Determining and Verifying Good Policies for Cloned Teleo-Reactive Agents

Krysia Broda and Christopher John Hogger

Department of Computing, Imperial College London
180 Queen's Gate, London SW7 2BZ, UK
kb@doc.ic.ac.uk, cjh@doc.ic.ac.uk

Abstract. This paper presents a new method for finding and evaluating policies for cloned multiple teleo-reactive agents operating in the context of exogenous events. The method is based upon discounted reward evaluation of policy restricted situation graphs. A teleo-reactive agent behaves autonomously under the control of a policy and is predisposed by that policy to achieve some goal. Our framework plans for a set of conjoint agents by focusing upon one representative of them and introduces generic situations in order to reduce the number of policies. Simulation results reported here indicate that our method affords a good degree of predictive power and scalability.

1 Introduction

This paper examines the problem of designing optimal or near-optimal policies for a group of *teleo-reactive* (TR) agents operating in the context of *exogenous* events. From the viewpoint of any individual agent an exogenous event is any change in the world not caused through its own actions. The characteristics of a TR agent are, typically, that it behaves autonomously under the control of a stored program or policy, that the policy alone instructs it how to react to perceptions of the world, that it possesses very limited computing resources for program storage and interpretation and that it is predisposed by the policy to achieve some goal. The main features of TR agents were introduced in [17] and further developed in [18]. We make two key assumptions about an agent: that it has (i) little or no access to cognitive resources, such as beliefs or reasoning systems, and (ii) only partial observational capability, in that its perceptions may not fully capture the whole environmental state. Such agents could find uses in those applications, for instance nano-medicine or remote exploration, where physical or economic constraints might preclude such access. Informally, a good policy is one which disposes an agent to perform well in pursuit of a defined goal whatever state it is currently in.

A reactive agent responding only to its current perception of the state can be modelled naturally by a Markov Decision Process (MDP). When that perception captures less than the entirety of the state the agent may be modelled using the framework of Partially Observable MDPs (POMDPs) [6, 10, 16]. Seeking good

policies for a given agent within the POMDP framework typically requires, in order to compensate for the lack of full state information, that the actual state be estimated using a history of previous events. When the history is owed *only to actions of known agents*, it is accurately representable. However, it is not accurately representable if arbitrary exogenous events can occur, which is exactly the case in which we, by contrast, are interested. Our approach is somewhat similar in motivation to that of [7], in that we plan for a set of conjoint agents by focusing upon some representative subset of them, or of [9], in that we focus on changes that affect a particular agent.

Our position must also be distinguished from that taken by many in the multi-agent community who equip their agents with complex theories about the environment and elaborate communication mechanisms, because of our focus on minimally equipped agents. Nevertheless, the problem remains of dealing with an agent having limited perception.

Choosing a good policy for a given goal is generally difficult. Even simple agents and worlds can offer huge numbers of policies to consider. Most ways of designing TR-policies are learning-based, as in [1] and [2], which apply inductive logic programming to determine advantageous associations between actions and consequences. Other learning schemes for TR-planning are [11], [15] and [20].

The core concepts in our approach, first proposed in [3], are as follows. The world in which the agent operates has a total set $\mathcal{O}$ of possible *states*. The agent is presumed to possess perceptors through which it may partially perceive these states; the set of all its possible *perceptions* is denoted by $\mathcal{P}$. The agent is also presumed capable of certain *actions* which form a set $\mathcal{A}$. In any state $o \in \mathcal{O}$, the agent's possible perceptions form some subset $P(o) \subseteq \mathcal{P}$, and in response to any perception $p \in P(o)$ the agent's possible actions form some subset $A(p) \subseteq \mathcal{A}$. A *policy* for the agent is any total function $f : \mathcal{P} \rightarrow \mathcal{A}$ satisfying $\forall p \in \mathcal{P}, f(p) \in A(p)$. The number of possible policies is the product of the cardinalities of the $A(p)$ sets for all $p \in \mathcal{P}$. A *situation* for the agent is any pair (o, p) for which $o \in \mathcal{O}$ and $p \in P(o)$. We denote by $\mathcal{S}$ the set of all possible situations, one or more of which may be designated *goal* situations. Associated with the agent is a unique *unrestricted situation graph* G in which each node is a situation. This graph has an arc labelled by an action a directed from node (o, p) to node (o', p') in every case that $a \in A(p)$ and execution of action a could take the state from o to o' and the agent's perception from p to p'. The possibilities that $o = o'$ and/or $p = p'$ are not inherently excluded. The policies are evaluated using discounted-reward principles [10] applied to the situation graphs.

A key feature of this approach is that the graph for the agent can represent, besides the effects of its own actions, the effects of exogenous actions, whether enacted by other agents or by other indeterminate agencies. This obviates analysing comprehensively the explicit combinations of all the agents' behaviours. This treatment is one contribution to the control of scalability and is in contrast to that presented in [19], who explicate entirely the joint perceptions of the agents. Scalability can be further controlled by abstraction, by way of replacing sets of concrete situations by abstract generic situations (see

Section 7). Situation graphs have also been employed to represent and exploit inter-agent communication [4].

The framework is not restricted to dealing with agents inhabiting physical worlds. It may instead be used as a particular kind of program development paradigm. In this context the world operated upon consists of data structures (or knowledge representations), whilst the agents serve as algorithms acting upon those structures. For instance, the world may consist of an array of values and each agent may be capable of transforming some region of that array within its prescribed range of perception. We have applied our framework successfully to simple cases of sorting and tiling problems, producing optimal agent policies which, under the control of a fixed governing meta-program, enact efficient strategies for solving those problems. Similar examples were investigated in [12], in which the authors report learning various "unusual algorithms" for sorting an array.

This paper demonstrates the use of situation graphs to predict optimal or near-optimal policies for groups of cloned TR-agents, and assesses their effectiveness on the basis of simulation results. The main contribution is to show that useful designs for agent clones can be obtained by analysing a single agent acting in the context of events instigated by others. Section 2 illustrates how TR-scenarios are formulated and Section 3 discusses the evaluation and testing of policies. In Section 4 we describe the treatment of multi-agent contexts and some case-studies are presented in Section 5. Section 6 uses the results of those studies to discuss some ramifications of the design method and Section 7 discusses an approach to abstractions. Finally, Section 8 summarizes our conclusions and compares our work with other approaches.

