Designing and Simulating Individual Teleo-Reactive Agents

Krysia Broda and Christopher John Hogger

Department of Computing, Imperial College London
kb@doc.ic.ac.uk, cjh@doc.ic.ac.uk

Abstract. A method is presented for designing an individual teleo-reactive agent, based upon discounted-reward evaluation of policy-restricted subgraphs of complete situation-graphs. The main feature of the method is that it exploits explicit and definite associations of the agent's perceptions with states. The combinatorial burden that would potentially ensue from such associations can be ameliorated by suitable use of abstraction. For various *BlocksWorld* examples, the agent's predicted behaviour is compared with simulation results to provide insight into the method's power and scalability.

1 Introduction

Teleo-reactive agents [10] react to their perceptions of the world by obeying an internal program (or policy) mapping perceptions to actions. The simplest policy structure is a set of mutually-exclusive production rules of the form *perception* $\rightarrow$ *action*, usually intended to control durative behaviour: given some current perception the agent performs the corresponding action until acquiring a new perception, whereupon it reacts likewise to that.

Such an agent may or may not have sufficient perceptive capability to know, at any instant, the entire state of the world. An agent of the kind described in [11] is presumed at least capable of perceiving an intended goal state whenever that state arises, and is accordingly designed with that capability in mind. Its program includes an explicit test for the goal state, whilst the nature and ordering of its rules are inferred by reductive analysis of that test. Its goal-orientedness is thus explicit in the program.

A teleo-reactive agent of the kind studied in this paper is, by contrast, presumed incapable of perceiving the entirety of any state, whether a goal state or otherwise. In particular, its program contains no rule specifically associated with a goal. The design process now relies not upon goal-reductive analysis but upon comparing the extents to which alternative programs dispose the agent towards achieving a goal – as judged, for instance, by a discounted-reward principle. A program identified on this basis is *implicitly* goal-oriented.

Our design process is based upon graphs that relate combinations of perception and objective state, and so differs from approaches that employ graphs relating perceptions alone. These typically estimate the distribution of states associated with a current perception using some species of history, whether of past actions, past states or past perceptions. The key presumption there is that the history is *reliable* in that no changes to the state ever occur except by the known actions of the agent. By contrast, our focus is specifically upon the opposite case in which the world may undergo unpredictable exogenous changes while the agent is pursuing its goal. These changes include, for example, results of failed actions and mistaken perceptions.

We first outline our basic formulation for teleo-reactive agents using objective states and perceptions, which are pairwise combined into situations. We then introduce a method for computing a value for a given policy using the discounted reward principle and show that the results obtained correspond well with simulation results (see Case Studies 1-4). In order that our method should scale to reasonably sized-problems we use a method of *abstraction.* We make an assumption that when there is a large number of situations a good policy can be found by aggregating situations into classes and treating each such class as a single abstract (or generic) situation. The results from the example in Case Study 5 vindicate this assumption.

Our core notions are formulated using the following notation. The set of all objective states that the world may assume is denoted by $\mathcal{O}$; the set of all perceptions that the agent may have of the world is denoted by $\mathcal{P}$; the set of all actions that the agent may take is denoted by $\mathcal{A}$. For each state $o \in \mathcal{O}$, the perceptions that the agent may have of o form some subset $P(o)$ of $\mathcal{P}$, and for any perception $p \in P(o)$ the actions that the agent may take in reacting to p for some subset $A(p)$ of $\mathcal{A}$. A *policy* for the agent is any total function $f : \mathcal{P} \to \mathcal{A}$ satisfying $\forall p \in \mathcal{P},\ f(p) \in A(p)$.

For any $o \in \mathcal{O}$ and $p \in P(o)$, the pair (o, p) is called a *situation.* A *goal* is a situation that the agent is required, by the designer, to achieve. The fundamental problem in identifying a suitable policy is the incompleteness of perceptions as state descriptors. If they fully described the states then any situation (o, p) could be contracted simply to o, and the design process would need only to compare conventional state-transition graphs. However, the realistic position is that each perception contains only limited knowledge of the world. The problem is therefore how to optimize, for a goal incorporating a state, an agent that (generally) cannot recognize that state.

The agents that we consider do not necessarily have any memory capability. However, our approach easily allows for such capability, by including memory as additional perceptions. Actions that affect such perceptions can be made available to the agents. In that way suitable policies for those problems commonly addressed in the literature, such as the "tiger problem" [7] can be modelled in our framework.

Example 1: *BlocksWorld* Formulation This example, like all others in this paper, employs *BlocksWorld* to illustrate the ideas involved. It is strongly representative of state-transition systems in a wide class of domains. This world comprises a surface and some number (here 4) of identical blocks. At most one such block may be held by the agent, whilst the other blocks are arranged as some configuration of towers standing upon the surface. A goal for this example might be the situation in which the configuration comprises a 4-tower and in which the agent is perceiving this tower. A configuration is represented by a list of integers each denoting the height (>0) of a tower. Figure 1(a) shows the possible configurations, each determining a state o. The states are named $1, \ldots, 8$ for brevity. In states 1-5 no block is being held, whilst in states 6-8 one block is being held.

