An Adaptive Robot Motivational System

George Konidaris and Andrew Barto

Autonomous Learning Laboratory
Department of Computer Science
University of Massachusetts at Amherst
{gdk, barto}@cs.umass.edu

Abstract. We present a robot motivational system design framework. The framework represents the underlying (possibly conflicting) goals of the robot as a set of drives, while ensuring comparable drive levels and providing a mechanism for drive priority adaptation during the robot's lifetime. The resulting drive reward signals are compatible with existing reinforcement learning methods for balancing multiple reward functions. We illustrate the framework with an experiment that demonstrates some of its benefits.

1 Introduction

Autonomy is central to intelligence—an intelligent system that requires the external specification of its goals is a tool, not an agent, because it fails the basic test of agency. To be autonomous, an agent requires an internal motivational system that appropriately values the actions available to it and generates its goals. In natural agents this system is evolved, but in artificial agents we must design it.

Reinforcement learning [17] is a learning, planning, and action selection paradigm based on maximising reward. Although it does not deal with the problem of designing the motivational system that generates those rewards, it is an intuitively appealing model of motivation-based learning. The importance of motivation to intelligent robot design was recognised early [7, 3], and building motivational systems based on reinforcement learning is still an area of active research (e.g. [6]).

In this paper we attempt to bridge the gap between motivation and action selection by outlining the properties that a motivational system should have, and by introducing a design framework based on them. We illustrate our framework with an experiment demonstrating its benefits.

2 Background

Reinforcement learning relies on the existence of a reward function that penalises bad actions and reward good actions. If we are to use it as a method of action selection for an autonomous robot, we require a reward generating mechanism (or *motivational system*) that expresses the robot's internal goals and motivations [3]. This mechanism will likely consist of multiple parts—real animals want more than one thing, and autonomous robots are likely to have multiple (possibly conflicting) simultaneously active concerns (at the very least, to keep running and simultaneously complete whatever task it is that we designed them for).

This leads to the concept of a drive as *the motivational unit underlying behavior*, a module that expresses one of the robot's purposes and produces motivational force as a *common currency* [12] for use in action selection.

Two types of drive are present in the artificial intelligence literature. Systems using homeostatic regulation [1] endow the agent with a set of internal physiological variables, each with an optimal range. Actions that move a physiological variable outside of this range are punished, and actions that move it toward this range are rewarded. Systems using Hullian drives (e.g. [10, 11]) maintain a set of drives, each with a drive level that varies between totally unsatisfied and fully satiated. Reward is generated by drive level difference, so actions that raise the level are rewarded and those that lower it are punished.

In this paper we use Hullian rather than homeostatic drives because not all drives are homeostatic, and any homeostatic drive can be simulated using a pair of Hullian drives (one to penalise going above the ideal range, and one to penalise going below). This increases the number of drives, but also adds flexibility because the two directions can be handled separately.

3 Desirable Properties

An ideal robot motivational framework would possess the following properties:

1. A **drive interface specification** that provides a well defined and consistent way of specifying drives.

2. A **reward generating mechanism** that allocates drive-specific reward to actions given their effect on the drive, rooting action selection in the agent's motivational state.

3. A **drive priority** mechanism so that the agent can adjust the relative urgency of each of its drives during its lifetime.

4. **Numerically comparable rewards and priorities**, so that drive rewards can be used as a common currency when comparing and balancing the demands of various drives.

5. An efficient **action selection mechanism** that balances the demands and priorities of multiple drives.

The split between reward and priority is not self evident, but we treat them separately because such a split is adaptive. Drives and their associated reward mechanisms will in most cases be fixed by design or evolution, and be grounded in the agent's "physiology". Priority, however, should be a property of the environment the agent finds itself in, reflected in its own history. Agents in an environment where water is scarce but food is not should be able to learn to value water over food (and therefore have a higher water drive priority). These aspects of the environment are difficult to predict at design time and may change during the agent's lifetime.

