own position x_i on the already-chosen positions $x_1, x_2, \ldots, x_{i-1}$ (Osborne, 1996).

In any such game-theory model, the question is what behaviour will tend to be adopted by each of the n parties. In particular, a collection of actions is a *Nash equilibrium* (Nash, 1951) if each party's payoff is marginally optimised, i.e. if no one party can increase their expected payoff by changing their action, assuming that all other parties' action strategies remain the same. In a sequential model, this can be thought of more directly: Party n chooses whatever action maximises their payoff, given the actions of Parties $n-1, n-2, \ldots, 1$. And, Party $n-1$ chooses whatever action maximises their payoff, given the actions of

Parties $n-2, n-3, \ldots, 1$, and also given that Party n will then subsequently choose their own optimal action as above. And Party $n-2$ chooses whatever action maximises their payoff, given the actions of Parties $n-3, \ldots, 1$, and also given that Party $n-1$ and n will then subsequently choose their own optimal actions as above. And so on. In particular, Party 1 must choose an action which will maximise their payoff, given that Party 2 will subsequently choose an action to maximise their payoff, assuming Party 3 will afterwards choose an action to maximise their payoff, etc.

For our model, if there are just $n = 2$ parties, then it is clear that both parties will choose to come in at the median value $1/2$, so that they each receive half the votes, and thus each receive a payoff of $1/2$. In that case, if either party deviated to a different action, then their payoff would change to either 0 (if they chose OUT), or -1 (if they chose some other position and therefore received less than half the vote). So, neither party can improve their position by deviating. Hence, the solution in which both parties choose $x_1 = x_2 = 1/2$ is a Nash equilibrium for this model.

However, if n is larger, then the situation becomes exponentially more complicated, since each possible action of Party i requires consideration of all possible further actions of Parties $i+1, i+2, \ldots, n$. If $n = 3$, then it is not too hard to see that Party 1 should again come in at the median voter position $1/2$; that way Party 2 cannot win (assuming best play from Party 3) and hence will stay out, after which Party 3 will also come in at $1/2$ so that Parties 1 and 3 each receive payoff of $1/2$ (and Party 2 receives payoff of 0). It has been conjectured (Osborne, 1996) that this pattern continues for larger n, i.e. that in equilibrium, only Parties 1 and n will contest the election, each with position equal to the median $1/2$, while the other $n-2$ parties will all stay out. This outcome is consistent with *Duverger's Law*, which states that democracies will tend towards having just two major parties contest elections (Duverger, 1951; Riker, 1982; Schlesinger and Schlesinger, 2006). The conjecture has been proven for $n \leq 4$ (Osborne, 1996), but it might not be true in general (de Vries, 2015; de Vries et al., 2016), and it is very challenging to approach this problem analytically for large n.

In this paper, we consider the possibility of running stochastic simulations of this sequential voting model, to attempt to verify the equilibrium behaviour directly without the need for complicated high-dimensional analysis. We shall describe an interactive graphical Java

applet created for this purpose (Rosenthal, 2015). We shall present some modifications of the algorithm and of the underlying game theory model, and shall study their properties through simulation results and some theoretical analysis. We shall see that although this simulation approach does not solve the problem completely, it does allow us to learn and verify various election behaviours, including some which are not at all obvious analytically.

2 Formal Set-Up and Assumptions

We assume that voters are distributed according to the Uniform[0,1] probability density on $\mathbf{R}$. In sequence, each of the parties $i = 1, 2, \ldots, n$ chooses a position $x_i \in \mathbf{R} \cup \{\text{OUT}\}$. If $x_i = \text{OUT}$, then party i does not contest the election and therefore receives a payoff of zero. Otherwise, each party i with $x_i \in \mathbf{R}$ receives a vote share w_i given by

$$w_i \;=\; \frac{|R_i|}{\#\{j : x_j = x_i\}}\,, \tag{1}$$

where $R_i = \{t \in [0,1] : |t - x_i| \leq |t - x_j| \text{ for all } 1 \leq j \leq n\}$ is the vote region won (or tied for winning) by Party i, of length $|R_i|$. Each Party i with $x_i \in \mathbf{R}$ receives payoff 1 if they have the highest vote share w_i, or $1/k$ if they tie with a total of k parties for the highest vote share, or -1 if their vote share is strictly less than that of some other party.

In this context, the conjecture of Osborne (1996) is that for any $n \geq 2$, in equilibrium (i.e., where no one party can increase their expected payoff by changing their action, conditional on all other parties continuing to behave optimally), we will have $x_1 = x_n = 1/2$, while $x_2 = \ldots = x_{n-1} = \text{OUT}$.

3 The Java Applet ("Vanilla" Version)

To study the stochastic simulation of the above voter model, we have developed a Java applet, freely available online (Rosenthal, 2015), which we now describe.

Our Java applet simulates the positions $x_1, \ldots, x_n \in [0,1]$ of each of n different parties. The positions are represented graphically as dots between 0 and 1, with the parties ordered from 1 (bottom) through n (top). The dots are green circles if the party is contesting the election (i.e. $x_i \in \mathbf{R}$), or red squares if the party stays out (i.e. $x_i = \text{OUT}$; here the dot's

location is not strictly relevant but still indicates where the applet was "considering" coming in). For each given set of positions, each party's win region is computed (and indicated with a horizontal line). Then each party's vote share and payoff are computed and displayed.

Iteratively, some Party j has their position tweaked some number M different times (e.g. $M = 50$) by the applet in some way (e.g. normally distributed around the previous position, or chosen freshly and randomly from the Uniform[0,1] density, or moving to or from OUT; specific choices are discussed below). Given this tweak, Party $j + 1$ attempts M different tweaks. And then, for each of those tweaks, Party $j + 2$ attempts their own M tweaks. And so on up to Party n.

Each of these tweaks is "accepted" if it gives the tweaked party a larger payoff (after taking into account the actions of all of the subsequent parties, each of whom also only accepts tweaks which increase their own payoff). Otherwise, the tweak is rejected, and the party's position is returned to its previous value.

Schematically, our algorithm may be described by the following recursive pseudo-code (see also the Java source code available from Rosenthal, 2015):

Algorithm 1 OPTIMISE(j; $x_1, \ldots, x_n$): Optimise the Actions of Parties $j, j+1, \ldots, n$

$S \leftarrow$ payoff of Party j in configuration $(x_1, x_2, \ldots, x_n)$.
for $\ell = 1, 2, \ldots, M$ **do**
  $x'_j \leftarrow$ some new randomised "tweaked" value (which could be OUT)
  if $j < n$ **then**
    // recursively optimise Parties $j+1, j+2, \ldots, n$
    $(x'_{j+1}, \ldots, x'_n) \leftarrow$ OPTIMISE($j+1$; $x_1, \ldots, x_{j-1}, x'_j, x_{j+1}, \ldots, x_n$)
  end if
  $S' \leftarrow$ payoff of Party j in configuration $(x_1, \ldots, x_{j-1}, x'_j, \ldots, x'_n)$
  if $S' > S$ **then**
    for $i = j, j+1, \ldots, n$ **do**
      $x_i \leftarrow x'_i$
    end for
    $S \leftarrow S'$
  end if
end for
return $(x_j, x_{j+1}, \ldots, x_n)$

In this way, the applet simulates each of the parties in turn attempting to maximise their own payoff, given that all subsequent parties will also attempt to maximise. Indeed, as

M gets large, the applet will do a better and better job of simulating each party's optimal action (given all of the previous parties' actions). However, this "vanilla" algorithm does have some limitations, as we discuss next.

4 The "Median-Finding" Limitation

As described above, a key component of equilibrium behaviour is that parties sometimes choose to come in right at the median voter position of 1/2.