(a) states

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]	$\{a, h\}$
7	[1, 2]	$\{a, b, h\}$
8	[3]	$\{c, h\}$

(b) perceptions

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

Fig. 1. States, perceptions and actions

A perception is represented by a list $[S, H]$, where S expresses what the agent is seeing and H expresses its holding status. The domain of S is $\{s0,\ s1,\ s2,\ s3,\ s4\}$ in which $s0$ denotes "seeing the surface" and sn $(n > 0)$ denotes "seeing an n-tower". The domain of H is $\{h,\ nh\}$ in which h denotes "holding a block" and nh denotes "not holding a block". Figure 1(b) shows the possible perceptions, named $a, \ldots, i$ for brevity. Figure 1(a) also shows alongside each state o the set $P(o)$ of possible perceptions. From this we can infer, for instance, that state 5 occurs in just two situations $(5, g)$ and $(5, i)$. Declaring perceptions and actions presumes certain physical characteristics of the agent. Here, we presume three possible actions – pick (`k`), place (`l`) and wander (`w`). In a `k`-action the agent removes the top block from a tower it is seeing, and afterwards holds that block and sees the resulting tower (or the surface, as appropriate). In an `l`-action the agent places a block it is holding upon whatever it sees (the surface or some tower) and afterwards is not holding a block and sees the resulting tower. A `w`-action merely updates the agent's perception without altering the state. Figure 1(b) shows alongside each perception p the set $A(p)$ of possible actions. The data in Fig. 1 and the assumed effects of actions jointly determine the transitions the agent could make between situations if no policy were employed to restrict its actions, i.e. if it were free to behave with maximum non-determinism. Figure 2 depicts all these transitions as an *unrestricted graph*, denoted generally by G. For brevity, each situation (o, p) is displayed there as op.

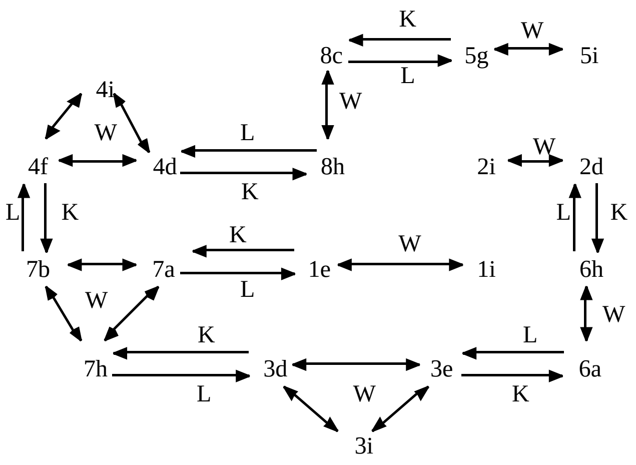

Fig. 2. Unrestricted graph G

In this example we suppress reflexive `w`-arcs, which can have no influence upon reachability. This effectively restricts the `w`-action to cause an immediate change in the agent's perception. Later, when we consider agents wandering incrementally in a positional context, we will permit a `w`-action to maintain the current perception.

A policy f renders the agent more deterministic. It prunes arcs from G to leave an *f-restricted graph* G_f such that all emergent arcs from any situation bear the same action label. Some non-determinism remains, in that a `w`-action from any situation offers (usually) arcs to several others. Assuming that the agent is to have some policy and that it can be assigned initially to any situation, our interest is in defining and identifying for it an optimal policy, given some intended goal. In the example above, the number of policies to choose from is 256, being the product of the cardinalities of all the $A(p)$ sets.

The remainder of this paper describes one particular method for predicting the merits of policies, and then tests this method against a set of case studies. The testing relies upon simulating agents in both positionless and positional modes and measuring their observed efficacies. The last of the case studies shows that abstraction at the formulation stage can enhance the method's scalability.

2 Predicting Policy Values

The value of a policy f should reflect both the agent's ability to achieve the goal and the overall benefit of it doing so, as it proceeds (implicitly) through the graph G_f. One way to predict a value for f is to use the discounted-reward principle [7]. This assigns a value to each situation on the basis of the expected rewards gained by traversing its emergent arc(s). Various versions of this principle give differing degrees of flexibility. We employ a *Teleo-Agent Policy Evaluator* which applies this simple version: a situation S having immediate successor-set SS in G_f has a value $V(S)$ defined by $V(S) = \Sigma_{s \in SS}(p_s \times (rwd(s) + \gamma \times V(s)))$, where p_s is the probability that from S the agent proceeds next to s, $rwd(s)$ is the reward it earns by doing so and γ is a discount factor such that $0 \leq \gamma \leq 1$; in the case that $SS = \emptyset$ we have $V(S) = 0$. More general versions parameterize their rewards and discount factors by the particular actions entailed in the S-s transitions. In our version we choose 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 difference between R and r governs the extent to which the method accords merit to the agent reaching a goal rather than not doing so, whilst γ controls the separation (but, generally, not the ranking) of policy values. The advantage of the method is that, provided $\gamma < 1$, the values of all situations are finite even when G_f contains cycles signifying non-terminating behaviour. Moreover, each value is a linear combination of other values and all may be computed efficiently by linear equation solving. On the other hand, it may not be easy to choose rewards and discount factors that relate intuitively to the physics of the agent, such as the resources expended on perceiving and acting. However, this is not necessarily a major impediment to the task of distinguishing good policies from bad ones.

