A TOURNAMENT OF PARTY DECISION RULES

James H. Fowler
University of California, San Diego

Michael Laver
New York University

ABSTRACT

In the spirit of Axelrod's famous series of tournaments for strategies in the repeat-play prisoner's dilemma, we conducted a "tournament of party decision rules" in a dynamic agent-based spatial model of party competition. A call was issued for researchers to submit rules for selecting party positions in a two-dimensional policy space. Each submitted rule was pitted against all others in a suite of very long-running simulations in which all parties falling below a declared support threshold for two consecutive elections "died" and one new party was "born" each election at a random spatial location, using a rule randomly drawn from the set submitted. The policy-selection rule most successful at winning votes over the very long run was declared the "winner". The most successful rule was identified unambiguously and combined a number of striking features. It satisficed rather than maximized in the short run; it was "parasitic" on choices made by other successful rules; and it was hard-wired not to attack other agents using the same rule, which it identified using a "secret handshake". We followed up the tournament with a second suite of simulations in a more evolutionary setting in which the selection probability of a rule was a function of its "fitness", measured in terms of the previous success of agents using the same rule. In this setting, the rule that won the original tournament pulled even further ahead of the competition. Treated as a discovery tool, tournament results raise a series of intriguing issues for those involved in the modeling of party competition.

Presented at the Annual Meeting of the American Political Science Association
Philadelphia: 31 August – 3 September 2006.
Panel 14-5 New Challenges to Partisan Strategies and Government Policies

Thanks are due to Robert Axelrod and Scott deMarchi for comments on the design of this tournament, and to well over 24 professional colleagues who devoted what was often considerable time and effort in designing tournament entries

1. INTRODUCTION

Consider a dynamic system of multi-party competition in a multidimensional policy space in which party leaders compete for votes at regular intervals by selecting a party location intended to appeal to as many voters as possible, given the policy positions of all other parties. The decision problem for the party leader is to select an optimal party location conditional on all available information, which is highly imperfect.

The number of different rules a party leader could use to select a position in a multidimensional policy space at any given time, conditional on the past history of the party system and the decision rules being used by other party leaders, is effectively infinite. This creates a serious analytical problem if we want to evaluate the relative effectiveness of different decision rules, a problem considerably more complex, for example, than that of evaluating different strategies for playing an iterated Prisoner's Dilemma (PD) game. At any given period of the iterated PD, the problem for the agent is to make a binary choice between co-operation and defection. At any given period of a party competition game, the problem is to pick one of an infinite number of possible locations in an *n*-dimensional real space.

One response to the complexity of modeling the effects of different policy-selection rules in a multi-dimensional multiparty dynamic setting has been to construct systematic computer simulation experiments, using agent-based models (ABMs) of party competition (Kollman, Miller and Page 1992; Kollman, Miller and Page 1998; De Marchi 1999; De Marchi 2003; Kollman, Miller and Page 2003; Kollman, Miller and Page 2003; Fowler and Smirnov 2005; Laver 2005; Laver and Schilperoord forthcoming 2007). However, these experiments have only explored interactions between a limited number of analyst-specified and predetermined decision rules; in no sense has the full set of *potential* rules been investigated.

Responding to a similar problem in relation to the much simpler task of evaluating potential decision rules (thought of by classical game theorists as "strategies") for use in the iterated PD, Robert Axelrod conducted a famous series of computer tournaments (Axelrod 1980a; Axelrod 1980b; Axelrod 1997). Interested individuals were asked to submit strategies for the iterated PD, which were then pitted against each other in a series of computer simulation experiments, with the highest-scoring strategy declared the winner. In the first tournament (Axelrod 1980a), with 14 entries, the winner was *Tit-for-Tat*, submitted by Anatol Rapoport. In a second tournament, new entries were invited in light of the results of the first tournament (Axelrod 1980b). There were 62 entries and *Tit-for-Tat* won again. In each of these experiments, all strategies investigated were predefined and immutable. In a more recent experiment, Axelrod explored the *evolution of new*

strategies for playing the iterated PD game. He started with a set of random strategies, as opposed to a set pre-defined by others, and applied the standard genetic operators of crossover and mutation to these over successive iterations, rewarding successful strategies with higher fitness scores and hence higher reproduction probabilities (Axelrod 1997). Strategies resembling *Tit-for-Tat* often emerged from this evolutionary process. However, completely new types of successful strategy also evolved which beat *Tit-for-Tat*. These were strategies no game theorist had submitted to earlier tournaments and all involved at least one initial defection, serving to probe an opponent's "type" in a more informative way than initial cooperation. In a 20th-anniversary re-run of the Axelrod tournament with 223 entries, the competition was won by a 60-entry portfolio of "master-slave" strategies submitted by a team from the University of Southampton led by the computer scientist Nick Jennings. Southampton strategies signed in with a distinctive and unusual sequence of early moves and were thus able to recognize each other during play, during which Southampton slaves offered themselves up for exploitation by a Southampton master whenever they encountered one.[1]

Here, we adapt this fertile "computer tournament" research design to the problem of evaluating different potential decision rules for the dynamic selection of party positions in a multi-party, multi-dimensional environment. We build on a recent dynamic implementation in an ABM environment (Laver 2005), of the canonical "static" spatial model of party competition. Published work on this model has investigated four different algorithms that party leaders might plausibly use to select a sequence of policy positions over time, conditional on their observations of past states of the party system. Adapting this ABM as a tournament "test-bed", we invited submissions of position-selection rules, advertising our tournament by word-of-mouth, on our websites and on the popular professional list-server POLMETH, offering a $1000 prize for the position-selection rule most successful in winning votes. The four rules investigated by Laver (2005) were declared as pre-entered in the tournament but not eligible to win the prize. We also published the test-bed code, programmed in R, for running simulations and evaluating submissions. (A full description of the tournament can be found in Appendix A.)

Following the submission deadline we loaded all submitted rules onto the simulation test-bed. Each was pitted against all others in a suite of very long-running simulations in which all parties falling below a ten percent support threshold for two consecutive elections "died" and a new party was "born" each election at a random spatial location, using a decision rule randomly drawn

[1] For details, see www.prisoners-dilemma.com. The Southampton portfolio of strategies was submitted as part of a research program investigating collusion and competition between software agents. See below for a discussion of "secret handshakes".

(with replacement) from the set of all rules competing in the tournament. The algorithm most successful at winning votes over the very long run was declared the "winner".

We describe the simulation test-bed for dynamic models of party competition in Section 2, together with the decision rules pre-entered in the tournament. In Section 3, we describe and discuss the submissions we received. We report and discuss the results of our tournament simulation experiments in Section 4. In section 5, we describe the results of extending the tournament into a much more evolutionary setting, where the probability that a decision rule will reproduce is a function of the past success of agents using the same rule. Finally, we conclude and lay out a program for future work.

2. AN ABM "TEST-BED" DESIGNED TO EVALUATE RULES FOR SELECTING SEQUENCES OF PARTY POSITIONS IN A MULTIDIMENSIONAL POLICY SPACE

Previous ABMs of multi-party party competition in multidimensional policy spaces

There are several reasons to change methodologies from the analytical approach that underpins the canonical static spatial model of party competition to the systematic interrogation of computer simulations using ABMs. The first concerns the *analytical intractability* of complex dynamic spatial models, which are nonetheless amenable to tractable systematic investigation using computer simulations. The second involves a reassessment, given the complexity of party policy selection in a multi-party, multidimensional dynamic environment, of plausible behavioral assumptions about agent rationality. Classical analytical theory tends to assume hyper-rational agents engaged in deep strategic look-forward; equilibrium strategy sets take account of every possible future choice that might be made by every agent. When agents face potentially bewildering complexity in their decision-making environment, however, an alternative behavioral assumption, implemented using ABM, stresses *adaptive learning rather than deep strategic prospective rationality*. A third reason for a shift to ABM concerns the quality of available information. Information requirements for many analytical spatial models are very high; many models, for example, assume all politicians know the ideal points of all citizens. In models where citizen ideal points are not known, it is typically assumed politicians have good parametric information about the distribution of these, such as the mean, the variance, and/or the functional form. Assuming much less information, adaptive learning, implemented using ABM, may offer an empirically more plausible description of *how political agents gather information* about the world in which they make choices. Thus the shift in methodology/epistemology from analytical theory to ABM involves several changes in how party competition is characterized – from a high-information static environment populated by prospectively rational agents, to a low-information dynamic environment

populated by adaptively rational agents. This shift is not just a change in method, it is a change in the fundamental characterization of party competition.

Building on earlier work by Kollman, Miller and Page, and by De Marchi, Laver developed an ABM of party policy selection in a two-dimensional real policy space (Kollman, Miller and Page 1992; Kollman, Miller and Page 1998; De Marchi 1999; De Marchi 2003; Kollman, Miller and Page 2003; Laver 2005). Working within the Downsian tradition, this assumed non-strategic "proximity" voters with ideal points randomly drawn from a bivariate normal distribution, each voter supporting the closest party.[2] Party leaders select policy positions without knowing the ideal point of any voter, basing their choices on the inferences they can draw from past positions and support levels of each party in the system. Laver defined four policy-selection rules that are suggested by traditional empirical literatures on intra-party decision-making:

- STICKER: never change position (an "ideological" party leader);
- AGGREGATOR**:** select party policy on each dimension as the mean preference of all party supporters (a "democratic" party leader responding perfectly to supporter preferences);
- HUNTER[3]**:** if the last policy move increased support, make the same move; else, reverse heading and make a unit move in a heading chosen randomly from the arc $\pm 90^{o}$ from the direction now being faced (an autocratic party leader who is a Pavlovian vote-forager);
- PREDATOR: identify largest party; if this is you, stand still; else, make a unit move towards largest party (an autocratic party leader seeking votes by attacking larger parties).

Laver's most striking finding was that party leaders using the *Hunter* algorithm, despite its simplicity, are systematically more successful at selecting popular policy positions than party leaders using any of the other algorithms investigated. Agents using *Hunter* tend to seek votes towards the center of the distribution of voter ideal points, but systematically to avoid the dead center of this distribution. Agents using the *Predator* algorithm, in contrast, do tend to go to the center of the space and to be unsuccessful in competition with *Hunters* at this location. When there was more than one *Predator* in the system, furthermore, any successful party using the Predator algorithm tends to be attacked by another *Predator*. Agents using the *Aggregator* algorithm tend to

[2] To model strategic voting in a multi-party multi-dimensional setting in which no party wins a majority of votes requires modeling the following: how voters forecast the election result without knowing the ideal points of other voters; how voters forecast which government will form following post-electoral coalition bargaining; how voters forecast the real-world policy outputs arising from this government; in light of the above how voters resolve the calculus of turnout problem. We extend our heartfelt good wishes to any analyst who sets out to do all of this.

[3] The biological analogue for Hunter behavior is "klinokinetic avoidance".

select significantly more eccentric (off-center) policy positions. Party systems comprising *only* agents using *Aggregator* always reach a steady state with no party movement, although this steady state is easily perturbed.[4]

Most existing work on ABMs of party competition takes both the set of parties in contention and the set of policy-selection algorithms deployed as both predefined and fixed. However, Laver and Schilperoord (2007), building on the arguments of "citizen candidate" models (Osborne and Slivinski 1996; Besley and Coate 1997), generalized the ABM of party competition to endogenize the number of parties, while using the same predefined set of policy-selection algorithms. They model the "birth" of new parties at points in the policy space where citizens have relatively high levels of cumulative dissatisfaction, measured in terms of their distance, aggregated over time, from their closest party. The "death" of existing parties is modeled in terms of parties' inability to maintain their support over some survival threshold. In a birth-evolved system such as this, for example, even unresponsive *Sticker* parties do not tend to occupy unpopular locations over the long run. *Stickers* at unpopular locations tend to die, while new *Stickers* tend to be born at more popular locations.

