Multiagent Policy Teaching

Lachlan Dufton
Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
ltdufton@cs.uwaterloo.ca

Kate Larson
Cheriton School of Computer Science
University of Waterloo
Waterloo, Canada
klarson@cs.uwaterloo.ca

ABSTRACT

Recently Zhang and Parkes [11, 12] introduced the idea of value-based policy teaching. In their framework, an interested party is able to provide incentives, by changing the environment, in order to encourage an agent to follow a particular policy. In this paper, we extend the Zhang-Parkes framework to a multiagent setting where all agents are in a common environment so that any modifications made by the interested party are experienced by all agents. We characterise when it is possible for the interested party to provide incentives so that all agents follow a particular desired policy. For the case where the interested party is unable to induce all agents to follow a particular desired policy, we propose that a *behaviour-based* policy comparison approach be used, where the interested party maximises the *similarity* of each agent's behaviour to some target behaviour.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Algorithms, Design, Performance

Keywords

Multiagent Learning, Co-ordination, Decision/Utility Theory, Markov Decision Processes

1. INTRODUCTION

Recently Zhang and Parkes [11, 12] introduced the idea of value-based policy teaching. In their framework, an interested party is able to provide incentives, by changing the environment, in order to encourage an agent to follow a particular policy. The focus of these papers was on a single-agent setting, and the authors studied computationally tractable methods for finding incentives.

In this paper, we study the problem of extending the environment design framework to a multiagent setting. In particular, we focus on the case where agents do not interact, but act in a common environment. Website management is a real-world example of such a scenario. The developer of a website may want visitors to view certain key pages in a particular order. Through well-chosen modifications to the website, it may be possible to lead many, and possibly all, visitors to act as the developer desires. Each visitor has different opinions and goals and none interact with each other, but the website is the same for all. The main challenge we examine in this paper is discovering the types of modifications or incentives that lead a heterogeneous collection of agents to act as desired.

In this paper, the agents are modelled as simple planners in a Markov Decision Process (MDP), and an interested party has a desired behaviour for these agents. While the interested party is unable to directly modify the behaviour of each agent, it can modify the environment, and thus agent behaviour.

We provide a linear programming solution for finding a minimal environmental modification that teaches a specific MDP policy to all agents. This is not always possible, and we show how to detect the feasibility of teaching a policy as well as a method for teaching the largest feasible subset of agents. In this case, attempting to teach the policy exactly can be inflexible. An interested party may not be interested in teaching an exact policy, but simply a policy with similar behaviour to a target policy. In this paper we look at how to quantify this behavioural similarity, based on short-term through to long-term behaviour.

This paper begins with an overview of policy teaching and environment design in the single agent setting, and a summary of other related work. Next, we discuss our multiagent model and our solution to the problem of teaching a specific policy to a set of agents. We then examine a value-based approach to policy teaching, that changes the policies of agents to maximise the aggregate value of each agents' policy. We present two methods of individual policy appraisal, based on the states visited when following the policy or by behavioural similarity to a target policy. We also provide a method of finding incentives that maximise the similarity of all agents' behaviours to the target behaviour. This is followed by empirical tests of our new methods and finally conclusions and directions for future work.

2. BACKGROUND

In this section we discuss the problem of policy teaching through the method of environment design. This begins with an overview of the problem in a single-agent setting. The single-agent model forms the foundation of our multiagent model and our work in this paper extends the solu-

Cite as: Title, Author(s), *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009)*, Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. XXX-XXX.

tion for this single-agent problem. We also examine some other works related to the problem of achieving a desired behaviour in a setting with multiple agents.