Indeed, this situation arises even with just $n = 2$ parties. In that case, if Party 1 comes in at 1/2, then Party 2 can do no better than also come in at 1/2 so both parties tie for the win and each receive payoff of 1/2. But if Party 1 comes in at any other position (e.g. $x_1 = 0.501$), then Party 2 can always respond by finding some clever subsequent position which is closer to the median than x_1 (e.g. $x_2 = 0.4995$, or $x_2 = 0.5005$). In that way, Party 2 will win a larger vote share, and thus a payoff of $+1$, while Party 1 will lose and thus receive a payoff of -1.

Now, if the Java applet only allows parties to choose their tweaked positions from continuous distributions, like Normal or Uniform as discussed above, then parties will never be able to find a precise position like 1/2. So, if a precise position like 1/2 is crucial to some equilibrium behaviour (as above), then the Java applet will fail to find that behaviour and will thus fail to accurately imitate the true continuous game (see Figure 1).

5 Incorporating medianProb and mimicProb

To deal with the above problem, we added a new parameter "medianProb" to our Java applet. This gives each tweaked position a small probability (e.g. 0.1) of being taken to be equal to the median 1/2, regardless of the other parties' positions or any previous history. This allows the simulation to "find" optimal behaviour even when it requires a party to come in at precisely the median as above.

This slight modification does indeed allow our simulation to find such optimal behaviour. For example, when $n = 2$, it quickly and easily finds the equilibrium in which $x_1 = x_2 = 1/2$ (see Figure 2).

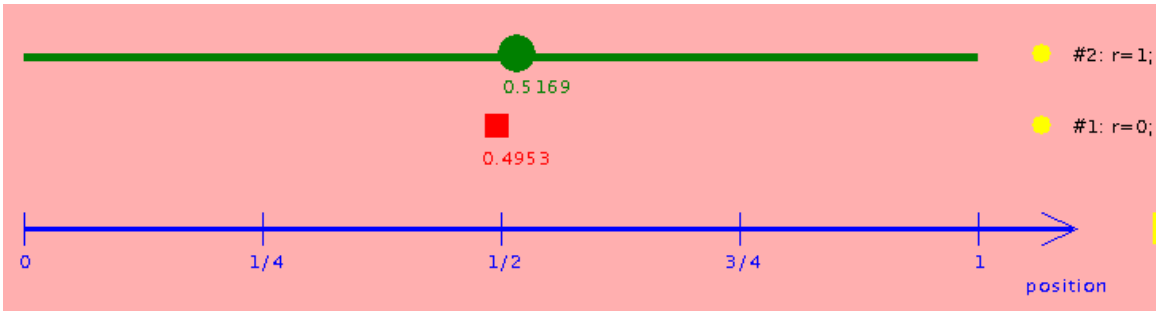

Figure 1: Output from the "vanilla" Voting Model Java Applet, with $n = 2$ parties, in which Party 1 comes in "near" $1/2$ but not exactly at $1/2$, thus allowing Party 2 to come in even nearer to $1/2$ and thus win outright, giving Party 1 a payoff of -1. (Here and throughout, parties are numbered from the bottom up, with a dot indicating each party's chosen position, which is a green circle if they are contesting the election, or a red square if they stay out, and with horizontal lines showing their vote win regions.)

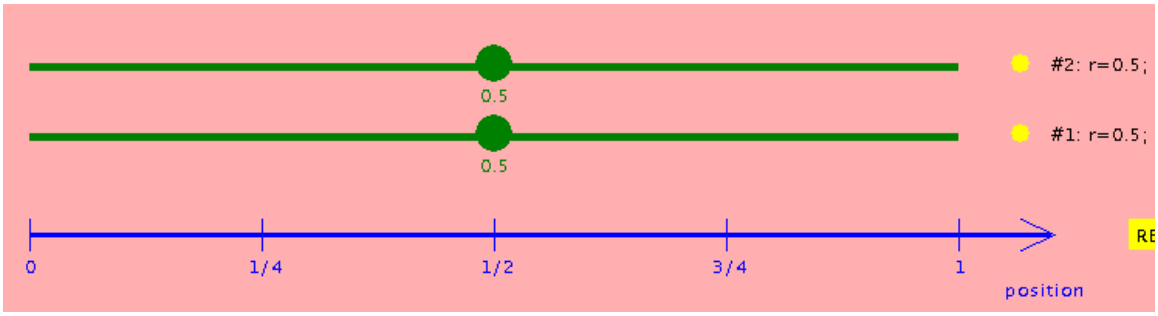

Figure 2: Output from the Voting Model Java Applet with $n = 2$ parties, and with a positive value of medianProb. This allows Party 1 to come in at precisely $1/2$, so that Party 2 can do no better than to also come in at $1/2$, giving each party a payoff of $1/2$.

Similarly, since some optimal actions might involve choosing exactly the same position as some other party, we also introduce a positive parameter mimicProb. This gives each tweaked position a small probability (e.g. 0.1) of being taken to be equal to the current position of one of the other parties. This allows the applet to consider the possibility that one party will need to choose precisely the same value as a previous party, even if that value is different from the median $1/2$.

6 Simulation Results for the Original Model

Armed with this new feature, our applet is able to investigate new results. For example, as mentioned, Osborne's conjecture was previously proven only for $n \leq 4$. However, our applet is able to simulate the full model with $n = 5$ and quickly find the conjectured equilibrium, in which the first and last parties come in at 1/2, and the remaining parties stay out, i.e. where $x_1 = x_5 = 1/2$ and $x_2 = x_3 = x_4 = \text{OUT}$ (Figure 3).

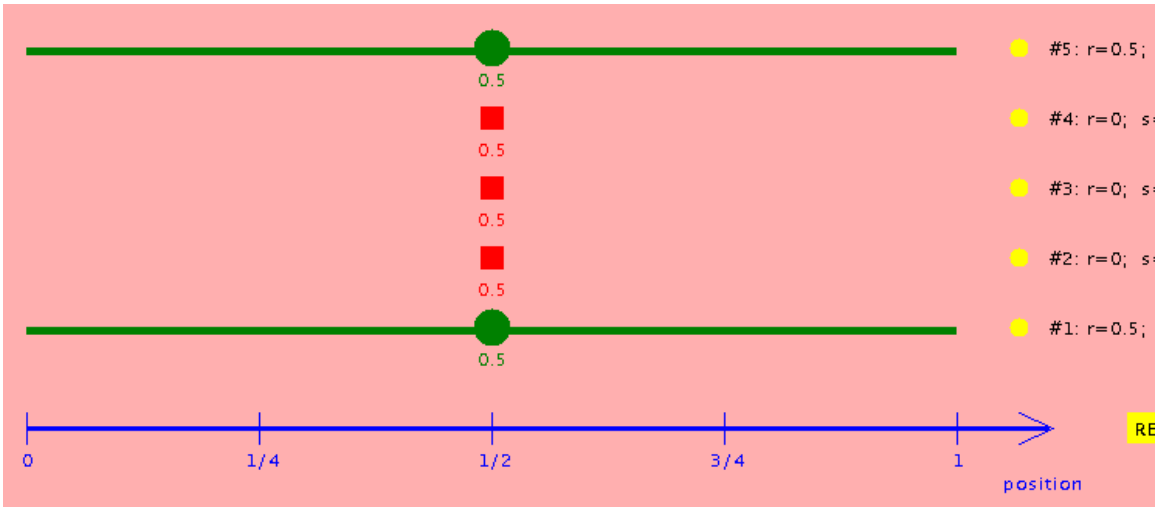

Figure 3: Output from the Voting Model Java Applet with $n = 5$ parties, and with a positive value of medianProb. The result is that $x_1 = x_5 = 1/2$ while $x_2 = x_3 = x_4 = \text{OUT}$, supporting Osborne's conjecture with $n = 5$ parties.