The ABM test bed for our tournament

Our intention in this paper is not to create the most realistic ABM of party competition we can devise, but to create a level playing field on the ABM test bed we use to evaluate different policy-selection algorithms. The most crucial requirement is that the tournament not be biased against any particular algorithm. We thus augmented the ABM developed by Laver (2005), adding provision for the "birth" of new parties and the "death" of existing parties. In contrast to the ABM in Laver and Schilperoord (2007) our test bed ABM exogenously forces party births at random locations, rather than allowing births to emerge endogenously at fertile locations as a function of the spatial configuration of existing parties. In our tournament, one new party is born every 20 periods, regardless of the state of the party system. The new party is given a random spatial location and assigned a policy-selection rule randomly drawn, with equal probability and with replacement, from the set of rules in the tournament. To simulate party death, we introduced a 10 percent survival threshold; parties die if they fall below 10 percent of the vote for two successive elections. Self-

[4] Analytically, this is because a system with a set of agents using *Aggregator* will be in steady state when party locations generate a "centroidal Voronoi tessellation" (CVT) of the policy space. A CVT is a Voronoi tessellation in which every stimulus point is at the centroid of its region. CVTs are useful in understanding, among many other things, the territorial behavior of animals, as well as image enhancement and the location of fast-food franchises. Steady-state simulation results for all-*Aggregator* systems reflect analytical existence results for CVTs (Du, Faber and Gunzberger, 1999). Indeed a dynamic all-*Aggregator* system is in effect an efficient algorithm for finding a CVT of the policy space.

evidently, if any party system systematically generates party births without at the same time having a *de facto* survival threshold, the number of parties in competition will grow relentlessly towards a *reductio ad absurdum* with an infinite number of parties.[5]

An important change from Laver's ABM, one that we consider substantively plausible, is that our model distinguishes between *inter-electoral* periods of party competition, in which parties adjust their policy positions in response to published polling information about levels of party support, and *election* periods in which real votes are cast by citizens, party vote totals are recorded, and parties may die and be born. There is an election every 20 periods of our model. At the end of each of the 19 inter-electoral periods, party positions and support levels are made public to all other parties, but these are not taken into account as part of the evaluation of the relative success of different decision rules. This gives parties time to adapt their positions prior to each election period in which the outcome will have consequences for their behavior.

Each simulation run started, artificially, with the birth of a single party. We repeated this initial state five times for each of the n algorithms entered into the tournament, yielding $5n$ simulation runs. Each simulation run lasted 220,000 periods, but the first 20,000 periods were discarded as "burn-in" to remove any possibility of artificial effects arising from initial conditions. Thus the scoring phase of each run lasted 200,000 periods and involved 10,000 elections. Since 25 policy-selection rules were submitted (see below) in addition to the four pre-submitted rules, making 29 in all, the full tournament involved 145 simulation runs, each of 200,000 periods and 10,000 elections after burn-in. Evaluation of party decision rules was thus based on a total of 29,000,000 simulation periods and 1,450,000 elections.

The length of these simulation runs was determined prior to the tournament. Using the four pre-entered rules, we conducted 20 simulations (in which each of the four rules was used five times by the first-born party) and let the simulation run to 100,000 periods (5,000 elections). We studied trace plots and used diagnostics to evaluate convergence in the party vote shares at each election. The Heidelberger convergence diagnostic, which uses the Cramer-von-Mises statistic to test the null hypothesis that the sampled values come from a stationary distribution, showed some nonstationary effects on the distribution up to 10,000 periods (500 elections) but none thereafter, so we chose to discard the first 20,000 periods of each chain. We also applied a Brooks-Gelman test to the set of simulations (Brooks and Gelman 1998). This indicates the point at which the output from all chains is indistinguishable, and thus at which initial values no longer influence the results.

[5] The ten percent survival threshold was selected because it tends, on average, to result in a system with what we take to be an empirically realistic number of seven to ten parties at any given time. See Laver and Schilperoord (2007) for simulations systematically investigating the "carrying capacity" of dynamic party systems with different survival thresholds.

This test is based on a comparison of within-chain and between-chain variances, and is similar to a classical analysis of variance.[6] The 'potential scale reduction factors' for each variable in our simulations were reduced to less than 1.1 by the 40,000th period, indicating convergence. However, to be safe, we decided to extend the simulation length to 200,000 periods. Longer runs of up to one million periods yielded identical results.

Information restrictions on submitted algorithms

As noted above, a key feature of existing ABMs of policy-based party competition is that they characterize decision-making agents, not as hyper-rational deep strategists, but as adaptive agents deploying bounded rationality in a world of very incomplete information. Thus, while the test-bed code for the tournament was published in advance, the tournament rules specified very clearly that policy-selection rules could only make use of the following information: any previously-announced policy position and level of voter support for any party; the mean or median position on each dimension of the ideal points of the party's own current supporters. Valid policy selection rules did *not* have access to the ideal point of any individual voter or to the decision rule being used by any other party; and they were *not* allowed to ask voters what they would do in counterfactual situations.

These information restrictions have the effect of ruling out the type of "hill-climbing" algorithms proposed by Kollman Miller and Page (1992, 1998, 2003), all of which imply what amounts to private opinion polling by party leaders about counterfactual policy positions. At issue here is not the intrinsic efficacy of such algorithms, but the substantive plausibility of their information requirements, a problem that becomes particularly evident when the KMP model is extended, as in KMP (2003), from a two-party incumbent-challenger setting to one with an arbitrary number of political parties. In this context the counterfactual question posed by each party in private polling must take the form: "assuming that each of the *n-1* other parties retains its current policy package but our party moves its policy package from x to $x \pm \delta$, which party would you then support?" Hill-climbing implies that a battery of similar questions, generating systematic sweeps of δ, must be asked by each party at each cycle of the competitive process (while all other parties stand still), to select the best policy in a given position. This seems a very complicated and unrealistic assumption about how parties gather information and use this to select policy positions.[7]

[6] Each chain starts at an over-dispersed starting point different from other chains. Between-chain variance should start high and *decrease* while within-chain variance starts low and *increases*. When the ratio of these variances is close to 1 then it suggests that the chains have reached their stationary distribution.

[7] This is in addition to violating every professional canon in survey research about the type of question that can validly be posed in an opinion survey.

In contrast, we regard as much more realistic our more conservative set of assumptions about the information available to those involved in setting party policy. At any stage of political competition, what all agents are assumed to know is what has been revealed about the system up until that point. At any given time point, this comprises information on past positions and support levels of all parties, obtainable from published opinion poll findings on levels of party support and/or the results of the most recent election. While position-selection algorithms may of course develop conjectures about the implications of potential future policy positions, conditional on the past history of the system, there is no way these conjectures can be tested, other than by making some policy move and then observing the consequences.

Features of our ABM test-bed that differ from earlier ABMs of party competition

Knowledge of pre-entered decision rules and anticipation of others

An infinite number of different policy-selection algorithms could be designed for the setting under investigation; the success of any given algorithm depends crucially on the nature of the other algorithms against which it must compete. Thus in the iterated PD for example, *Tit-for-Tat* is far from optimal if every other agent uses *Cooperate Unconditionally*, to which *Defect Unconditionally* is a much better response. To a large extent, much of classical game theory is about unraveling, in a given setting, the logic of the assumption that every agent chooses an optimal decision rule, conditional on his/her preferences and beliefs, while forecasting that all other agents will also chose an optimal decision rule, conditional on *their* preferences and beliefs. Computer tournaments, in contrast, set out to identify decision rules that are optimal within any given set, explicitly *not* assuming that all agents with the same preferences and beliefs use the same rule. Rather, they assume that relatively more effective decision rules will emerge from a Darwinian process of success-conditioned natural selection.

It is this very significant that we explicitly stated that the four decision rules defined and investigated in Laver (2005) would be pre-entered in the tournament. The designer of any new policy selection rule was thus aware that this would be pitted at the very least against *Sticker*, *Aggregator*, *Hunter* and *Predator*, and could take account of this when designing a competitor rule. In addition, rule designers knew that the four pre-entered rules were disqualified from winning, and that many other rival rules, of unknown (but somewhat foreseeable) content, would be entered by other rule designers. Given this, a more successful policy-selection rule would be one that would be likely to be relatively "robust" to the other types of rule it might encounter. Indeed one submitted rule (*Genety*) used a series of simulations running a genetic algorithm to optimize its parameter

settings against not only the four pre-submitted rules, but also against nine other hypothetical rules, taken to represent types of rule that *Genety* might be expected to encounter in the competition.

Diverse algorithm set / algorithms competing against themselves

Almost all investigations of ABMs of policy-driven party competition have considered competition between a set of parties using the same decision rule, although Laver (2005) looked at limited examples of competition between pairs of rules, including *Hunter-Predator*, *Hunter-Sticker*, and *Aggregator-Predator* competition. Thus one reason, as we have seen, why *Predator* was notably unsuccessful was that, in systems with two or more *Predators*, these tend to attack each other. *Predator* is notably more successful in a system with a lone *Predator* but then, if *Predator* is indeed successful, it must surely reproduce. Thus there remains huge unrealized potential for ABMs to investigate the evolution of party systems involving competition between agents using different decision rules.

One implication of the ten percent survival threshold in our test bed is that no more than ten, and more typically eight or nine, parties are in competition at any given simulation period. Given the set of 29 algorithms in the tournament, random selection of algorithms from this set for newborn parties, and an average of nine parties in competition at any one time with an average of one party dying each election, the probability that a randomly drawn decision rule differs from every surviving rule is $(28/29)^8$, or about 0.75. In other words, given the number of different rules entered into the tournament and the random selection mechanism, any given decision rule would typically not be competing with itself, reducing a relative disadvantage of rules, such as *Predator*, that do badly in competition against themselves. However, we also note that this situation is quite different for more successful rules. As we shall see below a highly successful rule, especially in a setting where the reproduction probability of a rule evolves as a function of its past success, becomes much more likely to find itself competing against other agents using the same rule.

Survival threshold

The ten percent survival threshold was one of the declared exogenous features of our tournament environment. In contrast with the rules investigated by Laver (2005), which faced no survival threshold, it is not surprising that a number of the submitted decision rules conditioned explicitly on this. Indeed some submissions were almost entirely focused on "staying alive", on keeping their agent above the survival threshold. The existence of a *de facto* survival threshold thus introduces a completely new criterion for policy-selection by political parties. In addition to vote-maximizing at any single election, policy-selection rules are also rewarded for keeping their agent alive over a

series of elections, as opposed to selecting policy positions that increase support at one election but also increase the risk of subsequent death. While essential to the research design of our tournament, it also seems to us to be entirely reasonable substantively to assume that real party systems have some kind of *de facto* survival threshold, with parties ceasing to exist in their current form if they fall below this threshold for some specified period. As we have already noted, furthermore, if this were not the case then enabling the birth of new parties would inevitably result in systems with an ever-growing number of parties.

The 20-period electoral process

The ABM models of KMP (1992, 1998, 2003) involved defined electoral campaign periods while that of Laver (2005) did not, modeling a continuous underlying process of party competition. In our tournament, as noted, elections are held and scores tallied every 20 periods. The intentions behind this were, first, to allow new parties time to adapt away from potentially unfavorable random birth locations and, second, to admit algorithms making use of information gathered during an inter-electoral period. A number of submitted decision rules, notably *Fisher*, made use of this opportunity.[8] The distinction between electoral and inter-electoral cycles in the process of party competition seems to us to be another important substantive problem that dynamic ABMs bring into sharp relief. Our test-bed models an extreme situation in which party leaders can choose any action at all, without cost, at an inter-electoral period; the moment of reckoning comes only at an election period. Inter-electoral maneuvering can thus be designed to explore the system with the intention of optimizing payoffs at an election. In any real process of party competition, it is clear that the election-day cycle is indeed critical in terms of party payoffs; but it is also clear that inter-electoral maneuvering is not costless. We are not aware of systematic theoretical work that addresses this matter, which becomes highly significant once we move our characterization of party competition into a dynamic setting.

3. A PORTFOLIO OF POLICY-SELECTION RULES FOR POLITICAL PARTIES

As we have seen, 29 different decision rules, including the four pre-submitted rules, were entered into the tournament. Entries are described in Appendix B. Here we attempt a broad-brush characterization and discussion of the most striking features of the rules that were submitted.

[8] Note that, while making use of inter-election published opinion polls allows party behavior that in one sense looks similar to hill-climbing, the difference with the KMP hill-climbing algorithms is that inter-electoral explorations in our ABM must assume all other parties may be in continuous motion; they thus do not involve parameter sweeping using counter-factual polling that assumes all other parties stand still.

Center-seeking

While explicit use of the center of the distribution of supporter ideal points was *not* permitted under our information restrictions, several submitted rules, notably *Median-voter-seeker*, were designed to find ways to move towards this point. One way of approximating this, for example, is to find the dimension-by-dimension median of party positions at the previous period, or the vote-weighted centroid of these, information for calculating which is available from previous published locations and support levels. There is, however, no reason to believe from investigations of previous ABMs that very central locations will maximize party support. Indeed a characteristic finding of previous work is that they do not. This tendency is considerably exacerbated if several competing algorithms all deploy center-seeking behaviors at the same time. *A priori*, therefore, we did not expect decision rules based solely on center-seeking to perform well, and this indeed proved to be the case.