The predicted value $V_{\mathrm{pre}}(f)$ for a policy f is then the mean of all the situations' values, assuming that the situations are equally probable as initial ones for the agent to be in. The predicted optimal policy is the one having the highest predicted $V_{\mathrm{pre}}(f)$ value. We need not always resort to such methods: for particular combinations of world, agent and goal, good policies may be ascertainable by intuition alone. Presently we will see examples where this is so, and others where it is not. Where the method is used, we (currently) evaluate all policies. This places some limitation on the method's effectiveness, as there may be many policies to examine. Various stances can be adopted on this. Firstly, if having an optimal agent is sufficiently important and if the design process is "once-and-for-all", it may be acceptable to expend heavy effort on policy evaluation. Secondly, intuitions can be applied at the start to eliminate clearly poor policies without needing to evaluate them. Thirdly, the number of policies can be reduced by the use of abstraction, that is, by the use of approximate rather than comprehensive problem formulations; this approach yields less predictive power but can still be an effective device, as we will later show by a concrete example.

If the set of situations is large, then the approach using abstractions separately partitions the set $\mathcal{P}$ of perceptions and the set $\mathcal{O}$ of objective states into classes. Each class so obtained is treated accordingly as a single abstract perception or abstract state. The partitioning is guided by intuition about the agent's lack of need to distinguish between the individual members in order to achieve the goal. Pairwise combinations of abstract states and perceptions yield abstract (or *generic*) situations and these become the nodes of the unrestricted graphs. Let (o_G, p_G) be a generic situation, then we impose the restriction, called the *principle of genericity* that some, and preferably many, of the situations belonging to the cartesian product $o_G \times p_G$ are concrete situations. This means that o_G can be associated with a set of perceptions $P(o_G)$ in the same way as for objective states. Moreover, we can associate with each generic perception p_G a set of actions $A(p_G)$. The principle of genericity then also requires that each action in $A(p_G)$ should belong to each perception in p_G. These restrictions on generic situations are made so that good policies predicted using the abstract graph translate into reliable policies in practice. Case Study 5 uses abstractions and discusses this issue further.

This kind of approach can be illustrated straightaway, by using an abstraction with just two abstract states $[e4]$ and $[ne4]$, where $e4$ denotes "a 4-tower exists" and $ne4$ denotes "no 4-tower exists", and three seeing perceptors, $s0$, $s4$ and sx, the latter denoting "seeing neither the surface nor a 4-tower". This abstraction suits the goal of building a 4-tower from an arbitrary but sufficient number of blocks. Figure 3 shows the restricted graph for the policy whereby the agent, if seeing a tower of height neither 0 nor 4 (sx), can `pick` if not-holding (nh) or `place` if holding (h), but in all other cases wanders. The intended goal is the situation $([e4], [nh, s4])$. In case there are 4 blocks, the generic state $[ne4]$ consists of all states except state 5 in Fig. 1. The generic perception $[nh, sx]$ consists of perceptions d, e, f from Fig. 1. The cartesian product includes 21 pairs, of which 6 correspond to concrete situations.

$$\begin{tikzcd}
 & & & & (ne4, nh, s0) \arrow[d, "W"] \\
(e4, h, s4) \arrow[ll, leftrightarrow, "W"] \arrow[dr, "W"] & & (e4, h, s0) \arrow[ll, leftrightarrow] \arrow[dl, "W"'] & & (ne4, nh, sx) \arrow[llll, bend right, "K"'] \arrow[d, shift right, "K"'] \arrow[dd, bend left=60, "K"] \\
 & (e4, h, sx) \arrow[rr, "L"] \arrow[dl, shift right, "L"'] & & \overset{\textit{goal}}{(e4, nh, s4)} & (ne4, h, sx) \arrow[l, "L"'] \arrow[u, shift right, "L"'] \\
(e4, nh, sx) \arrow[uu, "K"] \arrow[ur, shift right, "K"'] & & & (e4, nh, s0) \arrow[lll, "W"'] \arrow[u, "W"'] & (ne4, h, s0) \arrow[u, "W"']
\end{tikzcd}$$

Fig. 3. Using generic situations

If an unrestricted graph is formed using generic states, as opposed to objective states, it is likely that the degree of non-determinism present will be increased. This is because there is more opportunity for variation in the destination states. For instance, in the restricted graph shown in Fig. 3, after the action `pick` from the state $([e4], [nh, sx])$, the agent may be looking either at the surface ($x = 1$), or at a 4-tower ($x = 5$) or at some other tower ($x \neq 1$ and $x \neq 5$). Initially, probabilities on multiple arcs issuing from a situation are assigned uniformly, unless other information is available. However, our simulator is able to estimate the probabilities of traversing each arc and these empirical values can be used to give more reliable policy values.

