Learning Agents for Iterative Voting

Stéphane Airiau,[1] Umberto Grandi[2] and Filipo Studzinski Perotto[2]

[1] Université Paris-Dauphine, PSL Research University, LAMSADE
stephane.airiau@dauphine.fr
[2] IRIT, University of Toulouse
umberto.grandi@irit.fr, filipo.perotto@gmail.com

Abstract. This paper assesses the learning capabilities of agents in a situation of collective choice. Each agent is endowed with a private preference concerning a number of alternative candidates, and participates in an iterated plurality election. Agents get rewards depending on the winner of each election, and adjust their voting strategy using reinforcement learning. By conducting extensive simulations, we show that our agents are capable of learning how to take decisions at the level of well-known voting procedures, and that these decisions maintain good choice-theoretic properties when increasing the number of agents or candidates.

Keywords: Computational social choice, Iterative voting, Bandit algorithms

1 Introduction

In a situation of collective choice, we say that an agent is voting strategically, or that she is manipulating, when the agent does not submit her sincere view to the voting system in order to obtain a collective result that she prefers to the one that would be obtained had she voted sincerely. A classical result in social choice theory showed that all sensible voting rules are susceptible to strategic voting [6, 16]. In fact, strategic voting may be exploited to make better decisions in several situations where, for instance, agents are confronted with a sequence of repeated elections, from where an interesting compromising candidate can be elected.

The **plurality rule**, aka. first-past-the-post, selects the alternatives that have been voted for by the highest number of agents. Its computation is quick and its communication costs very low, but in view of its simplicity it suffers from numerous problems. For example, it is possible that the plurality winner would lose in pairwise comparison against all other alternatives[3]. If however the plurality rule is used in an iterative fashion, staging sequential elections in which at each point in time one of the voters is allowed to manipulate, then it constitutes an effective tool for selecting an outcome at equilibrium with good properties [11]. This setting is known as **iterative voting**, and several recent papers explored its convergence using different voting rules, and assessed the quality of the winner [9, 15, 7, 10, 12, 14].

Most works in this field suffer from two main drawbacks. First, agents are highly myopic in not taking into account the history of their interactions, and in having an

[3] See, e.g., [13] for a proof.

horizon for strategic thinking of one single iteration. It creates an artificial asymmetry between the available knowledge and the strategic behavior. Second, to ensure convergence it is required that agents manipulate **one at a time**, a property that is difficult to enforce.

In this paper, we tackle both aspects by studying a **concurrent manipulation process** in which agents have the capability of **learning from their past interaction**. In our setting, iterative voting is seen as a repeated game in which voters use reinforcement learning to cast their ballots. We limit the information available to the learning agents to only the winner of each iteration step (when classic iterative voting methods require more information). Our goal is to show that multiagent learning can be a solution in the context of iterative voting, even when the information available to agents is severly limited: a learning agent bases her decisions on the history of past interactions, and because of the learning rate, the choice of the vote is not purely myopic. In addition, in our model all the learning agents are allowed to change their ballot at the same time. Such possibility cannot be given to classic iterative voting methods because there may not be convergence to a single winner [11]. The question we ask in this paper is whether learning can help making a good collective decision [17]: do we observe convergence, and is the winner good according to choice-theoretic criteria?

We show experimentally that our learning agents are able to learn how to make collective decisions under standard measures of decision quality, such as the Condorcet efficiency and the Borda score. The **contribution** of this paper is twofold: we show that iterative learning 1) outdo all iterative voting methods using less information, and 2) is comparable to a well-known procedure called single transferable vote.

The paper is organized as follows. Section 2 provides the basic definitions and reviews the literature on iterative voting and multiagent reinforcement learning. Section 3 presents the specifics of our simulation setting, and Section 4 discusses the obtained results. Section 5 concludes the paper.

2 Basic Definitions and Related Works

We now provide all definitions that are needed for the construction of our setting. We introduce the basics of iterative voting and of multiagent reinforcement learning.