2 An Illustration

The above ideas are now illustrated using *BlocksWorld* for a single agent case. Of course, *BlocksWorld* is just a generic exemplar of a wide range of state transition systems. In *BlocksWorld* the world comprises a surface and a number of identical blocks. A state is an arrangement of the blocks such that some are stacked in towers on the surface whilst others are each held by some agent, and is representable by a list of the towers' heights. Suppose there are just 2 blocks and one agent which can, at any instant, see just one thing – the surface, a 1-tower or a 2-tower, denoted by $s0$, $s1$ and $s2$ respectively. Further, it can sense whether it is holding a block or not doing so, denoted by h and nh respectively. Its perception set $\mathcal{P}$ then comprises just 5 legal pairings of "seeing" status and "holding" status. At any instant the agent performs whichever action $a \in \mathcal{A}$ corresponds, according to its policy, to its current perception $p \in \mathcal{P}$; this behaviour is assumed to be durative – the action cannot change while perception p persists. The action set $\mathcal{A}$ in *BlocksWorld* is $\{\mathtt{k}, \mathtt{l}, \mathtt{w}\}$. In a `k`-action ("pick") the agent removes the top block from a tower it is seeing, and afterwards is holding that block and seeing the resulting tower (or the surface, as appropriate). In an `l`-action ("place") the agent places a block it is holding upon what it is

seeing (the surface or some tower), and afterwards is not holding a block and is seeing the resulting tower. In a `w`-action ("wander") the agent merely updates its perception without altering the state. Figure 1 shows these aspects of the formulation, labelling the states $1 \ldots 3$ and the perceptions $a \ldots e$.

(a) states and perceptions

o		$P(o)$
1	[1, 1]	$\{a, b\}$
2	[1]	$\{d, e\}$
3	[2]	$\{a, c\}$

(b) perceptions and actions

p		$A(p)$
a	$[s0, nh]$	{`w`}
b	$[s1, nh]$	{`k`, `w`}
c	$[s2, nh]$	{`k`, `w`}
d	$[s0, h]$	{`l`, `w`}
e	$[s1, h]$	{`l`, `w`}

Fig. 1. Formulation of the 2-block world

Figure 2a shows the unrestricted graph G, where each node (o, p) is abbreviated to op. There are 16 policies, one being

$$f = \{a \to \mathtt{w},\quad b \to \mathtt{w},\quad c \to \mathtt{w},\quad d \to \mathtt{w},\quad e \to \mathtt{l}\}$$

To adopt this policy is to eliminate certain arcs from G to leave the *f-restricted graph* G_f shown in Figure 2b. G_f gives the transitions the agent could make under policy f. In this case each action is *deterministic*. More generally, at least some actions are *non-deterministic*. For instance, in the 4 block world, shown in Figure 4, in the situation $([1, 3], [s0, nh])$ (*i.e.* $3i$), if the policy specified the `w` action for perception i, then the agent could afterwards be seeing either a 1-tower (in situation $3d$) or a 2-tower (in situation $3f$). For a given goal, choosing f

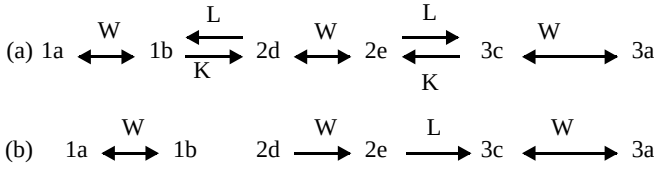

Fig. 2. (a) Unrestricted graph G; (b) f-restricted graph G_f

partitions the set $\mathcal{S}$ of nodes in G into two disjoint subsets T_f and N_f called the *trough* and *non-trough* respectively. T_f comprises exactly those nodes from which there is no path in G_f to the goal, and N_f comprises all others. Choosing, say, the goal $3c$ in the example yields $T_f = \{1a, 1b\}$ and $N_f = \{2d, 2e, 3c, 3a\}$. Here, no arc in G_f is directed from N_f to T_f. When such an arc does exist, G_f is described as NT-bridged. In that case an agent able in principle to reach the goal from some initial node may traverse the bridge and thereafter be unable to reach the goal,

since no arc directed from T_f to N_f can exist. Define $\lambda = 100|N_f|/|\mathcal{S}|$. Then for a series of experiments choosing initial situations randomly, the agent's predicted success rate is the proportion of experiments for which the agent reaches the goal. This is denoted by $SR_{\text{pre}}(f)$ and (as a percentage) is $< \lambda$ if G_f is NT-bridged but is otherwise exactly λ. For the example above $SR_{\text{pre}}(f) = 66.67\%$.

3 Predicting and Testing Policies

This section explains how policies are evaluated in a single-agent context. The extension to the multi-agent context then follows in the next section.

The predicted value $V_{\text{pre}}(f)$ of a policy f is the mean of the values of all situations in G_f, assuming they are equally probable as initial ones. In case some "initial situations" are known to be more likely than others they are weighted accordingly. The value of a situation S should reflect the reward accumulated by the agent in proceeding from S, favouring goal-reaching paths over others. If S has immediate successor-set SS then its value $V(S)$ can be measured by the discounted-reward formula

$$V(S) = \Sigma_{u \in SS}(p_u.(rwd(u) + \gamma.V(u))) \tag{1}$$

where p_u is the probability that from S the agent proceeds next to u, $rwd(u)$ is the reward it earns by doing so and γ is a discount factor such that $0 \leq \gamma < 1$. In case S is a situation with no successors Equation 1 forces $V(S) = 0$. This is made the case for a goal situation, reflecting that we are evaluating only the policy's ability to cause agents to reach the goal and not what may occur afterwards. In case S is a situation with a single successor transition to itself (called a *reflexive* arc), Equation 1 yields the value $r/(1-\gamma)$ for S. In addition, if every trough node has a successor then Equation 1 yields the value $r/(1-\gamma)$ for each one. In a single-agent context equal probabilities are assigned to those arcs emergent from S. We employ two fixed numbers R and r such that $rwd(S) = R$ if S is a goal and $rwd(S) = r$ otherwise, where $R \gg r$. The situations' values are therefore related by a set of linear equations which, since $\gamma < 1$, have unique finite solutions and so determine a finite value for $V_{\text{pre}}(f)$. In general, choosing $R \gg r$ ranks more highly those policies well-disposed to the reaching of the goal, whilst γ controls the separation (but not, in general, the ranks) of the policies' values. A *Policy Predictor* program is used to compute policy values by this method, and also to compute (as above) the upper bounds (λ) on their success-rates.

We tested the quality of the *Policy Predictor* by using a *Policy Simulator* program. Each *run* of the simulator takes an initial situation S and subsequently drives the agent according to the given policy f. The simulated agent then implicitly traverses some path in G_f from S. The run terminates when it either reaches the goal or exceeds a prescribed bound B on the number of transitions performed. As the path is traversed, the value $V(S)$ is computed incrementally on the same basis as used by the predictor. That is, in case the bound B is reached the situation S approximates a trough node and is given the value $r/(1-\gamma)$;

otherwise, the value is

$$1 + \gamma r + \ldots + \gamma^k R = r(1 - \gamma^k)/(1 - \gamma) + R\gamma^k$$

since all rewards have value r except the last, which has the value R. Equal numbers of runs are executed for each initial situation S and the mean of all observed $V(S)$ values gives the observed policy value $V_{\text{obs}}(f)$. The simulator also reports the observed success rate $SR_{\text{obs}}(f)$.