3 Simulating Agents

We test the effectiveness of policies by using a *Teleo-Agent Simulator* to simulate agents possessing them. A single *run* assigns the agent to an initial situation S and then drives its subsequent activity in accordance with the chosen policy f. The simulated agent is made to behave deterministically, implicitly traversing some single path in G_f from S. The run terminates when the agent either reaches the goal or exceeds a prescribed bound B on the number of transitions performed. As the path is traversed, the value of $V(S)$ is computed incrementally on the same basis as used in the predictive *Policy Evaluator*. Equal numbers of runs are executed for each initial situation S. The mean observed $V(S)$ over all runs for all S then gives the observed policy value $V_{\text{obs}}(f)$.

For a set $\mathcal{F}$ of n policies we can measure the correlation between their observed and predicted values as follows. A pair $(f, g) \in \mathcal{F} \times \mathcal{F}$ (distinct f, g) for which $V_{\text{pre}}(f) \leq V_{\text{pre}}(g)$ is *concordant* if $V_{\text{obs}}(f) \leq V_{\text{obs}}(g)$, but is otherwise *discordant*. If C is the number of concordant pairs and D the number of discordant pairs, then the *Kendall rank-correlation coefficient* [8] $\tau_{\mathcal{F}}$ for $\mathcal{F}$ is $2 \times (C - D)/(n \times (n-1))$. We can re-express this measure as a percentage $Q_{\mathcal{F}} = 50 \times (1 + \tau_{\mathcal{F}})$. In the best case $Q_{\mathcal{F}} = 100\%$, when the predicted and observed ranks of all policies agree perfectly. In the worst case $Q_{\mathcal{F}} = 0\%$, when they disagree maximally. The $Q_{\mathcal{F}}$ values cited in the case studies we report here all imply, with $> 99.8\%$ confidence, that the observed and predicted policy values are correlated.

The simulator also reports the observed success rate $SR_{\text{obs}}(f)$ for the policy, measured as the percentage of runs that reach the goal. The success rate can be predicted independently of the simulator by considering the reachability of the goal in G_f. Let *Sits* be the set of all situations in the problem formulation (and hence in G and in all its policy-restricted subgraphs). Then the choice of f partitions *Sits* into two disjoint subsets N_f and T_f called the *non-trough* and the *trough*, respectively. N_f contains the goal and all situations from which the goal is reachable under policy f. T_f contains all the other situations. By definition, there cannot exist any arc in G_f directed from T_f to N_f. However, there may exist one or more in the opposite direction, in which case we describe G_f as *NT*-bridged. The *Policy Evaluator* has the secondary function of determining N_f, T_f and the *NT*-bridged status for any policy f.

If G_f is not *NT*-bridged then the agent can reach the goal if and only if its initial situation is in N_f, so that its predicted success rate $SR_{\text{pre}}(f)$ (as a percentage) is exactly $C = 100 \times |N_f|/|Sits|$. If G_f is *NT*-bridged then we have only the weaker relationship $SR_{\text{pre}}(f) < C$. In either case C provides an upper bound on $SR_{\text{pre}}(f)$, which we may then compare with $SR_{\text{obs}}(f)$. Simulated runs curtailed by the bound B thereby fail to reach a goal that may have been reachable in principle. So in general $SR_{\text{pre}}(f)$ will overestimate $SR_{\text{obs}}(f)$ by an extent that depends on B.

The simulator supports both *positionless* and *positional* simulation modes. The former associates no positional data to the agent or the towers, and so precisely mirrors the information content of the original problem formulation. This means, for instance, that when 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 sees.

4 Case Studies in Positionless Mode

Positionless simulation and our prediction method explore an agent's behaviour in similar ways and so we expect them to give similar outcomes. It is still useful to test this by case studies. These studies also serve to introduce concepts and notations needed later in the paper when we present cases studies of agents in positional contexts.

Here we compare predicted results with those observed through positionless simulation, taking three different goals for an agent acting in a world of 4 blocks. When computing the predicted policy values we suppress any arcs emergent from the goal, in order to mirror the simulator's behaviour in terminating the agent when it reaches the goal; though instituted for the sake of rigour, this nuance has no discernible impact upon the outcomes in these particular examples. The prediction parameter values used throughout are R=100, r=-1 and γ=0.9; we determined by prior experiments that other choices would not have altered the results or conclusions reported here.

