Learning and Coordinating Repertoires of Behaviors with Common Reward: Credit Assignment and Module Activation

Constantin A. Rothkopf and Dana H. Ballard

Abstract Understanding extended natural behavior will require a theoretical understanding of the entire system as it is engaged in perception and action involving multiple concurrent goals such as foraging for different foods while avoiding different predators and looking for a mate. A promising way to do so is reinforcement learning (RL) as it considers in a very general way the problem of choosing actions in order to maximize a measure of cumulative benefits through some form of learning, and many connections between RL and animal learning have been established. Within this framework, we consider the problem faced by a single agent comprising multiple separate elemental task learners that we call modules, which jointly learn to solve tasks that arise as different combinations of concurrent individual tasks across episodes. While sometimes the goal may be to collect different types of food, at other times avoidance of several predators may be required. The individual modules have separate state representations, i.e. they obtain different inputs but have to carry out actions jointly in the common action space of the agent. Only a single measure of success is observed, which is the sum of the reward contributions from all component tasks. We provide a computational solution for learning elemental task solutions as they contribute to composite goals and a solution for how to learn to schedule these modules for different composite tasks across episodes. The algorithm learns to choose the appropriate modules for a particular task and solves the problem of calculating each module's contribution to the total reward. The latter calculation works by combining current reward estimates with an error signal resulting from the difference between the global reward and the

C.A. Rothkopf (✉)
Frankfurt Institute for Advanced Studies, Goethe University, Frankfurt am Main, Germany
e-mail: rothkopf@fias.uni-frankfurt.de

D.H. Ballard
Department of Computer Science, University of Texas at Austin, Austin, TX, USA
e-mail: dana@cs.utexas.edu

sum of reward estimates of other co-active modules. As the modules interact through their action value estimates, action selection is based on their composite contribution to individual task combinations. The algorithm learns good action value functions for component tasks and task combinations which is demonstrated on small classical problems and a more complex visuomotor navigation task.

1 Introduction

Making progress in understanding natural human visuomotor behavior requires considering the entire system as it is engaged in solving tasks involving natural perception and action sequences. A wealth of previous research has demonstrated that active perceptual strategies and perceptual states are highly dependent on the ongoing tasks (e.g., Ballard et al. 1995; Land and McLeod 2000; Yarbus 1967). It is therefore important to consider vision as it is performed within its natural setting of an embodied agent who is engaged in goal directed behavior. This requires developing models that inherently represent these behavioral goals. A particularly promising theoretical framework for addressing these questions is therefore reinforcement learning (RL) (Sutton and Barto 1998).

The promise of RL is to have an agent learn how to solve a task based on the experience they accumulate while interacting with an environment. The generality of RL leads to the hope that it may be applied to a large variety of problems in sequential perception and action. This hope has partially been nourished by successes in relating psychophysical and neuronal data obtained under laboratory settings in animals and humans to specific quantities in RL algorithms (Daw and Doya 2006; Schultz et al. 1997). Nevertheless, many open problems in relating everyday human visuomotor behavior in complex environments to RL remain, as the studied examples both at the theoretical and at the experimental level often are limited to worlds, which are simple with respect to how they evolve over time, how the observations of the agent are related to the states of the world, how the actions play out in the world, and how the feedback about the success in achieving the goal of actions is obtained.

One fundamental problem of RL algorithms is that they do not scale up well to large tasks since the state spaces, i.e. the representations of the states of the world, grow exponentially in the number of state variables per dimension: the so-called curse of dimensionality. This problem has made it difficult to apply RL to naturalistic settings, with the result that the state spaces considered are generally small in the number of dimensions and small in the number of states per dimension. Another problem is that RL has classically considered agents that learn to solve only a single task but animals and humans are faced with multiple concurrent and in part highly unrelated tasks. Further complicating the solutions is that new tasks may become relevant over time. A related issue is how to address the availability of different types of reward such as rewards related to different tasks, or related to different types of learners, or related to intrinsic versus extrinsic goals.

