Learning a Robot's Social Obligations from Comparisons of Observed Behavior

Colin Shea-Blymyer[1] and Houssam Abbas[1]

***Abstract*— We study the problem of learning a formal representation of a robot's social obligations from a human population's preferences. Rigorous system design requires a logical formalization of a robot's desired behavior, including the social obligations that constrain its actions. The preferences of the society hosting these robots are a natural source of these obligations. Thus we ask: how can we turn a population's preferences concerning robot behavior into a logico-mathematical specification that we can use to design the robot's controllers? We use non-deterministic weighted automata to model a robot's behavioral algorithms, and we use the deontic logic of Dominance Act Utilitarianism (DAU) to model the robot's social and ethical obligations. Given a set of automaton executions, and pair-wise comparisons between the executions, we develop simple algorithms to infer the automaton's weights, and compare them to existing methods; these weights are then turned into logical obligation formulas in DAU. We bound the sensitivity of the inferred weights to changes in the comparisons. We evaluate empirically the degree to which the obligations inferred from these various methods differ from each other.**

I. INTRODUCTION

Rigorous design of autonomous systems, such as social robots, requires that the system's requirements be formally expressed. That is, that they be expressed as formulas in a mathematical logic, for which automatic verification and control algorithms can be developed. Such algorithms come with correctness guarantees, thus significantly increasing trust in the robot. This paper focuses on a special class of requirements: the ethical and social obligations that constrain the robot's actions. *Deontic logic* has been developed especially to deal with obligations [1]. In prior work, we formalized the obligations and permissions of self-driving cars in Dominance Act Utilitarianism (DAU), a deontic logic that derives a system's obligations from utilitarian calculations [2]. We then demonstrated how to verify that an automaton has *given* DAU obligations via model-checking. In this work we tackle the question of how to elicit the DAU obligations in the first place: that is, how can we tell which obligations should constrain the robot, so we can design it to have and meet these obligations? This can be viewed as a special case of the more general *requirements elicitation* phase in software engineering.

Eliciting requirements starts by a conversation with the stakeholders. For social robots the stakeholders are the members of the society that will host these robots. A simple way to elicit requirements is to show a stakeholder two behaviors of the robot in the same situation, and ask which behavior they prefer. This pair-wise comparison has the advantages of being minimally burdensome to the respondents, and of being a quantitative assessment that can be processed mathematically. We follow this approach in this paper. The challenge then is to go from the pair-wise comparisons all the way to logical formulas that explain the respondents' preferences and synthesize the social obligations that are implicit in their minds. This paper exploits the particular structure of DAU formulas to perform this *obligation learning* task. Specifically, the pair-wise comparisons are used to assign a score to each behavior (the *scoring* step), such that if we order the behaviors by score, we approximately recover the population's preferences. (In general, exact recovery is not possible since different pair-wise comparisons need not be consistent). From the scores we then learn weights on the robot's actions, and from the weights DAU formulas can be derived.

The scoring process has been tackled in the preference learning literature in different contexts. We experiment with two methods from the literature: Bradley-Terry [3] and Borda [4]. However, our objective goes beyond scoring behaviors, since we want to ultimately infer DAU formulas. We also want to study the sensitivity of the final formulas to changes in opinion. For example, if one preference is flipped, or one respondent does not provide any answers, how much could the derived formulas differ? This sensitivity is an important consideration as it tells us whether our final learned obligations are robust to small shifts in opinion and to sampling artifacts. If the action weights, and therefore the obligations, change a lot, this tells us the learned obligations are practically arbitrary and cannot be taken as a reliable reflection of a population's social norms. We thus propose two additional, simple, scoring methods, for which we can provide a sensitivity analysis.

Related work This work has much in common with preference based reinforcement learning (PbRL). In PbRL, an agent demonstrates behavior to an expert and uses the expert's preferences regarding that behavior to learn a policy [5]. Techniques have been developed to learn from preferences over policies [5], actions [6], and trajectories [7]. We learn from preferences over trajectories. Core to many of these algorithms is a method of paired comparisons. Such a method takes pairwise comparisons of alternatives and assigns each alternative a score. Paired comparisons have a long history in psychometrics and economics ([8], [9], [10]). Scores arising from these methods are similar to the scores given to chess players via the Elo [11] and Glicko [12] rating systems, which express a competitor's likelihood of

[1]School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR, USA {sheablyc,houssam.abbas}@oregonstate.edu