2.1 Single Agent Environment Design

There are existing methods that modify an agent's environment to teach a specific policy to the agent [11] or to teach a policy that maximises the value of the policy from the perspective of an interested party [12]. In the single agent setting, the environment is simply a finite-horizon Markov Decision Process (MDP), M, and the agent follows the optimal policy, π. Formally, $M = \{S, A, R, P, \gamma\}$, where

- S is the finite set of states.
- A is the finite set of actions.
- $R : S \rightarrow \mathbb{R}$ is a reward function, giving the agent's utility for each state. This can be represented as a vector $R \in \mathbb{R}^{|S|}$.
- $P : S \times A \times S \rightarrow [0, 1]$ is the state probability transfer function. If the agent plays action a in state s_i, then the probability of transitioning to state s_j is $P(s_i, a, s_j)$.
- $\gamma \in (0, 1)$ is the discount factor.
- $\pi : S \rightarrow A$ is the policy that determines the action taken by the agent at each state.

In direct policy teaching, the interested party (IP) knows the MDP, M, and has a single target policy for the agent, π_t. The goal of the IP is to find some set of state incentives, $\Delta \in \mathbb{R}^{|S|}$, such that the optimal policy of the MDP $M_{R+\Delta} = \{S, A, (R+\Delta), P, \gamma\}$ is π_t. These incentives model environment modifications that increase the agent's reward for each state by the incentive amount.

Value based policy teaching [12] does not have a specific target policy for the agent, but instead uses a reward function for the IP, $G \in \mathbb{R}^{|S|}$. While the agent forms a policy that maximises the expected sum of discount rewards on $M_{R+\Delta}$, the interested party values the policy according to the expected sum of discounted rewards on $M_G = \{S, A, G, P, \gamma\}$.

In both direct and value based policy teaching, incentives are limited to *admissible incentives* [11]. This limits individual incentives to be non-negative (i.e. no punishments) and the expected discounted sum of incentives must be no greater than some value D_{max}. Stated formally, Δ is admissible if it satisfies the following constraints:

$$\begin{aligned} &\Delta(s) \geq 0 && \forall s \in S \\ &V^{\pi}_{\Delta}(start) \leq D_{max} \end{aligned}$$

where the expected discounted sum of incentives $V^{\pi}_{\Delta}(s)$ for an agent's policy π is defined as:

(1)

$$V^{\pi}_{\Delta}(s) = \Delta(s) + \gamma \sum_{s' \in S} P(s, \pi(s), s') V^{\pi}_{\Delta}(s') \quad \forall s \in S \quad (2)$$

Zhang and Parkes provided a linear programming solution for direct policy teaching with environment design [11]. This solution minimises the expected incentive spending, $V^{\pi_t}_{\Delta}(start)$, while satisfying constraints both for admissibility and those found through inverse reinforcement learning (IRL) [7]. IRL takes a policy, π, and an MDP without reward function, M_{-R} and calculates a space of reward functions, $IRL^{\pi} \subseteq \mathbb{R}^{|S|}$, such that any reward function in this space has an optimal policy π. This space is defined by the following constraints:

$$(P_{\pi_t} - P_a)(I - \gamma P_{\pi_t})^{-1} R \succeq 0 \ , \ \forall a \in A$$

where P_a and P_{π_t} are the state probability transfer matrices for action a and target policy π_t, respectively. The target policy becomes *uniquely* optimal in $IRL^{\pi} \subseteq \mathbb{R}^{|S|}$ if the inequality is replaced with a strict inequality.

The direct policy teaching linear program finds the incentive function with minimal expected cost that, when added to the agent's rewards, is in the space of reward functions determined by IRL. Zhang and Parkes also provide a mixed integer program for single-agent, value-based policy teaching [12]. Instead of minimising incentive costs, this program maximises the interested party's value of the policy taken by the agent. However, for the remainder of this paper we will focus on *multiagent* policy teaching.

2.2 Related Work

At a high level, multiagent policy teaching is related to the game theoretic technique of *mechanism design* [8]. Mechanism design seeks to achieve certain behaviours in the agents of a system by defining the rules of the system. This is similar to policy teaching and environment design, which seeks to achieve a specific behaviour in one or more agents by modifying the environment of the agents. However, mechanism design is more concerned with the interactions between agents, and in this paper we will assume that agents do not directly interact.