The simulator supports both *positionless* and *positional* simulation modes. For instance, in *BlocksWorld*, this means the former associates no positional data to the agent or the towers, and so uses the same information about the problem as the predictor. Thus, if the agent picks a block from the 2-tower in the state [1, 1, 2], it is indeterminate as to which tower it will see afterwards in the new state [1, 1, 1]. By contrast, the positional mode assigns discrete grid coordinates to the agent and the towers, and exploits these to effect transitions in a manner somewhat closer to physical reality through knowing precisely where the agent is located and what it is seeing. In the positionless case the graph does not have any reflexive wanders, since such arcs would not correspond to any physically meaningful action. On the other hand, they are included in the positional case to represent wandering from one place to another. In this paper there are results for both kinds of simulation, all using parameter values $R = 100$, $r = -1$ and $\gamma = 0.9$.

To visualize the correlation of predictions with test outcomes for a set $\mathcal{F}$ of n policies, those policies' observed values are charted against the ranks of their predicted values. Overall predictive quality is reflected by the extent to which the chart exhibits a monotonically decreasing profile. A precise measure of this can be computed as the Kendall rank-correlation coefficient $\tau_{\mathcal{F}}$ for $\mathcal{F}$[21], as follows. Let (f, g) be any pair of distinct policies in $\mathcal{F}$ satisfying $V_{\text{pre}}(f) \leq V_{\text{pre}}(g)$. This pair is said to be concordant if $V_{\text{obs}}(f) \leq V_{\text{obs}}(g)$, but is otherwise discordant. Then $\tau_{\mathcal{F}} = 2(C - D)/n(n - 1)$ where C and D are the numbers of concordant pairs and discordant pairs, respectively, in $\mathcal{F} \times \mathcal{F}$. Its least possible value is -1 and its greatest is +1. It is convenient to map it to a percentage scale by defining $Q_{\mathcal{F}} = 0\%$ when they disagree maximally.

Figure 3 shows the chart for the 2-block example above, choosing the goal as $(3, c)$, i.e. "build and see a 2-tower". Observed policy values are measured along the vertical axis, and predicted ranks (here, from 1 to 16) along the horizontal one. Arcs emergent from $(3, c)$ were suppressed from G to mirror the fact that the simulator terminates any run that reaches a goal. For each policy the simulator executed 1002 runs with $B = 100$. The chart is perfectly monotonic and $Q_{\mathcal{F}} = 100\%$. The optimal policy is $\{a \rightarrow \texttt{w},\ \ b \rightarrow \texttt{k},\ \ c \rightarrow \texttt{w},\ \ d \rightarrow \texttt{w},\ \ e \rightarrow \texttt{1}\}$, which picks a block only from a 1-tower and places a block only upon a 1-tower. Its success rate is 100% and observed policy value is 73.05%.

4 Policies for Multiple Agents

Whether or not it is useful to employ multiple agents in the pursuit of a goal depends upon the nature of the world and of the agents' interactions with the

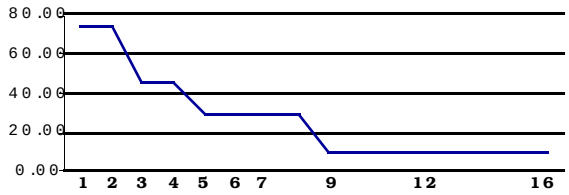

Fig. 3. Policy ranking chart for building a 2-tower

world and with each other. With limited perceptions, incognizance of the goal and lack of communication, simple TR-agents of the kind we have considered may cooperate advantageously only by serendipity. Predicting accurately the behaviour of multiple agents presents not only analytical difficulties, in attempting to assess the overall impact of their interactions, but also problems of scale – for just a modest case of a 4-block world with 2 agents there are potentially more than 13,000 policies to consider.

This section explains how a situation graph focusing upon one agent, termed "*self*", can express the effects of other agents acting in concert with it. The key question is whether such a graph enables prediction of good policies for the combined agents without requiring explicit analysis of all the combinations of the situations they occupy.

For this purpose we introduce a special domain-independent action named `x` that all agents possess in their $\mathcal{A}$ sets. So for *BlocksWorld* $\mathcal{A}$ is now $\{\mathtt{k}, \mathtt{l}, \mathtt{w}, \mathtt{x}\}$. In any situation graph an `x`-arc directed from (o, p) to (o', p') signifies that, from the viewpoint of *self*, the state transition from o to o' is effected *exogenously* by some other agent(s). So, when *self*'s policy prescribes an `x`-action for perception p, this is interpreted operationally as requiring *self* to "wait" (that is, to become inactive) until its situation is updated exogenously to (o', p'). On the other hand, when *self*'s policy prescribes some action other than `x` *self* may alternatively undergo an exogenous transition caused by the action of another agent. We call this *passive updating* of *self*.

It may be desirable to filter out policies that, for some reason or other, are unacceptable. For example, perhaps policies that would result in an agent never reaching the goal or policies that would result in assured deadlock are of this sort. We have considered a filter that reduces the chance of all agents waiting to be passively updated. One way to do this is to arrange that whenever an agent's policy requires it to wait, at least one of the transitions it awaits should be achievable in one step by some other agent. In the case that all agents have the same policy f (i.e. are clones) the filter is called the *clone-consistency principle* and operates as follows. Suppose f contains $p \rightarrow \mathtt{x}$ and G_f has an `x`-arc from (o, p) to (o', p'). In case G_f also has an arc from (o, q) to (o', q') labeled by an action `a` other than `x` or `w` and f contains $q \rightarrow \mathtt{a}$, then the `x`-transition is feasible and the arc is *enabled*. That is, *self* could be passively updated by another agent taking action `a` in situation (o, q). Any policy with rule $p \rightarrow \mathtt{x}$ for which no `x`-arcs

from situation (o, p) are enabled is called *clone inconsistent*. In this case, if an agent is in situation (o, p), then all agents would be stuck in state o, none able to change the state, although some may be able to move between perceptions in o. If all agents happened to be in (o, p) then true deadlock would ensue. Besides reducing the possibility of deadlock this principle also reduces very significantly the number of policies requiring to be examined.

In the example shown in Figure 4 a clone inconsistent policy is one which includes the rules $b \rightarrow \mathtt{x}$ and either $h \rightarrow \mathtt{x}$ or $h \rightarrow \mathtt{w}$. Consider an agent waiting in situation $(10, b)$. Other agents could be in situations $(10, h)$ or $(10, b)$. In neither of these situations can agent change the state, so the $\mathtt{x}$ transition to $(8, c)$ will never be taken. On the other hand, if the policy specified $h \rightarrow \mathtt{l}$, then the transition between $(10, h)$ and $(8, c)$ *is* possible. However, deadlock is not completely ruled out, since it might still happen that all agents were in situation $(10, b)$.