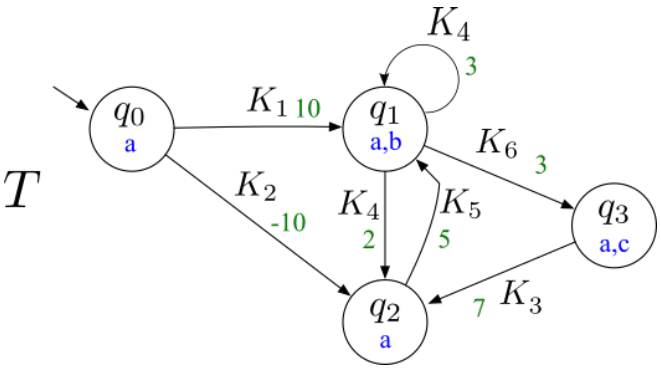

Fig. 1. A stit automaton T with weights in green and atomic propositions in blue. Let τ_1 be $(q_0, K_1, q_1)(q_1, K_6, q_3)(q_3, K_3, q_2)(q_2, K_5, q_1)\dots$ repeating the last three transitions ad infinitum. Assuming λ is DiscSum and $\gamma = 0.9$, then $\lambda(\tau_1) \approx 54.13$. In fact, τ_1 is the optimal execution of T. Further, there is no execution $\tau' \in \Xi_{q_0}^{K_2}$ such that $\lambda(\tau') > \lambda(\tau)$ for $\tau \in \Xi_{q_0}^{K_1}$. Thus, K_1 is the optimal action at q_0. Consequently, $\odot[T\, cstit\colon a]$ is an obligation at q_0 — the proposition "a" is a necessary result of any execution in $\Xi_{q_0}^{K_1}$. Less obviously, $\odot[T\, cstit\colon Xb]$ is an obligation at q_0, where "Xb" means that b holds true in the next state.

defeating (being preferred over) another competitor. Recent work in comparison methods has suggested that simply scoring an alternative by the number of comparisons won (the Borda count) has many attractive properties [13], while other work has sought to reconcile orderings inferred from pairwise comparisons and direct rating [14]. In our work, the scores are only an intermediate step towards learning optimal policies that yield DAU formulas.

II. System Model and Formalized Obligations

To develop a formal specification for a system, we must model the system, and choose a specification language.

A. System model

We model the robot's high-level behavioral algorithms, such as navigation or decision-making, as a weighted non-deterministic automaton, an example of which is given in figure 1. This model, and variations on it, are very common in the field of *formal methods and logic*. It captures reactive behavior, conditional executions, uncontrolled events in the environment, and costs of various actions.

Definition 1 (Stit automaton): Let AP be a finite set of atomic propositions. A *stit automaton* T is a tuple $T = (Q, q_I, \mathcal{K}, F, \Delta, L, w, \lambda)$, where Q is a finite non-empty set of states, q_I is the initial state, $\mathcal{K}$ is a finite non-empty set of actions, $F \subset Q$ is a set of final states, $\Delta \subset Q \times \mathcal{K} \times Q$ is a finite transition relation such that if (q, K, q') and (q, K', q') are in Δ then $K = K'$, $L : Q \to 2^{AP}$ is a labeling function, $w : \Delta \to \mathbb{R}$ is a weight function, and $\lambda : \mathbb{R}^\omega \to \mathbb{R}$ is an accumulation function.

The atomic propositions denote basic properties that hold in a state. E.g. 'Ignore-patient' or 'Respond-to-call'. We can interpret formulas of temporal logic, like CTL, over the non-deterministic executions of the automaton in the usual way [15]. Denote by $\mathcal{K}(q) = \{K \in \mathcal{K} \mid \exists (q, K, q') \in \Delta\}$ the set of actions available at q.

Definition 2 (Execution): Let T be a stit automaton and q_0 a state in Q. A *q_0-rooted execution* τ of T is a sequence of transitions of the form $\tau = (q_0, K_0, q_1)(q_1, K_1, q_2)\dots \in \Delta^\omega$. The *value of execution* $\tau = \tau[0]\tau[1]\tau[2]\dots$ is defined to be $\lambda(w(\tau[0])w(\tau[1])w(\tau[2])\dots)$, and abbreviated $\lambda(\tau)$. Because of non-determinism, a sequence of actions can yield multiple executions. Examples of function λ are the min and discounted sum functions:

$$\min(\tau) = \min_i w(\tau[i]) \tag{1}$$

$$\mathsf{DiscSum}(\tau) = \sum_{i=0}^{\infty} \gamma^i \cdot w(\tau[i]) \ , 0 < \gamma < 1 \tag{2}$$

where γ is the discount factor for the summation. We will write w_i for the weight $w(\tau[i])$.

B. The specification language

Deontic logics were specifically developed to formalize and reason about obligations and permissions, including ethical and social obligations. In this paper, obligations are formalized in the deontic logic of Dominance Act Utilitarianism (DAU) [16]. With DAU, we can describe temporal behaviors and reason about uncontrollable environments. Intuitively, in DAU, a system's obligations are derived from maximizing utility: an agent ought to do those things that are logically necessary for an optimal state of affairs.

We give the DAU basics which will suffice for our purposes in this paper. Let the set of executions starting with state q be Ξ_q, and let those executions in Ξ_q that begin with action $K \in \mathcal{K}(q)$ be denoted $\Xi_q^K = \{\tau \mid \tau[0] = (q, K, q')\}$. An action K^* in $\mathcal{K}(q)$ is *optimal* if there is no other action $K' \in \mathcal{K}(q)$ s.t. $\lambda(\tau') > \lambda(\tau)$ for all $\tau \in \Xi_q^K$ and $\tau' \in \Xi_q^{K'}$. In DAU, an agent's obligation is to enable an ideal state of affairs, which is characterized by the formulas that result from optimal actions. Specifically, let A be a formula in the underlying temporal logic, like CTL. An execution τ either satisfies or violates A. The basic obligation modality in DAU is $\odot[\alpha\, cstit\colon A]$, which says that robot α *ought to see to it that* A is true. Its formal semantics are as follows: $\odot[\alpha\, cstit\colon A]$ holds true in a state q iff for all optimal actions K^* at q, and for all executions τ in $\Xi_q^{K^*}$, τ satisfies A. That is, A is a necessary condition of all executions from optimal actions.