Policy teaching is also related to applications such as apprenticeship learning. Apprenticeship learning uses the actions of an expert to guide the behaviour of an agent. Abbeel and Ng [1] provided a method of extracting an expert's reward function given the expert's behaviour, and used this to determine the behaviour of an agent. The authors extend this work to solve apprenticeship learning when dynamics (state transition probabilities of the MDP) are unknown [2]. Syed and Schapire [9] modify the work of Abbeel and Ng to provide a method that not only learns from the expert, but also attempts to find a better policy than the expert. However, these apprenticeship learning techniques assume we have complete control over the agent's reward function, while we will be interested in only allowing limited changes to agent's rewards.

The problem examined by Monderer and Tennenholtz [6] does not assume complete control over an agent's reward function as in apprenticeship learning, but does still allow unlimited, non-negative incentives. In k-implementation, an interested party influences the behaviour of agents in a game through the use of incentives. This is closely related to multiagent policy teaching. However, the models of the environment and the incentives are different in k-implementation and our setting. Firstly, k-implementation deals with games of interaction between the agents in a single-shot setting, whereas policy teaching works in a sequential planning setting. More significantly, incentives in k-implementation are provided to agents based on their particular strategy, rather than state-based incentives. These incentives are conceptually in the form of bonus payments made specifically to agents rather than environmental modifications, and as such each agent can receive different incentives.

Eidenbenz et al. [4] discuss how incentives added to a game can be used to manipulate the outcome. An interested party can modify the outcome both to increase or to decrease the social welfare of the agents. In this paper, however, we assume the interested party has no interest in the actual utility of the agents. The interested party merely assumes the agents will act so as to maximise their utility in the current environment.

3. MULTIAGENT POLICY TEACHING

In *multiagent policy teaching*, the goal of the interested party is to direct a set of agents to a particular policy by adding incentives to the environment. When moving to the multiagent setting, several additional challenges occur that were not originally present in the single-agent approach. Incentives are no longer specifically tailored for a particular agent, but rather for several agents. A key point in the multiagent model is that each agent acts in the same environment and thus has the same incentives. Due to this, the multiagent problem cannot be solved by simply running the single agent policy teaching method separately for each agent.

With direct policy teaching, the IP has a single policy it wishes all the agents to adopt. However, in a multiagent setting, the IP may not be able to teach the target policy to all agents simultaneously. In this case the IP can instead look for the largest subset of agents that can be taught the target policy with a single incentive function. Alternatively, the IP can use a value-based approach, which we discuss in Section 4.

3.1 The Multiagent Model

Before progressing any further, we explicitly outline the model and assumptions used in this paper. The multiagent extension of policy teaching and environment design in this paper deals with *non-interacting agents*. Agents that interact must anticipate the strategies of the other agents and act accordingly. With non-interacting agents, each agent only needs to consider its own actions and the current environment. Without interaction, we can model all agents as planners in MDPs that have the same $M_{-R} = \{S, A, P, \gamma\}$. However, each agent, i, has a different reward function, R_i, and it is assumed that these values are known to the interested party.

The abilities of the interested party remain unchanged from the single agent setting. The IP performs policy teaching though environment design by adding an incentive function Δ to the reward function of each agent. However, this incentive function must be applied to all agents equally, rather than through individual incentive functions. The motivation for this restriction is that, conceptually, the incentives are modifications to the environment rather than additional rewards provided separately by the IP to the agents. Thus, a particular environmental modification affects all agents.

3.2 Teaching All Agents

If there is a single incentive function that can lead all agents to the desired policy, this can be found with a simple linear program. This linear program has the same number of constraints and variables as in the single agent case, regardless of the number of agents.