4 Overview

Figure 1 shows an overview of our motivational framework. The framework employs a collection of drives $d_1, ..., d_n$, where each drive d_i maintains a satiation level σ_i and a priority level ρ_i. The priority level determines the shape of a priority curve, which translates satiation level to drive priority κ_i. For each time step, the reward generated by the drive is obtained by multiplying the difference in satiation between time steps by the drive priority, and the agent's aim at any given time is to maximize the sum of these rewards. A detailed description of each of these elements is given the following sections.

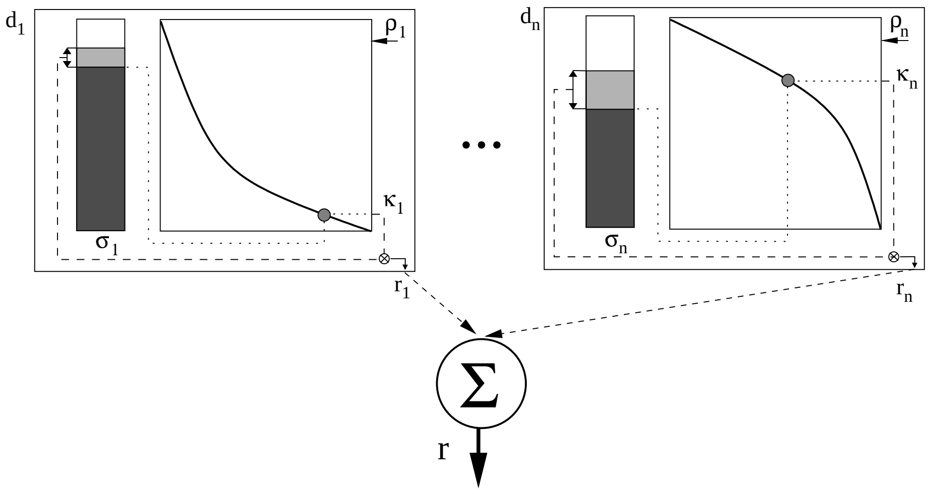

Fig. 1. An overview of our motivational framework. Each drive d_i maintains a satiation level σ_i, and the reward signal it generates is the difference in σ_i from one point in time to the next (shown in light gray) multiplied by a drive priority κ_i. Drive priority is determined by translating σ_i using a priority curve, the shape of which is determined by the drive's priority parameter ρ_i. The agent aims to maximise $r = \Sigma_i r_i$ for action selection.

5 Representing Individual Drives

Each individual drive d_i consists of the following components:

1. A *satiation level* $\sigma_i \in [0, 1]$, where at $\sigma_i = 0$ the drive is starved (and the agent may cease functioning), and at $\sigma_i = 1$ the drive is satiated and should have no effect on the agent's behavior.

2. A *drive process* that updates σ_i according to the drive's intended purpose.

3. A *priority parameter* $\rho_i \in (0, 1)$, which reflects the agent's long-term belief about the difficulty of raising σ_i. A high value for ρ_i indicates that d_i is difficult to satiate and should thus have a high priority, whereas a low value indicates that it is easy to satiate and should thus have a low priority.

4. A *priority process* that monitors d_i's satiation history and slowly increases or decreases ρ_i to reflect the drive's long-term priority.

The priority parameter ρ_i is used to affect the shape of a priority curve that determines the drive's priority κ_i given its current satiation level, according to the following equation:

$$\kappa_i = 1 - \sigma_i^{\tan \frac{\rho_i \pi}{2}} .$$

Thus ρ_i allows the agent to adjust its drive priorities without changing the underlying drive process. Figure 2 shows sample priority curves for a few values of ρ_i. A very high value of ρ_i means that the drive attains a high priority even when σ_i is near 1 (satiation), but a low value of ρ_i means that the drive reaches a high priority only when it is near 0 (starvation). As we would expect, drive priority is 1 at $\sigma_i = 0$ and 0 at $\sigma_i = 1$ irrespective of ρ_i, and at $\rho_i = 0.5$ the priority curve is a straight line.