Case Study 1: [*Goal=(5,g): construct and perceive a 4-tower*] This is the case introduced in Example 1. The predicted optimal policies are

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

for each of which $V_{\text{pre}}(f) = 37.30$ and $SR_{\text{pre}}(f) < 73.68\%$. Their graphs are NT-bridged. Owing to the suppression of arcs from $(5, g)$ the action for perception g is actually immaterial. From 1007 simulated runs per policy (thus, 53 for each initial situation) with bound B=100, the same two policies are identically observed as optimal, having $V_{\text{obs}}(f)$=37.32 and $SR_{\text{obs}}(f) = 70.75\%$.

A broader perspective for this set $\mathcal{F}$ of 256 policies is obtained by charting their observed values against the ranks of their predicted ones. For good predictive quality the profile should decrease monotonically. The chart obtained, in Fig. 4, does so and its Kendall measure $Q_{\mathcal{F}}$ is 99.56%. The optimal policies for the given goal are intuitive, directing the agent to place a block only upon an existing tower (so, not upon the surface) and to pick a block only from a tower of height < 3.

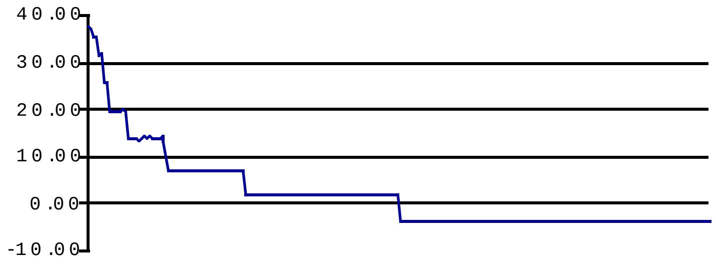

Fig. 4. Observed policy values in Case Study 1

Case Study 2: [*Goal*=$(2, i)$*: construct four 1-towers, perceive surface*] Here the predicted optimal policy is

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

for which $V_{\text{pre}}(f)$=41.71 and $SR_{\text{pre}}(f) = 100\%$. Its graph is not NT-bridged. From 1007 simulated runs with bound B=100, the same policy is optimal, having $V_{\text{obs}}(f) = 41.76$ and $SR_{\text{obs}}(f) = 100\%$. The ranking chart, shown in Fig. 5, is again almost perfectly monotonic, and $Q_{\mathcal{F}} = 99.88\%$. Again, the optimal policy is intuitive – place a block only upon the surface and pick a block only from a tower of height > 1.

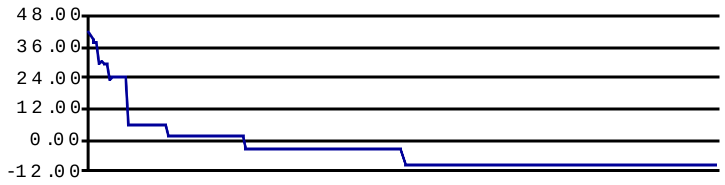

Fig. 5. Observed policy values in Case Study 2

Case Study 3: [*Goal* = $(3, i)$*: construct one 2-tower, two 1-towers, perceive surface*] Here the predicted optimal policy is

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

for which $V_{\text{pre}}(f)$=31.57 and $SR_{\text{pre}}(f) = 68.42\%$. Its graph is not NT-bridged. From 1007 simulated runs with bound B=100, the same policy is optimal, having $V_{\text{obs}}(f) = 31.37$ and $SR_{\text{obs}}(f) = 68.42\%$. Figure 6 shows the ranking chart, which is a little less monotonic. Here $Q_{\mathcal{F}} = 91.75\%$, not quite as high as before. More than half (144) of the policies have NT-bridged graphs, whereas the former cases have just 22 and 8 respectively. The many minor anomalies in Fig. 6, whose joint effect is to reduce $Q_{\mathcal{F}}$, arise only because the constraints upon sampling and bounding cause the simulations not to cover all those ways in which the agent may variously avoid or traverse a fateful bridge from the non-trough to the trough. The optimal policy in the present case – place a block only upon the surface, and pick a block only from a tower of height > 2 – is now not so obviously optimal. One might have guessed instead at, say, the policy identified in Case Study 2, on the grounds that in the state comprising two 2-towers the agent needs to pick from one of these in order to progress; but this is mistaken, for that state does not occur sufficiently often to justify this change, and this alternative policy turns out to have less than half the value and less than half the success rate of the optimal one.

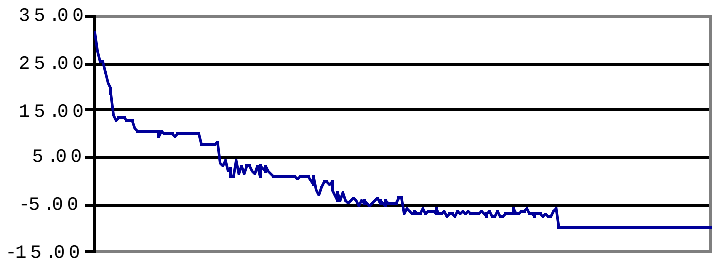

Fig. 6. Observed policy values in Case Study 3

In conclusion, the evaluator and the simulator (in positionless mode) correctly implement the underlying concepts, by mutually validating their outcomes.

