Optimizing Minimal Agents through Abstraction

Krysia Broda and Christopher J. Hogger

Department of Computing, Imperial College London
South Kensington Campus, London SW7 2AZ UK
{kb,cjh}@doc.ic.ac.uk
Technical Report 2006/4

Abstract. Abstraction is a valuable tool for dealing with scalability in large state space contexts. This paper addresses the design, using abstraction, of good policies for minimal autonomous agents applied within a situation-graph-framework. In this framework an agent's policy is some function that maps perceptual inputs to actions deterministically. A good policy disposes the agent towards achieving one or more designated goal situations, and the design process aims to identify such policies. The agents to which the framework applies are assumed to have only partial observability, and in particular may not be able to perceive fully a goal situation. A further assumption is that the environment may influence an agent's situation by unpredictable exogenous events, so that a policy cannot take advantage, of a reliable history of previous actions. The Bellman discount measure provides a means of evaluating situations and hence the overall value of a policy. When abstraction is used, the accuracy of the method can be significantly improved by modifying the standard Bellman equations. This paper describes the modification and demonstrates its power through comparison with simulation results.

1 Introduction

Our interest is in designing good policies for particularly simple autonomous agents. The simplest case is a memoryless reactive agent whose policy consists solely of some function that maps perceptual inputs to actions deterministically. A good policy disposes the agent towards achieving one or more designated goal situations, and the design process aims to identify such policies. We also consider modest extensions such as inclusion of finite memory, wireless communication and nondeterministic (relational) policies. The term *minimal agent* will be used loosely here to cover both the simplest case and these near-minimal extensions. The focus on minimal agents anticipates application contexts where physical or economic constraints make it impractical to deploy more sophisticated cognitive agents embodied in correspondingly sophisticated hardware. Examples include remote exploration and medical nano-robotics where the desired goals may be achievable by a large community of physically small and inexpensive primitive agents among which occasional losses and dysfunctionalities can be readily tolerated.

Our design method is based on discounted-reward analysis [11] applied to a directed policy graph whose arcs represent the transitions that occur under the policy being considered. Each transition takes the agent from some situation – a $(state, perception)$

pair – to some successor situation. The analysis yields an overall policy value that will depend upon, *inter alia*, the designated goal situation(s) and whatever rewards and probabilities are assigned to the graph's arcs by the designer. The method is called the *situation-graph-framework* (SGF) to reflect its reliance upon explicit situation graphs, and was first reported in [1]. The SGF framework can be distinguished from both the standard MDP and POMDP frameworks [7, 4]. It is non-Markovian, since an agent's next perception is conditional upon more than its current perception and action. The design process makes use of the full state, through the use of situations. The agents to which the framework applies are assumed to have only partial observability, and in particular may not be able to perceive fully a goal situation. This feature distinguishes the framework from other design methods that rely upon complete goal observation. The formulation in [6], where an agent's perception is treated as the state, is a special case of SGF. In particular, the various memoryless and single memory policies found by Q-learning are also found using SGF by the algorithm shown in Figure 5. The POMDP framework uses an estimation of the distribution of the full state, called a belief state, to guide the planning process [4]. This can result in policies in which the action taken when perceiving p may implicitly depend on the route taken to p – that is, an agent may follow a policy expressed by a graph. Agents in SGF are not equipped to follow such policies because they are designed for use in communities of agents, where unexpected events are the norm. The core assumption in using "belief states" is that remembrance of the past is a reliable basis on which to estimate an agent's current situation, which is a safe assumption in the specific circumstance that the environment can be impacted only by that agent. This assumption will not hold if the environment can be additionally impacted by exogenous events, including the actions of other agents. SGF represents such events by so-called x-arcs in the situation graphs and employs a particular elaboration of the discounted-reward analysis to deal with them. Experimental evaluation of policies designed in this way for communities of *cloned* agents was first presented in [2] and demonstrated strong empirical evidence for the efficacy of the design process. Communities of *differentiated* (non-cloned) multiple agents were investigated in [3], where it was shown how SGF could represent each species of agent by its own species of x-arc and extract, for each species, a so-called viewpoint graph representing the behaviour of that species in the context of all the others.