Tweaking *Hunter*

A number of submissions set out to refine *Hunter*, the most successful rule investigated in previous published work. One refinement, *Niche-Hunter*, involved having a *Hunter* move, if punished with static or declining vote share, towards the most recently successful adjacent party, rather than make a random-reverse move. Another, *Raptor*, involved giving *Hunter* a very distinctive step size so that one *Raptor* could recognize another, ensuring that two *Raptors* never attack each other while otherwise behaving like *Hunters*.

Tweaking *Predator*

Despite its lack of previous success in published simulation results, several entries set out to refine the *Predator* rule. While this might have been due to anticipations of the low probability of rules competing against each other given a large number of entries, it may also reflect the influence of Downsian theoretical results implying party convergence (Downs 1957). One refinement of *Predator*, *Niche-Predator*, had the predator moving towards the largest *adjacent* party – potentially alleviating the problem that unreconstructed *Predators* always converge on precisely the same point in the policy space.

Tweaking *Sticker*

Since *Sticker* – do nothing – is a completely unresponsive "baseline" decision rule it might seem hard to refine. However one submission, *Pick-and-Stick*, visits 19 random locations during the first interelectoral period and then "sticks" at the location that yielded the most votes when visited. In this context, note also that *Sticker* provides a critical benchmark for our tournament. A fundamental

criterion for evaluating any policy-selection rule is that it should perform systematically better than a randomly selected policy position. In the event, as we shall see below, a striking feature of our findings is that fewer than half of the submitted rules managed to achieve even this.

Inter-electoral exploration

The most explicit use of the 19-period inter-electoral period was made by *Fisher* – which systematically explores the adjacent policy space in the first half of the period, revisiting and refining promising locations in the second half of the period before selecting a final policy position for the election in period 20. As already noted, inter-electoral moves were (unrealistically) cost-free in our tournament. We will see below, however, that a striking feature of our findings is the emergence of an *endogenous* penalty for position-selection rules that change their parties' positions "too much".

Parasites

Not unrelated to *Predator* but with a far less flattering public image, a number of submitted decision rules were effectively parasites – indeed one was explicitly named *Parasite*. This rule, which was characteristic of the genre, stays put for the 19 inter-electoral cycles and then goes directly to a location very close to that of the party with the highest support in the period immediately preceding the election. All of the parasite rules submitted were "multi-host" parasites, not designed for one particular species of host. While the distinction between multi- and single-host parasites is unlikely to be significant in the context of the present tournament, it is likely to become much more relevant in the evolutionary environment to which we will be turning (see below), where different species of host might develop different types of defense against the same parasite, with obvious potential effects on the evolutionary stability of parasite rules (Gandon 2004).

The impact of parasites on party policy positions, as with real parasites, is likely to be complex. Parasite parties are unlikely ever to win more votes than any other party since, even if there is only one parasite in the system, it is likely to attract about half of the support of the largest host party, on average splitting the host's support down the middle. With more than one parasite party in contention, parasites will be forced to share the support levels of their hosts. However, parasites do systematically punish other parties for being successful, and as we shall see this did happen in our tournament. This could have a significant impact on the outcome of party competition in a system where parasites are common. Imagine a position-selection rule that is perfect in every other respect but takes no account of parasites; other things equal it always wins every election. In a competition with a set of parties including parasites, however, "success"

paradoxically almost guarantees failure. None of the submitted non-parasite rules anticipated parasites. None of the submitted parasite rules, however, anticipated the possibility of different species of parasite; most were not even designed to deal with other agents using the same parasite rule.

Satisficing and staying-alive

A number of submissions focused on *satisficing* by staying above the system's survival threshold, or some other internally specified threshold, rather than *maximizing* votes. A clear-cut example is *KQ-Strat*, which makes tiny random moves when above the survival threshold and only explores the space for a better location after it has fallen below the threshold for three or more periods. Several other algorithms, for example *Shuffle*, deploy different behaviors, depending on whether the party is over or under the threshold. This is not strictly speaking satisficing, but rather a more general rule feature that involves conditioning decision making on some reward threshold.

In the present context, there are several different rationales for satisficing rules. The first involves a substantive behavioral assumption, prefaced by an obligatory citation of the work of Herbert Simon, that real people satisfice rather than maximize. We find two different sub-rationales within this tradition. One is a psychological assumption about the motivations of real humans, that what they actually want is "enough", rather than "as much as possible", of some payoff. (Some people do stop looking for perishable food when they are full to bursting point.) The other has to do with decision-making in a complex and/or low-information environment, even by agents who seek to maximize payoffs. Such (Simonesque) satisficing involves selecting from the set of actions that can feasibly be evaluated, in the knowledge that this is not the universe of all possible actions. "Win-stay/lose-shift" decision rules have been shown to be effective in such settings (Nowack and Sigmund 1993). A win-stay/lose-shift rule maintains the same behavior if satisfied and changes behavior if dissatisfied, though note that maintaining the same behavior does not imply standing still. In the context of party competition, Bendor, Mookherjee and Ray model a dynamic two-party incumbent-challenger setting in which the winning party (incumbent) uses an "ain't broke, don't fix it" rule and sticks at its current policy position, while the loser (challenger) explores ways to increase its support (Bendor, Mookherjee and Ray forthcoming). In contrast, Laver's *Hunter* rule, which works in multiparty settings, continues to adapt policy in the same direction as long as this is rewarded with increased support levels, but switches to random local exploration when it has not been rewarded. In the latter case, maintaining the same behavior does not mean standing still, but moving in the same direction.

Dynamic, and especially evolutionary, models generate yet another rationale for satisficing. In a long-run dynamic setting, a decision rule designed to maximize payoffs in any single period may not maximize these over the long run, whereas a rule designed to satisfice at any given period may conceivably maximize payoffs over the long run. Our tournament highlights at least two ways in which this can happen. First, maximizing rules may well be "riskier" than satisficing rules, so that when they are good they are very, very good but when they are bad they are horrid. In long-run dynamic systems with a survival threshold, rules designed to maximize at every period may well generate higher variance payoffs than more "conservative" rules that satisfice in some way over the short run. A short-run maximizer may have higher long-run expectations than other rules if the survival threshold is ignored. Given a survival threshold, however, higher variance payoffs may imply lower survival probabilities for a short-run maximizer, and thus lower long-run expectations. Thus the need to stay above a survival threshold can mean that long-run maximizing implies short-run satisficing. A second reason to satisfice rather than maximize in a dynamic setting has to do with competitive multi-rule environments that include parasites. Since parasites systematically punish decision rules that would otherwise have been the most successful, a rule designed to keep its agent above a certain performance threshold, rather than to maximize in the short run (even in a world with perfect information and no survival threshold), might well yield greater long-run payoffs if it avoids the attention of parasites. Putting all of this together, the distinction between satisficing and maximizing blurs considerably in a long run dynamic setting

Cannibals and secret handshakes

Controversially, two submitted rules were programmed to ensure that an agent using the rule did not "attack" other agents using the same rule. Each used a distinctive and unusual step size that identified it secretly to other parties using the same algorithm. As noted, *Raptor* distinguished itself from *Hunter* in this way (with a step size of 0.178). A party using *KQ-Strat* jitters with tiny but distinctive random moves (step size 1.46e-4) when it is satisfied with its position, allowing other *KQ-Strat* parties to see it and avoid attacking it. The extent to which it is valid to allow decision rules to be hard-wired not to attack other agents using the same rule is another matter of considerable, and to our knowledge unexplored, interest in the substantive context of party competition. Two separate issues arise. The first is non-cannibalism *per se*, which may or may not be publicly declared. The second is the deployment of a "secret handshake" protocol which allows only other agents using the same rule to identify each other and condition behavior on this.

The presence of *cannibalistic decision rules* can have complex effects in dynamic evolutionary settings, especially in a setting such as ours in which one rule species is in competition

for finite energy resources (votes in our case) with other quite different species – Claessen et al provide a useful brief review (Claessen, Roos and Persson 2003). The prey provides energy for the same-species predator; cannibalism contributes to (highly non-linear) population control for the rule species (in our setting a decision rule for a party – *Predator* is a good example – may be more effective, the fewer other agents use the same rule); interaction of these effects can generate a bi-stable system. Thus it is not at all clear to us, *a priori*, that a decision rule hard-wired not to attack other agents using the same rule will necessarily prosper in any possible evolutionary setting.

Secret handshake protocols have recently become the subject of a burgeoning literature generated by computer scientists interested in public key cryptography – Balfanz et al. provide a very helpful formal definition (Balfanz, Durfee, Shankar et al. 2003). By the standards of this literature, the *Raptor* and *KQ* "secret" handshakes are extraordinarily primitive. They can be easily decoded by attackers analyzing agent behavior – though this would be a computationally intensive task in the context of our tournament, it would be trivial for a motivated attacker. They are irretrievably compromised by a defector from the group; and they cannot expel suspect group members. In short, no sane person would entrust credit card details to a software agent using a handshake such as those deployed by *Raptor* and *KQ*. Nonetheless, no other submitted rule anticipated the deployment of secret handshake protocols and thus none made any attempt to decode these. More striking, indeed, and despite the pre-announcement of the four Laver rules, no submitted rule made any attempt to infer the type of rule being used by any other agent (beyond the secret handshake device of finding other agents using the same rule.)

In substantive terms, we regard it as unrealistic to model a world of party competition in which agents using the same decision rule are hard-wired not to attack each other. Thus we do not regard it as particularly realistic to model a world in which two parties deploying *Hunter* (or any other rule) do not attack each other simply because they are both *Hunters*. Recall however that, for practical purposes in the context our own tournament, given random selection (with replacement) of decision-rules for new-born parties from a set of 29 rules, and given a typical number of eight or nine parties in the system at any one time, any particular rule will typically not be competing against itself so the issues of cannibalism and secret handshakes will typically not arise. However in a more evolutionary setting, converging on a situation in which only a small number of successful rules have a high probability of reproduction so that agents using the same rule have a much higher probability of interacting, these issues may well become critical. Since, as we shall see, the tournament was won by a rule with anti-cannibal programming and a secret handshake, we also report the results of a complete rerun of our tournament in which we edited the code of the rules

using a secret handshake, disabling this so that any attempt to infer the rule type of other agents had to be based on public information..

Overall

It will readily be seen from the diversity of the rules sketched above, and the range of intriguing issues raised by these, that an important feature of any successful decision rule must be the robustness of its performance when pitted against a diverse portfolio of different types of rule. We consider this to be entirely realistic in the context of a substantive account of party competition and to be a crucial component in the evaluation of any decision rule for political parties. The situation is complicated by the existence of parasites that exploit successful rules deployed by other parties, and of rules hard-wired not to attack themselves. Indeed both parasite and "non-cannibal" rules explicitly assume a range of different decision rules, and are degenerate when deployed only against themselves.

While the winning rule was the one that gained more votes than any other over the entire suite of simulations, in the next section we also evaluate rules in terms of the number of votes per party-using-rule. This will be a more interesting quantity, seen as a fitness parameter, when we move to an evolutionary model of party rule selection. We also evaluate rules in terms of the average life-span, in elections, of parties using each rule. As it happens, the same submitted rule was unambiguously the most successful, whichever criterion is applied.

4. SUCCESSFUL POSITION-SELECTION RULES

Statistical convergence of simulation results on a single winning outcome

Particularly given the large number of different decision rules under review, it is clearly very important to be sure that we ran "enough" simulations, in the sense that a different statistical inference about the winner would have been very unlikely had we run more simulations. To investigate this we revisited the convergence criteria we established prior to the tournament (see Section 2) and conducted Brooks-Gelman tests for each total vote outcome for each party for each set of five chains starting with each of the 29 rules in the tournament. Recall that these tests indicate the point when output from all chains is indistinguishable. They are based on a comparison of within-chain and between-chain variances, and are similar to a classical analysis of variance. Brooks and Gelman (1997) recommend potential scale reduction factors less than 1.2 and every one of our 841 test statistics was significantly less than 1.1 at the 95% confidence level. We also

applied Heidelberger convergence diagnostics, which indicated the chains had reached a stationary distribution.

The most successful position-selection rule: total votes/rule

Figure 1 shows a box plot of the totals of votes per simulation won by each rule over the entire suite of simulations. We can immediately see that 16 of the 25 position-selection rules entered in our tournament did not beat the static benchmark set by *Sticker*. In short, about two-thirds of the dynamic position-selection rules submitted systematically performed worse than a rule that picked a random policy position and never changed this, a very striking result. However, we can also see that seven of the submitted rules won significantly more votes than the most successful pre-announced rule which, as we had expected, was *Hunter*.