5 Case Studies in Positional Mode

Our interest here is in determining how well our *Policy Evaluator* – which knows nothing of positions or of the distinct identities of identical towers – predicts the behaviour of an agent in a position-sensitive world. In its positional mode the simulator manifests an initial situation as an assignment of the towers and the agent to distinct cells in a rectangular grid having prescribable dimensions. Thereafter the simulator drives the agent's behaviour as dictated by the policy f, and updates the agent's grid position as appropriate. Moreover, towers maintain distinct identities by having distinct positions.

A key decision for this mode concerns how the agent shall perform a `w`-action and what it shall see afterwards. We decided this as follows. A `w`-action moves the agent randomly to any vacant cell in its (immediate) neighbourhood (comprising 8 cells, unless the agent is next to a grid boundary). The agent then sees the surface if its new neighbourhood is entirely vacant, but otherwise randomly sees any one tower in that new neighbourhood. This (or any alternative) decision fixes the probability distribution over the possible `w`-transitions between the prior situation and its successors. By contrast, the *Policy Evaluator* treats, by default, those transitions as equi-probable, and thus over-simplifies the agent's behaviour. So we must expect some shortfall in its predictive power when applied to a position-sensitive world, and the question is how serious (or not) that shortfall is.

The grid dimensions are another factor in the shortfall. The smaller they are, the less freedom the agent has to wander. It may become surrounded by towers, particularly near boundaries, causing a run to fail on a path that the *Policy Evaluator* regards as successful. In our studies we used a 6×6 grid, large enough to prevent such possibilities from significantly affecting the comparisons made.

It is clear that in positional mode a `w`-action may leave the agent's situation invariant. In the earlier studies this possibility was denied by our excluding reflexive `w`-arcs from G. Now, however, we add such an arc to every situation in G in order that our predictions shall take account of reflexive transitions in positional simulation. The prediction parameter values used throughout were again $R = 100$, $r = -1$ and $\gamma = 0.9$.

Case Study 4: [*Goal*=$(5, g)$*: construct and perceive a 4-tower*] This repeats Case Study 1 but in positional mode and with bound $B = 200$ (increased from the former 100 to allow for more wandering). The predicted optimal policies are the same as before, but with a slightly reduced predicted value 30.47 (reduced precisely because of the negative rewards of reflexive wanders). However, they are not quite optimal among the observed values. The observed optimal policy is a little different:

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

On seeing a 2-tower when not holding (perception e), it marginally favours `w` over `k`. Figure 7 shows the ranking chart, for which $Q_{\mathcal{F}} = 66.91\%$. Minor sampling variations among many policies of similar value are the root cause of this reduced value. If we contract $\mathcal{F}$ to just the 20 best policies then $Q_{\mathcal{F}} = 76.32\%$, so the predictive quality is rather better in the region that matters.

The observed values in the chart are much lower than the predicted ones, since the simulated agent is wandering (in the sparse regions of the grid) much more than the prediction considers probable. However, it is the relative rather than the absolute values that are our main interest. The observed and predicted success rates remain in close agreement, since excess wandering does not alter the reachability of the goal, but merely delays its discovery.

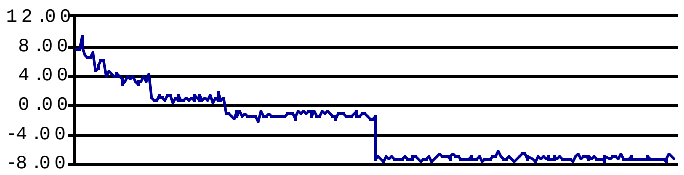

Fig. 7. Observed policy values in Case Study 4

Case Study 5: This is the study we feel is of greatest interest, since it bears directly upon the scalability of the predictive method. The goal is to construct (at least) one 2-tower and one 4-tower and to perceive the surface. If there were (say) 10 blocks, then a comprehensive formulation of this problem dealing with its 72 possible states would not be manageable. Nor might it be realistic to suppose the agent possessed 11 distinct perceptors $s0, \ldots, s10$; even if it did, the graph G would contain a huge number of situations and 2^{20} policies to consider.

Our approach to this problem was to assume that solving it did not require the planning process to distinguish between numerous slightly different situations. Instead, the formulation uses four generic states $[e2,\ e4]$, $[e2,\ ne4]$, $[ne2,\ e4]$ and $[ne2,\ ne4]$, where $e2$ denotes "a 2-tower exists" and $ne2$ denotes "no 2-tower exists" (and analogously for $e4$ and $ne4$). The seeing perceptors are $s0$, $s2$, $s4$ and sx, the latter denoting "seeing an x-tower" (x denotes "neither 2 nor 4"). This formulation gives just 24 situations and 128 policies.