Whether dealing with single agents or communities, SGF – like the other frameworks – must in general confront the issue of scalability, and the key to this is appropriate use of *abstraction*. Broadly speaking, abstraction amounts to ignoring many minor distinctions – such as between states, perceptions or agents – that are considered unlikely to have a significant bearing upon outcomes. It achieves this by collecting similar concrete entities (such as states, etc.) into single generic entities which then become the first-class elements of the formulations used in the design process. To a certain extent the SGF viewpoint treatment just mentioned is such an abstraction, in that an x-arc in a viewpoint graph signifies an event instigated by some other agent but without specifying which particular agent it is, and so avoids the explicit and cumbersome multi-agent vectors that some MDP designers [8] have resorted to in order to deal with communities. Here we shall concentrate instead on *situation-abstraction*, that is, the abstraction of both states and perceptions. This can be applied irrespective of whether

one is dealing with one agent or many, though in this paper we present it only in relation to the single-agent case. Its first benefit is to reduce the size of the situation-graphs being dealt with and hence to ameliorate the burden of estimating the probabilities on their arcs. Its second benefit is that abstraction of perceptions reduces the size of the policy-space over which optimization is pursued, since the number of possible policies depends exponentially upon the number of possible perceptions.

Section 2 outlines the basic features of SGF and its discounted-reward procedure in the absence of abstraction. Section 3 describes situation-abstraction and explains how its deployment can produce inaccurate predictions of policy value if one relies upon the standard discounted-reward procedure. Section 4 describes our new modification of that procedure to reduce those inaccuracies and Section 5 presents empirical simulation results for two simple case studies to show the improved predictive power of this modification. Together, those two sections contain what we consider to be the novel contribution of the paper. Section 6 concludes with an assessment of the method in comparison with related work on other frameworks.

2 Basic Features of SGF

A simple example serves to illustrate the basic features of SGF. It assumes a single agent in a world comprising a region G called the ground, two weights and a balance having left and right ends L and R. In any state of the environment the weights are distributed among L, R and G. The distribution determines whether or not some end of the balance is raised. The agent is equipped to perceive in any state just one of the following: that there are some weights on G, that there are no weights on G, that the L end of the balance is raised or that the R end is raised. Figure 1 shows a situation in

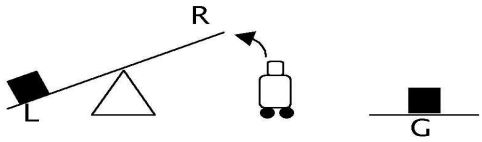

Fig. 1. A situation for a single agent

which one weight is on L, the other is on G and the agent perceives that R is raised. It is one of 10 possible situations, each being a pair (o, p) where o is a state and p is a perception that the agent may have of o. A state can be represented by a triple (l, r, g) giving the numbers of weights on L, R and G. The first two columns in Figure 2 show the possible states and perceptions, labelled 1-6 and a - d respectively. For compact presentation we use just these labels to denote situations: $1a$, $2a$, $2c$, $3a$, $3d$, $4b$, $4c$, $5b$, $5d$ and $6b$. Thus the situation ($(1, 0, 1)$, 'sees R is raised') shown in Figure 1 is denoted by $3d$.

The third column in Figure 2 shows for each perception p the set $A(p)$ of actions the agent might perform when perceiving p. Altogether there are 5 kinds of action, denoted

	o
1	(0, 2, 2)
2	(0,1, 1)
3	(1,0,1)
4	(0,2,0)
5	(2,0,0)
6	(1,1,0)

	p
a	sees G has weights
b	sees G has no weights
c	sees L is raised
d	sees R is raised

$A(p)$
$\{\mathtt{gr}, \mathtt{gl}, \mathtt{w}\}$
$\{\mathtt{w}\}$
$\{\mathtt{rg}, \mathtt{w}\}$
$\{\mathtt{lg}, \mathtt{w}\}$

Fig. 2. States, perceptions and action sets

by `gr`, `gl`, `rg`, `lg`, and `w`. The first four transfer a weight from G to R, from G to L, from R to G and from L to G, respectively, and thus effect a change of state. We must further stipulate what the agent perceives after performing one of these actions. After `gr` or `gl` it next perceives c or d unless no end is now raised, in which case it perceives b. After `rg` or `lg` it next perceives a. Informally, therefore, after putting a weight on the ground the agent next sees the disposition of the ground, whilst after putting a weight on the balance it next sees the disposition of the balance if the latter is tilted but otherwise sees the disposition of the ground. The `w` (wander) action leaves the state invariant and causes the agent to expend some time in updating its perception of that state; this includes the reflexive case of maintaining its current perception. Figure 3 shows the graph of possible transitions between situations, but to reduce clutter it suppresses each situation's reflexive w-arc. A policy for the agent is a total function from perceptions