2.1 Voting Rules

Let C be a finite set of m candidates or alternatives and N be a finite set of n agents. Based on their preferences, individuals in N need to make a decision on which alternative in C to choose. Agents are typically assumed to have preferences over candidates in C in the form of a *linear order*, i.e., a transitive, anti-symmetric and complete binary relation over C. We denote with $>_i$ the preference of agent i and with $\boldsymbol{P} = (>_1, \ldots, >_n)$ the profile listing all individual preferences. Hence, we write $b >_i a$ to denote that agent i prefers candidate b to candidate a. A (non-resolute) *voting rule* is a function w that associates with every profile $\boldsymbol{P}$ a non-empty subset of winning candidates $w(\boldsymbol{P}) \in 2^C \setminus \emptyset$. The simplest voting rule, and the one involving as little communication as possible among the agents, is the **plurality** rule: each agent votes for a single candidate, and the

candidates with the highest number of votes win. The **Borda** rule is a scoring rule in which a candidate c is given $m-j$ points for each voter that is ranking c in j-th position. The score of a candidate is the sum of her points over all voters. For the **Copeland** rule, the score of a candidate c is the number of pairwise comparisons she wins (i.e., contests between c and another candidate a such that there is a majority of voters preferring c to a) minus the number of pairwise comparisons she loses. For Borda and Copeland, the candidates with the highest score win. Finally, **Single Transferable Vote (STV)** can be viewed as an iterative process: at the first round the candidate that is ranked first by the fewest number of voters gets eliminated (ties are broken following a predetermined order). Votes initially given to the eliminated candidate are then transferred to the candidate that comes immediately after in the individual preferences. This process is iterated until one alternative is ranked first by a majority of voters.

2.2 Iterative Voting

Agents face the choice of submitting their truthful ballot, i.e., a ballot corresponding to their individual preferences, or to vote strategically. In iterative voting, agents start from a voting situation: they fill their ballots and a winner is announced. Then, **one at a time**, an agent changes her ballot, and a new winner is announced, creating a sequence of ballot profiles and consequent winners.

Iterative voting is guaranteed to converge for the plurality rule with linear tie-breaking [11] when agents know the score of each candidate at each round, though for most other voting rules convergence cannot be guaranteed [9]. Restricted dynamics, defined by limiting the possible actions available to players, have therefore been studied to guarantee convergence [15, 7, 12]. In this paper we focus on iterative voting with the plurality rule and on the following two strategies for individual manipulation:

Best response: at time t, given the plurality score for each candidate c at time $t-1$, the voter computes her best response(s) and votes for one;

3-Pragmatists [15]: at time t, given the top three plurality candidates at time $t-1$, the voter manipulates in favour of her preferred candidate amongst them.

Convergence with 3-pragmatists manipulation is guaranteed by the fact that the set of 3 most-voted candidates is not changed by every manipulation step, hence each agent will manipulate the election only once.

Two main critiques have been raised to the setting defined above. First, one agent at a time can change her ballot and the new winner is announced before another agent can change her ballot: this sequential aspect is unrealistic but is key to guarantee the convergence of iterative voting. As was already observed by [11], there is no convergence if individuals are allowed to move at the same time. Second, individuals are highly myopic, since their strategic horizon only considers one-step forward in the iterative process and they do not make use of the history of previous manipulations by other agents when making their next choice.

2.3 Multiagent Learning

Multiagent reinforcement learning has been used both in cooperative domains (where the set of agents share the same goal) and in non-cooperative ones (where each agent

is trying to optimize its own personal utility). For cooperative domains, the key issue is that learners obtain a local/personal reward but need to learn a decision that is good for the society of agents. For example, agents that try to optimise air-traffic [1] care about individual preferences as well as the global traffic. In this paper, agents are not concerned about the quality of the outcome for the entire population: each voter would like one of her favourite candidates to win. We are in a non-cooperative setting similar to the one of learning in games: the actions are the different ballots and agents have preferences over the joint actions (i.e. voters have preferences over the candidates). One key difference is that preferences are typically ordinal in voting whereas they are cardinal for games (see Section 3.1 describing how we generate cardinal utilities from ordinal ones).