In the multiple-agent context the predictor assigns probabilities to the arcs of a graph G_f as follows. Each node $S = (o, p)$ has emergent arcs for the action $\mathtt{a}$ that *self* performs according to the rule $p \rightarrow \mathtt{a}$ in its policy f. If $\mathtt{a} = \mathtt{x}$ then these $\mathtt{x}$-arcs are the only arcs emergent from S. The predictor counts, for each one, the number of distinct ways its transition could arise, taking account of the number of other agents that could effect it. Their probabilities are then required to be proportional to these counts and to have sum $\Sigma_{\mathtt{x}} = 1$. If $\mathtt{a} \neq \mathtt{x}$ then S additionally has arcs labeled $\mathtt{a}$ and these are assigned equal probabilities having sum $\Sigma_{\mathtt{a}}$. In this case $\Sigma_{\mathtt{a}}$ and $\Sigma_{\mathtt{x}}$ are made to satisfy $\Sigma_{\mathtt{a}} + \Sigma_{\mathtt{x}} = 1$ and $\Sigma_{\mathtt{x}} = (n-1)\Sigma_{\mathtt{a}}$ where n is the total number of agents, reflecting the relative likelihood in situation S of *self* being the agent selected to act or one of the others being selected to act.

In the multiple-agent context the simulator effects transitions between *multi-situations*, which generalise situations. A multi-situation is a *physically possible* assignment of the agents to situations that share a common state and differ (if at all) only in perceptions. An example of a multi-situation for the 2 agent example shown in Figure 4 is $\{(r1, (3, d)), (r2, (3, d))\}$. On the other hand, $\{(r1, (3, d)), (r2, (3, a))\}$ is not a multi-situation as it is physically impossible.

A run in a simulation begins with an initial multi-situation chosen randomly from the set of all multi-situations distinguishable up to agent identity. Each subsequent transition from a state o is made by randomly choosing some agent and performing the action $\mathtt{a}$ prescribed by its policy, causing the state to become some o'. If $\mathtt{a} \in \{\mathtt{w}, \mathtt{x}\}$ then $o = o'$ and the other active agents' perceptions remain unchanged. If (for *BlocksWorld*), $\mathtt{a} \in \{\mathtt{k}, \mathtt{l}\}$, then $o \neq o'$ and the other active agents are passively updated in a manner chosen randomly from all possible ways of updating them to state o', according to the distribution of probabilities for action $\mathtt{a}$ taken in state o.

A transition from the multi-situation $\{(r1, (2, d)), (r2, (2, d))\}$ (again for the example shown in Figure 4) might be: $r1$ is selected, its policy (say) dictates action $\mathtt{k}$ and the new multi-situation would be either $\{(r1, (6, h)), (r2, (6, i))\}$ or $\{(r1, (6, h)), (r2, (6, d))\}$. The situation change for $r2$ in this transition is an

example of a passive update corresponding to an x-arc from either $(2, d)$ to $(6, i)$ or $(2, d)$ to $(6, d)$. The simulator chooses randomly from these two possibilities.

A run is terminated when any active agent in the group reaches a goal situation or when the simulation depth bound B is reached. The simulator's successive random choosing of which agent acts next provides an adequate approximation to the more realistic scenario in which they would all be acting concurrently. Owing to the physical constraint that there can be only one state of the world at any instant, any set of concurrent actions that produced that state can be serialized in one way or another to achieve the same outcome. The simulator's randomness effectively covers all such possible serializations.

An x-arc in the graph thus represents two notions at once. On the one hand it represents the action of deliberate waiting in accordance with *self*'s own policy. On the other hand it indicates how *self* can be impacted by the actions of others.

5 Multiple Clone Examples

o		$P(o)$
1	[2,2]	$\{e, i\}$
2	[1,1,1,1]	$\{d, i\}$
3	[1,1,2]	$\{d, e, i\}$
4	[1, 3]	$\{d, f, i\}$
5	[4]	$\{g, i\}$
6	[1,1,1]	$\{a, d, h, i\}$
7	[1, 2]	$\{a, b, d, e, h, i\}$
8	[3]	$\{c, f, h, i\}$
9	[1,1]	$\{a, h\}$
10	[2]	$\{b, h\}$

p		$A(p)$
a	$[s1, h]$	{l, w, x}
b	$[s2, h]$	{l, w, x}
c	$[s3, h]$	{l, w, x}
d	$[s1, nh]$	{k, w, x}
e	$[s2, nh]$	{k, w, x}
f	$[s3, nh]$	{k, w, x}
g	$[s4, nh]$	{k, w, x}
h	$[s0, h]$	{l, w, x}
i	$[s0, nh]$	{w, x}

Fig. 4. Formulation of 4-block world with 2 agents

All the examples in this section are for a world having 4 blocks (see the formulation in Figure 4), but with various goals and various numbers of cloned agents. The total number of policies is 13,122 but only 810 are clone-consistent. The simulator operated in positionless mode. Each of these 810 was given about 1000 simulation runs with $B = 100$, and the reward parameters used were $R = 100$, $r = -1$ and $\gamma = 0.9$.

Example 5.1: [*Two agents building a 4-tower*] The formulation is shown in Figure 4. The unrestricted graph G has 30 nodes, among which is the goal $(5, g)$. Figure 5 shows a fragment of G ($4d$ appears twice only for drawing convenience). Even this fragment has some incident and emergent arcs omitted from the figure.

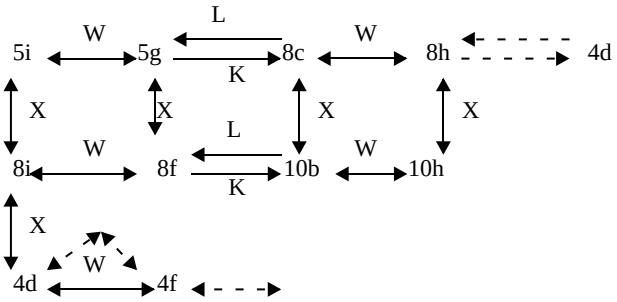

Fig. 5. Graph G for 4-blocks and 2 agents

Figure 6 shows the policy ranking chart for the set $\mathcal{F}$ of the best 240 policies. For these, $Q_{\mathcal{F}}$ is 87.90%. The best ones are ranked almost perfectly. The observed optimal one, having predicted rank 2, is

$$\{a \to \mathtt{l},\ \ b \to \mathtt{w},\ \ c \to \mathtt{w},\ \ d \to \mathtt{k},\ \ e \to \mathtt{k},\ \ f \to \mathtt{w},\ \ g \to \mathtt{w},\ \ h \to \mathtt{w},\ \ i \to \mathtt{w}\}$$

This picks only from a tower of height < 3 and places upon any tower of height < 4 (but not upon the surface).

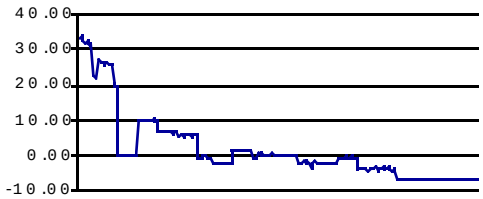

Fig. 6. Policy ranking chart for Example 5.1

Example 5.2: [*Three agents building a 4-tower*] The graph G is now a little larger, as there is an extra state. It has 36 nodes. Figure 7 charts the best 240 policies, for which $Q_{\mathcal{F}}$ is 89.94%. For the best 20 it is 85.79%. The observed optimal policy is the same as for 2 agents, and is the predicted optimal one.