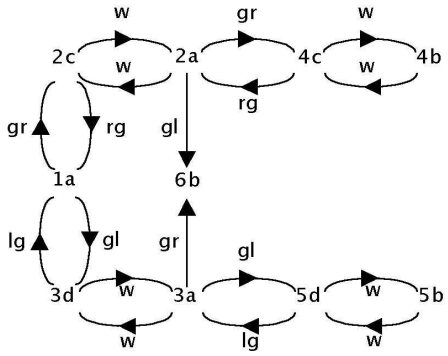

Fig. 3. Complete transition graph

to actions. The total number of possible policies is the product of the cardinalities of all the $A(p)$ action sets, which in the present case is 12. They include, for instance, the policy $\{a \rightarrow \mathtt{gr}, b \rightarrow \mathtt{w}, c \rightarrow \mathtt{w}, d \rightarrow \mathtt{lg}\}$. Each one corresponds to a restriction of the complete graph whereby the arcs emerging from all situations sharing a common perception all bear the same action label.

The value of a policy partly depends upon the chosen goal situation. we shall choose the goal to be $6b$, in which the balance bears one weight at each end and the agent is seeing the vacant ground. The status of the chosen goal is reflected in the assignment of numerical rewards to the arcs in the complete graph, each being a measure of the supposed benefit/disbenefit of effecting the associated transition. We might, for instance, assign a large positive reward $R = 100$ to each of the two arcs leading to $6b$ and a small but negative reward $r = -1$ to every other arc. A policy's value partly depends also upon the probabilities assigned to the arcs. If practicable, these are estimated from consideration of the particular problem domain. Otherwise, we can assign at each situation an equi-probable distribution to its emergent arcs for each action type.

Once the assignments of rewards and probabilities are in place, a value for each situation under a given policy can be calculated from $V(s) = \Sigma_{u \in SS}(P_{su} \times (\Upsilon_{su} + \gamma \times V(u)))$, the standard bellman formula, in which Υ_{su} is the immediate reward for the action that takes s to u, P_{su} is the probability that from s the agent proceeds next to u and the parameter $0 < \gamma < 1$ is a discount factor that ensures the resulting set of linear equations has a unique solution. If it is assumed that the agent may begin its activity at any situation then the overall policy value is just the mean of the situation values. Applying this to the above example, taking $R = 100$, $r = -1$, $\gamma = 0.9$ and assuming equi-probable distributions, it turns out that 4 of the 12 policies are co-optimal for the chosen goal, these being: $\{a \rightarrow \texttt{gl}, b \rightarrow \texttt{w}, c \rightarrow \texttt{rg}, d \rightarrow \texttt{lg/w}\}$ and $\{a \rightarrow \texttt{gr}, b \rightarrow \texttt{w}, c \rightarrow \texttt{rg/w}, d \rightarrow \texttt{lg}\}$, all having value 357.27. The worst 4 all have value 91 and include $\{a \rightarrow \texttt{w}, b \rightarrow \texttt{w}, c \rightarrow \texttt{w}, d \rightarrow \texttt{w}\}$.

In the above account the situations in the formulation were taken to be *concrete*, that is, corresponding one-to-one with the real situations arising in the problem domain. In the next section we turn to the use of abstraction, in which each generic situation in the formulation may encompass many concrete situations.

3 Situation Abstraction in SGF

Abstraction in SGF partitions the set of concrete states into subsets called generic states and partitions the set of concrete perceptions into subsets called generic perceptions. A generic situation (O, P) is a pairing of a generic state O with a generic perception P. This abstraction process is required to satisfy the following constraints:

1. if (o, p) is a concrete situation then there must exist exactly one generic situation (O, P) such that $o \in O$ and $p \in P$;
2. if (O, P) is a generic situation then it must contain at least one concrete situation (o, p) where $o \in O$ and $p \in P$;
3. if P is a generic perception then $\bigcap\{A(p)|p \in P\} \supseteq A(P)$.