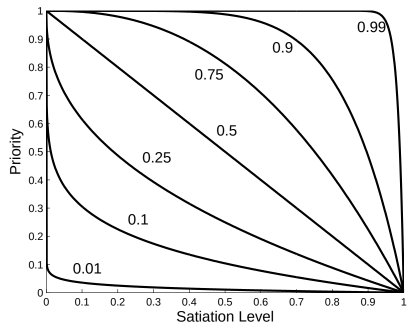

Fig. 2. Satiation level modulated by sample priority parameters.

When the agent performs an action, it results in a change in σ_i for drive d_i, and that change multiplied by the priority level for the drive results in drive reward r_i:

$$r_i(t+1) = \kappa_i(t)[\sigma_i(t+1) - \sigma_i(t)].$$

This results in two design problems per drive: defining a drive process that expresses the drive's intended purpose and defining an appropriate priority process. The first will differ for each drive, but will in most cases be grounded in the robot's physical state. The second will likely be based on the drive's satiation level history over a long period of time. Section 7 employs a simple priority adjustment heuristic, but in cases with more drives we expect more complex, drive specific rules will be required.

6 Combining Drives for Action Selection

We are faced with an action selection problem over a state space comprising both the internal state of the robot and the external state of the environment. More specifically, we are trying to find a policy π:

$$\pi : (s_E, s_I) \mapsto a,$$

where s_E is an external state descriptor, $s_I = (\sigma_1, ..., \sigma_n)$ is an internal state descriptor, and a is an action. We omit the priority parameters, $\rho_1, ..., \rho_n$, from s_I because we can consider them constant as they are expected to change slowly over the lifetime of the agent. This means that π is nonstationary, but it will only change gradually.

Learning this policy directly may be difficult because the resulting state space may be much larger than s_E, the state space of the problem the robot is solving. Furthermore, this space has significant redundant structure: only (s_E, a) is useful in predicting s'_E (s_E at the next timestep), and since drive satiation levels do not interact, (s_E, a, σ_i) uniquely determines σ'_i and thus r_i (and each satiation level may only depend upon a subset of s_E).

Unfortunately, it is easy to construct examples where varying just one drive (e.g., by starving it) drastically changes the optimal policy for a given environmental state. Therefore any individual drive's value function or policy that is not a function of both s_E and all of s_I must be an approximation.

There are two possible ways to exploit the structure of this state space. The first is to attempt to learn π directly, using a function approximator specifically designed to take advantage of this structure. Although learning would take place over both s_E and all of s_I, if the function approximator is well designed then the extra dimensions may not make the problem significantly harder.

Alternatively, if such a solution is not feasible, we can learn a value function Q_i for each drive simultaneously and independently (as a function of (s_E, a, σ_i) only), and combine them to form an overall value function Q. Several solutions to this problem have been proposed in the literature. Of these, Sprague and Ballard [16] show that using Sarsa (an *on-policy* reinforcement learning algorithm) to learn each Q_i and then setting $Q(s,a) = \Sigma_i Q_i(s,a)$ performs best. Using an on-policy learning algorithm prevents each Q_i from overestimation, since an off-policy algorithm (like Q-learning) would compute each drive's action values assuming that their own optimal policies will be followed thereafter.

Using this method is equivalent to treating the values of the other drives as hidden state, so the resulting state values for each drive will be the same as the correct value state values for that drive at the expected value of each of the other satiation levels.

7 Experiments

In this section we use Spier's domain [14] to illustrate the benefits of the priority aspects of our framework. An agent (of width 60 units) is placed in a $10,000 \times 10,000$ toroidal grid containing two types of uniformly scattered resources (of width 60 units). There are 14 of the first (dark) type of resource, but only 8 of the second (lighter) type. The agent is able to perceive the proximity (scaled from 0 to 1) and angular distance (scaled from -1 to 1) of the nearest three of each type of resource within a perceptual range of 1500 units, resulting in a sensor space of twelve continuous variables. The agent can move forward ten units in the direction it is facing or rotate in either direction by ten degrees, and has a high-level behavior for approaching and consuming each visible target. Consumed resources are randomly replaced. Figure 3 shows an example instance.