Example 5.3: [*Four agents building a 4-tower*] The graph G now has yet one more state (in which every block is being held). It has 39 nodes. Figure 8 charts the best 240 policies, for which $Q_{\mathcal{F}}$ is 90.02%. For the best 20 it is again 85.79%. The observed optimal policy is the predicted optimal one, being

$$\{a \to \mathtt{l},\ \ b \to \mathtt{l},\ \ c \to \mathtt{l},\ \ d \to \mathtt{k},\ \ e \to \mathtt{w},\ \ f \to \mathtt{w},\ \ g \to \mathtt{w},\ \ h \to \mathtt{w},\ \ i \to \mathtt{w}\}$$

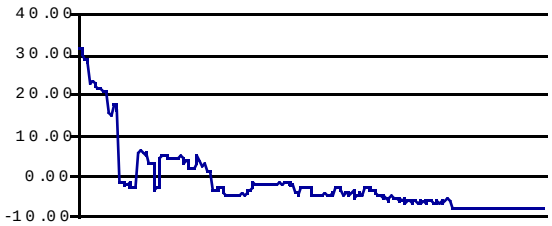

Fig. 7. Policy ranking chart for Example 5.2

which picks only from a tower of height 1 and places only upon a tower of height < 4. The best policy is slightly different from the three agent case as it is more likely that agents are holding blocks and consequently that towers of height 2 are less likely. If all 4 agents were treated individually, although there would still be 13122 policies, there would be several hundreds of situations, making each policy evaluation substantially more complex.

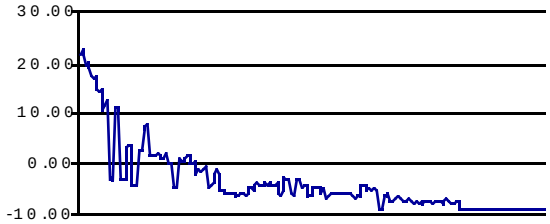

Fig. 8. Policy ranking chart for Example 5.3

Example 5.4: [*Two agents building a 2-tower and two 1-towers*] Here, G is the same graph as in Example 5.1. The goal is now $(3, i)$, i.e. "build one 2-tower, two 1-towers and see the surface". This example is different from the previous three, in that an agent cannot, using a single perception, recognise that the goal has been reached. Figure 9 charts the best 240 policies, for which $Q_{\mathcal{F}}$ is only 78.61%. For the best 20, however, it is 90.53%. The three observed best policies

$$\{a \to \mathtt{w},\ \ b \to \mathtt{w},\ \ c \to \mathtt{w},\ \ d \to \mathtt{w},\ \ e \to \mathtt{w},\ \ f \to \mathtt{k},\ \ g \to \mathtt{k},\ \ h \to \mathtt{l},\ \ i \to \mathtt{w}\}$$

(or $c \to \mathtt{x}$ or $c \to \mathtt{l}$) are the three predicted best ones, sharing the property that they pick only from a tower of height > 2 and can place upon the surface. They differ only in the case that an agent is holding a block and seeing a 3-tower (perception c), and their values are virtually identical. In the case $c \to \mathtt{x}$ the agent waits for another agent to pick from the 3-tower. The case $c \to \mathtt{l}$ is a retrograde choice as there is no merit in building a 4-tower, and it may seem

surprising that it occurs in a high-value policy. However, the probability of a 3-tower actually arising is made very low by the other rules in these policies, so that choosing $c \rightarrow \texttt{l}$ has no significant impact upon the policy value. The

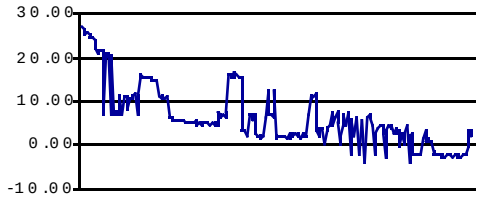

Fig. 9. Policy ranking chart for Example 5.4

chart is more volatile in this example. This may be due in part to the looser coupling of state and perception in the goal situation. In the 4-tower examples the perception "seeing a 4-tower" implies that the state contains a 4-tower. In the present example there is no single perception that implies that the goal state has been achieved, so the agents have poorer goal-recognition. However, the volatility in all these examples has other causes as well, as discussed in the next section.

Though limited in scope, these case studies suggest that, in order to select a good policy, the agent designer is unlikely to go far wrong in choosing from the best 10% according to the predictor. The predictor is clearly imperfect, and is inevitably so due to the approximations it makes, but unless the design imperatives are very demanding it is sufficient to seek policies yielding reasonable rather than optimal behaviour.

The option remains open to filter ostensibly high value policies using a more sophisticated predictor, in order to eliminate those that are actually worse than the basic one suggests. Such refinements can employ, for a small selection of policies, analyses that would be impractical in scale if applied to all policies.

Example 5.5: [*Two agents building four 1-towers*] In this example the goal is again not wholly perceivable by an agent, although it is more easily achievable than the goal of Example 5.4. Figure 10 charts the best 240 of them, for which $Q_{\mathcal{F}}$ is 87.60%. For the best 20 it is 90.00%. The two observed best policies

$$\{a \rightarrow \texttt{w},\ b \rightarrow \texttt{w},\ c \rightarrow (\texttt{w or x}),\ d \rightarrow \texttt{w},\ e \rightarrow \texttt{k},\ f \rightarrow \texttt{k},\ g \rightarrow \texttt{k},\ h \rightarrow \texttt{l},\ i \rightarrow \texttt{w}\}$$

are the two predicted best ones, sharing the property that they pick only from a tower of height >1 and place only upon the surface. In this case (unlike that in Example 5.4) the result is intuitive: the goal is clearly best achieved by the agents demolishing every non-unit tower they find whilst leaving unit towers undisturbed.

In these examples use of multiple clauses merely accelerates the achievement of what a single agent could accomplish on its own. However, we intend to explore domains in which communication, represented by appropriate perceptions, would expedite genuinely joint behaviour.