In this paper, we use a basic multiarmed bandit style reinforcement learning algorithm [18] for testing whether agents can learn to make a collective decision, experimenting with different exploration strategies (see Section 3 for a detailed description). Many reinforcement learning algorithms have been used for playing normal form games, e.g. joint-action learning [5], gradient-based algorithms such as IGA-WoLF or WoLF-PHC [4], to name a few. Since no algorithm can be claimed to be best, we focus on showing that the most basic learning algorithm is able to perform well. For a similar reason we also choose that agents will only get to observe the current winner, and no other information is available to them, such as the score of all candidates (as done in standard iterative voting).

2.4 Evaluation Criteria

Because there is no consensus on the quality of a collective outcome, we will study the results on multiple criteria. Given a profile of preferences $(>_1, \ldots, >_n)$, a *Condorcet winner (CW)* is a candidate that beats every other candidate in pairwise comparisons. A CW is not guaranteed to exists, but a first parameter in assessing a voting rule is the percentage of profiles in which it elects a CW when there exists one:

Condorcet efficiency: the ratio of profiles where a CW is elected out of all profiles where a CW exists.

Many voting rules are designed to elect a CW whenever it exists, such as the Copeland rule, which hence have a Condorcet efficiency of 1. Other voting rules, such as plurality, Borda and STV may elect a candidate that is not a Condorcet winner. Related work estimates the Condorcet efficiency of plurality and Borda for large electorates using Monte Carlo simulations [8].

A second parameter that can be used to measure the quality of the winner is the Borda score itself:

Borda Score: a candidate c is given $m - j$ points for each agent ranking c in j-th position in her truthful preference

The Borda score provides a good measure of how the rule compromises between top-ranked candidates and candidates ranked lower in the individual preferences. One interpretation of the Borda score is that it estimates the average rank of candidates, and

the Borda winner is the candidate with the highest average rank over all candidates. Obviously, the Borda rule is the best rule according to this criterion.When varying the number of voters or candidates, we measure the ratio between the Borda score of the elected winner and the maximal Borda score that can be obtained, i.e., if $B(c)$ is the Borda score of a candidate c then $BR(c) = B(c)/\max_{a \in C}(B(a))$.

3 Learning and Simulation Setting

We now describe the settings of our simulations. Each simulation is defined by the parameters $m = |C|$ (the number of candidates), $n = |N|$ (the number of voters), T as the number of iterations, or repeated elections, the agents dispose to learn. We use iterative voting with plurality rule and lexicographic tie-breaking. Note that the choice of the tie-breaking method has been shown to be an important factor in guaranteeing the convergence of iterative voting rules [9]. We also performed experiments with a randomised tie-breaking rule, obtaining comparable results.

3.1 Preferences and Utilities

While voting is based on ordinal information, reinforcement learning needs cardinal utility. Hence the need to translate a preference order $>_i$ of agent i into a utility function $u_i : C \to \mathbb{R}$. Given an ordering $>$, let $pos(c)$ be candidate c's position, where position 0 is taken by the most preferred candidate, and $|C| - 1$ by the least preferred. We considered three possibilities:

Linear utilities: $u_i^{lin}(c) = 1 - \frac{pos(c)}{|C|-1}$;
Exponential utilities: $u_i^{exp}(c) = \frac{1}{2^{pos(c)}}$;
Logistic utilities: $u_i^{sig}(c) = 1 - \frac{1}{1+e^{-k(pos(c) - \frac{|C|-1}{2})}}$,

The parameter k controls the steepness of the curve in the last definition. These three different methods represent distinct satisfaction contexts. Linear utilities correspond to the Borda values, meaning that the satisfaction with a given candidate decreases linearly following the preference order. Exponential utilities are a more realistic representation, especially in large domains where alternatives at the top bear more importance than those at the bottom. They can also be used to simulate partial orders, since the alternatives below a certain threshold of utility count as non-ranked. In this case, the voters have precise choices, and the satisfaction decreases quickly as soon as the winner is not the preferred candidate. In contrast, logistic utilities decrease slowly in a neighbourhood of the top preferred candidates.

3.2 Profiles Generation

Our experiments are averaged over 10.000 preference profiles generated using the following two distributions:

Impartial culture assumption (IC): linear orders are drawn uniformly at random.