Let P_a denote the state probability transfer matrix for playing action a in each state. Let P_{π_t} denote the state probability transfer matrix for following policy π_t. That is, $P_a(s_1, s_2)$ is the probability of transferring from state s_1 to

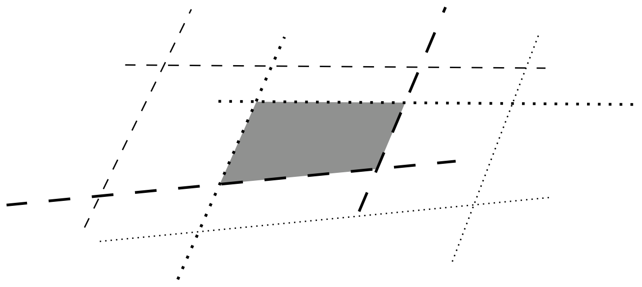

Figure 1: **The dotted lines and dashed lines are the constraints for two agents. Taking the most restrictive of each pair of parallel constraints defines the overlapping (shaded) region.**

s_2 given action a. $P_{\pi_t}(s_1, s_2) = P_a(s_1, s_2)\ \forall s_2$ if $\pi(s_1) = a$. For each agent $i \in N$, by applying inverse reinforcement learning, we obtain a set of up to $|S| \times |A|$ linear constraints. These are calculated as follows, starting with the constraints obtained from IRL [7].

$$\underbrace{(P_{\pi_t} - P_a)(I - \gamma P_{\pi_t})^{-1}}_{C_a}(R_i + \Delta) \succeq \epsilon\ ,$$

$$\begin{aligned}
& & \forall a \in A \setminus \pi_t(s), i \in N\\
&\Rightarrow C_a R_i + C_a\Delta \succeq \epsilon & \forall a \in A \setminus \pi_t(s), i \in N\\
&\Rightarrow C_a\Delta \succeq \underbrace{\epsilon - C_a R_i}_{D_{ia}} & \forall a \in A \setminus \pi_t(s), i \in N\\
&\Rightarrow C_a\Delta \succeq D_{ia} & \forall a \in A \setminus \pi_t(s), i \in N
\end{aligned}$$

Note that the $\epsilon > 0$ is to ensure that π_t is a unique optimal policy. If $\pi_t(s) = a$ then row $(P_{\pi_t}(s) - P_a(s)) = 0 \Rightarrow C_a(s) = 0$. Thus, these rows are removed from the set of constraints. All other rows ensure actions not in the target policy result in expected rewards that are at least ϵ lower than those of the target policy. This makes all π_t actions strictly preferred and so π_t is the unique optimal policy.

If we wish to find an incentive function to induce the desired policy in *all* agents, we must satisfy the $|S| \times |A|$ constraints for *each agent* simultaneously. That is, we must meet all $|S| \times |A| \times |N|$ constraints. However, an observation about the set of constraints allows us to greatly reduce the total number. Let $C_a(s)$ and $D_{ia}(s)$ denote the sth row of C_a and D_{ia} respectively. For any state s and action a the constraint $C_a(s)\Delta \geq D_{ia}(s)$ is parallel to $C_a(s)\Delta \geq D_{ja}(s)$ for every pair of agents $i, j \in N$, as $D_{ia}(s)$ is simply a scalar constant. This observation means that we can run a preprocessing step that reduces the number of contraints to $|S| \times |A|$ by keeping only the most restrictive constraint for each state-action pair (see Figure 1).

If we let

$$\hat{D}_a(s) = \max_{i \in N} D_{ia}(s) \qquad \forall a \in A, s \in S \qquad (3)$$

we get the following set of constraints:

$$C_a\Delta \succeq \hat{D}_a \qquad \forall a \in A \setminus \pi_t(s) \qquad (4)$$

Now, finding the minimal Δ to induce the desired policy

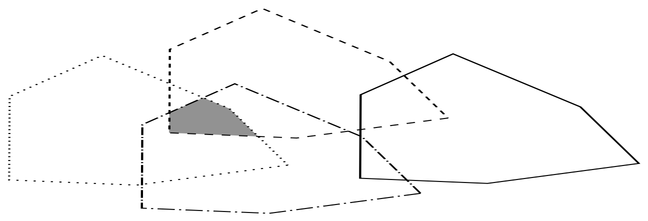

Figure 2: Each polygon represents the space of incentive functions that satisfy all constraints for a particular agent. There is no point that satisfies the constraints of all agents but the shaded region satisfies those of the greatest number of agents.