On the other hand, to optimise the action of Party i requires M^{n-i+1} simulation attempts. So, as the number of parties n gets large, the run time of the algorithm becomes exponentially long. For example, if $M = 50$ and $n = 7$, then the number of tweaks is $50^7 \doteq 7.8 \times 10^{11}$ which pushes the limits of what computers can compute in a reasonable time. However, by reducing the number of iterations M from 50 to 25, and letting the computer run for several hours, we eventually verified the Osborne conjecture for $n = 6$ (see Figure 4). Then, letting the applet run for several days, we eventually verified the Osborne conjecture for $n = 7$ as well (see Figure 5).

Finally, we note that the simulations sometimes illustrated surprising behaviour. For example, suppose we have $n = 7$ parties, and we start with positions $x_1 = 0.5$ and $x_2 = 0.1$ for the first two parties, and then optimise over the remaining five parties. Then we sometimes

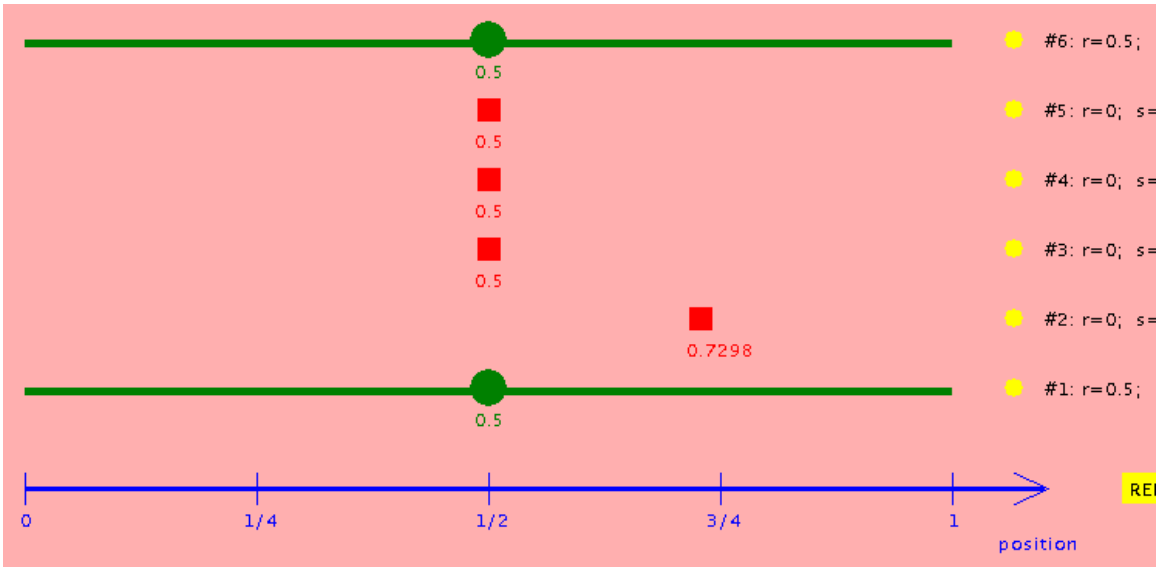

Figure 4: Output from the Voting Model Java Applet with $n = 6$ parties, and with a positive value of medianProb. The result is that $x_1 = x_6 = 1/2$ while $x_2 = x_3 = x_4 = x_5 = \text{OUT}$, supporting Osborne's conjecture even with $n = 6$ parties.

found that Party 3 would come in at some value $x_3 \in (0.7, 0.8)$, after which all remaining parties would stay out. This would allow Party 3 to win the election outright and receive a payoff of $+1$ (see Figure 6). At first we thought this was an artifact of the simulation, but we later determined that it is in fact optimal behaviour for Party 3 in this situation. (We later realised that similar situations are also considered in Appendix A1 of de Vries, 2015.)

7 The "Isolated-Point" Limitation

The addition of medianProb and mimicProb above go a long way towards helping the applet "find" all relevant actions. However, they do not completely solve the problem, as we now illustrate.

Inspired by the calculations of de Vries (2015), consider the following situation. Suppose there are $n = 7$ parties, and the first five parties have already chosen their positions as $x_1 = 0.5$, $x_2 = 0.1$, $x_3 = 0.9$, $x_4 = 0.4$, and $x_5 = 0.6$, respectively. Then what actions should parties 6 and 7 take?

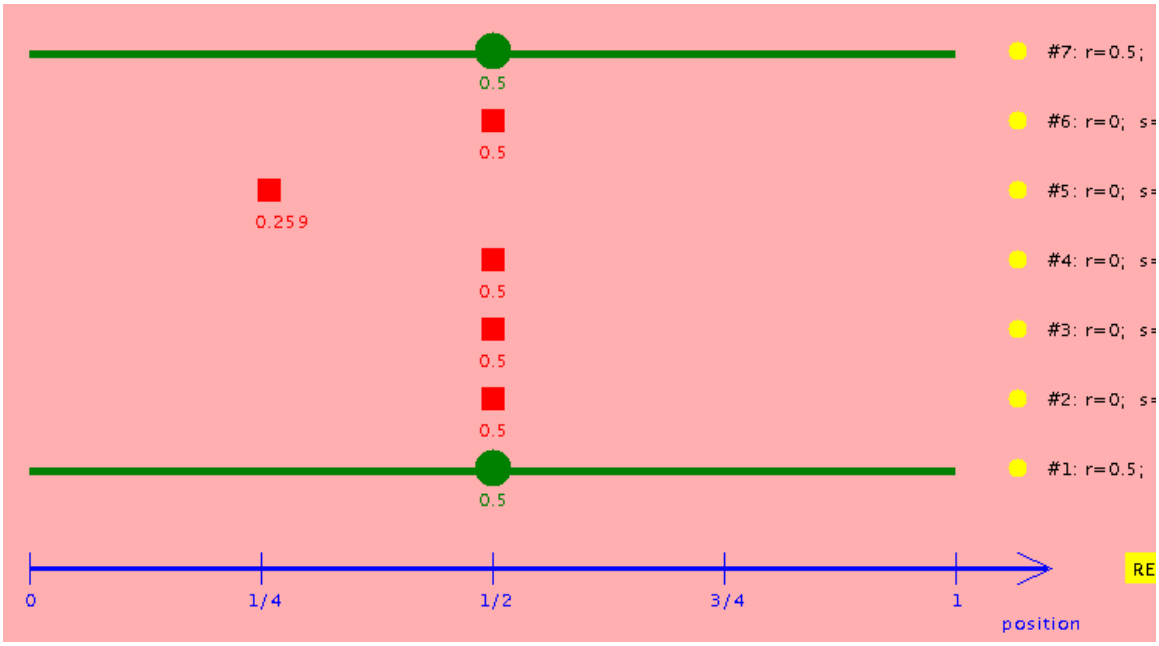

Figure 5: Output from the Voting Model Java Applet with $n = 7$ parties, and with a positive value of medianProb. The result is that $x_1 = x_7 = 1/2$ while $x_2 = x_3 = x_4 = x_5 = x_6 =$ OUT, supporting Osborne's conjecture even with $n = 7$ parties.

Well, it is not hard to see that if Party 6 comes in at $x_6 = 0.2$, and Party 7 comes in at $x_7 = 0.8$ (or vice-versa), then each of the Parties 2,3,4,5,6,7 will each receive a vote share of 0.15, while Party 1 will receive a vote share of 0.1. Thus, Parties 2,3,4,5,6,7 will tie for the most votes and each receive payoff of 1/6, while Party 1 will lose and receive payoff of -1. And given the actions of the first five parties, the choices $x_6 = 0.2$ and $x_7 = 0.8$ (or vice-versa) are indeed optimal for Parties 6 and 7.