Urn model with correlation $\alpha \in [0, 1)$: The preference order of the first voter is drawn with uniform probability among all possible linear orders that are present in an urn. A number of copies of the first drawn preference is then put into the urn depending on the parameter α (more precisely, $\frac{m!}{(\frac{1}{\alpha}-1)}$), and the preference of the second voter is then drawn. The process is repeated until all n preference orders have been selected.

The urn model is also known as the Polya-Eggenberger model [3]. The interest of such scheme is to have some correlation between voters, where some observed preference is more likely to be observed again. The higher the correlation parameter α the more likely it is that a Condorcet winner exists, and the less likely it is that a single voter can change the winner of the plurality rule with a single manipulation. Note that IC is equivalent to the Urn model with $\alpha = 0$.

We developed a generator for the urn model that avoids the manipulation of all permutations over the set of candidates, based on the following intuition. At first the probability of creating a random preference order is equal to 1, and the list of defined voters is empty. At each iteration a new preference is generated, either as a copy of a previous preference, or by generating a random preference. The probability of using a random preference decreases with the number of preference orders generated. The method is described in Algorithm 1, the function *CreateRandomPreference()* returns a sample from the uniform distribution over all possible linear orders, and *CopyPreferenceFrom(N)* picks a voter uniformly at random from N, returning a copy of her preferences.

Algorithm 1 Efficient Urn Model Generator (α, C, n)

/* α is correlation, C is the set of candidates, n is the size of the profile */
$N \leftarrow \emptyset$ // preference profile
$\beta \leftarrow \frac{1}{\alpha} - 1$
for i from 0 to $n - 1$ **do**
  $v \leftarrow \begin{cases} \textit{CreateRandomPreference()}, & \text{with prob. } \frac{\beta}{\beta+\lvert N\rvert} \\ \textit{CopyPreferenceFrom(N)}, & \text{otherwise} \end{cases}$
  $N \leftarrow N \cup \{v\}$
end for

3.3 Learning Algorithm

Since the voting rule is plurality and only the winner is announced at each iteration, the learner has to decide, at each iteration, the candidate she will vote for. From the point of view of each voter, the proposed setting correspond to the well-known multiarmed bandit problem (MAB), a wide studied case of computational reinforcement learning (RL). A MAB is equivalent to a Markovian Decision Process (MDP) with a single state [18, 2].

The learning mechanism must evaluate the utility of each possible action during the sequence of interactions, only based on the feedback suggested by the reward. In our

case, the reward is the preference value of the elected candidate (the winner) for the agent. The agent then learns a function $Q : C \to \mathbb{R}^+$ that estimates the expected utility of voting for each candidate. Once a voter i has voted for candidate c and knows the winner w, it can compute its reward r (the elected winner is w and we have $r = u_i(w)$), and from there, i updates its Q value using the following update rule:

$$Q(c) \leftarrow \beta r + (1 - \beta) \cdot Q(c).$$

where α is the learning rate used to control the impact of new information: when $\beta = 0$, the new information is not used, when $\beta = 1$, only the new information matters. Initially, we fix $\beta = 0.1$.

For controlling the exploration, we have several choices (e.g. using ϵ-greedy, softmax exploration schemes or UCB [2]). We experimented several mechanisms, and the one described below obtained the best results. We use a simple implementation of the "optimism in face of uncertainty" principle for exploration: the voter always picks the candidate with the highest Q-value. To ensure exploration, the Q-values are initialized with a copy of the preference values of the agent, and in case of ties the agent prefers to chose the action that has been tested the least. Thus, a voter is most likely to vote for candidates she prefers at the beginning. If a different candidate wins the iteration, the bad reward decreases the Q-value, and the agent has incentives to try other candidates in the coming iterations.

Both *ϵ-greedy* and *softmax* algorithms make the agents very explorative at the beginning, and when an agent explores, sub-optimal actions are chosen (even the least preferred ones), disturbing the learning progress of all the agents. The same happens with classic version of the *optimistic-greedy method*, where all the utilities are equally initialized to the maximal reward possible. The *UCB* method is also very conservative, the exploration is made sufficient to guarantee near optimal performance in stationary MAB problems. As the environment is not stationary (agents are concurrently learning), exploration is no longer adequate and UCB performs poorly.