in all agents can be found with the following linear program:

$$\min_{\Delta} V_{\Delta}^{\pi_t}(start) \tag{5}$$

subject to:

$$\begin{aligned} &C_a \Delta \succeq \hat{D}_a && \forall a \in A \setminus \pi_t(s) \\ &0 \leq \Delta \leq \Delta_{max} \\ &V_{\Delta}^{\pi_t}(start) \leq D_{max} \end{aligned}$$

The final two constraints ensure the incentives are admissible. Here $V_{\Delta}^{\pi_t}(start)$ is defined as in Equation (2), as the expected discount incentive spending. The linear program has the same number of constraints and variables as the solution for a single agent [11], and this remains constant with respect to the number of agents.

3.3 Determining If All Agents Can Be Taught

As shown in the previous section, we can define the space of possible incentives that induce the target policy in a particular agent. Combining all the constraints gives the space of possible incentives that induce the target policy in all agents. This space is defined by the constraints in Equation (4).

If these constraints, coupled with the admissibility constraints, define an empty region, then no such admissible incentive function exists that can teach all agents the desired policy (as in Figure 2). Linear programming techniques such as the simplex method or extensions to the interior point method will detect the infeasibility of a set of constraints [10]. If, when running the linear program in Equation (5) on the set of constraints for all agents, it declares the problem infeasible, then no incentive can teach the policy to all agents. However, if all agents can be taught, the linear program will return the incentive function that achieves this.

3.4 Teaching The Largest Subset

Given a set of agents, N, where $|N| = n$, the interested party may desire to find the largest subset of N such that all agents follow the desired policy, given an incentive function. Geometrically, let $K = \{K_1, K_2, \ldots, K_n\}$, where K_i is the space of admissible incentives functions that induce the desired policy for agent $i \in N$. The goal is to find the largest subset of K such that the intersection of all elements (e.g. the shaded region in Figure 2) is non-empty.

Given the results in Subsection 3.3, a brute force approach to this problem is to find the largest set of agents that leads to a feasible linear program by testing all possible subsets of agents in order of decreasing cardinality. As soon as a feasible linear program is found based on the constraints of the subset of agents, the algorithm returns the solution to the linear program. If a specific application is likely to only have feasible subsets of size strictly less than $\frac{|N|}{2}$, then the algorithm should test subsets in increasing order of cardinality.

4. MULTIAGENT VALUE-BASED POLICY TEACHING

In the previous section, the interested party was only concerned with agents following the target policy exactly. Where the requirements of the interested party aren't so strict, a value based approach to policy teaching is more appropriate. As an example, consider a simple grid world where an agent's actions allow it to move to adjacent tiles in the grid. The interested party has a desired path for the agents to traverse the grid from the start state to a goal tile. Instead of only accepting solutions where the agents follow the path precisely, a value-based approach permits deviations. The larger the deviation from the target policy or behaviour, the lower the value the IP places on the agent's policy.

In the value-based setting, the IP has a policy valuation function $V : A^S \to \mathbb{R}$ that assigns a value to each policy. A state-based valuation has a value for each state in the MDP, and calculates a policy value based on the states it visits. An alternative measure is to evaluate each policy based on its *behavioural similarity* to the target policy. In such a setting, the IP needs a function to aggregate these individual policy valuations for the group of agents. While this is not an issue in the single agent setting as there is only one policy to evaluate, it becomes very important in the multiagent case. There are two intuitive methods of determining a total value for a set of policies, motivated by social welfare functions. Where the policy valuation directly represents the utility of the interested party, it makes sense to use a utilitarian approach and sum over all valuations. Thus, the value of a set of policies $\{\pi^i\}$ is calculated as $\sum_{i \in N} V(\pi^i)$. Alternatively, the interested party may prefer to have reasonable values for all agents rather than high values for some and poor values for others. In this case, the egalitarian value of a set of policies is determined by the worst policy, that is $\min_{i \in N} V(\pi^i)$.