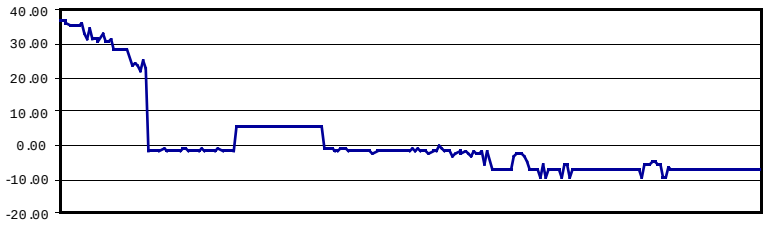

Fig. 10. Policy ranking chart for Example 5.5

6 Factors Affecting Predictive Quality

We dealt with a multi-clone context by optimizing a single clone using only the `x`-action to represent the effects of other clones. This is our alternative to analysing comprehensively how the clones behave conjointly. The latter analysis would need an unrestricted graph in which each node were a complete multi-situation having emergent arcs for all the real (non-`x`) actions its members could perform. From this *multi-graph* one could then seek the best policy for any one clone. The combinatorial nature of such an analysis makes it generally impractical. Figure 11 repeats the chart for Example 5.4 but highlights two of its many

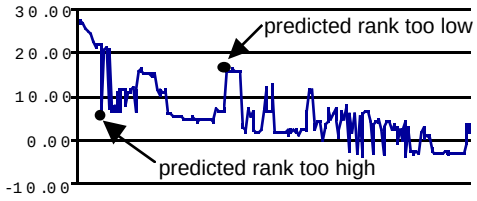

Fig. 11. Prediction anomalies in Example 5.4

anomalies. The policy indicated with too high a predicted rank is

$$\{a \to \mathtt{w},\ \ b \to \mathtt{l},\ \ c \to \mathtt{w},\ \ d \to \mathtt{w},\ \ e \to \mathtt{w},\ \ f \to \mathtt{k},\ \ g \to \mathtt{k},\ \ h \to \mathtt{l},\ \ i \to \mathtt{w}\}$$

If we examine how its predicted value is calculated, we find that a node $S = (4, f)$ in G_f is assigned a positive value $V(S)$ even though, in reality, the goal is

unachievable from S. As noted earlier, the calculation of policy values must take into account passive transitions of *self* (`x`-updating). Our method sees paths from S to the goal that contain steps in which *self* is assumed to be `x`-updatable by the other agent, although, in fact, the goal is unachievable from S. In situation S no block is held and *self* is seeing a 3-tower and must pick. From S there is the following ostensible path in G_f that reaches the goal:

(a) *self* is in situation $(4, f)$ and – because of what happens next to it in this path – the other agent must also be in $(4, f)$;
(b) the other agent takes action `k` and moves to $(7, b)$, whilst *self* is passively updated to $(7, e)$;
(c) *self* performs two wander actions taking it to $(7, i)$, whilst the other agent remains at $(7, b)$;
(d) the graph shows an `x`-action from $(7, i)$ to the goal $(3, d)$, which is correct since another agent could be in situation $(7, h)$. However, if this particular path is followed, the other agent would not be in the required $(7, h)$ but in $(7, b)$.

This path is infeasible because to achieve it would require *self* to perform actions and undergo `x`-updates that the other agent could never fulfil. In the multi-graph this joint scenario would not be feasible and any group situations containing $(4, f)$ would be assigned negative node values thereby reducing the policy's rank.

Figure 11 also indicates a policy with too low a predicted rank. Anomalies of this kind arise primarily from inaccuracy in the probabilities on the `x`-arcs in G_f, again owing to insufficient information about the situations actually occupied by *self*'s partners.

7 Scaling Using Abstractions

Although the approach discussed in earlier sections will work for small sets of perceptions, when this set becomes large it is clear that the number of policies is too great to evaluate them all. The *method of abstractions* groups the perceptions such that a single action can apply to all perceptions in a group, thereby reducing the number of policies that need to be considered. So far, we have applied this approach only to a case involving just one agent, although the method can easily be combined with the approach taken for groups of cloned agents.

In this section, we illustrate the method for the case of an agent aiming to build a 10-tower from 10 blocks. If each tower size were considered separately as a perception there would be 21 possible perceptions for 1 agent, making more than one million possible policies and approximately 100 situations. Instead, we choose to consider the set Gp of just 8 grouped perceptions,

$$\{(s0, h), (s0, nh), (s5, h), (s5, nh), (< s5, h), (< s5, nh), (> s5, h), (> s5, nh)\}$$

which yields 256 policies only. Each perception in Gp, called a *generic perception*, consists of one or more concrete perceptions. For example, $(< s5, h)$ means the

agent is looking at a non-empty tower with fewer than five blocks and holding a block. Similarly, $(> s5, h)$ means the agent is looking at a tower containing at least six blocks and the generic perception contains the four concrete perceptions $(s6, h)$, $(s7, h)$, $(s8, h)$, $(s9, h)$. We impose the restriction that, for a generic perception $P = \{p_1, \ldots, p_m\}$, the set of actions $\mathcal{A}(p_i)$ is the same for each i. In this way it is possible to associate a single action with P that is always possible whatever concrete perception actually pertains. In this example, this restriction forbids combining the concrete perceptions $(0, nh)$ with any other, since its action set is $\{\mathtt{w}\}$, whereas all other perceptions include some other action.

It is likely that the objective states can also be abstracted, since there may be several states that behave in essentially similar ways when paired with a generic perception. In general, therefore, there is a set of *generic situations*, given by $S_a = O_a \times P_a$, where O_a and P_a are, respectively, the sets of generic objective states and generic perceptions. Each situation (O, P) in S_a is given by a set of concrete situations: $(O, P) = \{(o_i, p_j) | o_i \in O \text{ and } p_j \in P\}$. The only other criterion we impose is that each generic situation in S_a should be non-empty and that the abstractions partition the sets of concrete perceptions and concrete states.

(a) states

O		$\mathcal{P}(O)$
1	[10]	$\{d, h\}$
2	[one5]	$\{a, b, e, f, g, h\}$
3	[5,5]	$\{f, h\}$
4	[no5no10]	$\{a, b, c, d, g, h\}$

(b) perceptions

P		$\mathcal{A}(P)$
a	$[< s5, h]$	$\{\mathtt{l}, \mathtt{w}\}$
b	$[< s5, nh]$	$\{\mathtt{k}, \mathtt{w}\}$
c	$[> s5, h]$	$\{\mathtt{l}, \mathtt{w}\}$
d	$[> s5, nh]$	$\{\mathtt{k}, \mathtt{w}\}$
e	$[s5, h]$	$\{\mathtt{l}, \mathtt{w}\}$
f	$[s5, nh]$	$\{\mathtt{k}, \mathtt{w}\}$
g	$[s0, h]$	$\{\mathtt{l}, \mathtt{w}\}$
h	$[s0, nh]$	$\{\mathtt{w}\}$

Fig. 12. States, perceptions and actions

Figure 12 shows the generic perceptions and states for the 10 blocks example. The state $[one5]$ consists of the eleven concrete states in which there is exactly one 5-tower. The state $[no5no10]$ consists of all concrete states that have neither a 5-tower nor a 10-tower. In general, the unrestricted graph includes an arc labelled by action a between generic situations $(O1, P1)$ and $(O2, P2)$ if there is an arc between some concrete situations $(o1, p1)$ and $(o2, p2)$, where $(o1, p1) \in (O1, P1)$ and $(o2, p2) \in (O2, P2)$. It is clear, therefore, that there will be more non-determinism present in graphs for generic situations, since there may be several ways to select $(o1, p1)$.