The most successful submitted rule, *KQ-Strat*, submitted by Kevin Quinn, had a very clear lead over all other rules. Furthermore, Figures 2 and 3 show that, had we decided to use as criteria for tournament success either mean votes/party-using-rule or mean elections survived, the results would have been essentially the same. Under any of these criteria, the set of rules beating *Hunter* and the set of rules losing to *Sticker* were the same. The overall winning rule, *KQ-Strat*, was the same under all criteria.

The submitted rules in the set that beat *Hunter,* the most successful pre-announced rule, are a mixed bag. Some condition position-selection on staying above the survival threshold. Others exploit the 19-period inter-election phase when no payoff is awarded. Some are at least partially parasitic, capitalizing on positions set by the most successful rules. Others were programmed to prevent cannibalistic attacks on other agents using the same rule. The winning rule, *KQ-Strat*, incorporated three of these features, being in essence a satisficing parasite hard-wired not to attack itself. *KQ*-parties jittered, using small random moves with a distinctive "secret handshake" step size, when over the threshold. When under the threshold (for three or more periods), a *KQ*-party moved very (very) close to the position of another party over the threshold, provided this was not another *KQ*-party.

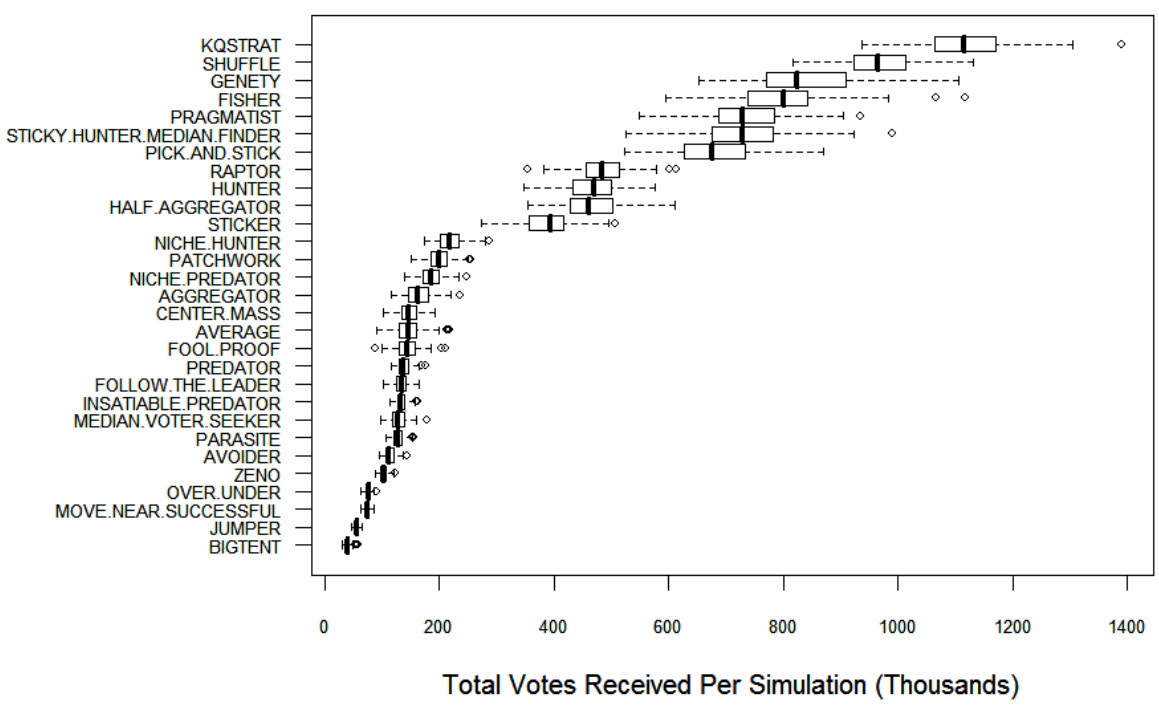

Figure 1: Total votes received, by rule

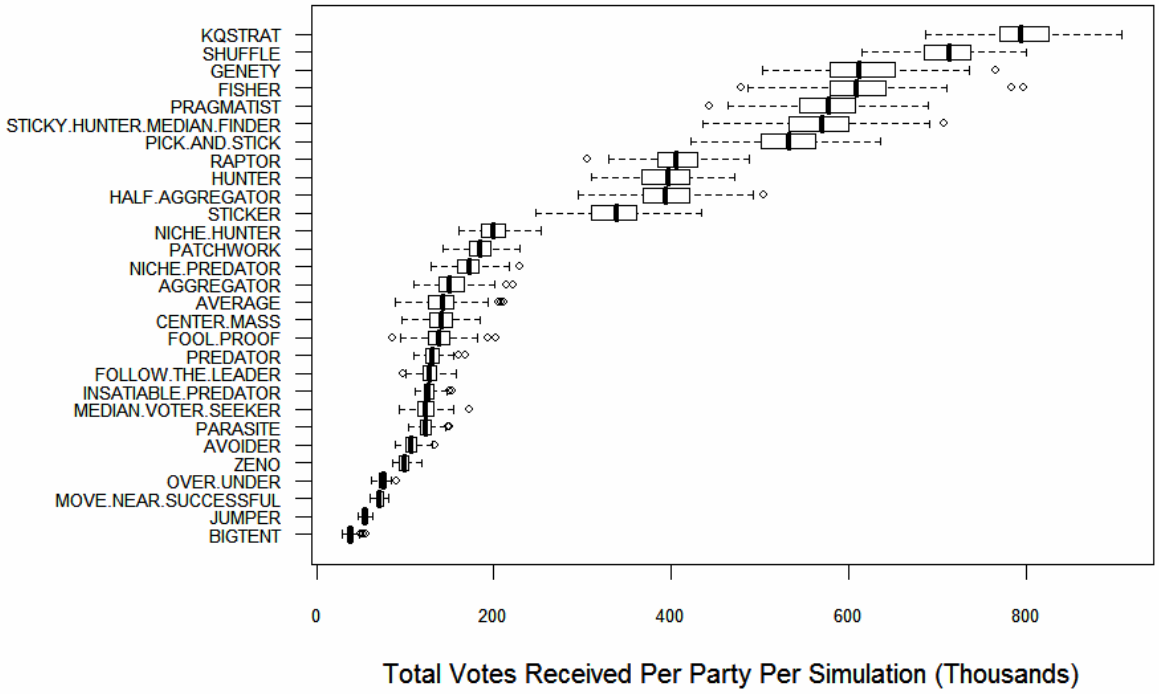

Figure 2: Total voted received, per party-using-rule

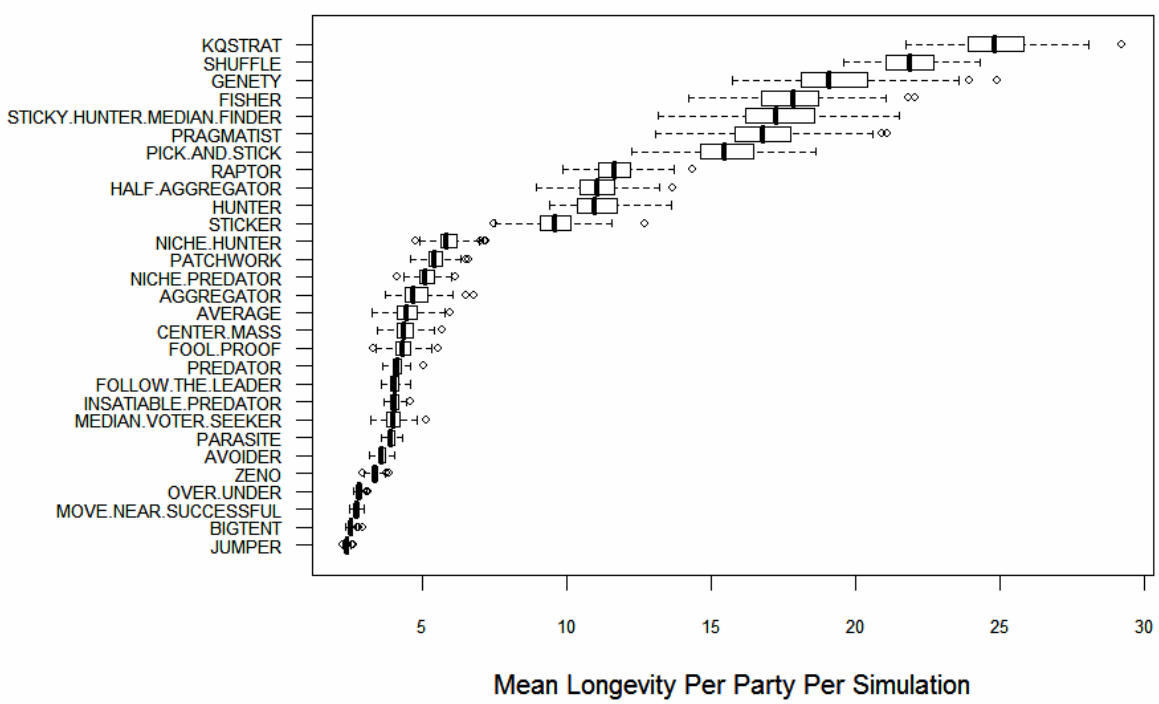

Figure 3: Mean number of elections survived by each rule

However others in the top set of successful rules deployed none of the distinctive features we have noted. Indeed one of the rules that beat *Hunter* was *Pick-and-Stick*, a very simple tweak of *Sticker*. It simply explored random policy positions for each of the first 19 pre-election periods and then became a *Sticker* for all time at the position that had yielded most votes. It is also of interest that the submitted rule that was by far the most computationally intensive, *Bigtent*, was in the event the least successful rule in the entire tournament. (*Bigtent* devoted most of its energy to a sophisticated method for estimating where the "unknown" voter ideal points were located by analyzing past party positions and vote shares. Its problem was that it was killed off before it came close to achieving this.) Note that the rules of our tournament imposed no penalty on decision rules that were computationally expensive. A more realistic dynamic model of party competition might well penalize any rule that takes far longer to decide what to do than its rivals. However, a striking feature of our results is that, even without such a penalty, successful rules were neither computationally intensive nor convoluted in their logic.

Given the large number of spectactularly unsuccessful submissions, measuring success against the static *Sticker* benchmark of maintaining a fixed random policy position, one obvious possibility is that successful submissions were successful precisely because they were typically pitted against ineffective decision rules. This is easily investigated by rerunning the tournament using only the top set of successful rules, specifically the set of seven rules that performed

Table 1: Vote percentages, by rule for run-off simulations using only the top set of submitted rules

Rule	Runoff		Tournament	
	Mean vote	*Median rank*	*Mean vote*	*Median rank*
KQ-Strat	19.6	1	11.2	1
Pick-and-Stick	15.4	2	6.8	6
Sticky hunter	15.0	3	7.3	5
Genety	14.0	4	8.4	4
Pragmatist	13.6	5	7.4	5
Shuffle	11.6	6	9.7	2
Fisher	10.9	7	7.9	4

significantly better in statistical terms than *Hunter*, the most successful pre-submitted rule. Seven more 220,000-period simulation runs were performed, with each of the seven rules in the top set allowed to be the initial party in one simulation. *KQ-Strat* ranked first in every one of these runs, which the Brooks-Gelman test showed had converged statistically. Vote shares in this suite of "run-off" simulations are reported in Table 1, and compared with vote shares for the same rules in the full tournament with 29 different rules.

Table 1 shows that *KQ-Strat* pulled ahead of the competition in the runoff. Note that it is the only decision rule in the top set that is programmed not to attack itself, and that the *a priori* probability of *KQ-Strat not* encountering itself has now dropped from about 0.75 in a setting with 28 other rules to about 0.25 in a setting with six other rules. Thus the "anti-cannibal" programming in *KQ-Strat* is likely to be much more effective in this setting. Of the other rules, *Shuffle* and *Fisher* declined strikingly in relative success, suggesting that their success in the full tournament was at least in part the result of typically being pitted against ineffective rules. Perhaps the most surprising result is the significantly increased success in the run-off environment of the very simple *Pick-and-Stick* rule, which now came second behind *KQ-Strat*. As we will see in the next section, a general emergent feature of our tournament, despite the fact that changing policy position was cost-free, was that successful decision rules did *not* make dramatic changes of policy position.