4.1 State-Based Policy Valuation

In state-based policy valuation, the value of a policy is calculated as the discounted sum of future rewards, using the reward function of the interested party. This approach was used in the single agent setting by Zhang and Parkes [11]. A multiagent solution is simply a basic extension to the mixed integer program (MIP) used for the single agent problem.

Unlike in direct policy teaching, the additional constraints for multiple agents can not be trivially collapsed together. The MIP needs to calculate the value of each agent's policy, which also requires constraints for each agent to determine this policy. However, as the agents do not interact, the number of constraints and variables only grow linearly with the number of agents. The multiagent MIP has a copy, for each agent, of all the constraints of the single agent solution, except for those bounding the state incentive values, $\Delta(s)$. The minimised function of the MIP aggregates the policy values for all agents, for example, by summing over all values.

4.2 Comparison-Based Policy Valuation

In some cases, the interested party has a single desired policy or behaviour, but may be satisfied if agents follow a *similar* policy. In such a setting, it is easier for the interested party to appraise policies using some similarity score to the target function than to calculate a state-based reward function that captures this. Unfortunately, there is no clear definition of policy similarity. For policy teaching, we propose that appropriate comparison methods involve looking at the short-, medium-, or long-term behavioural similarities of the policies. In the rest of this section we describe our proposed methods.

Two policies are similar in short-term behaviour if their state transition probabilities are close for most states. This captures similarity in how an agent moves from each particular state and takes into account different actions that have similar state transition probabilities. Stated formally, the short-term difference between policies π_1 and π_2 can be defined as $\sum_{i,j\in S}\left(P_{\pi_1}(i,j)-P_{\pi_2}(i,j)\right)^2$, where $P_\pi(i,j)$ is the probability of transitioning from state i to state j playing action $\pi(i)$. To convert this difference into a policy value based on similarity to a target policy π_t we have:

$$V(\pi)=e^{-\sum_{i,j\in S}\left(P_{\pi_t}(i,j)-P_\pi(i,j)\right)^2}$$

Another measure of similarity between policies is based on the medium-term behaviour. In this metric, two policies are similar if they follow similar paths through the state space, or visit states with similar probabilities. An example where such a measure is useful is when an interested party wants the agents to explore a grid world. The desired policy may visit every state with probability at least p after t time steps. There are multiple ways of exploring a grid world, and the interested party is happy with any policy that visits each state with probability close to or greater than p after t time steps. This behaviour is less well defined and thus more difficult to quantify formally.

The long-term behaviour of a policy is measured by the probability distribution over final states as the number of time steps approaches infinity. In this case, the interested party is only concerned with where the agents are likely to end up and not how they get there. A policy's transition matrix P_π defines a Markov chain, and the long term behaviour can be captured by the stationary distribution, r.

Given the stationary distribution for two policies, r_{π_1} and r_{π_2}, the long-term difference between the two policies can be defined as sum of squared difference in stationary distributions: $\sum_{s\in S}|r_{\pi_1}(s)-r_{\pi_2}(s)|^2$. A policy valuation for a policy π compared to a target policy π_t based on long-term behaviour is:

$$V(\pi)=e^{-(r_{\pi_t}-r_\pi)(r_{\pi_t}-r_\pi)^T} \tag{6}$$

For policy teaching, the incentive function is found by maximising some enumeration of policy valuations over all agents. Using the valuation in Equation (6), the maximisation is:

$$\max_{\Delta}\sum_{i\in N}e^{-(r_t-r_i)(r_t-r_i)^T}$$

where r_t is the stationary distribution of the target policy π_t, a constant value, and r_i is the stationary distribution

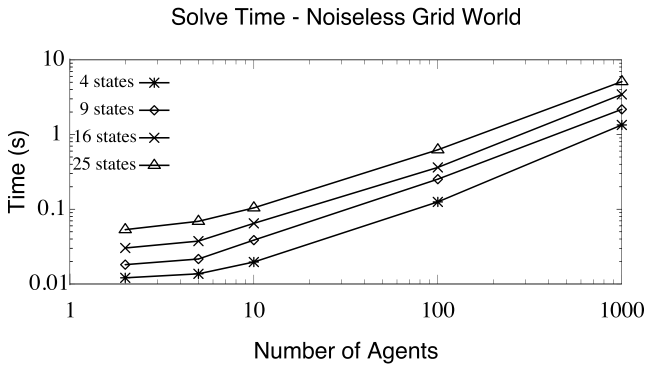

Figure 3: Time taken to find the incentive function to induce a target policy in all agents.