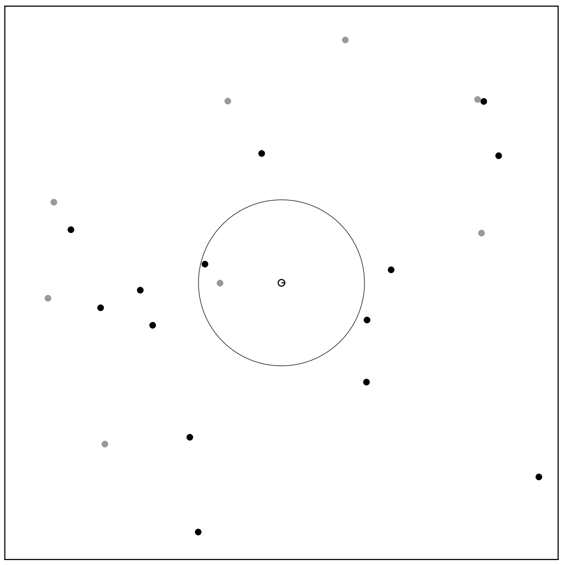

Fig. 3. An example domain instance. The large circle indicates the agent's perceptual radius, so it can see one of each resource type.

The agents were given two drives (one for each resource type), each using a satiation level penalty of 0.00015 for a movement and a satiation level gain of 0.1 for each successful consumptive act. Each drive's priority parameter and satiation value was initially set to 0.5. Learning was by Sarsa(0) and gradient descent using a linear function approximator ($\alpha = 0.01, \gamma = 0.9$) representing separate value functions for each drive, each over the twelve continuous-valued sensory attributes plus one continuous satiation level. Actions were chosen from the available high-level approach and consume behaviors. We randomly generated 50 sample environments, and ran three types of agents for 250 episodes of 10 consumptive acts each. The first type of agent used drive reduction as a reward, the second used our framework but with fixed priority parameters ($\rho = 0.5$), and the third moved each drive's ρ value after each episode toward the ratio of the two drives' number of successful consumptive acts (using $\alpha = 0.1$).

The results are shown in Figure 4. The agent that does not take priority into consideration (Figure 4a) first satiates (at about episode 100) the drive associated with the plentiful resource, and only then (when further consumptive acts on this resource create less reward because it is near satiation) makes progress bringing its second drive towards satiation, although by the end of 250 episodes the second drive has not reached 80% satiation. The agent using priority curves (Figure 4b) allocates a higher reward to the lower drive—because it is higher on the priority curve by virtue of its low satiation value—and therefore increases both drives simultaneously, with the second drive reaching well over 80% satiation by 250 episodes. This occurs even though both drives have the same priority parameters. Finally, the agent with flexible ρ values (Figure 4c) is able to better balance the two satiation levels, even though they cannot be made to match because of differences in resource frequency. It also reaches well over 80% satiation by 250 episodes.

Note that (as can be seen in Figure 4d) the final ρ values inversely reflect the frequency of each resource, with the ρ value for the light (scarce) resource converging to approximately double that of the dark (plentiful) one.

8 Related Work

The relevant work in motivational system design[1] is primarily split into two threads: research on motivational models based on drives and research on balancing multiple reward functions.

8.1 Motivational Models Based on Drives

Cañamero [4] introduced a motivational model using homeostatic drives where each drive has an error signal (similar to a difference in satiation) which translated to an activation level (similar to a priority level), which could also be

[1] We note that a great deal of research exists on computational models of natural motivational systems. Since this paper is concerned with motivational system *design* rather than modelling, we refer the interested reader to Savage's review [13].