III. Learning Obligations from Pairwise Comparisons

The fact that DAU obligations are derived from value maximization, and the value function λ depends on the weights w_i, allows us to focus on learning the weight function w of the automaton in Def. 2. On the other hand, when comparing different methods, we will need to measure the difference between formulas. This section explains how both steps are performed.

Automata generation. We generate system automata randomly via ErdosRenyi model [18] $T \sim G(|Q|, \rho)$, where the probability of adding a transition between two given states is

ρ. Transitions are carefully added to avoid absorbing states. Weights assigned to each transition are chosen uniformly at random, and the executions τ are the result of random walks of finite (but relatively great) length. We focus on the case where the accumulation λ is DiscSum (eq. 1), with a discount rate $\gamma = 0.9$.

Generating preference data. Given a stit automaton T, we generate n random executions τ_i of T from an initial state q_I via random walks. Then, we simulate a population making comparisons as follows. For each pair τ_i, τ_j s.t. $i \neq j$, for each member p of the population P, we draw values for the *latent scores* $s_p(\tau_i) \sim \mathcal{D}_i$ and $s_p(\tau_j) \sim \mathcal{D}_j$, where the mean of the distribution $\mathcal{D}_i$ is $\lambda(\tau_i)$. In the preference learning literature, $s_p(\tau_i)$ captures an implicit, or latent, score that a member of the population assigns to a trajectory, based on which it judges it to be better or worse than another. Of course, we do not ask the population for the latent scores, we only ask which execution they prefer. Let $R^{n\times n}$ be our response matrix, where

$$r_{ij} = \sum_{p\in P}(1 \text{ if } s_p(\tau_i) > s_p(\tau_j) \text{ else } 0) \tag{3}$$

i.e. r_{ij} is the number of comparisons where τ_i is preferred over τ_j. R may also be thought of as an adjacency matrix for a weighted directed graph where an edge from τ_i to τ_j is weighted by the number of times τ_i won against τ_j. The matrix R is then passed to each of the methods of paired comparison discussed in section III-A. Each method then returns an (explicit) score $s(\tau_i)$ for each execution.

Weights from scores. To estimate the values of the weights $w \in T$ from the scores, we need knowledge of the accumulation function $\lambda \in T$, and of the executions τ. When λ is $\min$, we can estimate the weights by a mixed-integer linear feasibility problem (eq. 4). Each weight must be equal to, or larger than, the score of any execution they appear in. Additionally (and the reason this problem requires integer constraints) each execution must have a weight that appears in it that equals that execution's minimum.

$$\begin{array}{lrcll} \min\limits_{\vec{w}} & 0 & & & \\ \text{s.t.} & w_j & \geq & s(\tau_i) & \forall w_j \in \tau_i \ \forall \tau_i \\ & w_j & \leq & s(\tau_i) + (1 - b_{i,j})M & \forall w_j \in \tau_i \ \forall \tau_i \\ & \sum\limits_{j=0}^{n} b_{i,j} & \geq & 1 & \forall \tau_i \end{array} \tag{4}$$

When λ is DiscSum, the problem of estimating the weights $w \in T$ can be formulated as a least-squares problem (eq. 5). Namely, we create the matrix $A^{n,m}$ where $m = |\Delta|$ - i.e. it has one column per transition in Δ. Initially A is the zero matrix. For each transition $\tau_i[k] \in \tau_i$ we add γ^k to a_{ij} where j is the index of transition $\tau_i[k]$ in Δ. We can now estimate the weights by minimizing the squared sum of the difference between an execution's accumulated value, and its score.

$$\min_{\vec{w}} \sum_{i=0}^{n} (\vec{a_i}^T \vec{w} - s(\tau_i))^2 \tag{5}$$

With these estimated weights, we can employ value iteration [17] to determine the optimal action to take from each state in T, and, therefore, we can determine what obligations hold in each state. The solution w^* to Eq 4 obeys :

$$\|\delta\vec{w}\| \leq \|A^+\|\|\delta\vec{s}\|$$

where A^+ is the pseudo-inverse of A. This will be used for the sensitivity analyzes for the various scoring methods in subsection III-A.

Measuring the distance between formulas. To compare the various scoring methods, we will need to quantify the difference between the DAU formulas they yield. In the absence of a fixed labeling map L for the automaton (Def. 1), this is an ill-posed problem, since we could always choose a labeling that makes syntactically different formulas, logically 'close'. In our experimental setup, we generate random automata, so there is no fixed labeling map. Luckily, the DAU formulas that hold at a given state q depend on which actions are optimal at q, so we can quantify the difference between optimal actions. A *policy* is a list π of optimal actions, with one entry $\pi[i]$ per automaton state. As a state may have multiple optimal actions available to it, $\pi[i]$ is a set of actions, e.g.$\pi[i] = \{K_1, K_2\}$. We measure difference between two policies π_1 and π_2 by taking the Jaccard distance [19] between each pair of entries and then taking the L_2 norm of the vector of distances $\vec{d}$, as shown in equation 6

$$d_i(\pi_1, \pi_2) = \frac{|\pi_1[i] \cap \pi_2[i]| - |\pi_1[i] \cup \pi_2[i]|}{|\pi_1[i] \cap \pi_2[i]|} \tag{6}$$

so $D(\pi_1, \pi_2) = \|\vec{d}(\pi_1, \pi_2)\|_2$ is the difference between the two policies.