Since *KQ-Strat's* anti-cannibal and secret handshake programming are controversial and may well be seen as substantively unrealistic, we reran the entire tournament, making the sole change that we edited the *KQ-Strat* and *Raptor* code to disable secret handshakes, and thus to prevent any agent from recognizing another using the same rule. Figure 4 reports the results and

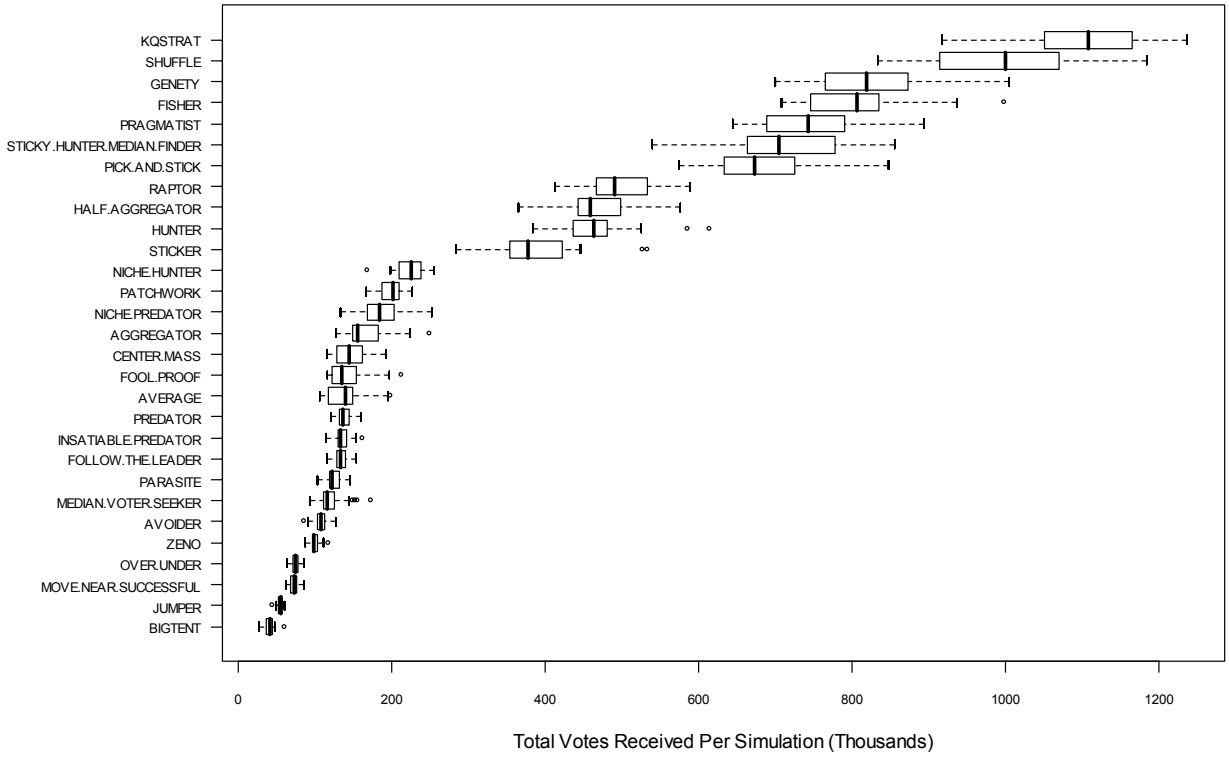

Figure 4: Total votes received, by rule (KQ-Strat secret handshake disabled)

shows that *KQ-Strat* won the tournament even with its secret handshake disabled, although its margin of victory was noticeably reduced.

Which rules did better and why?

It is difficult in such a complex environment to isolate factors that characterize the relative success of different decision rules. However, in spite of the diversity of approaches in the submitted rules, there were some striking regularities. First, there was a "sweet-spot" in the policy space for position-selection rules, which we can characterize in terms of how far from the center of the voter distribution parties tended to locate. Figure 5 plots the mean vote share of each party type in each simulation against its mean *eccentricity* – Euclidean distance from the center of the voter distribution, measured in standard deviations of the voter distribution. This yields 145 observations for each rule entered,[9] and we use a full-bandwidth loess procedure to draw a line representing the mean relationship across all observations. It is immediately apparent that the most successful position-selection rules tend to locate their parties about one standard deviation away from the center of the space. This echoes findings by Laver (2005) that Hunters tend to hunt for votes at

[9] We label the mean location in the figure for several rules, omitting some labels for clarity.

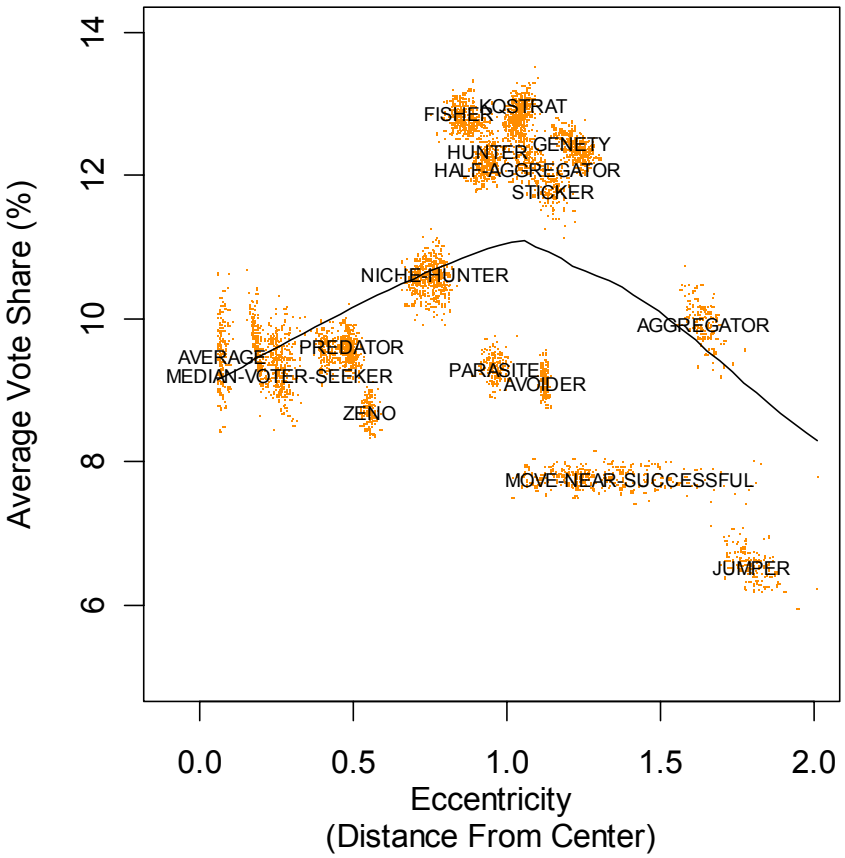

Figure 5: Relationship between average vote share per party and eccentricity

somewhat less that one standard deviation of the voter distribution away from the center, and from benchmarking simulations by Laver and Shilperoord (2007), showing that the mean distance of voters from their closest party tends to be minimized when randomly scattered parties tend to locate about one standard deviation from the center. In contrast, center-seeking parties (like those using *Average*) as well as extreme parties (like those using *Aggregator*) fared much worse.

Second, our simulated system of party competition systematically generates an *endogenous* penalty for parties that tend to change their positions by large amounts from election to election – recall that no such penalty was programmed into the system. Figure 6 plots the average vote share of each party type in each simulation against *motion* – the mean Euclidean distance each party moves from election t to election t+1. The relationship is unambiguous. Decision rules that generate a lot of movement from one election to the next (for example *Parasite*, *Avoider*, and *Move-Near-Successful*) are much less successful than rules that tend to stay put. Recall that that *Pick-and-Stick* performed remarkably well despite the fact that it never adapts at all after the first election. This rule, and other low-motion rules, capitalize on the fact that surviving parties, by

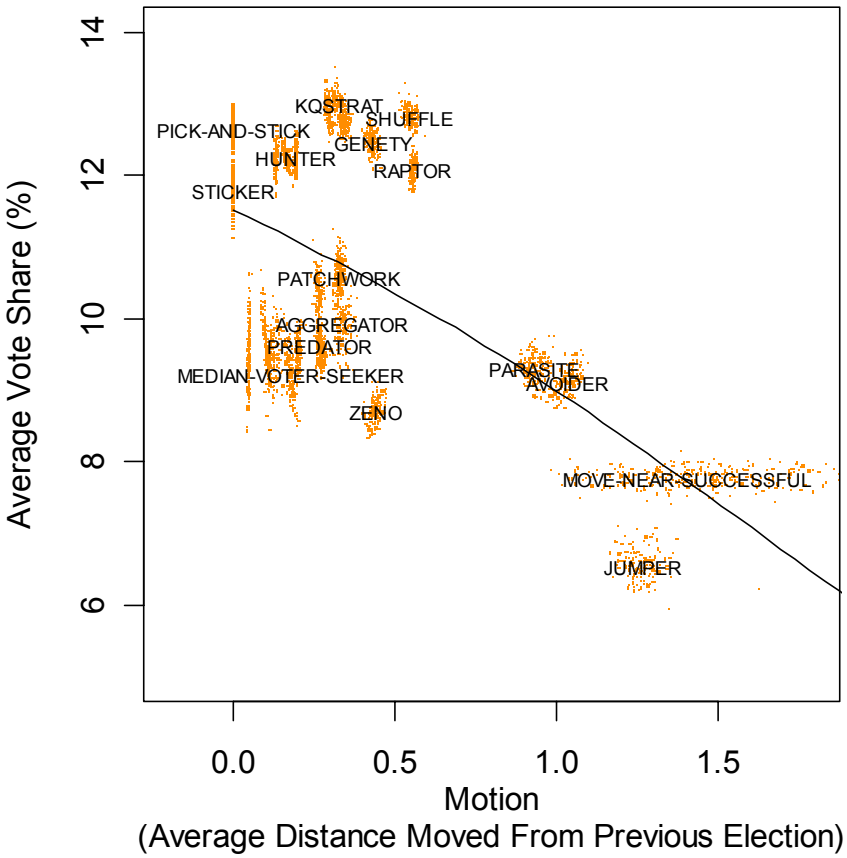

Figure 6: Relationship between average vote share per party and motion

definition, have found a spot in the electoral space in which they can be successful. They face the classic exploration-exploitation tradeoff of reinforcement learning, between exploiting a position that has previously been successful and exploring for a new position that might improve on this, but might also be worse. What is remarkable is how little there is to gain from exploration in this particular setting, given the risks of doing this. Although a few parties fared better than *Pick-and-Stick* in the full tournament, only *KQStrat* performed better in the runoff. And parties using *KQStrat* typically make near-zero adjustments to their position, changing policy position only when facing a shift in electoral conditions that puts them under the survival threshold for more than three inter-electoral periods. This provides a way to understand real-world political parties that are very resistant to changing their policy positions.

Analyzing how well pairs of decision rules performed when pitted against each other, we notice that these exhibit strong clustering by type. Figure 7 shows a visual representation of the set of pairwise correlations between vote shares received by a given party type (vertical axis) and the

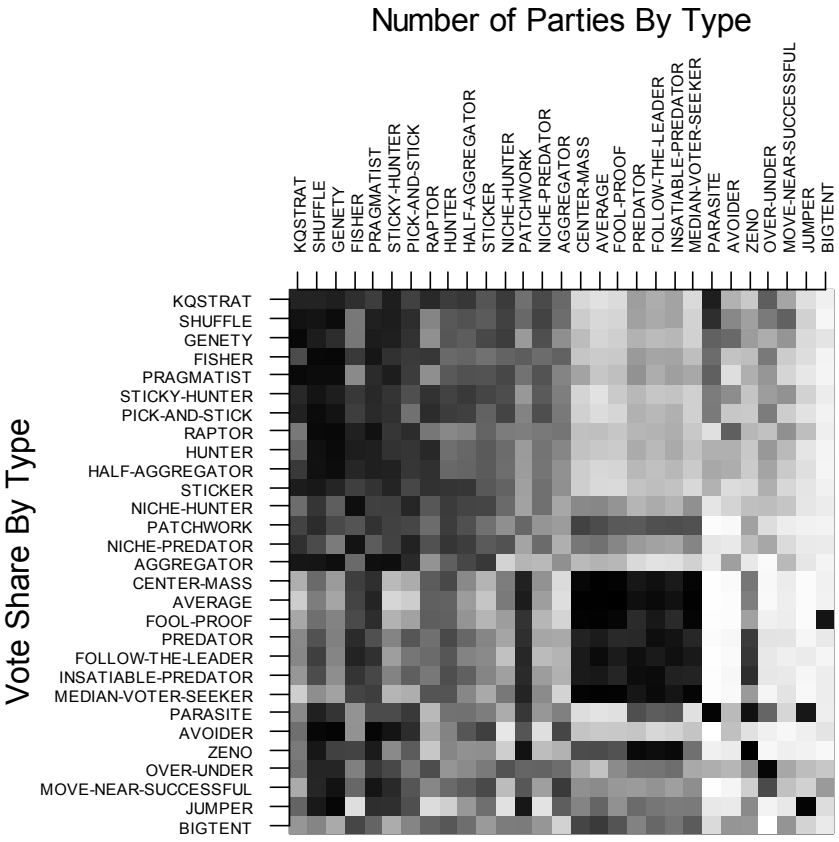

Figure 7: Effect of other party types on average vote share per party

number of parties of a given type (horizontal axis).[10] Lighter shades indicate increasingly positive correlation and darker shades indicate increasingly negative correlation. Parties are ordered by their final performance in the tournament, best to worst. Thus going from left to right of the top row, we see that *KQStrat* performed relatively worse against the more effective rules and relatively better against the less effective rules. We also notice a vertical "spike" for *Parasite* near the top right hand corner of the plot. As we anticipated earlier, while not doing particularly well in the tournament overall, *Parasite* inflicted systematically more damage on the more successful decision rules when pitted against these. Indeed *KQStrat,* the most successful of all the rules, was particularly vulnerable to the presence of *Parasites*. However *Parasite* was also vulnerable to other rules, and most notably to itself. If we see anti-cannibal programming and secret-handshakes as unrealistic when modeling party decision rules, then this will be an inherent weakness in any parasitic decision rule, so that parasite parties, if they exist, are likely to be relatively small. Nonetheless we can see

[10] Each election permits an observation of vote share for each party type that competes in that election and an observation of the number of parties of each type for all parties. The correlation matrix looks very similar if we convert number of parties to a dichotomous variable coded 1 for any positive number of parties and 0 otherwise.

that, if parasites do nonetheless exist in the party system, they are likely to have a considerable effect on the evolution of party competition – in particular by systematically punishing otherwise successful decision rules.