As a further challenge to the predictive method the simulations were done in positional mode, using 10 blocks and making 1008 runs per policy with bound B=200. Figure 8 shows the ranking chart which, though having many fluctuations, is reasonably monotonic overall and has a surprisingly high overall quality measure $Q_{\mathcal{F}} = 86.55\%$.

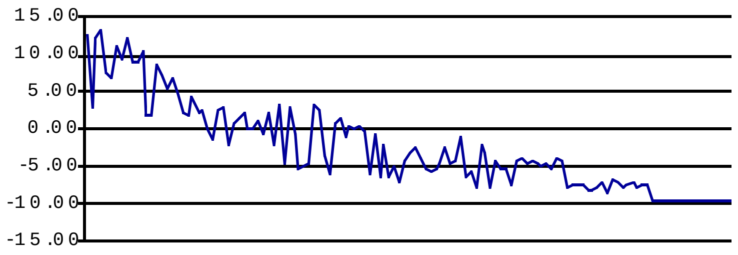

Fig. 8. Observed policy values in Case Study 5

The observed optimal policy is that which places a block only upon the surface or upon an x-tower and never picks a block. It has rank 4 among the predicted values. The observed second-best policy places a block only upon an

x-tower and picks a block only from an x-tower. This has rank 1 among the predicted values.

The fluctuations in this chart arise partly from the prediction method misjudging the probabilities on the arcs, for the reasons noted at the start of this section. In principle we can improve the method by determining more accurate probabilities (by considering how the agent actually behaves) and attaching them as data to the arcs in G_f to be used in the discounted-reward equations. For the current example it is not easy to ascertain accurate probabilities analytically, though it would no doubt be possible, by considering the density distribution of blocks, to obtain reasonable ones with sufficient effort. To show in principle that the method can be improved in this way, we made the simulator count, for each policy f, the number of times each arc was traversed, and used these totals to assign probabilities to the arcs in G_f. The *Policy Evaluator* was then re-run using these more realistic probabilities to provide updated predictions. Comparing these with the simulations' observed values yields the less volatile chart in Fig. 9.

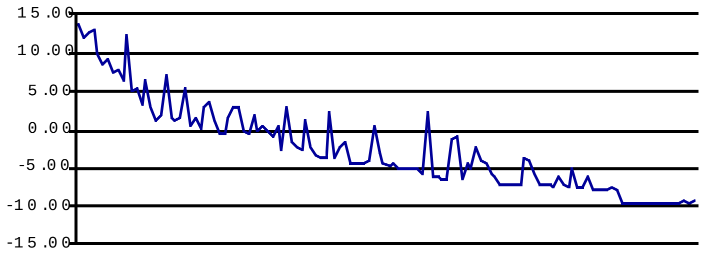

Fig. 9. Result of using better probabilities in Case Study 5

The quality $Q_{\mathcal{F}}$ over the full range has risen to 91.73%, whilst for the best 20 policies it has improved radically from 69.48% to 86.32%. In particular, the observed optimal policy is now ranked 1 among the predicted values. Some volatility remains because (i) each graph G_f offers transitions from generic situations that cannot be realized by all their concrete instances and (ii) the number of runs (42) for each initial generic situation insufficiently tests each of its many concrete instances. Cause (i) exacts an inevitable penalty for any approximate problem formulation, whilst cause (ii) merely calls for more simulation runs per policy to achieve better sampling.

A more challenging study – of building a 10-tower – used just four generic state: the state [10], the state [5, 5], states with one 5-tower and all other states. The agent could distinguish only three cases when seeing an n-tower: $n = 5$, $n < 5$ and $n > 5$. With this degree of abstraction the predictive quality $Q_{\mathcal{F}}$ over the 128 policies was 70.83%. The observed best policy, to pick only from a tower of height < 5 and place only upon a tower of height ≥ 5, was the one predicted as best. This further encouraged the use of abstraction for addressing scalability.

6 Discussion and Conclusions

Our work on teleo-reactive agents was inspired by their pioneer and originator [10, 11]. The main approach to constructing them automatically has been to employ various strategies based upon learning. An example is the work of [1] which uses inductive logic programming to learn from experience useful relations between action and effect. Related work on agent planning by learning is given in [9] and [12]. The formulation of teleo-reactive programs as restrictions of situation graphs was introduced in [2]. The discounted reward principle [7] has recently been used to plan cooperative and communicative teleo-reactive agents [3, 4]. The contributions of this paper include the following:

(i) statistical comparison of predicted discounted-reward values of teleo-reactive policies with the results of simulations;
(ii) evidence that sole reliance upon idealized situation graphs affords good predictive power for simulated agents in position-sensitive cellular worlds;
(iii) elucidation of factors involved in the differences between predictions and observations; and
(iv) evidence that the approach remains fairly robust when, for the sake of scalability, the situation graphs employ generic descriptors of states and perceptions.