A. The four scoring methods.

We now introduce four methods for turning pairwise comparisons of alternatives into scores for each alternative.

a) Bradley-Terry: The Bradley-Terry method of paired comparison [3] is a probabilistic model of the result of a match-up between two alternatives. This model is well known in the literature on paired comparisons and was discovered as early as 1860 by Fechner [20], and again in 1929 by Zermelo [21], before being popularized by Bradley and Terry. The Bradley-Terry model finds use today in machine learning research as a popular choice of comparison method, evidenced by its use in [22] and [23].

The Bradley-Terry model estimates that the probability of i defeating j is:

$$Pr(i > j) = \frac{\beta_i}{\beta_i + \beta_j} \tag{7}$$

The parameter β_i for each alternative i can be solved by maximum likelihood estimation from the comparisons data, though faster spectral methods have been developed as well [24]. This necessity for parameter estimation makes formal sensitivity analysis difficult, so an empirical approach is taken in section IV. In this work, the score returned by the Bradley-Terry method of an execution τ_i is $s(\tau_i) = \beta_i$.

b) Borda: We apply the Borda count [4] as a method of paired comparisons. Originally conceived as an election method, and still in use as one today, this method has gained attention as a method of paired comparison for its simplicity and its capability to recover the latent ordering of alternatives with high fidelity [13].

The Borda method simply counts the number of times an alternative i defeats any other alternative. More formally, given a response matrix $R^{n\times n}$ where r_{ij} is the number of people that prefer i over j, the Borda score for i is $b(i) = \sum_{j\neq i} r_{ij}$. In this work we normalize the maximum returned Borda score to one, so $s(\tau_i) = \frac{b(\tau_i)}{\max(b(\vec{\tau}))}$.

c) Deterministic Depth-Based Scoring: We are also interested in paired comparison methods that maximally preserve transitivity in the superiority of an alternative over another. To avoid the analytic complexities of model-fitting methods, we introduce two novel methods for which we have calculated sensitivity results.

Our first such method, the Deterministic Depth-Based Scoring (or *d-comp*) method, constructs a weighted directed graph G over the n alternatives using the response matrix $R^{n\times n}$ as an adjacency matrix.

If G is acyclic, then it will be clear which histories are the most preferred, and which are the least. However, it is likely that G will contain cycles, and so, to arrive at a transitive ordering of histories, we remove cycles from G using a variant of [25]. We find in our experiments that the wall-time for this operation is negligible for even large problems in this domain.

We call the weighted directed acyclic graph that results from applying this algorithm $\hat{G} = (V, \hat{E})$.

Now that we've obtained the weighted DAG $\hat{G}$, we can use the transitive nature of the graph to establish a ranking of traces. To do so, we first obtain a topological ordering of $\hat{G} = (V, \hat{E})$.

Given the topological ordering $\hat{G}'$, we may assign values to the traces represented by the vertices in $\hat{G}'$ that best obey the preferences expressed by our population. To avoid arbitrarily large rewards, we will normalize all rewards to the range $[0, 1]$. We begin by finding the longest path between root and leaf vertices — that is, the longest path between a vertex with no incoming edges and a vertex with no outgoing edges. The target (v_0) of that path is assigned a value of 0 and the source (v_1) of that path is assigned a value of 1. The predecessor (v_{p0}) of v_0 on that path is given the value of $1/l$ where l is the length of the path. Then the predecessor of v_{p0} on that path is given the value of $2/l$, and so on, until v_1 (which would receive a value of $l/l = 1$). Next, assign all root vertices the value of 1. Then, every vertex is assigned a value equal to $1 - d/l$ where d is its maximum distance from a root vertex. So $s(\tau_i)$ is the value assigned to the vertex associated with τ_i. We note that this method biases scores towards 1. We explore one way to mitigate this with our second novel method.

d) Stochastic Depth-Based Scoring: In sub-section III-A.0.c, we propose a simple method to assign values to traces in a given extended linear order. We now introduce a modification of that procedure in order to investigate the solution's sensitivity under a different valuation procedure. In comparison to the previous method, this second method (*s-comp*) harbors no bias towards scores of 1 due to the introduction of random score sampling.