4 Simulation Results

We now present the main results, showing that a society of agents provided with very simple learning capabilities can make a "good" collective decision, comparable to that taken by well-known voting rules and often better than what standard iterative voting would recommend. In all results we present next, we use the urn model with $\alpha = 0.1$ for generating the preferences. With the exception of Section 4.3, all experiments in this section use exponential utilities, but the results for linear and logistic utilities are similar and not presented here.

4.1 Learning Dynamics

By considering the result of iterated plurality with learning agents as a voting rule *per se* (recall that in our setting iterated plurality is guaranteed to converge), we are able to evaluate its performance in social-choice-theoretic terms, measuring both its Condorcet

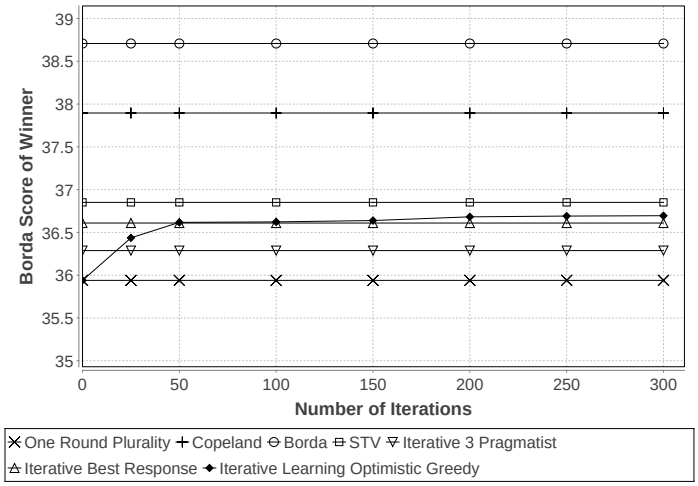

Fig. 1. Performance of learning agents in terms of Borda score of the winner (9 voters, 7 candidates). The number of profiles which actually have a Condorcet winner is 9359 profiles.

efficiency and the Borda score of the winner at the end of the iteration. In Figures 1 and 2 we plot the progress of the first parameter depending on the number of learning iterations that is allowed (a similar figure can be obtained for the second parameter). In order to interpret our findings, we also plot the Condorcet efficiency (CE) and Borda score of one-round plurality, best-response iterative voting, 3-pragmatists iterative voting, STV, Copeland and Borda. We present the results in one scenario with a population of 9 voters and 7 candidates. Utilities of voters are generated using exponential utilities and the results averaged over 10,000 elections.

First, let us consider the Borda score as evaluation criteria in Figure 1. By design, the Borda voting rule is best. The averaged rank of the winner over all the elections is about 1.7. The next best mechanisms are Copeland and STV. All these voting rules require the knowledge of the complete ordinal preference of the agents. We observe that iterative voting comes next, either in the standard mechanism or our new mechanism with learning agent. Note that our learning agents are using less information as they only know the current winner whereas standard iterative voting uses the plurality score of all candidates. The level of performance is still quite acceptable (the winners' averaged rank is about 1.94). Finally, we observe that our mechanism outperforms one round plurality and 3-pragmatists.

Now, let us turn to Figure 2 in which we evaluate the performances using Condorcet efficiency. Among the voting rules we consider, only Copeland always elects a Condorcet winner when it exists. For the other rules, STV performs well, followed by iterative voting with learning agents. The best results are obtained for a high number of candidates and low number of voters. Previous work showed that a 10% increase could be obtained with restricted iterative voting in a similar setting (25 candidates and 10 vot-