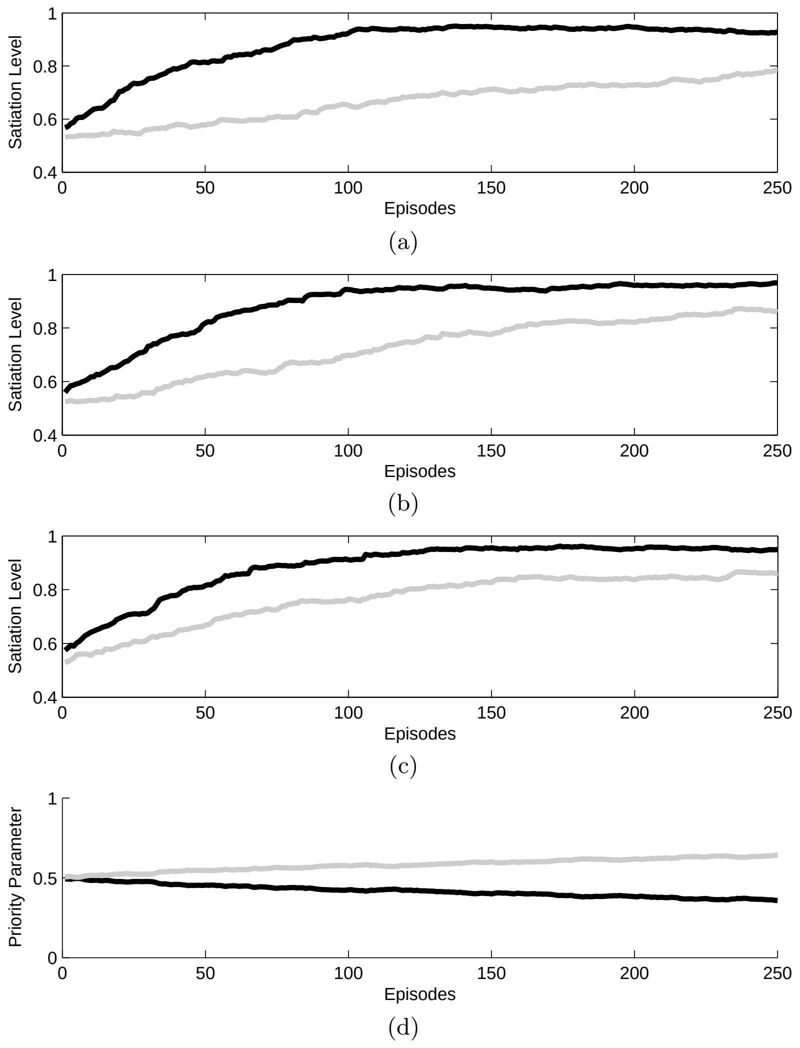

Fig. 4. Average (over 50 trials) drive satiation levels for two drives given an uneven distribution of resources. The first graph (a) shows satiation for an agent using drive difference directly as reward. The second (b) shows satiation for an agent with fixed priority parameters, and the third (c) shows the same for an agent that adjusts each drive's priority parameter to match observed resource frequency. The final figure (d) shows the third agent's priority parameters changing over time.

increased by an incentive stimulus (the presence of a goal object) or an activation modifier. The behavior attached to the drive with the highest activation is selected for execution. This system did not use learning for action-selection, but

Cos Aguilera, Cañamero and Hayes [5] built a similar system using a simpler model and reinforcement learning with dominant-drive action selection.

Blumberg [2] described a behavior-based architecture that uses internal Hullian drives, where action selection uses the behavior with the highest activation (internal motivation multiplied by incentive stimulus). Reward generated by drive difference is used to associatively learn the value of incentive stimuli using reinforcement learning, but not for action selection.

Spier and MacFarland [15] empirically compared five different models of decision-making in a two-resource domain using Hullian drives. This work did not use reinforcement learning (although some of the decision models are similar), and the resources were available in equal quantities so the notion of drive priorities was absent.