Many of these issues with RL have been attacked in the past by trying to use additional structure present in the respective domain so as to somehow factor the problem. This idea has been proposed by several authors early on and has reappeared in many different settings (see, e.g., Dayan and Hinton 1992; Guestrin et al. 2003; Humphrys 1996; Kaelbling 1993; Kok and Vlassis 2004; Russell and Zimdars 2003; Sallans and Hinton 2004; Schneider et al. 1999; Singh and Cohn 1998; Sprague and Ballard 2003). One specific approach, that we pursue here too, is to start with learners that represent separate non-overlapping parts of the state space. The main idea is that there will be a large number of situations that can be handled by different sets of independent modules used concurrently (e.g., Humphrys 1996; Singh and Cohn 1998; Sprague and Ballard 2003). This requires the agent to have access to a collection of independent variables describing the state of the world. In the context of human visuomotor behavior, one navigation module requires a learner to represent the position of obstacles independently from a second module that represents the position of goal positions. Such individual modules can be obtained by solving individual problems separately (Singh and Cohn 1998), or in our case, by having the modules learn their action values when activated concurrently.

The idea of a modular organization of cognitive processes has appeared in the literature in many different variations (see e.g. Barrett and Kurzban 2006; Brooks 1986; Fodor 1983; Minsky 1988; Pinker 1999). Concrete and direct evidence for a modular organization of task solutions in the human brain conform with the modular RL framework used in the present chapter comes from experiments by Gershman et al. (2009). Human subjects made simultaneous decisions with their left and right hands and received separate rewards for each hand movement. Not only was the choice behavior better described by a modular learning model that decomposed the values of bimanual movements into separate values for each component but also blood oxygen level-dependent activity in reward learning related areas of the subjects' brains reflected specific values of modular RL models. This suggests that the human brain can use modular decompositions to solve RL problems allowing for factorization.

Having a collection of individual modular solvers for separate concurrent tasks requires to specify how the rewards from different sources are handled by the agent. In multi-criterion RL a learner tries to maximize the return not only for a single reward source but also for multiple separate rewards (Gábor et al. 1998; Mannor and Shimkin 2004; Natarajan and Tadepalli 2005). Here instead we treat extrinsic and intrinsic rewards as well as rewards from different sources together by assuming that a common currency exist for the comparison of many different rewards. Such a setting has been considered previously within RL by several authors (e.g., Russell and Zimdars 2003; Singh and Cohn 1998; Sprague and Ballard 2003). Empirical evidence for a single currency in biological systems is abundant (see Kable and Glimcher 2009 for a discussion).

But observing only a single composite reward also introduces a computational problem, which is a variant of the classical credit assignment problem. Individual learners have their respective state representations and all observe the global reward, which we model here as the sum of the individual rewards. The goal is to find such sets of learners across different task combinations so that the global reward obtained is divided up correctly among the learners, given their respective state representations. This means that individual learners do not have access to the full state representation of all other learners. Again, in the context of human visuomotor behavior, the navigation learner avoiding obstacles may not represent how other learners approach goals or carry out hand movements within reaches. Thus, different active reinforcement learning modules have the problem of dividing reward up between them. A solution to this credit assignment problem for the modular case considered here has been previously given in Rothkopf (2008) and Rothkopf and Ballard (2010).

A second problem introduced by the use of individual modules stems from the fact that at different times, different sets of modules may be active within their respective tasks. One therefore needs to specify a module activation policy that attempts to pick a good set of modules for any particular task combination and an action selection process for active modules. In the past, Doya et al. (2002) proposed to use modules that consist each of a state predictor and a controller in an actor-critic-type architecture (Jacobs et al. 1991). After learning, the modules all contribute to the action of the system on the basis of how well each module is able to predict the next state of the world using each of their explicitly learnt models of the world transitions. Thus, the better an individual module is predicting the dynamics of the world the more it contributes to the composite action, irrespective of value or overall outcome. A different approach was used by Daw et al. (2005), who considered selecting a single action from different types of learners. In their system, the learner with the smallest uncertainty associated with its value prediction is selected. Thus, a single controller selects the next action alone based solely on the uncertainty of the current value estimates of a state so that each module needs to represent the same, full set of state variables. Here we propose to learn the selection of module combinations by the respective value of an action that these are predicting. Note that this is fundamentally different, as it allows arbitration of actions at the level of their expected returns, thereby avoiding conflicts that result from arbitration at the level of the actions. As learning of the task solutions by individual modules progresses, better and better estimates of the total discounted reward that each module expects will be available and therefore those modules that promise to give larger returns will be picked probabilistically more often whereas those modules that promise high returns that are not attainable will make progress in learning so. This finally leads to the scheduling of module combinations that obtain the highest combined reward and select individual actions jointly, as their value estimates are combined, and not their actions.