However, our applet as designed cannot "find" this solution! Indeed, if tweaks of x_6 are generated variously from normal distributions, uniform distributions, medianProb, and mimicProb, then they will include values which are *close* to 0.2 and 0.8, but they will never include *exactly* 0.2 or 0.8. And, if x_6 differs from 0.2 or 0.8 by even the smallest amount, then Party 6 will receive a (slightly) lower vote share than at least one of the other parties, so Party 6 will receive payoff of -1 which is much worse than 1/6. As a result, when running our applet for this situation, Parties 6 and 7 instead stay OUT and receive payoff of zero (Figure 7).

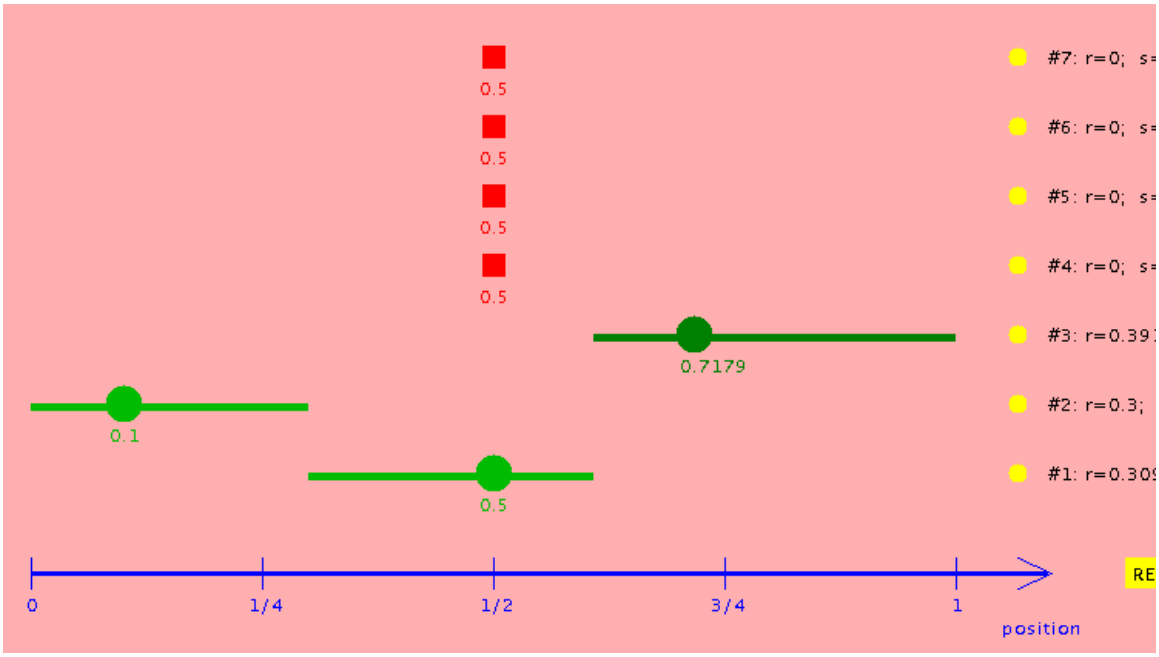

Figure 6: Output from the Voting Model Java Applet (Original Model), with $n = 7$ parties. In this case, starting with $x_1 = 0.5$ and $x_2 = 0.1$ and optimising over Parties 3 through 7, the applet finds a surprising solution in which Party 3 comes in between 0.7 and 0.8 (in this case, at $x_3 = 0.7179$), thus forcing all the remaining parties to choose OUT so that Party 3 wins outright.

The reason for this limitation is that the payoff values are highly *discontinuous* functions of the positions x_i. For example, in the above situation, if $x_6 = 0.2$ or $x_6 = 0.8$ then Party 6 will receive a payoff of $1/6$, but if x_6 equals any other value then Party 6 will receive a payoff of -1. Motivated by this, we next consider a related model in which the payoff values do not have this discontinuity.

8 A Related Continuous Model

Motivated by the above, we wish to modify our original model to make the payoffs have fewer discontinuities. We do this in two stages. In each case, we leave the computation of the vote shares w_i exactly the same as in (1). And the payoff is always zero for parties who stay OUT. All that is modified is the payoff rule for parties who contest the election.

For clarity, we let "Original Model" (OM) refer to our previous model above, with payoff

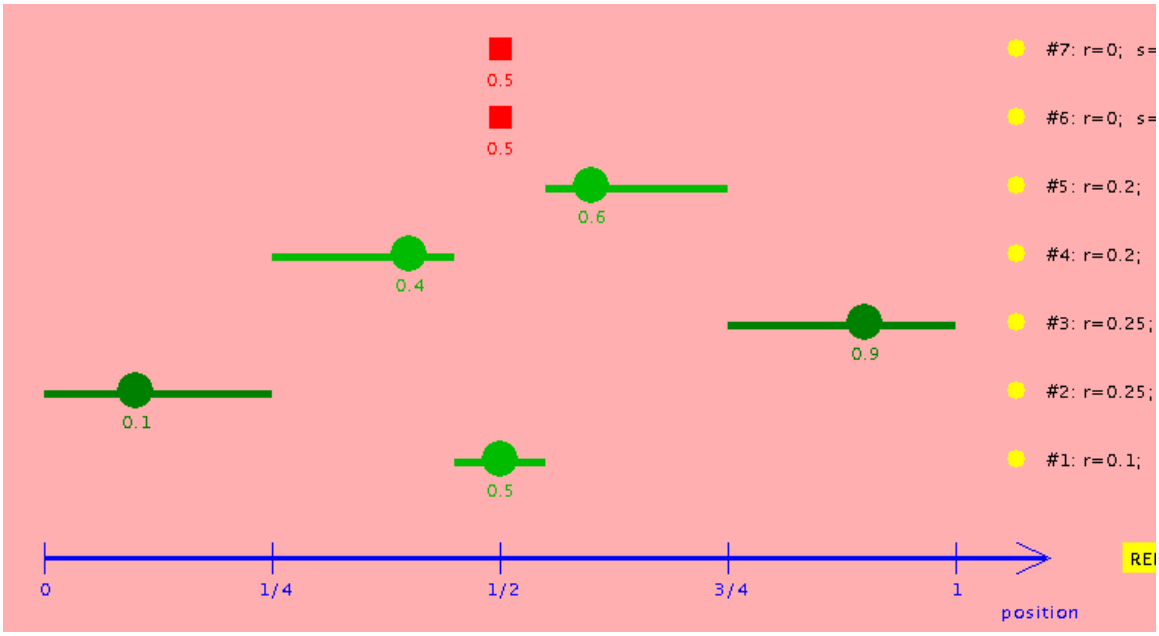

Figure 7: Output from the Voting Model Java Applet (original model) with $n = 7$ parties, assuming we begin with $x_1 = 0.5$, $x_2 = 0.1$, $x_3 = 0.9$, $x_4 = 0.4$, and $x_5 = 0.6$. The result is that Parties 6 and 7 stay OUT, rather than finding their optimal actions $x_6 = 0.2$ and $x_7 = 0.8$.

0 for parties which stay out, and -1 for parties who contest the election and lose, and $1/k$ for parties who contest the election and tie for highest vote share with a total of k parties.

We next define a "Modified Original Model" (MOM) as having the same vote shares as in (1), and still giving payoff 0 to parties which stay out, but now giving payoff $-c$ to parties which contest the election and lose (for some fixed penalty value c, assumed to satisfy $0 < c < 1/n$), and payoff $(1/k) - c$ to parties which go in and tie for highest vote with a total of k parties.