Finally, Konidaris and Hayes [10, 11] recently built a situated reinforcement learning system based on Hullian drives similar to ours (but without a priority mechanism), using a circadian switching mechanism for drive selection.

8.2 Balancing Multiple Reward Functions

To the best of our knowledge, Whitehead, Karlsson and Tenenberg [19] were the earliest to recognise that a reinforcement learning robot might have multiple goals expressed as separate reward functions and need to balance them. They introduced the idea of reward functions that could be "switched off" and proposed both setting Q to the highest individual Q_i value and setting it to the sum of the Q_i values as modular action selection mechanisms. They pointed out that these methods must respectively understimate and overestimate the monolithic (true) value function, and that using the maximum Q_i does not perform drive balancing, while summing the Q_i values may lead to actions where no drive receives a reward at all. However, their empirical results suggest that the difference between either modular method and the monolithic method is small, and may be more than made up for by the resulting increase in learning speed.

Sprague and Ballard [16] pointed out that these problems arise from using off-policy methods, which compute each drive's action values assuming that its own optimal policies will be followed thereafter, and show improved performance using Sarsa(0) (an on-policy learning algorithm). However, unlike Whitehead et al. [19] they assume that each value function is always active, in which case their method is not an approximation.

Humphrys [8] surveys several methods for balancing multiple reward functions, placing them on a continuum from single-minded to cooperative, and introduced W-learning, where agents explicitly learn a weight for each drive in each state expressing the extent to which that drive would suffer if its preference is not taken.

Although all of these papers are strongly related to this research, none of them employed drive mechanisms. Thus they did not include satiation (beyond on or off) or priority levels, and could not guarantee numerically comparable rewards.

9 Discussion

9.1 The Multiplicity of Drives

Even the most intuitive drives may not be atomic upon closer inspection. For example, sodium-depleted rats displays an enhanced appetite for food containing sodium [18], which suggests that they have a separate sodium seeking drive and possibly other nutrient seeking drives, instead of a single hunger drive. However, a system approaching the complexity of an animal will need to maintain so many internal variables that creating and balancing separate drives for each of them is not likely to be feasible.

One way to get around this would be for each drive to represent many internal variables likely to be systematically reduced by the same activity (e.g., nutrient levels are all modified by eating). The drive process could then modify what changes its satiation level according to the system's current needs (e.g., fruit becomes more rewarding when the agent needs sugar). We could even view each agent-level drive as composed of several subdrives, so that a hunger drive is composed of a salt drive, a sugar drive, etc., each with a very small state space.

9.2 Motivational Systems as a Basis for Further Learning

Once the motivational system described in this paper is present in an agent, it could also be used a way of focusing other types of learning. For example, Cos Aguilera, Cañamero and Hayes [5] learn object affordances based on changes in motivational state, where the effect of a behavior is quantified in terms of its effect on the agent's drives. Another example is provided by Konidaris and Hayes [10], where a robot learns associations between reward and the sensations present at reward states to speed up reinforcement learning in novel environments. This results in guided (as opposed to blind) searches in new environments [9], using a form of heuristic that can be learned autonomously. We expect that robot control architectures based on a motivational system will provide further opportunities for motivationally grounded learning.

9.3 Non-Physiological Drives

When designing a sufficiently complex autonomous robot, we may wish to include motivational aspects that do not involve "physiological" attributes. For example, we may wish to motivate the robot to seek social interaction, or to avoid verbal reprimand. Such motivations can be represented in our framework, and thus integrated and balanced against physiological factors, although their implementation may not be as natural as physiological drives. In such cases the drive satiation level or priorities may be kept within a smaller range than usual so that the robot's "physiological" needs are never completely overridden for less immediate motivations. Alternatively, constraining drive priorities to a particular ordering may be a useful way to build safety measures or sanity checks into the robot.

10 Summary

We have presented a robot motivational system framework that provides a simple interface specification for drives, a mechanism for reward generation that guarantees numerically comparable rewards, and a natural method for adjusting drive priorities. The resulting reward structure is compatible with existing reinforcement learning methods for balancing multiple drives.