Here, 1) and 2) ensure that the sets of concrete states and perceptions are partitioned such as to result in a partitioning of the complete set of concrete situations, whilst 3) ensures that every generic perception offers at least one action among those offered by each of its concrete members.

With fewer situations to deal with at the abstract level than at the concrete level, the equations relating situation values for any given policy are correspondingly fewer.

Perhaps more importantly, having fewer perceptions at the abstract level than at the concrete level reduces the number of policies to be evaluated.

Intuitively, a good abstraction is one whose discounting of differences at the concrete level yields a ranking of abstract policies that is approximately commensurate with the ranking of the concrete policies that would be obtained from concrete analysis. At present we do not have a clear prescription for reliably identifying such abstractions in SGF, and indeed the question of what constitutes a good abstraction *in general* remains an open one in AI research [9]. However, given any abstraction in SGF, we will elucidate the manner in which the standard Bellman formula is susceptible to inaccuracy when applied to it and we will show how to obtain improved accuracy by suitably modifying that formula.

We illustrate these issues with an example, again assuming a single agent. The environment in which this agent operates is called *Token World* and contains some fixed number N of tokens. It is organized as one or more heaps of tokens together with a single region named *void* in which there are no tokens. As its perception, the agent always sees either a heap or *void* and always knows whether or not it is holding a token. Its possible actions are just the following: `gr`: grab a token from a heap; `dr`: drop a token onto a heap or onto *void*; `w`: wander.

Prior to performing `gr` the agent must be not holding and seeing a heap, and is afterwards holding a token and seeing the reduced heap. Prior to performing `dr` it must be holding a token and seeing a heap or *void*, and is afterwards not holding and seeing the heap containing the token just dropped. Prior to performing `w` it can be holding or not and seeing anything, and is afterwards seeing a heap or *void* with its (not)holding status preserved. The goal will comprise some specified configuration of heaps and some perception, not necessarily perceivable in its entirety by the agent. This *Token World* may alternatively be viewed as a simple analogue of *Blocks World* or as a system for incrementally transforming partitions of the number N.

In this formulation the region in which there are no tokens is treated as indivisible. The alternative would be to represent a plurality of tokenless regions, e.g. as the vacant cells in a grid-space. Although that is notionally more realistic as a model of an environment in which an agent may wander from place to place, it turns out to confer no material benefit in assessing the relative merits of policies, whether that be done through analysis or through simulation. Using our single *void* representation, an agent dropping a block onto *void* thereby creates a new 1-token heap leaving *void* preserved, whilst an agent grabbing the token from a 1-token heap merely eliminates that heap. Figure 4 shows some legal transitions in the case that $N = 10$. Taking the above notions to define the concrete representation, a concrete formulation of the complete situation graph would entail 72 states, 21 perceptions, 236 situations and 220 policies.

We now consider one possible abstraction for the case when the goal is to achieve a configuration having exactly 3 identical heaps and not exactly 2 identical heaps, with the agent seeing *void* and not holding. The concrete states are partitioned into abstract states 1-4 and the concrete perceptions into abstract perceptions $a - h$:

1. exactly 3 identical heaps and not exactly 2 identical heaps
2. exactly 2 identical heaps and not exactly 3 identical heaps
3. the heaps are all different

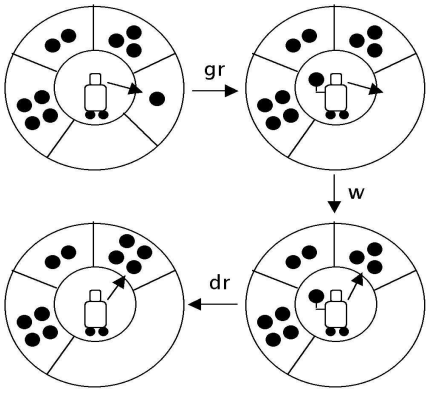

Fig. 4. Transitions in *Token World*

4. all other cases

Examples of states 1 and 2 are $\{2, 2, 2, 1, 1, 1, 1, \textit{void}\}$ and $\{2, 2, 1, 1, 1, 1, 1, 1, \textit{void}\}$ respectively.