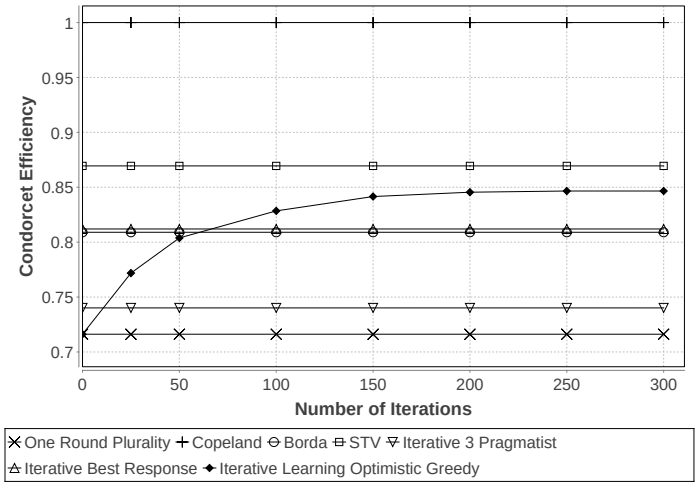

Fig. 2. Performance of learning agents in terms of Condorcet efficiency (9 voters, 7 candidates). The number of profiles which actually have a Condorcet winner is 7331 profiles.

ers, with preferences generated using the urn model) [7], and comparable figures with 50 voters and 5 alternatives using the impartial culture generation of preferences [15].

With standard iterative voting, only one voter at a time can manipulate, and it will do so only if it changes the outcome. If the winner c is winning by two votes or more, standard iterative voting will not be able to elect the Condorcet winner. In our framework, all voters can manipulate. However, some learners may not notice they have a chance to improve their utilities (because they voted for cw in the past but cw never won, so the Q-value for cw is low), others may decide to explore. In addition, voters do not know whether a Condorcet winner exists. But the improvement we observe shows that learners manage to coordinate their vote which results in electing a Condorcet winner.

Observe that, while Borda and Copeland are obviously scoring the maximum, respectively, in Borda score and CE, the Borda rule can score worse than iterative learning in terms of Condorcet efficiency, and a complex voting rule such as STV can score worse under both parameters (this occurs in simulations performed with 3 voters and 15 candidates). Note that we also conducted the same experiments under the impartial culture assumption, obtaining similar results. As a last remark, observe that in view of our initialization the first election is always truthful, i.e. its result coincide with one-round plurality. This is not the case anymore when using other exploration strategies, for which we obtained slower but similar learning dynamics.

4.2 Scalability

One drawback of using learning agents is the number of iterations for convergence. Obviously, it is not reasonable for a human agent to participate in such an iterated process. In the results presented in this section simulations are run with 500 iterations,

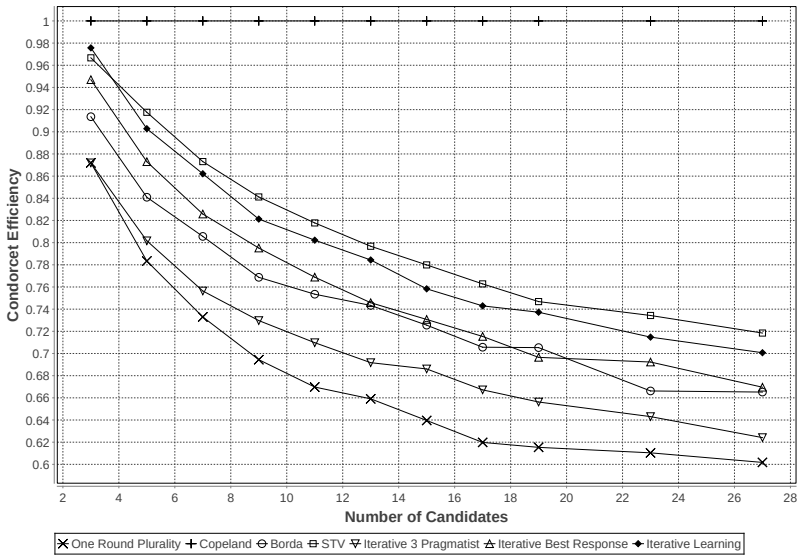

Fig. 3. Scalability of the performance of iterative voting with learning agents at 500 iterations, increasing candidates (Condorcet efficiency, 9 voters).

and we show that learners can still perform well. In addition, the number of voters and the number of candidates are two parameters that, when increased, could significantly deteriorate the performance of learning agents in iterative voting. Figures 3 and 4 shows instead that the deterioration is comparable to those voting rules we considered.