Acknowledgements

We would like to thank Gillian Hayes, John Hallam and Ignasi Cos Aguilera for their useful input. Andrew Barto and George Konidaris were supported by the National Science Foundation under Grant No. CCF-0432143. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. H. Bersini. Reinforcement learning for homeostatic endogenous variables. In *From Animals to Animats 3: Proceedings of the Third International Conference on the Simulation of Adaptive Behavior*, pages 325–333, 1994.
2. B.M. Blumberg, P.M. Todd, and P. Maes. No bad dogs: Ethological lessons for learning in hamsterdam. In *From Animals to Animats 4: Proceedings of the Fourth International Conference on the Simulation of Adaptive Behavior*, pages 295–304, 1996.
3. R.A. Brooks. The role of learning in autonomous robots. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory (COLT '91)*, pages 5–10, 1991.
4. L. Cañamero. Modeling motivations and emotions as a basis for intelligent behavior. In W. Lewis Johnson, editor, *Proceedings of the First International Conference on Autonomous Agents*, pages 148–155, New York, NY, 1997. ACM Press.
5. I. Cos-Aguilera, L. Cañamero, and G.M. Hayes. Motivation-driven learning of object affordances: First experiments using a simulated khepera robot. In F. Detjer, D. Dörner, and H. Schaub, editors, *The Logic of Cognitive Systems: Proceedings of the Fifth International Conference on Cognitive Modeling (ICCM'03)*, pages 57–62, 2003.
6. K. Doya and E. Uchibe. The cyber rodent project: Exploration of adaptive mechanisms for self-preservation and self-reproduction. *Adaptive Behavior*, 13(2):149–160, 2005.
7. J.R.P Halperin. Machine motivation. In *From Animals to Animats: Proceedings of the First International Conference on the Simulation of Adaptive Behavior*, pages 213–221, 1990.
8. M. Humphrys. Action selection methods using reinforcement learning. In *From Animats to Animats 4: Proceedings of the Fourth International Conference on the Simulation of Adaptive Behavior*, pages 135–144, 1996.

9. G.D. Konidaris and G.M. Hayes. Anticipatory learning for focusing search in reinforcement learning agents. In *The Second Workshop on Anticipatory Behavior in Adaptive Learning Systems*, July 2004.
10. G.D. Konidaris and G.M. Hayes. Estimating future reward in reinforcement learning animats using associative learning. In *From Animals to Animats 8: Proceedings of the 8th International Conference on the Simulation of Adaptive Behavior*, pages 297–304, 2004.
11. G.D. Konidaris and G.M. Hayes. An architecture of behavior-based reinforcement learning. *Adaptive Behavior*, 13(1):5–32, 2005.
12. D. McFarland and T. Bosser. *Intelligent Behavior in Animals and Robots*. MIT Press, Cambridge, MA., 1994.
13. T. Savage. Artificial motives: a review of motivation in artificial creatures. *Connection Science*, 12(3/4):211–277, 2000.
14. E. Spier. *From Reactive Behaviour to Adaptive Behaviour: Motivational Models for Behaviour in Animals and Robots*. PhD thesis, Balliol College, University of Oxford, 1997.
15. E. Spier and D. McFarland. Possibly optimal decision-making under self-sufficiency and autonomy. *Journal of Theoretical Biology*, 189(3):317–331, 1997.
16. N. Sprague and D.H. Ballard. Multiple-goal reinforcement learning with modular Sarsa(0). In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 03)*, pages 1445–1447, 2003.
17. R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.
18. F. Toates. *Motivational Systems*, chapter 4. Cambridge University Press, Cambridge, UK, 1986.
19. S. Whitehead, J. Karlsson, and J. Tenenberg. Learning multiple goal behavior via task decomposition and dynamic policy merging. In J.H. Connell and S. Mahadevan, editors, *Robot Learning*, pages 45–78. Kluwer Academic Publishers, 1992.