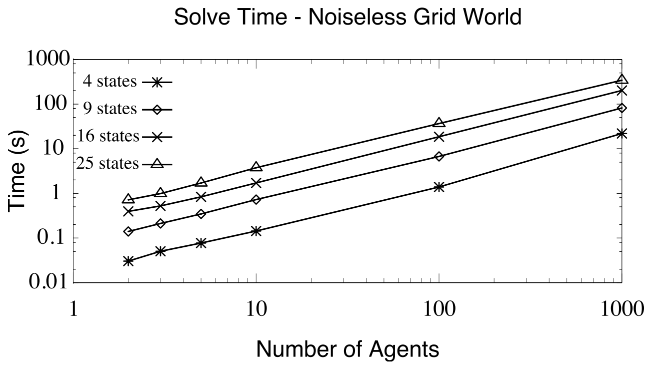

Figure 4: Time taken to find incentives that maximise long-term behavioural similarity of the agents to the target policy.

for the optimal policy played by agent i on the MDP with added incentives Δ.

While this can be converted into a mixed integer program, it requires $O(|S|^2|A||N|)$ constraints and variables. Given that mixed integer programs are typically NP-hard to solve [3], this solution rapidly becomes infeasible. This complexity is further worsened using more sophisticated objective functions that appraise agent policies according to a mix of short-, medium-, and long-term behaviour.

As shown in the next section, good empirical performance is possible by optimising incentives through metaheuristic methods, such as simulated annealing [5].

5. EXPERIMENTAL RESULTS

We conducted empirical tests to verify the solution quality and time requirements of our methods. All tests were run using GNU Octave version 3.0.3 on a 2.4 GHz MacBook Pro with 2 GB of RAM.

The test domain is that of a simple grid world. An agent can either move in one of the four cardinal points or stay put. There is uncertainty added to the environment, where an agent may move in a direction different to the action.

First, we tested the linear program for teaching all agents (Equation (5)). The target policy in these tests was to take the shortest path from the current state to the centre of the grid (with ties broken arbitrarily). The agents' reward functions were generated randomly, with one random "preferred" state having a value of 1 and all others having a value drawn uniformly from $[0, 0.001]$. The discount factor γ was 0.99. Figure 3 shows the time required to solve the linear program with varying numbers of states and agents. The time required was roughly linear on the number of agents due to calculating the maximum values in Equation (3). The value of D_{max} was chosen such that it was possible to teach all agents the desired policy. In each run, all agents followed the target policy precisely. The difference in solving time on an MDP where an action always moved in the intended direction and where an action had a 10% chance of moving in an unintended direction was not significant.

We also tested the efficacy of a simulated annealing approach to solving comparison-based multiagent policy teaching, as presented in Subsection 4.2. These tests used the same set up of target policy and rewards as in the linear program tests. However, in this comparison-based approach, we looked at policy similarity based on long-term behaviour (using Equation (6)). A high-value policy should have a high long-term probability of being in the centre state with lower probabilities further from the centre. The plot in Figure 4 shows the time required to find the optimal set of incentives, where the algorithm was halted as soon as the method had found a solution within 0.01% of the optimal value. Here, the optimal value was when all agents follow the target policy precisely, and each had a value of 1. The tests used for Figure 4 aggregated agent policy values through summation, but tests using an "egalitarian" measure showed the same linear trend with number of agents. As the simulated annealing algorithm is non-deterministic, the values plotted are the average over 100 different runs. These results show that, while simulated annealing doesn't have the guaranteed performance of a mixed-integer formulation, it is still able to find good solutions in reasonable time.