When we keep the number of voters fixed and we add more candidates, the Condorcet efficiency decreases at a similar rate as the other voting rules (results are equivalent to the ones of STV, and learning agents perform better than iterative best response and 3-pragmatist with a similar margin). Typically, in learning in games adding additional actions requires more iterations for learning well (and we usually observe a slight drop in performance). When the number of candidates is high, candidates low in the ranking will have utilities that are negligible compared to the top candidates. Therefore, under those circumstances, the loss in Condorcet efficiency is acceptable.

What is perhaps the most interesting result is that the number of voters does not affect significantly the performance of the learning agents. This is surprising since the environment is less stationary and the noise level is higher with more agents trying to learn concurrently. Typically, it is much more difficult to reach convergence with a high number of voters. On the other hand, with the increasing number of agents, the likelihood of being pivotal decreases, which may help convergence.

4.3 Social Welfare

The last criteria we want to consider are the measures of the social welfare that aggregate the individual utility. We considered the following two definitions:

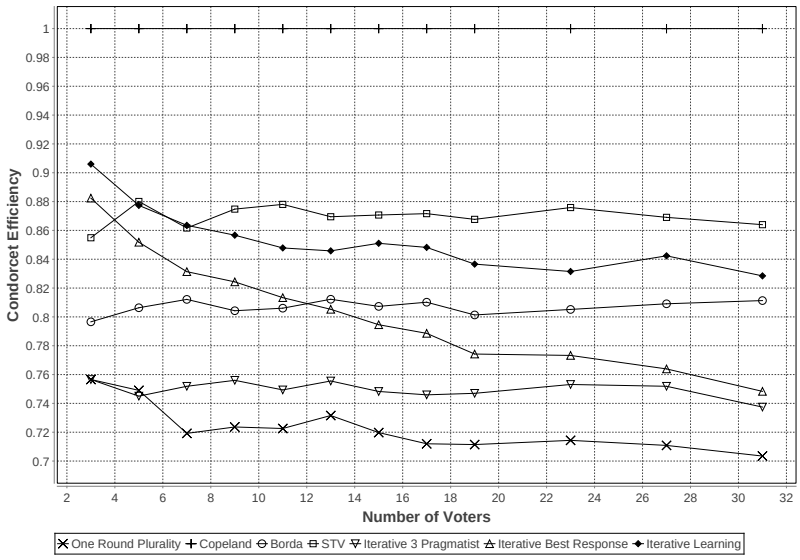

Fig. 4. Scalability of the performance of iterative voting with learning agents at 500 iterations, increasing voters (Condorcet efficiency, 7 candidates)

Utilitarian social welfare: $USW(c) = \sum_{i \in N} u_i(c)$
Egalitarian social welfare: $ESW(c) = \min_{i \in N} u_i(c)$

Observe that if individual utilities are defined as the Borda score, i.e., giving $m - j$ points to the individual in j-th position, then the USW of a candidate corresponds to its Borda score. The Borda score, Borda ratio and Condorcet Efficiency measure the performance of the voting rules. On the other hand, USW is a measure of efficiency, often used to study the performance of a (cooperative) multiagent system.

Each learning agent is trying to maximise its private utility function. Using plurality at each round, a majority of agents will be satisfied, so we do not necessarily expect to maximise USW, but we should observe that a majority of agents improves their utilities. Indeed, this is what we observed in our simulations, in which initially the performance of the learning agents is comparable with those of other voting rules but the USW quickly deteriorates. Remember that the utilities of each candidate (from the most preferred to the least preferred) are 1, $\frac{1}{2}$, $\frac{1}{4}$, etc. If we look at the distribution of utilities, we have more very high values initially (but less than a majority) and at convergence, we have a majority of middle or high utility values. The overall sum decreases over time, but more agents are happier. As a proof of this observation we measured the egalitarian social welfare - the utility of the poorest voter - observing that initially ESW starts pretty low and raises with the number of iterations. Using that criterion, iterative voting with learning agents scores third behind the Borda rule and Copeland.

Our method seems to perform poorly in terms of USW, although this should not be interpreted as a negative result. Note that the second worse voting rule is the Borda rule for the same reason. We also ran experiments with linear and logistic utilities, observing

an increasing social welfare as measured by the Borda score (recall that USW with linear utilities is the same as the Borda score).