Figure 13 shows the restricted graph for the policy

$$\{a \to \mathtt{w},\ b \to \mathtt{k},\ c \to \mathtt{l},\ d \to \mathtt{w},\ e \to \mathtt{l},\ f \to \mathtt{k},\ g \to \mathtt{w},\ h \to \mathtt{w}\}$$

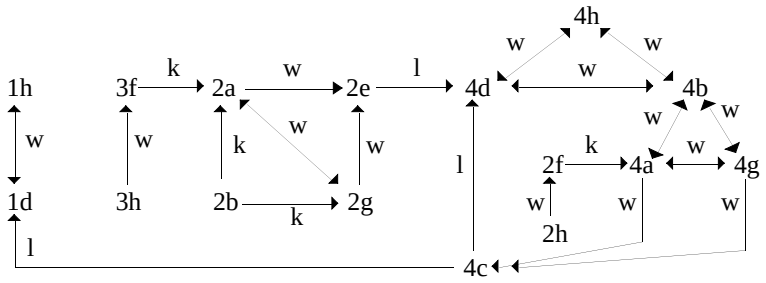

Fig. 13. Restricted graph for abstraction

The graph omits various reflexive arcs to keep it readable. For example, when in situation $2b$, it could be that the concrete state is $[5, 2, 2, 1]$ and the agent is seeing a 2-tower. After a `pick` action the agent would still be in abstract situation $2b$ (although in a different concrete situation). There would be a reflexive arc $(2b, 2b)$ to reflect this possibility (and others). If this policy were to be used by several cloned agents there are several transitions by which *self* might be passively updated. They are the following (abbreviated) transitions:

$$(2a, 2g), (2a, 4a), (2b, 2h), (2b, 4b), (2e, 4a), (2e, 4c), (2f, 4d), (2g, 4g),$$
$$(2h, 4h), (3f, 2b), (3f, 2f), (3h, 2h), (4a, 4g), (4b, 4h), (4d, 1d), (4h, 1h)$$

For example, if the agent *self* is in situation $2b$, it could be in several different concrete situations. If *self* were looking at some 1-tower, another agent might also be looking at the same tower and act before *self* to pick the block. Then *self* would find itself in situation $2h$. Alternatively, *self* might be looking at a 4-tower, in a concrete situation consisting of a 5-tower, 4-tower and another agent could be looking at the 5-tower and holding a block. In this case the second agent would place the block and *self* would passively update to situation $4b$.

The simulation for this example was made using positional mode on a 5 x 4 cellular grid using a single agent. The chart was extremely volatile, owing partly to inaccuracy in the probabilities assigned to the non-deterministic arcs. It had a Q_F of 64.87%, and 51.43% for the first 20. The analytical calculation of probabilities described in Section 4 is difficult to apply in the abstract context because it has to consider the multitude of underlying concrete situations that all the agents might occupy. To circumvent this difficulty we used the simulator to assist in probability estimation by counting, for each policy, the number of occasions each arc was traversed. The probabilities inferred from these counts were then supplied to the *Policy Predictor* The policy chart for about 1000 simulation runs is shown in Figure 14. It has a better Q_F of 70.83%, with a value of 69.47% for the first 20.

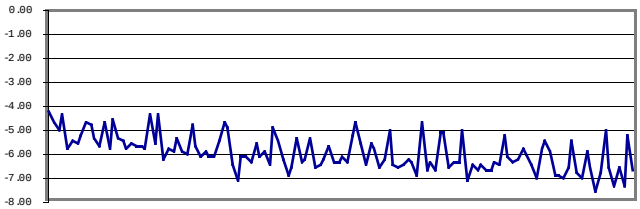

Fig. 14. Chart for abstraction

This method of estimation is nonetheless imperfect because information is lost when one conveys, to the abstract graph, the simulator's observations (counts) of concrete transitions. For instance, in the graph in Figure 13 there is a path from $(2, f)$ to $(4, c)$ via $(4, a)$. Both transitions in this path are feasible, the first since the agent can go from the concrete situation of looking at the only 5-tower and not holding a block to the concrete situation of looking at a 4-tower and holding a block, and the second since the agent can wander from the concrete situation of a 7-tower and 3-tower with the agent looking at the 3-tower to the agent looking at the 7-tower. However, these two transitions cannot actually occur in sequence: to make the transition from $(2, f)$ to $(4, a)$ with action `k`, means the agent must end up in some concrete situation looking at a 4-tower, not holding a block and where all towers have height <5. Therefore, it is impossible to wander to a situation where the agent is looking at a tower higher than 5.

We call this phenomenon of a restricted graph *piecewise incoherence*. In other words, a restricted graph offers, between generic situations, paths that may not be piecewise-coherent for all concrete instances of their arcs. It is this property that leads to the values given to policies being either underestimated or overestimated. Underestimation occurs when futile piecewise incoherent paths are present, whereas overestimation occurs when a piecewise incoherent path leads quickly to the goal.

A final reason for volatility is due to the choice of abstraction. For example, another abstraction might split state 4 into two states, one in which there is a tower containing more than 5 blocks and one in which there is no such tower. So far, we have always chosen an abstraction that distinguishes the goal. That is, the generic goal situation contains only concrete goal situations. If this is not the case, then under- and over-estimation of policy values will again occur.

8 Conclusions and Related Work

We have presented a method using situation graphs for predicting policies for TR-agents in multi-agent contexts. The use of situation graphs enables policies to be evaluated taking account of objective states, yielding greater discrimination than approaches (e.g. [10]) that focus primarily upon perceptions. This discrimination is necessary due to our assumption that exogenous events are not only possible but an inherent fact in a multi-agent context, so that we cannot

rely on using histories to estimate current situations. Furthermore, since we wish to find policies to achieve goals not detectable by a single perception, we cannot use the method of back-chaining and the so-called regression property used in [17], since a given perception may not be unique to the goal.

However, inclusion of objective state information poses a greater need for scalable methods, especially in multi-agent contexts. We have demonstrated here that in such contexts good policies are obtainable by analysing how any single agent responds to exogenous effects caused by other agents. This applies whether or not the agents are cloned. We have shown in [5] that the approach continues to yield good results in positionally sensitive formulations, even when abstraction is also employed.

Our future work will seek to understand better the relationship between our "self-based prediction graph" and the multi-graph and to investigate how the use of the former might be refined to give even better predictive power. The multi-graph approach seems to be similar in spirit to the approach described in [13], although they do not restrict their agents to be clones, and future work will further investigate this relationship. In this paper we have restricted agents to behave without communicating with one another. However, communication, and local and global memory, might be modelled as additional perceptions. Thus an agent may communicate with another using a kind of telepathy. Future work will seek to include the modelling of telepathy within the multiagent framework. Increasing the number of perceptions increases exponentially the number of policies and so in addition we will investigate a branch and bound strategy, based on an idea in [14] that an upper bound for the value of a partially determined policy is obtained by taking the "best" action from all situations not yet constrained by the policy. However, in [14] the policy value is determined as the shortest path to a single goal and for actions that are apparently deterministic. In our case none of these can be assumed: there may be multiple goal situations, actions need not be deterministic and our policy is dependent on the value of the tree of paths from each node. Furthermore, the impact of `x`-arcs in partial policies requires careful consideration.