Given a directed acyclic graph $\hat{G}$, find all vertices with no incoming edges (i.e. the roots). All root vertices r_i receive the same value $s(r_i) \sim U(0, 1)$. These root vertices make up the first layer of the graph — L_0. Assign each other vertex to layer L_d where d is that vertex's maximum distance from a root vertex. Assign all vertices in L_d the same value $s(L_d) \sim U(0, s(L_{d-1}))$.

$$\|\delta\vec{s}\|^2 = \left|\frac{1}{2^{d_u+1}} - \frac{1}{2^{d_v}}\right| \cdot \sum_{k=0}^{m} \frac{|D_k|}{4^k} \tag{8}$$

where $|D_0|$ is 1.

e) Comparison methods.: For the comparison of policies arising from T, we define a *baseline* and an *ideal* method for determining optimal policies. The baseline method, against which the others are measured, directly applies value iteration to the true weights of T. The policy generated by this process is the true optimal policy of the T. The ideal, or *no-comparisons* method solves the minimization problem in eq. 5 by replacing $s(\tau_i)$ with $\lambda(\tau_i)$. That is, the no-comparisons method minimizes the sum-squared difference between an execution's accumulation of unknown weights and the true value of that execution, as opposed to the score for that execution. The policy returned by the no-comparisons method is the theoretically ideal policy that a method using pairwise comparisons could return.

f) Theoretical Sensitivity Analyzes: It is important to place theoretical bounds on how much the learned weights change, when there's an opinion change in the responses. Below we present sensitivity analyzes for the scoring methods we introduced, s-comp and d-comp, and for Borda. To the best of our knowledge, there is no known sensitivity analysis for Bradley-Terry. (In the next section, we study empirically the sensitivity or the *final policy* to changes in responses and experimental parameters.)

d-comp The scores returned by this method may change only when an edge in $\hat{G}$ is added or removed, so we can bound the change in $\vec{s}$ above by impact of the worst case edge edit on this graph. Let d_u be the depth of a vertex in $\hat{G}$, defined by the maximum distance of u from a root vertex. Let there be an edge addition uv to the edge set of $\hat{G}$ such that $d_u < d_v$. Let the descendants of u be partitioned into sets $D_1, D_2, \ldots, D_m$ such that D_1 is composed of the children of u that are one layer beneath u, D_2 is composed of the children of u that are two layers beneath u, and D_m is composed of the children of u that are m layers beneath u. If the edge addition uv is worst case, then we find the change in this method's returned score $\|\delta\vec{s}\|$ is given by equation 9.

$$\|\delta\vec{s}\| = \left|\frac{d_v * d_u - 1}{n}\right| (1 + \sum_{k=1}^{m} |D_k|) \tag{9}$$

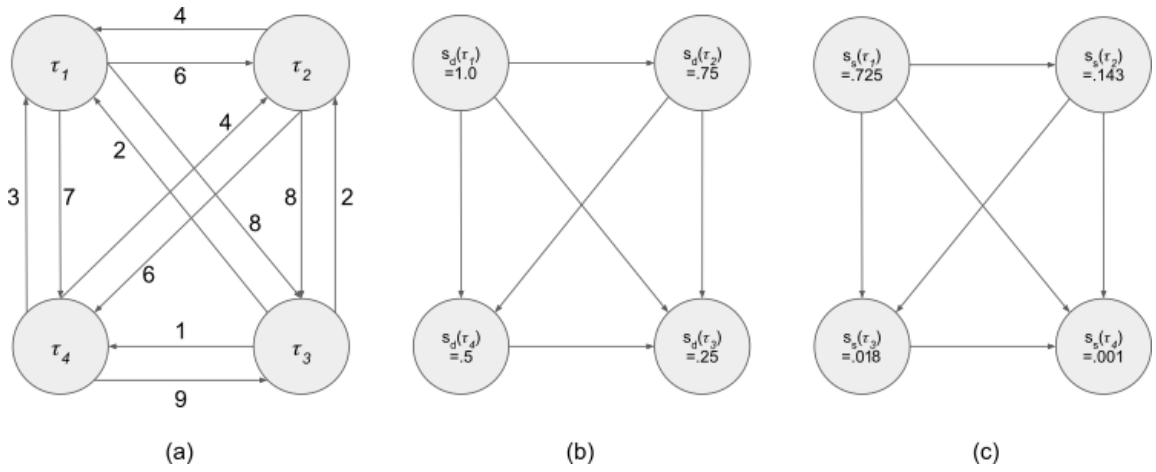

Fig. 2. (a) The *tournament graph* G constructed with adjacency matrix R. (b) and (c) the DAG $\hat{G}$ obtained from G with values on vertices as given by the deterministic depth-based scoring method, and the stochastic depth-based scoring method, respectively. Given this tournament, the Borda method would assign scores: $s(\tau_1) = 1.0$, $s(\tau_2) = 0.857$, $s(\tau_3) = 0.238$, $s(\tau_4) = 0.762$, and the Bradley-Terry method would return: $s(\tau_1) = 0.723$, $s(\tau_2) = 0.357$, $s(\tau_3) = -1.135$, $s(\tau_4) = 0.045$.

s-comp Again we find that the scores may only change due to edge edits on $\hat{G}$. Let d_u be the layer of u defined by the maximum distance of u from a root vertex. Let there be an edge addition uv to the edge set of $\hat{G}$ such that $d_u < d_v$. Let the descendants of vertex v be partitioned into sets $D_1, D_2, \ldots, D_m$ such that D_1 is composed of the children of v that are one layer beneath v, D_2 is composed of the children of v that are two layers beneath v, and D_m is composed of the children of v that are m layers beneath v. We find the expected value given to layer n is $E[s(L_n)] = \frac{1}{2^{n+1}}$. so, in expectation, $\delta\vec{s}[v] = \frac{1}{2^{d_u+2}} - \frac{1}{2^{d_v+1}}$. For children of v in the next level, $\delta\vec{s}[D_k] = \frac{1}{2^{d_u+1+k}} - \frac{1}{2^{d_v+k}}$. Thus, in total, the sensitivity to a worst case edge edit for this method's returned score is given in equation 8.