$a(b)$. sees a heap of size < 4 and holding(not holding)
$c(d)$. sees a heap of size > 4 and holding(not holding)
$e(f)$. sees a heap of size $= 4$ and holding(not holding)
$g(h)$. sees *void* and holding(not holding)

The goal is therefore $(1, h)$. There are then 32 abstract situations in the graph and just 128 policies to evaluate. Each of the latter maps abstract perceptions to actions and is therefore an abstract policy spanning some set of concrete policies. In effect, the abstraction process partitions the concrete policy space as well as the concrete situation space. Simulation experiments indicate that the optimal abstract policy is $\{a \rightarrow \mathtt{dr}, b \rightarrow \mathtt{w}, c \rightarrow \mathtt{dr}, d \rightarrow \mathtt{w}, e \rightarrow \mathtt{dr}, f \rightarrow \mathtt{w}, g \rightarrow \mathtt{dr}, h \rightarrow \mathtt{w}\}$.

The standard Bellman formula takes the probabilities on the arcs to be Markovian: the value of any situation is calculated using the expectation over its emergent arcs without regard to how that situation was reached. Consider situation $1a$ in the graph for some policy in which $a \rightarrow \mathtt{w}$ and $c \rightarrow \mathtt{w}$. The successors of $1a$ are then $[1a, 1c, 1e, 1g]$. If the probabilities are estimated simply as the mean, over all concrete instances i of $1a$, of the probability that i can transit by $\mathtt{w}$ to these successors, their values are about [0.75, 0.025, 0.042, 0.183]. This takes the view that if the agent arrives by any means at $1a$ then the probability that it will next wander to $1e$ is 0.042. But this is not the case. Had the agent arrived at $1a$ from $1c$, for example, the probability of it next wandering to $1e$ would be zero: the only concrete state in which the agent can transit by $\mathtt{w}$ from $1c$ to $1a$ is $\{6, 1, 1, 1, \textit{void}\}$ in which it is impossible for the agent to wander to see a heap of size 4 (perception e).

Therefore, if the formula is applied to an abstract policy graph with probabilities estimated as just indicated above, it will perceive paths through the graph that might not

be concretely traversable at all or traversable with quite different probabilities, yielding misleading policy values. We refer to this property of the paths as *piecewise incoherence*. The next section discusses our modification of the formula with the aim of ameliorating this deficiency.

4 Evaluating Abstract Situations and Policies

The inaccuracies from piecewise incoherence in the abstract context can be reduced by modifying the standard Bellman formula. As it stands, the latter yields a set of linear equations expressing each value $V(i)$ of abstract situation i in terms of the values of the successor set $SS(i)$ of i. The first stage of our modification reformulates $V(i)$ as

$$V(i) = \Sigma_{j \in SS(i)} p_{ij}(r_{ij} + \gamma V(j|i))$$

where $V(j|i)$ is the contribution made to $V(j)$ by all those concrete transitions that transit to j from i, and p_{ij} is the average probability of those transitions. This stage therefore introduces a new set of conditional variables of the form $V(j|i)$. The second stage, which is somewhat more subtle, inter-relates these new variables as follows:

$$V(j|i) = \Sigma_{k \in SS(j)} p_{ijk}(r_{jk} + \gamma V(k|j))$$

where p_{ijk} is the probability that k is reached from i via j. Together these two formulations yield a new set of linear equations, involving more variables and probabilities than before, from which the various $V(i)$ values can be calculated. The abstract policy value is then again the mean of these. The modification thus gives recognition to the fact that, in the abstract context, the value of a situation has a non-Markov dependence upon its immediate predecessors. In principle one could extend that recognition still further by considering dependences upon remoter predecessors, but any further accuracy so obtained would generally incur much greater computational expense.

Even significantly abstract formulations may, with this modification, offer a large number of equations, so it is important that an efficient procedure be used to extract the best policies. For this we use a branch-and-bound algorithm, adapted from Littman [5], which, for some number $n > 0$, develops a tree of partially-constructed policies whilst pruning those that could not - if fully extended to become complete policies - be among the n highest-value policies. To find only some optimal policy, n is chosen as 1. This algorithm, whose pseudocode is shown in Figure 5, can be used whether the problem formulation is abstract or concrete. The next section returns to the *Token World* context to demonstrate that the modification improves predictive quality in relation to empirical simulation results.