6. CONCLUSIONS AND FUTURE WORK

This paper examined multiagent extensions to environment design and policy teaching. These procedures allow an interested party to achieve a desired behaviour in a set of agents through environmental modifications. The environmental modifications are in the form of value-increasing incentives added to each state in the environment.

In the case where a policy is being taught to all agents, the paper provided a linear program to solve this problem. The number of constraints and variables in the linear program does not change as the number of agents increases. The linear program is feasible if and only if the policy can be taught to all agents with a single incentive function.

This paper also examined the problem of maximising the aggregate value of agents' policies, from the perspective of the interested party, in particular, when the value of a policy reflects its similarity to some target policy. Policy similarity is defined as the behavioural similarity of two policies, ranging from short-term to long-term behaviour. A comparison-based approach gives added flexibility to the interested party, as a target policy may not need to be explicitly stated. Using long-term behavioural similarity, the IP can compare policies to a target stationery distribution rather than a target policy. The aggregation of values for each agent's policy can be a summation of values, or the minimum of all values. This problem can be solved with metaheuristic optimisation techniques, as a mixed integer programming solution rapidly becomes infeasible. Empirical tests showed that simulated annealing is one such technique that is effective in solving this problem.

A key assumption in this paper was that of no agent interaction. There is added complexity when agents interact, in modelling the environment, determining policies and finding incentives. In this setting, the MDP model is no longer sufficient, but a stochastic game framework would be one appropriate option to examine. Stochastic games are sufficiently general that a conceptually simple, but perhaps computationally difficult environment design method could be developed by adding the interested party as an extra agent to the game. In this method, the interested party's actions do not affect agents' payoffs in the current stage game, but lead to different sets of stage games with modified agent rewards.

Another assumption that can be relaxed in future work is that of complete knowledge of agent rewards. If rewards are unknown, then the policy teaching methods need to incorporate preference elicitation techniques to narrow the space of possible rewards. A multiagent extension of active indirect elicitation [11] will let the interested party narrow the space of possible agent rewards until enough information is known to determine appropriate incentives.

7. REFERENCES

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 1, New York, NY, USA, 2004. ACM.

[2] P. Abbeel and A. Y. Ng. Exploration and apprenticeship learning in reinforcement learning. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 1–8, New York, NY, USA, 2005. ACM.

[3] R. E. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. Mixed-integer programming: A progress report. In *The Sharpest Cut: The Impact of Manfred Padberg and His Work*. SIAM, 2004.

[4] R. Eidenbenz, Y. Oswald, S. Schmid, and R. Wattenhofer. Manipulation in games. *Algorithms and Computation*, pages 365–376, 2007.

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[6] D. Monderer and M. Tennenholtz. k-implementation. In *EC '03: Proceedings of the 4th ACM conference on Electronic commerce*, pages 19–28, New York, NY, USA, 2003. ACM.

[7] A. Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *in Proc. 17th International Conf. on Machine Learning*, pages 663–670. Morgan Kaufmann, 2000.

[8] S. Parsons and M. Wooldridge. Game theory and decision theory in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 5(3):243–254, 2002.

[9] U. Syed and R. Schapire. A game-theoretic approach to apprenticeship learning. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 1449–1456. MIT Press, Cambridge, MA, 2008.

[10] M. J. Todd. Detecting infeasibility in infeasible-interior-point methods for optimization. In

Foundations of Computational Mathematics, Minneapolis 2002, London Mathematical Society Lecture Note Series 312, pages 157–192. University Press, 2004.

[11] H. Zhang and D. C. Parkes. Enabling environment design via active indirect elicitation. In *Proc. Workshop on Preference Handling*, Chicago, IL, July 2008.

[12] H. Zhang and D. C. Parkes. Value-based policy teaching with active indirect elicitation. In *Proc. 23rd National Conference on Artificial Intelligence (AAAI'08)*, Chicago, IL, July 2008.