Borda. The Borda score can be rewritten as $\vec{b} = R \cdot \mathbf{1}^n$, where $\mathbf{1}^n$ is an n-dimensional column vector whose entries are all 1. In this form, the sensitivity of this method is clear: $\|\delta\vec{b}\| \leq \|\delta R\|$. Changes to the score are closely bounded above by changes in the response matrix.

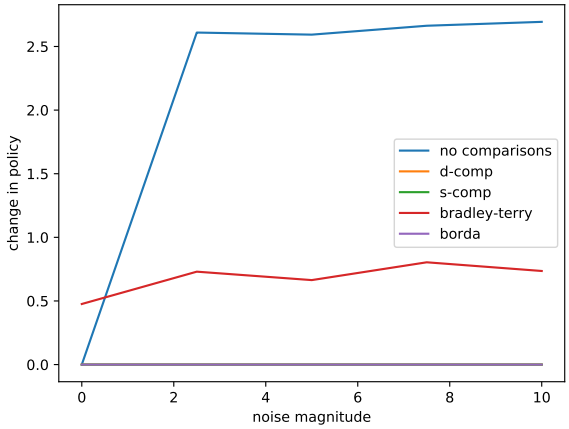

Fig. 3. Change in policy as the magnitude of change in trajectory values increases. The "d-comp", "s-comp", and "borda" curves are overlapping.

IV. EXPERIMENTAL RESULTS

We explore the robustness of the four methods presented in section III-A. We begin with an empirical measure of the sensitivity of an output policy to perturbations in the given values of trajectories. We then investigate how changing experimental parameters alters the performance of these methods.

Policy sensitivity to changes in preferences. To explore the sensitivity of a method, we first use it to find a policy π_u on a 25-state automaton. This policy is our unperturbed policy. We then perturb the latent score $s'(\tau_i) = s(\tau_i) + \delta_i$ where δ_i is a random value, and $\vec{\delta}$ is of a fixed magnitude. Next we use the method in question to learn a new policy π_p from these perturbed values. This is our perturbed policy. Changing the latent scores results in changes to the preference matrix R. The empirical sensitivity to this perturbation is taken as the difference $D(\pi_u, \pi_p)$ between the unperturbed and perturbed policies. In figure 3 we show the empirical sensitivity of each method (averaged over 50 batches) as the fixed magnitude $\|\vec{\delta}\|_2$ of perturbation increases. Figure 3 shows that the no-comparisons ideal method is highly sensitive to noise — allowing noise to noticeably change the returned policy. The Bradley-Terry method is also sensitive to noise, but not as dramatically. The other methods of comparison all seem insensitive to this degree of noise, and thus are preferable from a sensitivity-to-preferences perspective.

A reliable method behaves predictably when the experimental parameters, like population size or automaton size, change. For this study, we measure a method's performance as the difference $D(\pi_{\text{method}}, \pi_{\text{true}})$ between its inferred policy and the optimal policy derived from the true weights of the automaton.

In each of the following analyses, 50 datasets are generated

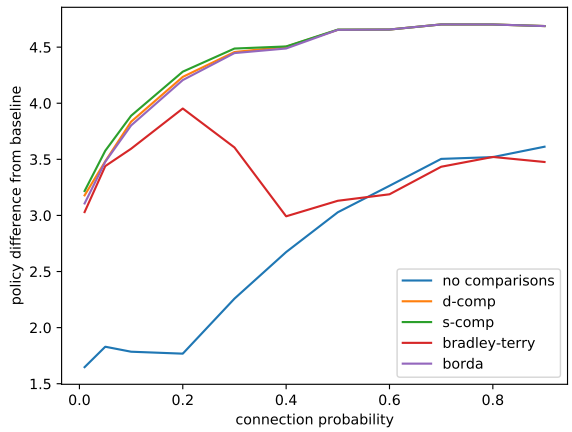

Fig. 4. Policy difference as the number of edges in the automaton increases.