5 Conclusions and Future Work

Motivated by the emergent works on iterative voting, this paper assesses the learning capabilities of autonomous agents in making good collective decisions. Voting theory tells us that agents always have incentives to manipulate. The idea of iterative voting is then to use a simple voting rule such as plurality, and ask each voter one at a time whether she wants to change her ballot to obtain a preferred winner in the next round.

In this paper we address two weaknesses of existing models of iterative voting: it is not realistic that voters change their ballot one at a time, nor is the myopic-agent assumption that does not allow voters to profit from past interactions. In our model we allow all agents to change their ballots at the same time if they wish to do so. In order to avoid a completely chaotic process, we use learning agents as a mean to learn a good compromise. We show that by using a simple learning algorithm with scarce information (only the winner of the current election is shown to learning agents), the performance of winning candidates are quite good. We evaluate the winner of the iterative process using extensive simulations both in terms of Borda score and of Condorcet efficiency against various voting rules, and we show that we obtain reasonable performances from around 300 iterations. While this number is of course too large for any human to use this method, it is manageable for artificial agents.

We leave it for future work to use more sophisticated learning mechanisms for decreasing the number of iterations to obtain a reasonable performance, and to explore settings in which more information is available to the agents. For instance, using the score of each candidate appears quite reasonable, but has consequences on the learning space.

Bibliography

[1] Adrian K. Agogino and Kagan Tumer. A multiagent approach to managing air traffic flow. *Autonomous Agents and Multi-Agent Systems*, 24(1):1–25, 2012.

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.

[3] Sven Berg. Paradox of voting under an urn model: The effect of homogeneity. *Public Choice*, 47(2):377–387, 1985.

[4] Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

[5] Caroline Claus and Craig Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-1998)*, 1998.

[6] Allan Gibbard. Manipulation of voting schemes: A general result. *Econometrica*, 41(4):587–601, 1973.

[7] Umberto Grandi, Andrea Loreggia, Francesca Rossi, Kristen Brent Venable, and Toby Walsh. Restricted manipulation in iterative voting: Condorcet efficiency and Borda score. In *Proceeding of the 3rd International Conference on Algorithmic Decision Theory (ADT-2013)*, 2013.

[8] Dominique Lepelley, Ahmed Louichi, and Fabrice Valognes. Computer simulations of voting systems. *Advances in Complex Systems*, 3(1-4):181–194, 2000.

[9] Omer Lev and Jeffrey S. Rosenschein. Convergence of iterative voting. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2012)*, 2012.

[10] Reshef Meir, Omer Lev, and Jeffrey S. Rosenschein. A local-dominance theory of voting equilibria. In *Proceedings of the 15th ACM Conference on Economics and Computation (EC-2014)*, 2014.

[11] Reshef Meir, Maria Polukarov, Jeffrey S. Rosenschein, and Nicholas R. Jennings. Convergence to equilibria in plurality voting. In *Proceedings of the Twenty-fourth conference on Artificial Intelligence (AAAI-2010)*, 2010.

[12] Svetlana Obraztsova, Evangelos Markakis, Maria Polukarov, Zinovi Rabinovich, and Nicholas R. Jennings. On the convergence of iterative voting: How restrictive should restricted dynamics be? In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, (AAAI-2015)*, 2015.

[13] Eric Pacuit. Voting methods. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Stanford University, 2012.

[14] Zinovi Rabinovich, Svetlana Obraztsova, Omer Lev, Evangelos Markakis, and Jeffrey S. Rosenschein. Analysis of equilibria in iterative voting schemes. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, (AAAI-2015)*, 2015.

[15] Annemieke Reijngoud and Ulle Endriss. Voter response to iterated poll information. In *Proceedings of the 11th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2012)*, June 2012.

[16] Mark Allen Satterthwaite. Strategy-proofness and Arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10(2):187 – 217, 1975.

[17] Yoav Shoham, Rob Powers, and Trond Grenager. If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171(7):365 – 377, 2007.

[18] R.S. Sutton and A.G. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.