Figure 7 highlights two distinctive blocks of party types. The darker square in the top left hand corner, extending from *KQStrat* to *Aggregator* shows, unsurprisingly, that dynamically successful rules[11] had most difficulty in competition with other parties using dynamically successful rules. The very dark square towards the bottom left, running from *Center-Mass* to *Median-Voter-Seeker*, is generated by rules designed to move either towards the center of the policy space or towards other parties. This was thus a set of rules that tended to compete either with themselves or with other similar rules, and not with the top set of position-selection rules which, as we have seen, tended to locate about one standard deviation away from the center. This emphasizes the point that, while prospecting the center of the voter distribution can be rewarding for a party acting alone or in competition with one other party, as the number of interacting parties increases, the more center seeking parties there are, the worse things get for each of them, in competition with rules that are not center-seeking.

5. "EVOLUTIONARY" COMPETITION BETWEEN RULES

The simulations reported thus far were explicitly designed to provide a "level playing field" for all decision rules entered into the tournament. Of much more substantive interest, however, is an evolutionary environment in which the "reproduction" probabilities of different rules are not constant across the entire set of rules, but evolve dynamically as functions of each rule's past success. These "replicator dynamics" are designed to model a process in which new parties entering the system would be more likely to adopt decision rules that had been more successful in the past. If we make a rule's reproduction rate proportional to its *fitness*, measured in terms of its past success, then how we define the "past success" of each rule becomes critical. In what follows, we measure a rule's instantaneous success at any single point in time as the *mean percentage vote share won by all parties using that rule at that time*. The crucial issue concerns the manner in which success at each previous instant in history is aggregated into some memory of past success – which is then used to condition rule fitness. If the fitness regime has "too long" a memory, then rule fitness may be overly influenced by long-gone outcomes that no longer have relevance to the current state of the system. If it has "too short" a memory, then it may overreact to quite possibly random short-term fluctuations. This is another manifestation of the "exploitation-exploration"

[11] That is, rules more successful than a rule picking a random policy positions and never changing this.

trade-off discussed in the reinforcement learning literature, to which Sutton and Barto provide an excellent introduction (Sutton and Barto 1998). One obvious solution is to discount past success exponentially when computing current rule fitness. This can be computed rather elegantly using a recursive updating algorithm of a type common in the reinforcement learning literature. Let F_{rt} be rule *r*'s fitness at the beginning of period *t*. This is recursively updated as follows by $V_{r(t-1)}$ – vote share/party-using-*r* at the end of period *t-1*:

$$F_{rt} = \alpha F_{r(t-1)} + (1 - \alpha) V_{r(t-1)} \qquad : \qquad 0 \leq \alpha \leq 1$$

Thus α is the memory parameter of the regime under which reproductive fitness evolves. If α = 0 we have a "goldfish memory" regime which conditions fitness only on success in the immediately preceding period. When α = 1 we have a regime in which fitness never updates. Set in this more general context, our tournament implicitly assumed a fitness regime for which α = 1, with priors such that all rules had equal reproduction rates. A fitness regime for which α = 0.5 remains highly reactive, with an agent's fitness level eight periods previously contributing only about one percent of the information in its current fitness. In the evolutionary simulations we report below, we model a fitness regime for which α = 0.9 under which, to get a sense of things, an agent's fitness level 30 periods previously contributes about five percent of the information in its current fitness.

The final issue to consider in this context is the extent to which there are shocks in the evolutionary system, in the sense than random births might occur that are not in accord with the evolving fitness regime. Thus, even though one rule might have been very successful and have a consequently high reproduction rate, while another might have been very unsuccessful and have a reproduction rate that is effectively zero, there may remain some finite probability that the unsuccessful rule will be born into the system. Successful decision rules must continue to prosper against random invaders, even after they have become very fit. In the current setting we model this by setting a probability, π, that a decision rule for new born party at period *t* is selected with a probability proportional to current evolved rule fitness, F_{rt}. With probability (1 – π), the decision rule at birth is selected by setting probabilities equal for all rules in the tournament set. In results we report below, π = *0.9*. In other words, about 90% of the time rules for newborn parties are selected on the basis of evolved rule fitness, while 10% of the time every rule in the tournament has an equal chance of being chosen by a newborn party. The results of this new tournament, including all 29

Table 2: Vote percentages, for evolutionary simulations using success-updated rule fitness

Rule	Success-updated fitness $\pi = 0.9$; $\alpha = 0.9$	Original tournament $\pi = 0.0$; $\alpha = 1$
KQ-strat	17.0	11.2
Genety	13.9	8.4
Sticky-hunter	11.7	7.3
Shuffle	11.2	9.7
Pick-and-stick	10.4	6.8
Pragmatist	9.6	7.4
Fisher	8.5	7.9
Raptor	4.1	4.9
Hunter	3.7	4.7
Half-aggregator	2.6	4.7
Sticker	1.2	3.9

rules from the original tournament and run in an evolutionary setting in which $\pi = 0.9$ and $\alpha = 0.9$, are reported in Table 2.[12]

Table 2 shows that, in an environment under which rule reproduction probabilities evolve as a function of past success, the gap between successful and unsuccessful rules widened, as might be expected. The winning rule, *KQ-strat,* pulled ahead of the competition. Each rule for which results are *not* reported in Table 2 won on average less than one percent of the vote. Note the anti-cannibal programming in *KQ-strat* could well have contributed to its enhanced success in a more evolutionary setting. *KQ*'s *a priori* probability of meeting another *KQ*-party in the original tournament, given a reproduction probability of 1/29 (=0.03), was about 0.25. However, as *KQ*'s reproduction probability converges on 0.17 in the evolutionary setting, the probability of one *KQ*-party meeting another balloons to about 0.77.

Table 2 offers another very important finding. Successful as it was, *KQ-strat* did not come even close to driving out all of the other decision rules that were submitted to our tournament. Setting $\pi = 0.9$ meant that each submitted rule, no matter how miserable its performance, had a 1/29 chance of being selected for use by on average one-tenth of all newborn parties. The long tail of rules winning on average less than one-percent of the vote can no doubt be explained by this feature of the evolutionary environment and it seem very likely that these rules would have been

[12] This evolutionary tournament involved 29 separate runs, each of 200,000 periods after discarding a 20,000-period burn-in, each run forcing-in one of the submitted strategies as the one to be selected in the very first election. The total number of recorded periods involved in this simulation is thus 14,500,000. Brooks-Gelman tests confirmed statistical convergence of the 29 chains.

driven out of the system completely had we set $\pi = 1$ and thereby conditioned birth probabilities entirely on past success. What is very striking, however, is that each of the top set of seven rules prospered in this evolutionary setting. Thus the evolutionary simulations converged on a situation in which elements in the diverse top set of rules coexist with one another. We also note that each rule in this top set exploited a different feature it its decision-making environment. For us, the evolutionary stability of this diverse set of rules for setting party policy positions is one of the most significant results of our work to date.

6. CONCLUSIONS AND FUTURE WORK

The evolutionary simulations we described in the previous section implement simple replicator dynamics, making reproductive fitness a function of past success and assuming that all "children" are perfect clones of their "parent" rules. Obviously, this setting will never allow completely new decision rules to evolve—results are determined by the identity of a portfolio of rules that is exogenously set at the start of the process. The purpose of these simulations is thus to investigate the evolved relative fitness of each of the rules in the starting portfolio, and in particular to look for situations in which one decision rule drives out all others, or in which there are ergodic states involving combinations of successful rules within the starting portfolio. This environment is clearly less intriguing and suggestive than one in which the actual content of party decision rules is not fixed exogenously, but itself evolves endogenously. We can model such an environment using the *genetic algorithm*, although implementing the genetic algorithm in the substantive context of position-selection rules for political parties presents robust challenges.

Consider the classic genetic operators of *mutation* and *crossover* and begin by thinking about an evolutionary setting in which party decision rules reproduce *asexually*, with mutation but no crossover. This in effect involves a small but vital modification to the replicator dynamics we described in the previous section. Reproductive fitness remains a function of past success; however, "child" strategies are no longer bound to be perfect clones of their "parents", but may be subject to random mutations at the point of reproduction. Substantively, this models a situation in which new-born parties select a decision rule based on their observations of the past success of decision rules used by other parties, but that random mutations (which we might think of in a political context as innovations) may be made to particular rules. Many mutations which will be abject failures but some may be unexpected successes in an evolutionary setting driven by replicator dynamics.

The challenge in implementing such a research design is to produce a full parameterization of each of the rules in the portfolio, which can serve as the rule's "genetic material" that becomes subject to random mutation. Take *Hunter*, for example. *Hunter* is in effect not so much a single rule, but a *species* of decision rule, within which there are many possible breeds. The breed of *Hunter* entered in the tournament had an unconditional step size of 0.05 (where the metric is defined in standard deviations of the distribution of supporter ideal points). Thus, for breeds of *Hunter* with unconditional step sizes, step size is an obvious parameter. If punished with lower support, the breed of *Hunter* entered into the tournament reverses direction (adds 180 degrees to its heading) and chooses a random heading from the arc +/- 90 degrees from the direction it is now facing. Thus there are two more parameters, one specifying the number of degrees to be added to the heading if punished and one specifying the width of the arc (if this is symmetrical) within which a new heading is now sampled. Other parameters might determine the memory regime under which past success is measured, the extent to which step size is conditioned on the level of reward and punishment, and so on. Careful deconstruction of the *Hunter* rule would yield a general characterization of *Hunter*. This characterization would yield a parameter string that would in some sense be a genome for the *Hunter* species of rules. A particular set of parameter values would define particular settings for a type of *Hunter*. This string of parameter values could then be expressed as an array of binary numbers, with particular parameter values occupying particular places in the array. When a particular type of *Hunter* was due to be replicated under the replicator dynamics outlined in the previous section, there would be a finite though tiny probability that each of the binary numbers in this array would be inverted, creating a "new" type of mutant strain of *Hunter*, defined by new parameter values. The ongoing replicator dynamics would then determine whether the mutant strain was successful, and thus reproduced and prospered, or was a failure, and thus died out. Note that this evolutionary setting would allow us to model the evolution of the fittest *breed* of "super-*Hunter*", or of any other fully parameterized decision rule. No new rule *species* would evolve.

Continuing our focus on asexual reproduction, a difficult issue even to think about concerns whether it is conceivable in some sense to develop a general characterization, and hence parameterization, the entire universe of rules for setting party policy positions in *n*-space. If this were to prove possible, then the evolutionary process that we could model would transcend the starting portfolio of "rule species" (each species characterized by a distinctive set of parameters), to allow for the evolution of completely new rule species that no analyst had ever though of. While we seem fairly far from being able to do this at present, this seems to us to be an extraordinarily attractive goal for future work.