Then it is easy to see that:

Proposition 1 *MOM and OM are exactly equivalent in terms of the parties' behaviour. In particular, a sequence of positions $x_1, x_2, \ldots, x_n$ is a Nash equilibrium for MOM if and only if it is a Nash equilibrium for OM.*

Proof. This follows because the payoffs in the two games are ordinally equivalent. More formally, let $S_i(x_1, \ldots, x_n)$ be the payoff received by Party i according to OM when Party i

takes position x_i for $1 \leq i \leq n$, and let $T_i(x_1, \ldots, x_n)$ be the payoff received by Party i according to MOM. Then for two sets of positions $x_1, x_2, \ldots, x_n$ and $x'_1, x'_2, \ldots, x'_n$, $S_i(x'_1, \ldots, x'_n) > S_i(x_1, \ldots, x_n)$ if and only if $T_i(x'_1, \ldots, x'_n) > T_i(x_1, \ldots, x_n)$. This means that Party i has the same set of action preferences in the two models. In particular, Party i can profitably deviate from a configuration under MOM if and only if they can profitably deviate from it under OM. The result follows. ∎

Since MOM has equivalent actions as OM, it will not help with the discontinuity problems mentioned earlier. To deal with those, we instead introduce a new "Continuous Model" (CM). This model still has the same vote shares as in (1), and still gives payoff 0 to parties which stay out. It also gives payoff $-c$ to parties which contest the election and lose (for the same fixed penalty value $c \in (0, 1/n)$ as above). However, for parties who contest the election and receive a vote share w_i, their payoff score is now equal to

$$s_i \;=\; \frac{(w_i)^\alpha}{\sum_j (w_j)^\alpha} - c \tag{2}$$

for some fixed (large) power value $\alpha > 0$.

A key observation is that for large α, this new CM model is similar to the previous MOM model. More precisely, MOM is the *limit* of the CM models as $\alpha \to \infty$. (This was the motivation for introducing MOM; by Proposition 1, MOM and OM are equivalent, but only MOM is the limit of the CM models.)

Proposition 2 *As $\alpha \to \infty$, CM converges to MOM, in the sense that under the same actions, each party's payoff under CM converges to the corresponding payoff under MOM.*

Proof. Again let $T_i(x_1, \ldots, x_n)$ be the payoff received by Party i according to MOM when Party i takes position x_i for $1 \leq i \leq n$. Also let $U_i^{(\alpha)}(x_1, \ldots, x_n)$ be the payoff received by Party i according to CM with a particular power value $\alpha > 0$.

Obviously if $x_i = \text{OUT}$ then $U_i^{(\alpha)}(x_1, \ldots, x_n) = T_i(x_1, \ldots, x_n) = 0$ for all α, so clearly $\lim_{\alpha \to \infty} U_i^{(\alpha)}(x_1, \ldots, x_n) = T_i(x_1, \ldots, x_n) = 0$.

If Party i contests the election but receives a smaller vote share than some other Party j_*, so that $w_i < w_j$, then $T_i(x_1, \ldots, x_n) = -c$, while $U_i^{(\alpha)}(x_1, \ldots, x_n) = \frac{(w_i)^\alpha}{\sum_j (w_j)^\alpha} - c \geq -c$, but

on the other hand

$$\lim_{\alpha\to\infty} U_i^{(\alpha)}(x_1,\ldots,x_n) \;=\; \lim_{\alpha\to\infty} \frac{(w_i)^\alpha}{\sum_j (w_j)^\alpha} - c \;\le\; \lim_{\alpha\to\infty} \frac{(w_i)^\alpha}{(w_{j_*})^\alpha} - c \;=\; 0 - c \;=\; -c\,.$$

Hence, in this case $\lim_{\alpha\to\infty} U_i^{(\alpha)}(x_1,\ldots,x_n) = -c = T_i(x_1,\ldots,x_n)$.

Finally, suppose that Party i contests the election and ties for highest vote share with a total of k parties. Then there is a subset $\Gamma \subseteq \{1,2,\ldots,n\}$ with $i \in \Gamma$ and $|\Gamma| = k$, such that $w_i = w_j$ for all $j \in \Gamma$, and $w_i > w_j$ for all $j \notin \Gamma$. Then we compute that

$$\lim_{\alpha\to\infty} U_i^{(\alpha)}(x_1,\ldots,x_n) \;=\; \lim_{\alpha\to\infty} \frac{(w_i)^\alpha}{\sum_j (w_j)^\alpha} - c \;=\; \lim_{\alpha\to\infty} \frac{(w_i)^\alpha}{k(w_i)^\alpha + \sum_{j\notin\Gamma} (w_j)^\alpha} - c$$

$$=\; \lim_{\alpha\to\infty} \frac{1}{k + \sum_{j\notin\Gamma} (w_j/w_i)^\alpha} - c \;=\; \lim_{\alpha\to\infty} \frac{1}{k+0} - c \;=\; (1/k) - c\,,$$

so yet again $\lim_{\alpha\to\infty} U_i^{(\alpha)}(x_1,\ldots,x_n) = T_i(x_1,\ldots,x_n) = 0$, as claimed. ∎

Propositions 1 and 2 together show that our Continuous Model (CM) for large α is similar to the Modified Original Model (MOM), which in turn is ordinally equivalent to the Original Model (OM). This suggests that studying CM will lend additional insights into OM.

9 Simulation of the Continuous Model (CM)

In contrast to the situation with OM, our Java applet for CM is indeed able to avoid the above isolated-point limitation and find (nearly) optimal actions.

For example, consider the above situation where there are $n = 7$ parties, and the first five parties have already chosen their positions as $x_1 = 0.5$, $x_2 = 0.1$, $x_3 = 0.9$, $x_4 = 0.4$, and $x_5 = 0.6$. For OM, our Java applet was unable to locate the optimal remaining actions $x_6 = 0.2$ and $x_7 = 0.8$ (or vice-versa). However, for the continuous model, the applet does indeed find values very close to the optimal ones (see Figure 8).

Similar results were found in other simulations, too. In each case, the applet for CM (in contrast to OM) was able to find optimal actions even when they were isolated points. We conclude from this that CM is more amenable to simulation in some sense, as we discuss in the next section.

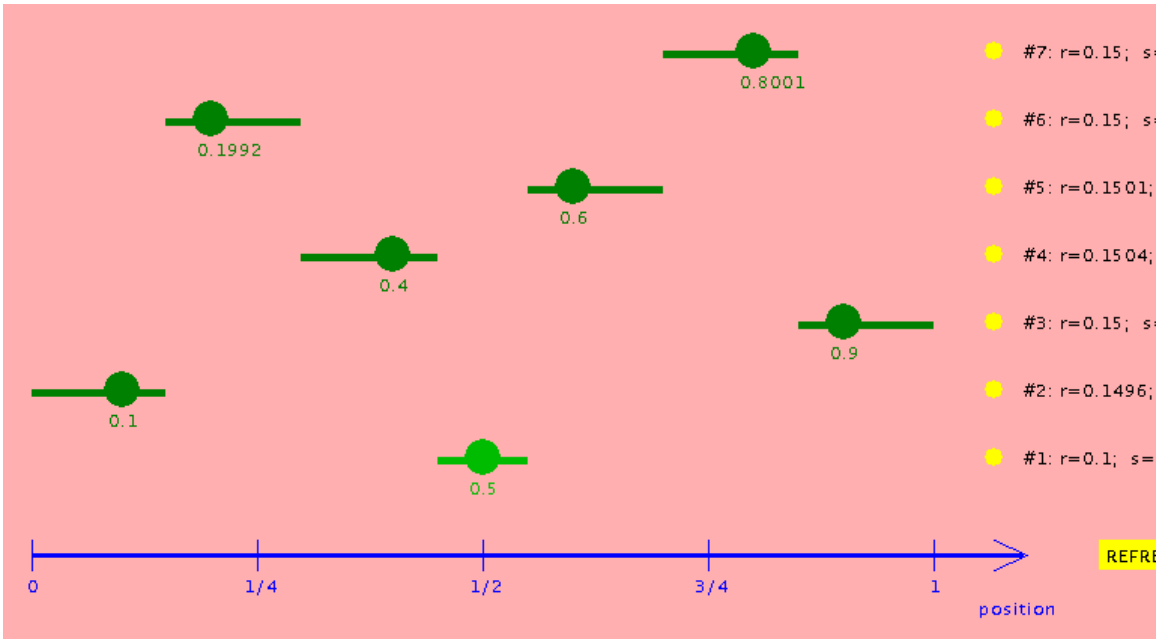

Figure 8: Output from the Voting Model Java Applet (continuous model) with $n = 7$ parties, assuming we begin with $x_1 = 0.5$, $x_2 = 0.1$, $x_3 = 0.9$, $x_4 = 0.4$, and $x_5 = 0.6$. In this case, Parties 6 and 7 find actions $x_6 = 0.1992$ and $x_7 = 0.8001$, which are very close to their optimal actions $x_6 = 0.2$ and $x_7 = 0.8$. And, since the model is now continuous, they accept those values and contest the election there and receive a positive payoff. In this way, the applet finds a set of actions which is very close to the optimal actions.