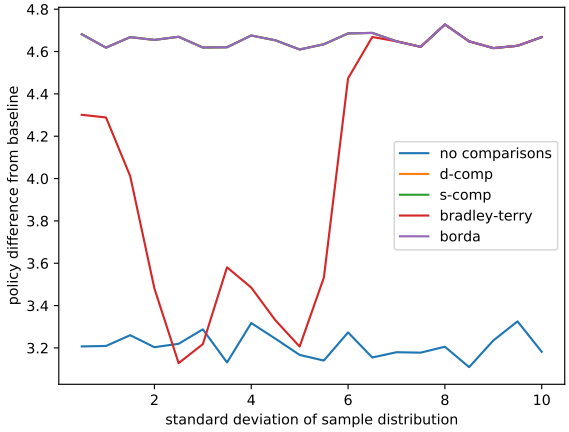

Fig. 5. Policy difference as the standard deviation of the sampling distribution grows. The "d-comp", "s-comp", and "borda" curves are overlapping.

for every value of the experimental parameter, and we report a method's average performance over those sets. Further, the default state size $|Q|$ of an automaton is 25, with edge probability ρ of 0.6, and weights w assigned to each edge with uniform probability in the range 0 to 10. The number n of trajectories sampled from each automaton is equal to its number of transitions, and the length of each trajectory is $1.5 \times |Q|$. The latent score of a trajectory is sampled from a normal distribution with a standard deviation σ of 2.5. The default population size $|P|$ is 500.

Robustness to automaton connectedness. We begin by varying the connectedness ρ of the automaton while keeping the number of states constant. Increasing ρ increases the nondeterminism of the automaton and creates a larger execution space. As seen in figure 4, the no-comparisons ideal decreases in performance as the connectedness increases. Our novel methods and the Borda method remain approximately a constant factor away from the ideal and level off. The Bradley-Terry method begins to follow suit, but sharply increases in performance until it meets the performance of the ideal method, at which point the two worsen in performance at the same rate.

Under the suspicion that our choice of normal distributions $\mathcal{N}(\mu;\sigma)$ for sampling the latent scores was of particular advantage to Bradley-Terry, we performed the experiment with a uniform distribution and observed similar results. We then increased the standard deviation σ (fig. 5). Methods s-comp, d-comp and Borda performances do not change. While Bradley-Terry's performance varies with σ, it always performs better than, or equal to the other (non-ideal) methods.

Robustness to number of automaton states. We next show the performance of these methods as the number $|Q|$ of states in the automaton grows. Figure 6 shows that the Bradley-Terry method is consistently competitive with the no-comparisons ideal method, while the other methods fall behind. Indeed, the difference between the performance of the Bradley-Terry method and the other non-ideal methods increases monotonically as the automaton's size grows.

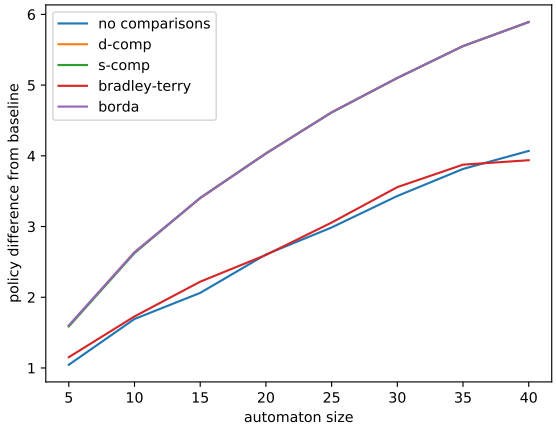

Fig. 6. Policy difference as the number of states in the automaton grows. The "d-comp", "s-comp", and "borda" curves are overlapping.

Robustness to population size. It is also desirable that the policies obtained through these methods are reliable reflections of a population's preferences regardless of that population's size. Figure 7 suggests that the population size $|P|$ is not a major factor in the performance of any of the methods.

Robustness to number of behaviors. Finally, we would like a sense of how many executions we must sample from an automaton and submit to judgment before we see reasonable policies being returned. Figure 8 shows that the no-comparisons ideal and Bradley-Terry methods require about $n = 350$ executions before their performance ceases to increase. This implies that these methods are optimal when $n \approx |\Delta|$. Meanwhile, other methods do not seem to vary greatly based on the number of trajectories compared.

V. CONCLUSIONS

DAU has a computational structure that allows the learning of DAU formulas from pair-wise comparisons of robot

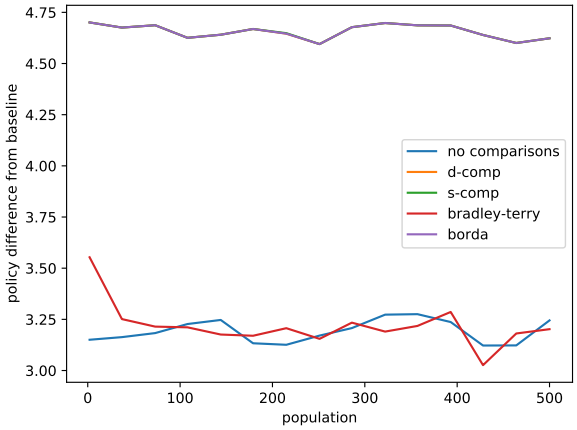

Fig. 7. Policy difference as the sample population grows. The "d-comp", "s-comp", and "borda" curves are overlapping.