5 Prediction and Simulation Results

We have applied the modified treatment just described to a range of examples over different domains and have observed in all these cases an improvement in the correlation between predicted and simulated policy values. We illustrate this for two goals in *Token World*.

```
procedure BB(c:node, B:int):(G:int, g:node)
//returns G=best policy value\&g=best policy
   {if c is a leaf-node
        //c represents a full-policy
     then {if value(c)>B return (value(c),c)}
       //c could be the optimum policy
     elseif upperbound(c)>B then
       {for each cc:=next-child-of-c
         {if lowerbound(c)>B
            then return BB(cc,lowerbound(c))
            else return BB(cc,B)}
          }
     else return (B,b)}
    //do not search below cc if cc worse than B
    }
```

Fig. 5. Branch and Bound Algorithm for searching a policy tree

5.1 Goal: Achieving 3 Identical Heaps

Here the example is that described in Section 3, where the world has 10 tokens and the goal is to transform any initial situation to one having exactly 3 identical heaps but not exactly 2, with the agent seeing *void* and not holding. The abstraction used is the one shown there, having 32 situations and 128 policies. We evaluated all the policies using first the standard Bellman equations and then our modified equations. The probabilities were calculated by analysing all the concrete transitions. The predicted policy values were then compared with the results of simulating all the policies. Each one was run 500 times, with random initialization, for up to 50 transitions. (For economy, any run achieving the goal was terminated at that point, and to reflect this in the prediction we suppressed the goal's emergent arcs). The reward parameters were $R = 100$, $r = -1$ and $\gamma = 0.9$.

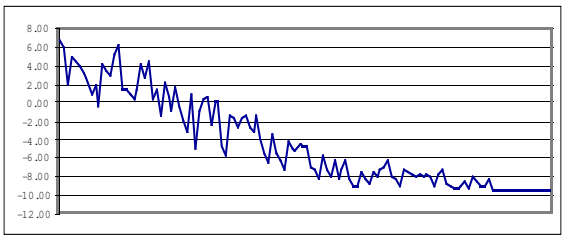

Fig. 6. Using standard Bellman for goal 5.1

Figure 6 charts the simulation values (vertical axis) against the increasing ranks of the values predicted by standard Bellman (horizontal axis), so that the predicted-best policies are on the left. If the prediction were perfect the chart would decrease mono-

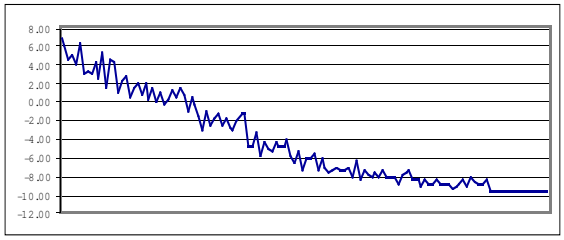

Fig. 7. Using the modified equations for goal 5.1

tonically from left to right. The correlation between predicted and simulated values is measured by the Kendal coefficient as a percentage ranging from 0 (worst) to 100 (best). For Figure 6, Q is 91.4% over all 128 policies and 69.5% across the first 20. Figure 7 shows the results using the modified equations. There, Q is 94.8% for all 128 and 78.4% for the first 20. In both cases the predicted optimal policy is the one that is optimal in simulation.

5.2 Goal: Achieving Exactly 1 Heap

Here there are again 10 tokens but the goal is the much harder one of arranging them into a single heap. For this problem we used a different abstraction, partitioning the states into 4 cases: one heap of 10; two heaps of 5; exactly one heap of 5 plus anything else; no heap of 5 or of 10. The perceptions were partitioned according to whether the agent was seeing *void*; seeing a heap less than 5; equal to 5; greater than 5. This yields 16 abstract situations and 128 policies.

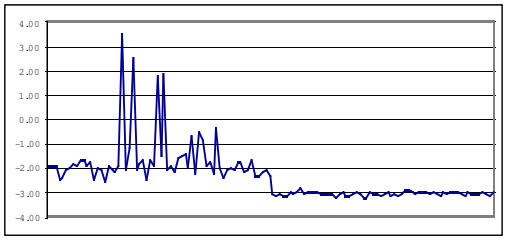

Fig. 8. Using the modified equations for goal 5.2

Proceeding now as in the previous case, Figure 8 shows the chart using standard Bellman prediction, where Q is 66.4% for all 128 policies but only 31.1% for the first 10. Figure 9 shows the chart using the modified equations, where Q is 64.7% (not quite as good) for all 128 but 66.7% (radically improved) for the first 10. Moreover, the best policy from simulation is now predicted as best, whereas in the standard prediction it is ranked 21.