Now consider the genetic operator of *crossover*, the essence of sexual reproduction, under which the genetic material of the child is some combination of the genetic material of each parent. We might first want to think about whether there is a substantive story about politics, whereby party leaders not only have interactions that are the political equivalent of sex but that the result of this interaction is a new set of decision rules, each of which is on some sense a combination of the decision rules of its parents. It seems to us that such a characterization is neither impossible nor implausible. Once achieved and subjected to the genetic algorithm, it would enable us to model situations where, for example, next-generation party decision rules would blend entire chunks of code from successful party decision rules that had been among their ancestors.

Returning, finally, to the outcome of our existing tournament, we believe that this already contains some "big" news for those who are interested in modeling policy-driven party competition in a dynamic setting. First, recall that a number of the most successful rules focused on satisficing, keeping their party above some *de facto* survival threshold that must inevitably be an intrinsic feature of any dynamic system that enables the birth and death of political parties. This is something quite new that arises when we move from a static to a dynamic environment since, on many substantively plausible criteria, a party can be more successful if it satisfices with low variance rewards over the long term rather than maximizes with high variance rewards, and thus a higher risk of extinction, over the shorter term.

The top set of successful position-selection rules also included some that made effective use of the inter-election period to look for policy positions that would win votes at election time. Our simulation test-bed was almost certainly unrealistic by making such exploration costless. It is also important to note, however, that while our simulations contain no exogenous penalty for changing policy positions, a strong emergent pattern of party competition in our system was that decision rules generating relatively large changes in policy positions tended strongly to fare much worse than rules generating relatively small changes in positions. Thus, without explicitly punishing excessive policy movement in our simulated party system, excessive policy movement was nonetheless punished endogenously as part of the emergent pattern of party competition.

A third headline result from our tournament, rerun in its more evolutionary setting, is that no single decision rule in the starting portfolio drove out all others, with the result that a stable top set of different rules emerged to coexist with each other, each successful rule tending to exploit a different feature of the competitive environment. The results of this tournament have thus strengthened our conviction that further investigations of position selection rules in models of party competition should be set in the context of how each rule performs against a heterogeneous rule set, rather than looking at the dynamics of single-rule systems.

A single tournament such as the one we have described is clearly only a single step down a long path of evaluating the relative effectiveness of different decision rules in an evolving system of multi-party competition. To a large extent, our conclusions depend upon the set of decision rules that happened to be submitted and we could not conceivably come to robust general conclusions about the relative effectiveness of different decision rules in an evolutionary setting, on the basis of our work so far. In our view there are two main ways forward. One is to continue with "empirical" experiments involving further tournaments like the one we describe here. If we were to re-run the same tournament, and it is our firm intention to do this, rule designers will be able to analyze the results of the first tournament and attempt to improve rule design on the basis of these findings. This process could be iterated many times. The other way forward is to develop the more general parametrizations of party decision rules that we discussed in the first part of these conclusions. Given such parameterizations, a comprehensive and systematic suite of simulations using the genetic algorithm, over the very long run, offers the hope of being able to explore the space of potential party decision rules to find rules, and rule combinations, that evolve as being evolutionarily "fit", in the sense of being better than other rules at attracting the votes of ordinary decent citizens.

APPENDIX A: TOURNAMENT RULES

OVERVIEW

In the spirit of Robert Axelrod's tournament for strategies in the repeat-play prisoner's dilemma game, we announce a tournament for party strategies in a dynamic agent-based spatial model of party competition. The winner will receive $1000. We herewith call for submissions of party strategies by 15 April 2006. Each strategy submitted will be pitted against each other strategy in a series of long-running simulations of a two-dimensional multiparty spatial model in which the number of parties is endogenously determined. All parties falling below a certain size threshold for two consecutive elections will "die". One new party will be "born" each election at a random spatial location, using a strategy randomly selected from the portfolio of submitted strategies. The strategy most successful at winning votes over the very long run will be declared the "winner".

THE ENVIRONMENT OF PARTY COMPETITION

The policy space
The policy space will be defined by 1000 voters with two-dimensional ideal points randomly drawn from a standard normal distribution with a mean of zero and a standard deviation of 1 in each dimension (a bivariate normal distribution). Voters never abstain and always vote sincerely for the party specifying the policy position closest to their ideal point.

Periods and elections
Each simulation in the series of simulations (see below) will run for 200,000 periods, after discarding results from 20,000 initial periods ("burn-in"). An "election" will be held every 20 periods. The period immediately after each election will begin with the "birth" of a new party, using a strategy randomly selected from the set of strategies submitted for the tournament. The initial policy position of each newborn party will be determined by a random draw from a normal distribution with a mean of zero and a standard deviation of 1 in both dimensions.

Each period
At the start of each period, each party strategy, subject to the information requirements set out below, must specify a policy position for the party. (The initial policy position in the first period after party birth will be random – see above). In the middle of each period, each voter supports the party specifying the policy position closest (in terms of Euclidean distance) to the voter's ideal point. At the end of each period, the policy positions of each party and the number of voters supporting each party are announced. As with an opinion poll, this announcement provides information to parties that they can use to adapt their position to the current competitive environment. Each party thus has 20 periods to adapt before each election.

Each election
At the end of every twentieth period, an election is held. The number of voters supporting each party is announced and recorded. If a party falls below a survival threshold of 10 percent of votes cast for two consecutive elections, it "dies" and disappears from the competition. Note that more than one party may die in an election, but only one new party is born immediately after each election. After votes have been recorded and party deaths have taken place, a new party is "born" at the beginning of each period following an election. The new party will use a strategy randomly selected, with equal probability, from the set of strategies submitted for the tournament, and will be

given a random initial policy position (see above). (The only exception to this is that the first party selected for the first period of each simulation will be rotated between strategies - see below).

The series of simulations

To determine the winner of the tournament, we will analyze results from 5n simulations, where n is the number of strategies submitted. Each submitted strategy will be identified as the "starting" strategy for five simulations to be sure initial conditions are not influencing the tournament results. The simulation environment is programmed in R. Code for the simulation test-bed is available at http://jhfowler.ucsd.edu/tournament.htm. Each simulation in the series will run for 220,000 periods, with the first 20,000 periods (the "burn-in") discarded from the analysis. In preliminary testing of the simulation environment, using the four "pre-submitted" party strategies (see below), this is about five times as long as was necessary to reach convergence as determined by standard statistical tests. Each submitted strategy will thus start "first" in simulations running for a total of 1,000,000 periods (50,000 elections) after burn-in.

CONSTRAINTS ON PARTY STRATEGIES

Party decision rules must specify a party position for the beginning of each period using only the following information:

1. Any previously-announced policy position and level of voter support for any party.

2. The mean or median position on each dimension of the ideal points of the party's own current supporters.

3. Note that party strategies cannot have access to the ideal point of any individual voter, nor can they interrogate voters about what they would do in hypothetical situations.

4. Note also that this implies that, on being born, a party using any strategy does not know its initial position in the space, and has no data about the state of the system in previous periods.

See http://jhfowler.ucsd.edu/tournament.htm for a table of variables that correspond to this information that have already been programmed in R.

PRE-SUBMITTED STRATEGIES

There will be four "pre-submitted" strategies, not eligible for the prize. These will be the four strategies described in Michael Laver, "Policy and Dynamics of Political Competition", *American Political Science Review*, 99:2 (May 2005) 263-281, viz:

1. STICKER – the party does not move from its original position;

2. AGGREGATOR – the party sets policy at the mean position on each dimension of the ideal points of current party supporters;

3. HUNTER – If party support in the previous period was higher than in the period before that, then the party moves a step of 0.05 in the same direction it moved previously. Otherwise it randomly chooses a direction in the opposite 180 degree half space from its previous move and moves a step of 0.05 in that direction;

4. PREDATOR – the party observes the current sizes and policy positions of all parties. If it is the largest party it stands still, otherwise it moves a step of 0.05 towards the position of the largest party.

RULES FOR SUBMISSION

Each party strategy submitted must be fully specified so it can be programmed. Since the tournament will be run in an environment programmed in R, strategy submissions will be particularly welcome if they are specified in the form of program code snippets in R. However, any strategy submitted in the form of a clearly written and programmable algorithm, or programmable piece of pseudo-code, will be accepted. Please suggest a short and informative name for your party strategy. Only one submission is allowed per person. If several entrants independently submit an identical strategy, then this strategy will be entered only once in the tournament. If this strategy wins the prize, a single winner will be chosen by lot from among those who submitted the winning entry. All entries must be received by 15 April, 2006. Send them to jhfowler@ucsd.edu.

WINNERS

The prize of $1000 will be awarded to the strategy winning the most votes in all elections over the series of 5*n* simulations described above. That is, for every election in every simulation after burn-in, the total votes won by all parties using each strategy will be calculated. The winner of the tournament will be the strategy accumulating the most votes over the entire suite of simulations. Although these strategies will not win prizes, we will also calculate and report the strategy winning the most votes per party using each strategy over the suite of simulations, as well as the strategy used by parties surviving, on average, for the largest number of periods. The prize will be awarded at American Political Science Association's 2006 annual meeting in Philadelphia, PA.

APPENDIX B: TOURNAMENT ENTRIES

ID#	Name	Author	Description
5	Average	Grynaviski, Jeff*	Adopt the weighted average of all parties' locations, where the weights are given by the number of votes given to the party in the most recent round.
6	Avoider	Thomas Plümper* Martin, Christian	Given the calculated weighted mean of party distributions $\overline{x}, \overline{y}$, let our party take the following position $x = \overline{x} + a \cdot b$ $y = \overline{y} + \sqrt{(1-a)} \cdot c$ where *a* is drawn from a uniform distribution of the interval [-1,1] and *b* and *c* take the value -1 or +1 with equal probability.
7	Bigtent	Curran, John P Daryl Posnett	The algorithm consists in two parts. The first forms an estimate of voter distributions, and the second searches for a party position that is optimal subject to certain constraints. To estimate the voter distribution, a bounding box is determined that

			contains all current and historical party positions, including those of extinct parties. The box is partitioned into a coarse grid, and the current position for each active party is mapped to the nearest grid point. Thus, each grid point is assigned a fraction of total votes based on the most recent poll. Each grid point maintains a finite-horizon moving average of the vote fraction assigned to it. These historical averages are used as the estimate of the true voter distribution in the search stage. In the search stage, the algorithm tries to find a position within the bounding box that will give its party the largest fraction of votes, assuming (i) that the voter distribution is given by the estimate calculated in the first part of the algorithm, and (ii) the other parties do not move from their most recently revealed positions. Given a candidate position for its party, the algorithm computes what fraction of votes the party will receive if (i) and (ii) hold. The candidate position is then revised using a standard optimization routine. (The submitted version uses R's optim routine with the BFGS option, which is a modified Newton's method.) Ideally, several initial points are tried within the bounding box, due to the possibility of local maxima. Because finding the optimum is computationally expensive in the context of the competition, however, the length of the search has an upper limit that probably stops short of finding a local maximum.
8	Center-mass	Zimmerman, Matt*	Move one-twentieth of the distance towards the weighted average of the parties each period.
9	**Fisher**	Mayer, Alex*	A party using the fisher strategy fishes by taking steps at near right angles to its last step (random angles within 45 degrees of a right angle). At each position the level of voter support is recorded. Overview: When a party is born with the fisher strategy, it stores its position and level of voter support, then moves to a policy position calculated to be approximately the mean voter position. From this position, it fishes throughout the policy space taking large steps. During the 11th period (just before 3/5 of the total number of periods in an election cycle - election frequency [efreq] = 20 periods), the party identifies the 4 (efreq/5) policy positions that yielded the most voter support during the initial fishing periods. During this period and the next 3 periods (total of efreq/5), the party revisits those positions and records voter support at each. During the 15th period (just before 4/5 of efreq), the party returns to the top position among the 4 re-visited positions. If that position yields voter support below the threshold for survival, the party returns to the second-ranked position from among the 4 re-visited positions. Once the party has chosen the first or second ranked position, it fishes with very small steps (5% of the step size at birth), starting from that position. During the 19th period (the period immediately before the election), the party moves to the position that produced the most voter support during the last 4 periods (these include at least 2 fishing periods). If the party's vote total in an election satisfies the threshold for survival, then the party reduces the size of steps it takes to test the policy space to 20% of the size it used at birth, staying relatively near the successful position. If the party's voter support at an election drops below the threshold, it instead uses the original step size used at birth until it finds a position with support that satisfies the threshold.
10	Follow-the-Leader	Kasdin, Stuart* Glasgow, Garrett	The first period after being born (the second period in the game), calculate the shortest Euclidian distance between the two leading vote gainers from the previous period. Move OWNPARTY to the point midway along the shortest line in Euclidian space between these two leading vote gainers.