Despite Propositions 1 and 2 above, the optimal behaviour under CM does not always imitate that of OM. For example, suppose there are $n = 3$ parties, and we optimise the actions of all three.

For OM, as discussed above, it is optimal for Parties 1 and 3 to each come in at 1/2, and for Party 2 to stay out. And indeed, our applet (with a positive value of medianProb) quickly finds this equilibrium; see Figure 9.

On the other hand, our applet for $n = 3$ for CM finds a somewhat different optimal outcome. This time, Parties 2 and 3 come in at 1/2, while Party 1 stays out; see Figure 10.

In light of Propositions 1 and 2, the differing behaviour in these two cases might initially be dismissed as mere simulation error. However, to our initial surprise, it actually indicates fundamentally different behaviour of the OM and CM models, as we now explain.

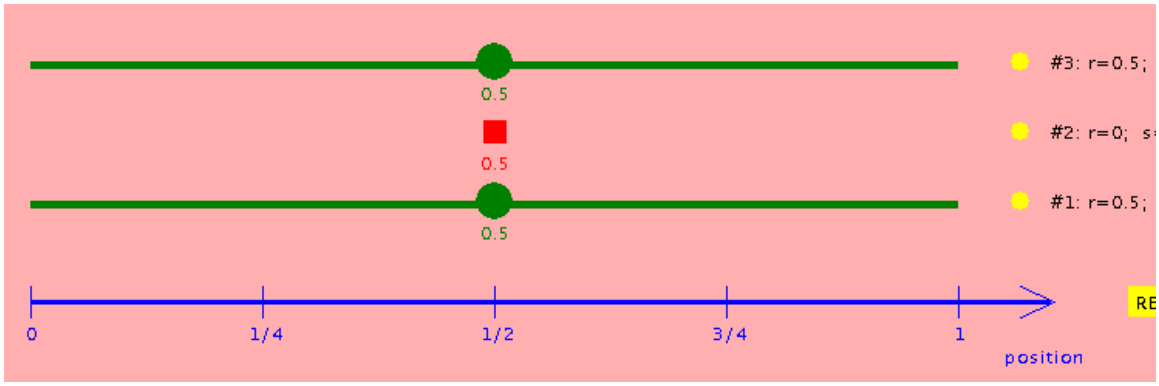

Figure 9: Output from the Voting Model Java Applet (Original Model), optimising over the actions of all of the $n = 3$ parties. The chosen optimal actions are $x_1 = x_3 = 1/2$ and $x_2 = \text{OUT}$, consistent with Osborne's conjecture.

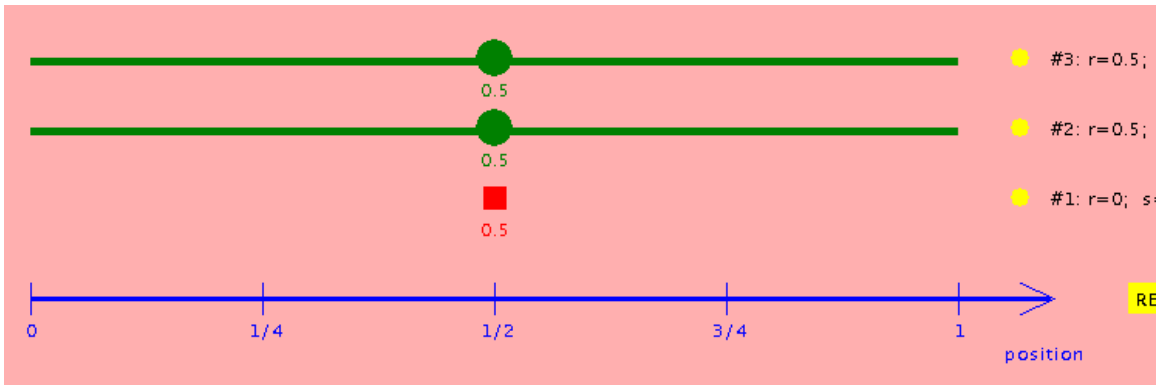

Figure 10: Output from the Voting Model Java Applet (Continuous Model), optimising over the actions of all of the $n = 3$ parties. In this case, the chosen optimal actions are $x_2 = x_3 = 1/2$ and $x_1 = \text{OUT}$, somewhat different from the Original Model (but still consistent with Duverger's Law).

Indeed, with $n = 3$ parties, suppose Party 1 comes in at $x_1 = 1/2$, and then Party 2 comes in very near to $1/2$, say at $x_2 = 0.501$. Then under OM, Party 3 can come in even closer to $1/2$ but on the other side, say at $x_3 = 0.4995$. If so, then Party 3 will win the largest vote share, and thus a payoff of $+1$, while Parties 1 and 2 will each lose and thus receive a payoff of -1. (Or, if Party 2 comes in at $x_2 = 1/2$, then Party 3 can come in at say $x_3 = 0.51$ and win the largest vote share.) This shows that it is actually *not* optimal for Party 2 to enter. Instead, it is optimal for Party 2 to stay out, and then for Party 3 to come in at $x_3 = 1/2$, as per Osborne's conjecture.

But under CM, the situation is different. In that case, suppose again that $n = 3$ and $x_1 = 1/2$ and $x_2 = 0.501$, and then Party 3 comes in at $x_3 = 0.4995$. Then Party 3 still wins the largest vote share, but Party 2 receives nearly as much, so they each receive payoff of about 1/2. (To be precise, if $x_1 = 1/2$ and $x_2 = 0.501$ and $x_3 = 0.4995$, and say $\alpha = 20$, then the vote shares are $w_3 = 0.49975$, $w_2 = 0.4995$, and $w_3 = 0.00075$, so Party 3 receives payoff of about 0.5025, and Party 2 about 0.4975, with Party 1 receiving approximately 10^{-57}.)

Now, it is still true as per Propositions 1 and 2 that as the power value $\alpha \to \infty$, the payoffs under CM converge to those under OM. So, in this case, as $\alpha \to \infty$, the payoff for Party 2 does indeed converge to zero. However, because the vote shares of Parties 2 and 3 are so close, α has to be very large before the asymptotics kick in. Indeed, if $\alpha = 1,000$, then Party 2's payoff is still about 0.3775. But if $\alpha = 10,000$ then it is about 0.000667, and if $\alpha = 100,000$ then it is about 10^{-22}. More importantly, for any fixed choice of $\alpha > 0$, if $x_1 = 1/2$, then Party 2 can always choose x_2 sufficiently close to 1/2 to obtain a positive payoff of nearly 0.5 (assuming Party 3 then chooses a position x_3 on the opposite side of 1/2 from x_2). This indicates, surprisingly in light of Propositions 1 and 2, that behaviour similar to that in Figure 10 can persist even for arbitrary large α.