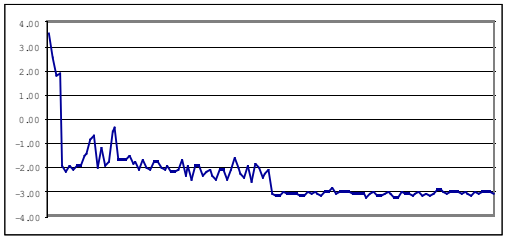

Fig. 9. Using the modified equations for goal 5.2

6 Discussion and Conclusion

Policy design frameworks can be compared in terms of their tradeoffs between ease of problem formulation, complexity of policy optimization and predictive accuracy. Some are not directly comparable as they assume different agent architectures. MDP/POMDP methods assume agents capable of holding and consulting perception-action graphs whose paths represent the episodes an agent may experience only when undisturbed by exogenous events. By contrast, our target agents presume a simpler policy structure and maintain optimal behaviour in all situations whether these have arisen by their own actions or not. The backward-planning design method of Nilsson [10] for teleo-reactive agents also assumes a different agent architecture: there, the agent must have sufficient observability to take the best action in any situation and, crucially, in a goal situation. In many realistic contexts, however, the state component of a goal is too delocalized to be fully perceivable in practice. All the above methods, including SGF, involve estimating probabilities, in contrast with those based upon learning. Q-learning [11, 6] can discover optimal policies having our structure but, like MDP methods, requires agents to have full observability.

SGF has the distinctive feature that probability estimation is a once-only task for the given complete situation graph, independently of all policies and goals that might then be considered. In a POMDP framework each change of goal demands an evaluation of a new set of belief state probability distributions to find an optimal policy for achieving it.

The use of abstraction in SGF assumes that policy ranking is not overly sensitive to the small variations between the concrete situations spanned by an abstract one. The equations we employ are expected to deliver for each abstract policy a value to which all its concrete policy instances closely approximate. Our simulation studies of abstraction using the standard Bellman equations showed that they cannot be relied upon to have this property. However, the modified equations in the cases we have tested, including those presented here, have manifested this property. In future work on SGF we shall investigate how robust the property is in relation to the choice of abstraction.

References

1. K. Broda, C.J. Hogger and S. Watson, Constructing Teleo-reactive Robot Programs, *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI-2000)*, Berlin,

pp. 653-657, 2000.
2. K. Broda and C.J. Hogger, Policies for Cloned Teleo-Reactive Agents, 2nd Conference on Multi-Agent System Technologies, Ehrfurt, LNAI, 3187, Springer Verlag, pp. 328 - 340, 2004.
3. K. Broda and C.J. Hogger, Abstract Policy Evaluation for Reactive Agents, *SARA-05, 6th Int. Symposium on Abstraction, Reformulation and Approximation*, Springer, LNAI 3607, pp. 44-59, 2005.
4. L.P. Kaelbling, M.L. Littman and A.R. Cassandra, Planning and acting in partially observable stochastic domains, *Artificial Intelligence*, 101, pp. 99-134, 1998.
5. M. L. Littman, Memoryless policies: theoretical limitations and practical results, *Proceedings of the 3rd International Conference on Simulation of Adaptive Behaviour* , MIT Press, pp. 297-305, 1994.
6. J. Loch and S. Singh, Using Eligibility Traces to find the Best Memoryless Policy in Partially Observable Markov Decision Processes, *Proceedings of the 15th International Conference on Machine Learning*, pp. 323-331, 1998.
7. T. Mitchell, *Machine Learning*, McGraw Hill, 1997.
8. R. Nair, M. Tambe, M. Yokoo, D. Pynadath and M. Marsella, Taming Decentralised POMDPs: Towards Efficient Policy Computation for Multiagent Settings, *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*, pp. 705-711, 2003.
9. M. Lauer and M. Riedmiller, Generalisation in Reinforcement Learning and the Use of Observation-Based Learning, *Proceedings of the FGML Workshop 2002*, pp. 100-107, 2002.
10. N.J. Nilsson, Teleo-Reactive Programs and the Triple-Tower Architecture, *Electronic Transactions on Artificial Intelligence*, 5, pp. 99-110, 2001.
11. R.S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.