			After each period, move OWNPARTY 0.05 toward the party with the highest level of support in the most recent period parallel to the axis on which the distance between it and OWNPARTY is greatest. If this increases vote share from the previous period, move again next period 0.05 along the same axis in the same direction. Do not move on the other dimension. When OWNPARTY no longer gains electoral support after a period ends, in the next period move 0.05 along the other axis, so move on x if OWNPARTY's last move was on y, or y if OWNPARTY's last move was on x. If OWNPARTY is the strongest party do not move.
11	Fool-proof	Adams, Jim*	Shift to the weighted mean position of all the parties' positions. At each period t, employ the following two-step procedure to determine the policy position: First, compute the average policy shift of all rival parties at the previous period, compared to the period before that, with these shifts not weighted by the parties' vote shares. Thus at time t, for instance, this calculation might yield the conclusion that between periods t-2 and t-1 the rival parties shifted on average .05 units to the left along the horizontal axis and .15 units downward along the vertical axis, or whatever. Second, the party's decision rule at time t is to shift its position in the same direction as the rival parties shifted between periods t-2 and t-1, but the focal party's shift will be of only one fifth the magnitude of the average shift of the rival parties. So for instance in the above example, where the rival parties shifted on average .05 units to the left along the horizontal axis and .15 units downward along the vertical axis between period t-2 and t-1, the focal party at time t would shift .01 units to the left on the horizontal axis and .03 units downward along the vertical axis.
12	**Genety**	Larson, Eric* Cook, Tyson	Weighs three factors to choose where to move: 1) location relative to the origin, 2) aggregation, and 3) hunting. Party calculates the vector toward the origin, the vector toward the mean position of party supporters, and the vector emerging from applying the pre-submitted hunting algorithm, and moves according to a weighted sum of these vectors. The static weights applied to these vectors were determined by applying a genetic algorithm, and were optimized in competition with: 1-4) The four pre-submitted strategies 5) A brownian motion strategy 6) A random bivariate normal draw strategy 7) The hole-finding algorithm (developed by another Kalamazoo College student, Joel Haas) 8-11) Four different genetic algorithm parties 12) A circling algorithm that stayed outside the radius of the farthest party from the origin 13) A party that oscillated inside the r=1 circle but outside the r=0.8 circle Each generation of the genetic algorithm had approximately 100 candidate strategies, run against each other in randomized order in sets of four as above.
13	Half-aggregator	Williams, Daniel*	Arbitrarily select one dimension, call it X. With respect to dimension X, the party selects a position that is at the mean ideal point of its current party supporters. After the first round, dimension X remains fixed. Call the other dimension, dimension Y. With respect to dimension Y, if party support in the previous period was higher than in the period before that, then the party moves a step of 0.05 in the same direction it moved previously, otherwise it moves 0.05 steps away.

14	Insatiable-predator	Nyblade, Ben*	The basic strategy is identical to that of Predator, but Insatiable Predator never stays in one spot. The only change to the code is what Predator does when it is the largest party. Instead of staying put like Predator, the Insatiable Predator moves .05 towards the nearest other party when it is the largest party.
15	Jumper	Smirnov, Oleg*	Party policy is located on the straight line formed by the locations of the two parties with the highest vote totals on the previous round. Regardless of whether or not it is the largest party, Jumper occupies a point on this line such that the former largest party is in between the jumper and the 2nd place party. The distance between the former largest party and the jumper is 0.5 of the distance between the winner and the 2nd place party.
16	**KQ-strat**	Quinn, Kevin*	If alive for 3 iterations and votes < pthresh for 3 iterations or more and haven't moved more than 1.46e-4 units in 3 iterations or more then set new position equal to p + e, where p is the position of a randomly chosen party that did not previously move exactly 1.46e-4 units and whose current vote total is greater than pthresh and e is a N(0, .0625^2) random variable. Otherwise move 1.46e-4 units in random direction.
17	Median-voter-seeker	Gartner, Scott	The party locates the dimension-by-dimension median of the policy space, defined by the location and vote share of the parties in the previous period. It moves a step of 0.05 in that direction.
18	Move-near-successful (aka Satisficing Explorer)	Zick, Ken*	"First employ a satisficing rule: if my vote total was at or above the election threshold (e.g. 10%) on either of the last two periods (periods, not elections), then maintain the current position. Otherwise, explore the space, as follows: 1) Create a list of all parties that are at or above the election threshold on the current period 2) Randomly choose one party location from this list 3) Jump to a location that is 0.3 units away from the chosen party location. The angle between my new location and the chosen party location is random."
19	Niche-hunter	Money, Jeannette*	If party support in the previous period was higher than in the period before that, then the party moves a step of 0.05 in the same direction it moved previously. Otherwise, it examines the party support of each ADJACENT party in the two prior periods (that is, it compares the electoral support at t - 1 with the electoral support at t - 2). It moves a step of 0.05 in the direction of the ADJACENT party with the largest gains (smallest losses) in party support. To identify the adjacent party in a two-dimensional space, generate a four quadrant grid centered on the NICHE HUNTER party's policy position. Unless a quadrant contains no parties, there will be one adjacent party in each quadrant, which is the party closest to the NICHE HUNTER in Euclidian distance. Using this definition, the NICHE HUNTER may have 1-4 "adjacent" parties. If the "adjacent" party with the largest electoral gains is less than 0.05 units away from the NICHE HUNTER, then the NICHE HUNTER moves one half of the distance between itself and the "adjacent" party.
20	Niche-predator	Andrews, Jo*	The party observes the current sizes and policy positions of all ADJACENT parties. If it is the largest party it stands still; otherwise, it moves a step of 0.05 toward the position of the largest ADJACENT party.

			If the largest "adjacent" party is less than 0.05 units away from the NICHE PREDATOR, then the NICHE PREDATOR moves one half of the distance between itself and the "adjacent" party. To identify the adjacent party in a two-dimensional space, generate a four quadrant grid centered on the NICHE PREDATOR party's policy position. Unless a quadrant contains no parties, there will be one adjacent party in each quadrant, which is the party closest to the NICHE PREDATOR in Euclidian distance. Using this definition, the NICHE PREDATOR may have 1-4 "adjacent" parties.
21	Over-under	Lo, James*	Go to (x_i+10, y_i+10) until election period, where x_i and y_i are the coordinates of the position of the highest vote getter in the previous election. In an election period given random perturbation e~U(0, 0.05), go to (x_i+e, y_i+e) or (x_i-e,y_i-e), alternating between going just "above" and just "below".
22	Parasite	Schleicher, David*	For 19 periods, the policy position reflects my initial position. For the 20th period (the election period), I observe the largest party and move to a position that is .01 away from the largest party's position in a random direction.
23	Patchwork	Guarnieri, Fernando	As predator, identifies largest party and moves towards it, then becomes an aggregator if it is the largest party.
24	**Pick-and-stick**	Schulz, Evan	In the periods before the first election, locate party at random points in the space. Then return to the point at which it received most votes and stay there for subsequent elections.
25	**Pragmatist**	Calvo, Ernesto F*	Combination of the vote-weighted mean location of all parties and the party's own median voter. A normally distributed noise term prevents parties with the same strategy from overlapping with each other. $ploc_i = \alpha\left(\sum_j v_j\, ploc_j + e\right) + (1-\alpha)^*\, pmloc_i$ where α=.2, 0≥v_j≥1, and e~N(0,.2).
26	Raptor	Jones, David Hugh*	The strategy is based on the pre-submitted Hunter rule but with two differences. 1. Each turn, the party moves a distance of 0.178 rather than 0.05 2. Each turn, the party calculates the distance all other parties have moved. If any of these other parties has also moved a distance of 0.178, is now less than 0.3 away from the first party, and is weakly to the right of the first party, then the party behaves as if its votes had decreased, ie moves randomly backwards in the opposite 180 degree half space to its previous direction.
27	**Shuffle**	Takeuchi, Kan*	This behaves like Aggregator if it won 11.5% or more votes in previous round, Hunter if won 11.5%-8.0%, or it relocates itself into the least populated area as follows: Divide the space into 4 quadrants around the center of the gravity, which is derived from ploc and ptotals, and find one in which the votes per party is the highest; then jump to any point there randomly using rnorm function.
28	**Sticky-hunter-**	Bramson, Aaron	If party receives at least 10% last election, then use Sticky Hunter, else Median Finder.

	median-finder		The Sticky Hunter Strategy: If the party collected less than 10% of the votes in both of the last two periods then it steps according to the following rule. If its last step improved its percentage of votes then step forward a distance of 0.135. If the last step did not improve its vote percentage then pick a new direction from the first and last third of the opposite 180 degrees (i.e. cutting out the middle third that includes a total U-turn) and make a 0.135 step. If the agent did collect at least 10% of the votes last turn then do nothing (i.e. stick). The Median Finder: Each period, determine the median position of my supporters, and move to that location.
29	Zeno	McGovern, Geoff*	Like PREDATOR, "the party observes the current sizes and policy positions of all parties. If it is the largest party it stands still." Otherwise, ZENO will move half the distance between itself and the largest party in the direction of the largest party.

* = author approved/supplied description of the impact of submitted code

(Du, Faber and Gunzburger 1999)

REFERENCES

Axelrod, R. (1980a). "Effective choice in the prisoner's dilemma." Journal of Conflict Resolution **24**: 3-25.

Axelrod, R. (1980b). "More effective choice in the prisoner's dilemma." Journal of Conflict Resolution **24**: 379-403.

Axelrod, R. (1997). The evolution of strategies in the iterated prisoner's dilemma. The complexity of cooperation: agent-based models of competition and collaboration. R. Axelrod. Princeton, Princeton University Press**:** 14-29.

Balfanz, D., G. Durfee, N. Shankar, et al. (2003). "Secret handshakes from pairing-based key agreements." Proceedings of the 2003 IEEE Symposium on Security and Privacy (SP '03).

Bendor, J., D. Mookherjee and D. Ray (forthcoming). "Satisficing and selection in electoral competition." Quarterly Journal of Political Science.

Besley, T. and S. Coate (1997). "An Economic Model of Representative Democracy." Quarterly Journal of Economics **112**(1): 85-106.

Brooks, S. P. and A. Gelman (1998). "General methods for monitoring convergence of iterative simulations." Journal of Computational and Graphical Statistics **7**: 434-455.

Claessen, D., A. M. d. Roos and L. Persson (2003). "Population dynamic theory of size-dependent cannibalism." Proceedings of the Royal Society B **271**: 333-340.

De Marchi, S. (1999). "Adaptive models and electoral instability." Journal of Theoretical Politics **11**(July): 393-419.

De Marchi, S. (2003). A computational model of voter sophistication, ideology and candidate position-taking. Computational models in political economy. K. Kollman, J. H. Miller and S. E. Page. Cambridge, Mass., MIT Press.

Du, Q., V. Faber and M. Gunzburger (1999). "Centroidal Voronoi tessellations: applications and algorithms." Society for Industrial and Applied Mathematics Review **41**(4): 637-676.

Fowler, J. H. and O. Smirnov (2005). "Dynamic parties and social turnout: an agent-based model." American Journal of Sociology **110** (4): 1070–1094.

Gandon, S. (2004). "Evolution of multihost parasites." Evolution **58**(3): 455-469.

Kollman, K., J. Miller and S. Page (1992). "Adaptive parties in spatial elections." American Political Science Review **86**(December): 929-937.

Kollman, K., J. Miller and S. Page (1998). "Political parties and electoral landscapes." British Journal of Political Science **28** (January): 139-158.

Kollman, K., J. H. Miller and S. E. Page (2003). Computational models in political economy. Cambridge, Mass., MIT Press.

Kollman, K., J. H. Miller and S. E. Page (2003). Political institutions and sorting in a Tiebout model. Computational models in political economy. K. Kollman, J. H. Miller and S. E. Page. Cambridge, Mass., MIT Press**:** 187-212.

Laver, M. (2005). "Policy and the dynamics of political competition." American Political Science Review **99**(2): 263-281.

Laver, M. and M. Schilperoord (forthcoming 2007). "Spatial models of political competition with endogenous parties." Philosophical Transactions of the Royal Society B.

Nowak, M. and K. Sigmund (1993). "A strategy of win-stay, lose-shift that outperforms tit-for-tat in the Prisoners' Dilemma game." Nature **364**: 56–58.

Osborne, M. J. and A. Slivinski (1996). "A model of competition with citizen candidates." Quarterly Journal of Economics **111**(1): 65-96.

Sutton, R. S. and A. G. Barto (1998). Reinforcement Learning: An Introduction. Cambridge, MA, MIT Press.