Our use of graphs (here termed *op-graphs*) relating situations contrasts with the use of simpler graphs (*p-graphs*) relating perceptions alone. Such p-graphs are traditionally used to represent decision processes and to reason about them [6]. In a p-graph each arc denotes an action transition from one perception to another. Clearly an op-graph distinguishes between situations that share a common perception, whereas a p-graph does not. If in a p-graph a perception p occurs in both goal and non-goal situations, then the discounted reward method overestimates the value of p by attributing the higher reward associated with goal-reaching arcs to arcs not reaching the goal.

There is a further feature that differentiates the two kinds of formulation and is not immediately obvious. This is *accidental non-determinism*, which can occur in the case when a perception occurs in more than one objective state. Suppose action `a` takes $o1p1$ to $o2p2$ and $o3p1$ to $o4p4$ and further suppose these are the only transitions when `a` is the action in states $o1p1$ and $o3p1$. In the p-graph there will be two outgoing arcs labelled by `a` from $p1$ – one to $p2$ and one to $p4$. Thus the p-graph makes `a` into a non-deterministic action when in perception $p1$, whereas the op-graph does not. The consequence of accidental non-determinism when using a p-graph is that the overall policy value may be over or under-estimated. A node in the trough of a given restricted op-graph may, for the corresponding p-graph, lie on a path to the goal perception resulting in overestimation. Or, a short deterministic path to the goal in an op-graph might project onto a non-deterministic set of paths, including long and futile ones, in the p-graph, resulting in an under-estimation of policy value. Furthermore, in the same way that our use of op-graphs extends p-graphs, so also our abstraction of situations extends approaches, as in [5], that abstract perceptions alone.

It is therefore apparent that there are circumstances in which these factors render the p-graph approach less well suited to predicting good policies using discounted reward methods. Accidental non-determinism introduced by using a p-graph can be removed in some cases by extending perceptions (e.g. transition histories [6]) to enable detection of which objective state the agent is in for some particular perception. However, this "solution" increases the number of possible policies requiring to be considered and evaluated. Our use of op-graphs, kept to manageable proportions as necessary through the use of abstraction, also enables us to find policies that maintain an agent's pursuit of the goal even in a context of state changes induced by unpredictable exogenous factors. This is because our planning method judges how best the agent should behave whatever situation it is currently in, regardless of whether that situation results from the agent's previous action or from arbitrary exogenous disturbance.

The key aspects of teleo-reactive agents are their autonomy and their lack of dependence upon on-board computing power and sophisticated software. It is plausible that they will play a key role in some nano-technological applications, and the quest for practical ways to design them is, we believe, a worthwhile pursuit.

References

1. Benson, S. Learning Action Models for Reactive Autonomous Agents, PhD, Dept. of Computer Science, Stanford University, 1996.
2. Broda, K., Hogger, C.J. and Watson, S. Constructing Teleo-Reactive Robot Programs, *Proceedings of the 14th European Conference on Artificial Intelligence*, Berlin, pp 653-657, 2000.
3. Broda, K. and Hogger, C.J. Designing and Simulating Individual Teleo-Reactive Robots, Technical Report TR 2003/8, Dept. of Computing, Imperial College London, UK, 2003.
4. Broda, K. and Hogger, C.J. Policies for Cloned Teleo-reactive Robots, To be presented to the *2nd German Conference on Multiagent System Technologies*, Erfurt, 2004.
5. Dearden, A. and Boutilier, C. Abstraction and approximate decision-theoretic planning, *Artificial Intelligence* 89: pp 219-283, 1997.
6. Dutech, A. Solving POMDPs using selected past events, *Proceedings of the 14th European Conference on Artificial Intelligence*, Berlin, pp 281-286, 2000.
7. Kaelbling, L.P., Littman, M.L. and Cassandra, A.R. Planning and Acting in Partially Observable Stochastic Domains, *Artificial Intelligence* 101: pp 99-134, 1998.
8. Kendall, M.G. A New Measure of Rank Correlation, *Biometrika* 30: pp 81-93, 1938.
9. Mitchell, T. Reinforcement Learning, Machine Learning, pp 367-390, McGraw Hill 1997.
10. Nilsson, N.J. Teleo-Reactive Programs for Agent Control, *Artificial Intelligence Research* 1: pp 139-158, 1994.
11. Nilsson, N.J. Teleo-Reactive Programs and the Triple-Tower Architecture, *Electronic Transactions on Artificial Intelligence* 5: pp 99-110, 2001.
12. Ryan, M.R.K. and Pendrith, M.D. An Architecture for Modularity and Re-Use in Reinforcement Learning, *Proceedings of the 15th International Conference on Machine Learning*, Madison, Wisconsin, Morgan Kaufmann, pp 481-487, 1998.