We conclude from all of this that the Continuous Model is closely related to the Original Model (cf. Propositions 1 and 2), and it is more amenable to simulations (see above, and as discussed further below), but it does lead to somewhat different optimal behaviour in some cases.

10 Attainable Equilibria

We finish with a more theoretical discussion of the extent to which our Continuous Model (CM) is more amenable to simulation than the Original Model (OM). To do this, we wish to capture the notion that it is possible for our simulation to "find" actions which are "close" to the optimal ones. We begin with some definitions.

Let $\mathcal{X}$ be the set of all possible actions by all parties, i.e. $\mathcal{X} = (\{\text{OUT}\} \cup [0, 1])^n$.

Let D be a distance metric on $\mathcal{X}$. In our case, we use the L^1 distance $D(x, y) = \sum_{i=1}^{n} |x_i - y_i|$, with the understanding that if $x_i = y_i = \text{OUT}$ then we take $|x_i - y_i| = 0$, while if

$x_i = \text{OUT}$ but $y_i \in \mathbf{R}$ or vice-versa then we take $|x_i - y_i| = 1$.

Given a list $x = (x_1, x_2, \ldots, x_n)$ of positions of all parties, and $\epsilon > 0$, we define $\Omega(x, \epsilon)$ to be the "ϵ-neighbourhood" of x, i.e. the set of all position lists $x' = (x'_1, x'_2, \ldots, x'_n)$ such that (a) the party positions x' are within ϵ of the corresponding positions x, i.e. $D(x', x) \leq \epsilon$, and (b) each party's payoff from x' is within ϵ of the corresponding payoff from x, i.e. if $s_1, \ldots, s_n$ are the party payoffs from x, and $s'_1, \ldots, s'_n$ are the party payoffs from x', then $|x'_i - x_i| \leq \epsilon$ for all i.

To continue, we borrow the notion of "ϕ-irreducibility" from the Markov chain Monte Carlo literature (see e.g. Tierney, 1994; Meyn and Tweedie, 1993; Roberts and Rosenthal, 2004). We begin with a reference measure ϕ on $\mathcal{X}$, i.e. any countably-additive measure on the set $\mathcal{X}$ of all possible actions. Then we define a position list x to be *ϕ-attainable* if for any $\epsilon > 0$, the ϵ-neighbourhood $\Omega(x, \epsilon)$ of x has positive measure under ϕ, i.e. $\phi(\Omega(x, \epsilon)) > 0$. Intuitively, this means that the set of all position lists "near" to x has positive measure, according to ϕ.

One specific reference measure of interest is $\phi_{\text{ind}} = (\frac{1}{2}\,\delta_{\text{OUT}} + \frac{1}{2}\,\text{Uniform}[0,1])^n$. This measure corresponds to each party independently choosing the same strategy "$\frac{1}{2}\,\delta_{\text{OUT}} + \frac{1}{2}\,\text{Uniform}[0,1]$", which has the interpretation of staying out with probability 1/2, otherwise coming in with a position distributed as Uniform[0,1] (more formally, Lebesgue measure on [0,1]). We have the following.

Proposition 3 *Let $x = (x_1, \ldots, x_n)$ be a position list in which no two parties come in at the same position, i.e. if $x_i \in \mathbf{R}$ and $x_j \in \mathbf{R}$ and $i \neq j$, then $x_i \neq x_j$. Then under the Continuous Model (CM), the position list x is ϕ_{ind}-attainable.*

Proof. If the $\{x_j\}$ are all distinct, then each vote region R_i and vote share w_i and payoff is a *continuous* function of the $\{x_j\}$. Hence, for all configurations x' sufficiently close to x, the payoffs are still within ϵ of their original values, so that $x' \in \Omega(x, \epsilon)$. This gives the result. ∎

Proposition 3 may be interpreted as saying that, if a computer program simulates potential position lists from ϕ_{ind}, and x has no two parties at the same position, then for any $\epsilon > 0$, it has a positive probability of eventually proposing a position list which is within ϵ of x.

If desired, by carefully bounding the various derivatives, it is possible to obtain a (weak) *quantitative* version of Proposition 3, as follows (proved in the Appendix).

Proposition 4 *Let x be a configuration for which $|x_i - x_j| \geq \Delta$ for all $i \neq j$, and $s_i \geq \delta > 0$ for each positive score function s_i. Then under the Continuous Model, for any $\epsilon > 0$, the ϵ-neighbourhood $\Omega(x, \epsilon)$ includes all configurations x' for which $D(x', x_i) \leq \gamma := \min[\Delta,\ \delta/2,\ \epsilon\,\delta^{\alpha-1}/\alpha(n-1)]$. Furthermore, $\phi_{\text{ind}}\big(\Omega(x,\epsilon)\big) \geq (\gamma/2)^n\,(2^n/n!) > 0$.*

To make Propositions 3 and 4 more concrete, consider again the position list discussed earlier as inspired by de Vries (2015) and approximated in Figure 8 above, namely $x_1 = 0.5$, $x_2 = 0.1$, $x_3 = 0.9$, $x_4 = 0.4$, $x_5 = 0.6$, $x_6 = 0.2$, and $x_7 = 0.8$. Under the Original Model (OM), this configuration is clearly *not* ϕ_{ind}-attainable, since if e.g. x_7 is changed by any arbitrarily small amount, then Party 7's payoff is changed from $+1/6$ to -1 which is a difference more than ϵ for all small $\epsilon > 0$. However, by Proposition 3, this configuration is indeed ϕ_{ind}-attainable in the Continuous Model (CM), and indeed a (weak) lower bound on $\phi_{\text{ind}}(\Omega(x,\epsilon))$ could then be computed from Proposition 4. This further illustrates the difference between OM and CM in terms of simulation.

Now, Propositions 3 and 4 do not apply to position lists where multiple parties come in at the same position. To deal with such cases, we need to modify our reference measure to a new one ϕ_{mix} which gives positive probability to configurations with multiple equal positions. Specifically, given a partition of $\{1, 2, \ldots, n\}$ into subsets $\mathcal{S} = (S_0, S_1, \ldots, S_k)$, we let $\mu_{\mathcal{S}}$ be the probability measure under which all parties in S_0 choose OUT (with probability 1), and for $1 \leq i \leq k$, all of the parties in S_i all choose the same equal position $x_i \in \mathbf{R}$, where x_i is chosen independently from the Uniform[0,1] distribution. Then we define ϕ_{mix} to be the sum of the $\mu_{\mathcal{S}}$, over all partitions $\mathcal{S} = (S_0, S_1, \ldots, S_k)$ of $\{1, 2, \ldots, n\}$. That is,

$$\phi_{\text{mix}} \;=\; \sum_{\mathcal{S}=(S_0,S_1,\ldots,S_k)} \mu_{\mathcal{S}}\,,$$

with $\mu_{\mathcal{S}}$ as above. This definition of ϕ_{mix} is rather cumbersome, but it does allow us to attain position lists even if they have multiple identical positions:

Proposition 5 *Under the Continuous Model (CM), any position list x is ϕ_{mix}-attainable.*

We now apply the attainable concept to simulation algorithms. Consider a simulation which proposes new strategies according to some random rules, keeping track of the best strategy so far. Given any reference measure ϕ on $\mathcal{X}$, we say that the simulation procedure *covers* ϕ if there is $\epsilon > 0$ and $N \in \mathbf{N}$ such that for any subset $A \subseteq \mathcal{X}$ of positions lists satisfying $\phi(A) > 0$, and any $t \geq 0$, and any previous history of the simulation up to iteration t, the simulation still has probability at least ϵ of proposing a state within the subset A at some iteration between $t+1$ and $t+N$. Roughly speaking, a simulation covers ϕ if it has positive probability of proposing tweaks to any ϕ-positive subset of configurations. The connection between covering and attainability is as follows. (Note that Proposition 6 applies to any game, not just the voter models discussed herein.)