A different approach to finding policies for agents with partial observability is to use the method of *belief states* as described in [6, 10]. In this case the beliefs of an agent are represented by real-valued n-dimensional vectors, $n \geq 1$. For instance, for a problem which can be modelled using four states a 4-dimensional vector would be used. The belief state $[0.25, 0, 0.5, 0.25]$ indicates that the agent believes it is more likely to be in state 3 than in states 1 or 4 and definitely is not in state 2. Each node is now a belief state and the arcs are labelled by actions. In order to obtain an optimal policy for traversing this infinite graph, an iterative algorithm is used to find an approximation by computing

$$V(b) = max_{a \in \mathcal{A}}(r + \gamma \times e)$$

where r is the expected reward for action a from state b and e is the estimated expected value of successor states, using the current estimates. In [22] it was shown that the optimal value function can be represented using some finite set

of vectors $\mathcal{V}$, such that the value of a node is computed as $V(b) = max_{v \in \mathcal{V}} \langle b \cdot v \rangle$, where $\langle b \cdot v \rangle$ is the inner product of vectors b and v. In [6, 10] it is shown how a policy can be found using this result.

The above approach and many others similar to it are useful in problems where the environment changes only under the actions of the agent. Otherwise, there is no reason why the agent should assume its previous belief state is any guide to the recent past and hence to the current state. When several agents are acting, an agent's environment undergoes exogenous changes due to other agents' actions so this assumption is not valid. In effect, the use of belief states is an encoding of the recent past, or the history of the agent's earlier actions and observations. This is demonstrated next for a simple example.

We have taken a very simple example from [6], with just one agent, and applied the method described in this paper. The problem concerns an agent that can move left or right through a linear sequence of four cells to look for a star which is always in the same place, in this case at cell 3. The set of states is $\{1, 2, 3, 4\}$ and each perception is a triple $[o, a1, a2]$, where o is either "e(mpty)" or "s(tar)" and $a1$ and $a2$ are the two most recent actions (either "move left" or "move right"). We assumed the agent stops when it sees the star. The predicted optimal policy is

$$[e, ml, ml] \rightarrow r, [e, ml, mr] \rightarrow r, [e, mr, ml] \rightarrow r, [e, mr, mr] \rightarrow l, [s, -, -] \rightarrow l$$

The previous actions are not relevant when seeing a star, so $[s, -, -]$ is, in fact, a generic situation. A similar policy has $[s, -, -] \rightarrow r$ and both correspond to the policy in [6]. This shows that, for small-scale single-agent contexts, at least, our method gives the same results as the POMDP method but uses much simpler machinery to obtain them. For multi-agent contexts our use of the *self*-oriented `x`-arc framework appears simpler and more scaleable than the corresponding "decentralized" POMDP approach of [19], which has to analyse conjointly how all the agents are acting upon the environment.

Throughout, our work so far has concerned only functional policies. In contrast, a relational policy [8] would allow choice of action for each perception according to some probability distribution, which may be given or learned. These represent two different engineering approaches to agent applications and it is of key interest to determine how well communities of (various) functional agents can subsume the capabilities of relational agents.

References

1. Benson, S. Inductive Learning of Reactive Action Models, *Proceedings of the 12th International Conference on Machine Learning*, (1995) 47-54
2. Benson, S. and Nilsson, N. Reacting, planning and learning in an autonomous agent, Machine Intelligence 14, Eds. Furukawa, K., Michie, D. and Muggleton, S, Clarendon Press, Oxford, (1995) 29-64
3. Broda, K., Hogger, C.J. and Watson, S. Constructing Teleo-Reactive Robot Programs, *Proceedings of the 14th European Conference on Artificial Intelligence*, Berlin, (2000) 653-657

4. Broda, K. and Hogger, C.J. Designing Teleo-Reactive Programs, Technical Report 2003/8, Dept. of Computing, Imperial College London, UK, (2003)
5. Broda, K. and Hogger, C.J. Designing and Simulating Individual Teleo-Reactive Agents, *Poster Proceedings, 27th German Conference on Artificial Intelligence*, Ulm, (2004) 1-15
6. Cassandra, A.R., Kaelbling, L.P. and Littman, M. Acting Optimally in Partially Observable Stochastic Domains, *Proceedings 12th National Conference on AI (AAAI-94)*, Seattle, (1994) 183-188
7. Chades, I., Scherrer, B. and Charpillet, F. Planning Cooperative Homogeneous Multiagent Systems using Markov Decision Processes, *Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS 2003)*, (2003) 426-429
8. Dickens, L. Learning through Exploration, MSc Dissertation, Department of Computing, Imperial College London, (2004)
9. Guestrin, C., Lagoudakis, M. and Parr, R. Coordinated Reinforcement Learning, *AAAI Spring Symposium*, Stanford, California, (2002)
10. Kaelbling, L.P., Littman, M.L. and Cassandra, A.R. Planning and Acting in Partially Observable Stochastic Domains, *Artificial Intelligence* 101, (1998) 99-134
11. Kochenderfer, M. Evolving Hierarchical and Recursive Teleo-reactive Programs through Genetic Programming, *EuroGP 2003*, LNCS 2610, (2003) 83-92
12. Lauer, M. and Riedmiller, M. Generalisation in Reinforcement Learning and the Use of Observations-Based Learning, *Proceedings of FGML workshop*, (2002) 100-107
13. Lauer, M. and Riedmiller, M. Reinforcement Learning for Stochastic Cooperative Multi-Agent-Systems, *AAMAS-04*, Columbia, New York, (2004) 1516-1517
14. Littman, L. Memoryless Policies: theoretical limitations and practical results, *Proceedings of the 3rd International Conference on Simulation of Adaptive Behaviour*, MIT Press (1994)
15. Mitchell, T. Reinforcement Learning, Machine Learning, McGraw Hill, (1997) 367-390
16. Nair R., Tambe, M., Yokoo, M., Pynadath, D. and Marsella, M. Taming Decentralised POMDPs: Towards Efficient Policy Computation for Multiagent Settings, *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, (2003) 705-711
17. Nilsson, N.J. Teleo-Reactive Programs for Agent Control, *Artificial Intelligence Research* 1, (1994) 139-158
18. Nilsson, N.J. Teleo-Reactive Programs and the Triple-Tower Architecture, *Electronic Transactions on Artificial Intelligence* 5, (2001) 99-110
19. Rathnasabapathy, B. and Gmytrasiewicz, P., Formalizing Multi-Agent POMDPs in the Context of Network Routing, *Proceedings of 36th Hawaii International Conference on System Sciences (HICSS'03)*, (2003)
20. Ryan, M.R.K. and Pendrith, M.D. RL-TOPS: An Architecture for Modularity and Re-Use in Reinforcement Learning, *Proceedings of the 15th International Conference on Machine Learning*, Madison, Wisconsin, (1998) 481-487
21. Snedecor, G.W. and Cochran, W.G. Statistical Methods, Iowa State Univ. Press, (1972)
22. Smallwood, R.D. and Sondik, E.J. The optimal control of partially observable Markov decision processes over a finite horizon, *Oper. Res.* 26 (2), (1978) 1071 - 1088