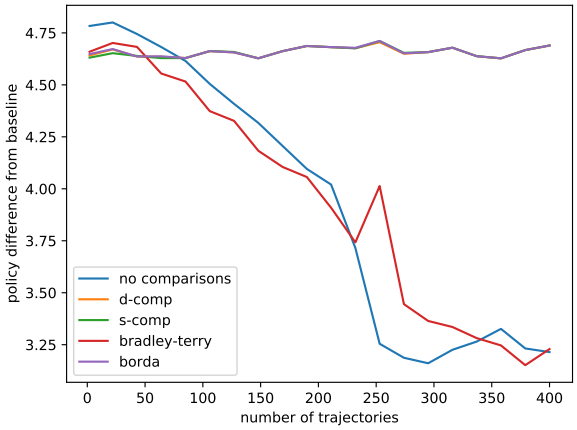

Fig. 8. Policy difference as the number of trajectories sampled from the automaton increases. The "d-comp", "s-comp", and "borda" curves are overlapping.

behavior, when the robot is modeled by a non-deterministic automaton. The learning algorithm allows the use of different scoring methods. The two methods we introduced in this paper allow a theoretical analysis of their sensitivity to changes in population preferences. Next it will be interesting to extend our approach to cases where the automaton is only partially available, or is unknown altogether. This can happen when the learning task is performed early in the design stage.

REFERENCES

[1] G. H. von Wright, "Deontic logic," *Mind*, vol. 60, no. 237, January 1951.

[2] C. Shea-Blymyer and H. Abbas, "A deontic logic analysis of autonomous systems' safety," in *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, 2020, pp. 1–11.

[3] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: I. the method of paired comparisons," *Biometrika*, vol. 39, no. 3/4, pp. 324–345, 1952.

[4] J. d. Borda, "Mémoire sur les élections au scrutin," *Histoire de l'Academie Royale des Sciences pour 1781 (Paris, 1784)*, 1784.

[5] R. Akrour, M. Schoenauer, and M. Sebag, "Preference-based policy learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2011, pp. 12–27.

[6] W. Cheng, J. Fürnkranz, E. Hüllermeier, and S.-H. Park, "Preference-based policy iteration: Leveraging preference learning for reinforcement learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2011, pp. 312–327.

[7] A. Wilson, A. Fern, and P. Tadepalli, "A bayesian approach for policy learning from trajectory preference queries," in *Advances in neural information processing systems*, 2012, pp. 1133–1141.

[8] L. L. Thurstone, "A law of comparative judgment." *Psychological review*, vol. 34, no. 4, p. 273, 1927.

[9] B. Trawinski and H. David, "Selection of the best treatment in a paired-comparison experiment," *The Annals of Mathematical Statistics*, vol. 34, no. 1, pp. 75–91, 1963.

[10] H. Buhlmann and P. J. Huber, "Pairwise comparison and ranking in tournaments," *The Annals of Mathematical Statistics*, vol. 34, no. 2, pp. 501–510, 1963.

[11] A. Elo, "The rating of chessplayers past and present. arco pub (1978)," *Glickman, ME, Paired comparison models with time-varying parameters, Tech.*

[12] M. E. Glickman, "A comprehensive guide to chess ratings," *American Chess Journal*, vol. 3, no. 1, pp. 59–102, 1995.

[13] N. B. Shah and M. J. Wainwright, "Simple, robust and optimal ranking from pairwise comparisons," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 7246–7283, 2017.

[14] M. Perez-Ortiz, A. Mikhailiuk, E. Zerman, V. Hulusic, G. Valenzise, and R. K. Mantiuk, "From pairwise comparisons and rating to a unified quality scale," *IEEE Transactions on Image Processing*, vol. 29, pp. 1139–1151, 2019.

[15] E. M. Clarke and E. A. Emerson, "Design and synthesis of synchronization skeletons using branching-time temporal logic." in *Logic of Programs*, 1981, pp. 52–71.

[16] J. Horty, *Agency and Deontic Logic*. Cambridge University Press, 2001.

[17] R. Bellman, "A markovian decision process," *Indiana Univ. Math. J.*, vol. 6, pp. 679–684, 1957.

[18] P. Erdős and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, no. 1, pp. 17–60, 1960.

[19] P. Jaccard, "The distribution of the flora in the alpine zone.1," *New Phytologist*, vol. 11, no. 2, pp. 37–50, 1912. [Online]. Available: https://nph.onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-8137.1912.tb05611.x

[20] G. T. Fechner, *Elemente der psychophysik*. Breitkopf u. Härtel, 1860, vol. 2.

[21] E. Zermelo, "Die berechnung der turnier-ergebnisse als ein maximumproblem der wahrscheinlichkeitsrechnung," *Mathematische Zeitschrift*, vol. 29, no. 1, pp. 436–460, 1929.

[22] B. Ibarz, J. Leike, T. Pohlen, G. Irving, S. Legg, and D. Amodei, "Reward learning from human preferences and demonstrations in atari," *Advances in neural information processing systems*, vol. 31, pp. 8011–8023, 2018.

[23] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, "Deep reinforcement learning from human preferences," in *Advances in Neural Information Processing Systems*, 2017, pp. 4299–4307.

[24] L. Maystre and M. Grossglauser, "Fast and accurate inference of plackett–luce models," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/2a38a4a9316c49e5a833517c45d31070-Paper.pdf

[25] G. Even, J. S. Naor, B. Schieber, and M. Sudan, "Approximating minimum feedback sets and multicuts in directed graphs," *Algorithmica*, vol. 20, no. 2, pp. 151–174, 1998.