Proposition 6 *Let ϕ be any reference measure on $\mathcal{X}$, and let $x \in \mathcal{X}$ be any action configuration. Suppose x is ϕ-attainable, and a computer simulation covers ϕ. Then for any $\epsilon > 0$, as the number of iterations converges to infinity, the probability that the computer simulation will propose a configuration within the ϵ-neighbourhood $\Omega(x, \epsilon)$ of x converges to 1.*

Now, the Java applet simulation described above clearly covers ϕ_{ind}, since e.g. it has positive probability of proposing a tweak from the Uniform[0,1] distribution (and also of switching to and from OUT) at each iteration. Furthermore, provided mimicProb > 0, the applet also covers ϕ_{mix}. This shows:

Proposition 7 *For any $\epsilon > 0$, and any fixed configuration list x, as the number of iterations goes to infinity, the Java applet described herein for the Continuous Model, with mimicProb > 0, will eventually propose tweaks within the ϵ-neighbourhood of x.*

Now, Proposition 7 does not specify that the applet's proposed tweak to the ϵ-neighbourhood will be *accepted*. However, if x is a configuration in which each party behaves optimally, then any proposed tweak leads to a payoff within ϵ of optimal for each party. So, such a tweak will be accepted by the algorithm, unless it has already found another configuration with even larger payoffs. This proves:

Proposition 8 *For any $\epsilon > 0$, as the number of iterations goes to infinity, the Java applet described herein for the Continuous Model, with mimicProb > 0, will eventually find a configuration giving payoffs for all parties which are within ϵ of being optimal.*

Proposition 8 verifies that our Java applet approach is valid for the Continuous Model (at least), in the sense that it is guaranteed to eventually find configurations which are within ϵ of being optimal.

On the other hand, if our algorithm is just run for a *finite* amount of time, then it still might miss an ϵ-optimal solution. And in any case, the above guarantees are only for the Continuous Model, not for the Original Model. For both of these reasons, although our results are suggestive, they do not suffice to actually *prove* Osborne's conjecture, which thus still remains just a conjecture for all $n > 4$.

Appendix: Proof of Proposition 4

Finally, we prove Proposition 4. We proceed in stages.

Beginning with the vote regions R_i, we see from the definition of R_i that if any x_j changes by a small amount, then (since $x_i \neq x_j$) R_i can change by at most half that amount at each endpoint. It follows from (1) that the vote shares w_i satisfy $\left|\frac{\partial w_i}{\partial x_j}\right| \leq 1/2$ for any i and j (whether $i = j$ or not). Also we must always have $\left|\frac{\partial w_i}{\partial w_j}\right| \leq 1$.

Next we consider the payoff functions w_i from (2). We first note that in general, for any non-negative functions f and g,

$$\left|\frac{d}{dx}\,\frac{f(x)}{f(x)+g(x)}\right| \;=\; \left|\frac{d}{dx}\,\frac{1}{1+g(x)/f(x)}\right| \;=\; \left|-\frac{\frac{d}{dx}[g(x)/f(x)]}{(1+g(x)/f(x))^2}\right| \;\leq\; \left|\frac{d}{dx}\,\frac{g(x)}{f(x)}\right|.$$

Applying this with $f = (w_i)^\alpha$ and $g = \sum_{j\neq i}(w_j)^\alpha$ (and taking $w_i = 0$ for parties who do not contest the election) gives

$$\left|\frac{\partial s_i}{\partial w_j}\right| \;\leq\; \sum_{j\neq i} \alpha(w_j/w_i)^{\alpha-1}\,.$$

In particular, $w_j \leq 1$, so

$$\left|\frac{\partial s_i}{\partial w_j}\right| \;\leq\; \sum_{j\neq i} \alpha(1/w_i)^{\alpha-1} \;=\; \alpha(n-1)/w_i^{\alpha-1}\,.$$

Then since $\left|\frac{\partial w_i}{\partial x_j}\right| \leq 1/2$, it follows that

$$\left|\frac{\partial s_i}{\partial x_j}\right| \;\leq\; \alpha(n-1)/2w_i^{\alpha-1}\,.$$

Now, if $D(x', x) \leq \min(\Delta,\, \delta/2)$, then $w'_i \geq \delta/2$, so at any such x' we must still have

$$\left|\frac{\partial s_i}{\partial x_j}\right| \;\leq\; \alpha(n-1)/\delta^{\alpha-1}\,.$$

So, if $D(x', x) \leq \min(\Delta,\, \delta/2,\, \beta)$ for some $\beta > 0$, then each s'_i is within $\beta\alpha(n-1)/\delta^{\alpha-1}$ of s_i. Solving, this bound equals ϵ if $\beta = \epsilon\,\delta^{\alpha-1}/\alpha(n-1)$. Furthermore this $\beta \leq \epsilon$, too. So, if $D(x', x) \leq \min(\Delta,\, \delta/2,\, \beta) =: \gamma$, then we must have $x' \in \Omega(x, \epsilon)$ as claimed.

For the second result, recall that $\phi_{\text{ind}} = (\frac{1}{2}\,\delta_{\text{OUT}} + \frac{1}{2}\,\text{Uniform}[0,1])^n$, so $\phi_{\text{ind}}\big(\Omega(x,\epsilon)\big)$ must be at least as large as $(1/2)^n$ times the Lebesgue measure of an n-dimensional L^1 ball of radius γ. By rescaling, the latter is the same as $\gamma^n(1/2)^n$ times the Lebesgue measure of a *unit* n-dimensional L^1 ball. But it is known (e.g. Wang, 2005) that the volume of a unit n-dimensional L^1 ball is equal to $2^n/n!$. Hence, $\phi_{\text{ind}}\big(\Omega(x,\epsilon)\big) \geq \gamma^n(1/2)^n(2^n/n!)$, as claimed. ∎

Acknowledgements. I thank Martin J. Osborne for introducing me to this topic and for many helpful discussions, and thank the anonymous referee for a careful reading and many excellent suggestions. This research was partially supported by NSERC of Canada.

References

J.-P. de Vries (2015), Duverger's (f)law: Counterproof to the Osborne Conjecture. MSc thesis. Available at: https://thesis.eur.nl/pub/17645/

J.-P. de Vries, J.J.A. Kamphorst, M.J. Osborne, and J.S. Rosenthal (2016), On a conjecture about the sequential positioning of political candidates. Work in progress.

M. Duverger (1951), Les Partis Politiques. Paris: Armand Colin.

S.P. Meyn and R.L. Tweedie (1993), Markov chains and stochastic stability. Springer-Verlag, London. Available at `probability.ca/MT`.

J. Nash (1951), Non-cooperative games. *The Annals of Mathematics* **52(2)**, 286–295.

M.J. Osborne (1996), A conjecture about the subgame perfect equilibria of a model of sequential location. Available at: https://www.economics.utoronto.ca/osborne/research/conjecture.html

M.J. Osborne (2003), An Introduction to Game Theory. Oxford University Press.

W. Riker (1982), The two-party system and Duverger's law: an essay on the history of political science. *American Political Science Review* **76(4)**, 753-?766.

G.O. Roberts and J.S. Rosenthal (2004), General state space Markov chains and MCMC algorithms. *Probability Surveys* **1**, 20–71. http://www.i-journals.org/ps/viewarticle.php?id=15

J.S. Rosenthal (2015), Voting Model Java Applet. Available at: www.probability.ca/voting

J.A. Schlesinger and M.S. Schlesinger (2006), Maurice Duverger and the study of political parties. *French Politics* **4**, 58–68.

L. Tierney (1994), Markov chains for exploring posterior distributions (with discussion). Ann. Stat. **22**, 1701–1762.

X. Wang (2005), Volumes of Generalized Unit Balls. *Mathematics Magazine* **78(5)**, 390–395.