The modular framework that we propose naturally accommodates several desirable properties. First, it allows for continual learning (see, e.g., Ring 1994) as individual modules will learn to solve specific task components when they appear, and solutions learnt by other modules in previous tasks can be reused. This is valid under the assumption that the independent state representations are available to the learners. Secondly, by learning which modules should participate in individual task combinations the system learns to coordinate the elemental task solutions learnt by the individual modules in order to achieve more complex goals in the composite tasks. By combining all rewards from different sources into a single internal currency, the respective contributions can be weighted by their expected contribution to the current goals and action selection can be carried out on the basis of the sum of the expected reward. Finally, the system is able to adjust to new external and internal reward structures, as it learns and combines the values of individual actions from the individual modules. As rewards change, the corresponding value functions are just rescaled (Von Neumann et al. 1947) instead of learning new task solutions or a new action arbitration.

We describe a solution to the credit assignment problem and the selection of appropriate learners that avoids previous restrictions by introducing additional structure in the form of a module activation protocol and by making the additional assumption that each module has access to the estimated sum of the reward estimates of other active modules. We derive formulas for estimates of reward that, assuming properties specified in the activation protocol, converge rapidly to their true values. Furthermore, it is shown that such learning can be implemented both in the setting where individual modules have independent action spaces as in Chang et al. (2004) and also in the single-agent case where individual modules' policies are combined to specify an action (Russell and Zimdars 2003; Sprague and Ballard 2003). We demonstrate that the algorithm can solve the credit assignment problem in variants of classical problems in the literature (Singh and Cohn 1998; Sutton and Barto 1998) as well as a more complex case of an avatar learning to navigate in a naturalistic virtual environment (Sprague et al. 2007).

2 Background

The problem setting is that of a Markov Decision Processes. An individual MDP consists of a 4-tuple (S, A, T, R) with S being the set of possible states, A the set of possible actions, T the transition model describing the probabilities $P(s_{t+1}|s_t, a_t)$ of reaching a state s_{t+1} when being in state s_t at time t and executing action a_t, and R is a reward model that describes the expected value of the reward r_t, which is distributed according to $P(r_t|s_t, a_t)$ and is associated with the transition from state s_t to some state s_{t+1} when executing action a_t.

Reinforcement learning attempts to find a policy π that maps from the set of states S to actions A so as to maximize the expected total discounted future reward through some form of learning. The dynamics of the environment T and the reward function R may not be known in advance. In the present case an explicit reward function R is learned from experience so that individual modules can learn from experience about their contributions to the global reward. The central goal in model free RL is to assign a value $V^{\pi}(s)$ to each state (s), which represents this expected total discounted reward obtainable when starting from the particular state s and following the policy π thereafter:

$$V^{\pi}(s) = E^{\pi}\Big(\sum_{t=0}^{\infty} \gamma^t r_t\Big) \tag{1}$$

Alternatively, the values can be parametrized by state and action pairs, leading to the Q-values $Q^{\pi}(s,a)$. Where Q^* denotes the Q-value associated with the optimal policy π^*, the optimal achievable reward from a state s can be expressed as $V^*(s) = \max_a Q^*(s,a)$ and the Bellman optimality equations for the quality values can be formulated as:

$$Q^*(s,a) = \sum_r rP(r|s,a) + \gamma \sum_{s' \in S} P(s'|s,a) \max_{a'} Q^*(s',a') \tag{2}$$

Temporal difference learning (Sutton and Barto 1998) uses the error between the current estimated values of states and the observed reward to drive learning. Evidence for temporal difference learning in animals comes from a multitude of studies (e.g., Schultz et al. 1997). The values of state-action pairs can be updated by this error δ_Q using a learning rate α:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\delta_Q \tag{3}$$

Evidence for the representation of action values has also been found (e.g., Samejima et al. 2005). Two classical learning rules for the Q-values $Q(s,a)$ are (1) the original Q-learning rule (Watkins 1989) and (2) SARSA (Rummery and Niranjan 1994). While the Q-learning rule is an off-policy rule, i.e. it uses errors between current observations and estimates of the values for following an optimal policy, while actually following a potentially suboptimal policy during learning, SARSA is an on-policy learning rule, i.e. the updates of the state and action values reflect the current policy derived from these value estimates. The temporal difference for SARSA is accordingly:

$$\delta_Q = r_t + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t). \tag{4}$$

Empirical evidence for learning in animals consistent with the SARSA learning rule has similarly been found (Morris et al. 2006).

3 Multiple Modules as Individual Task Solutions

Within the composite state space both the relationship between the optimal value functions for each of the individual component tasks and the global task in which multiple objectives are pursued depend on the overall structure of the problem and can be very complex when considering the most general case in which arbitrary dependencies between state transitions and between rewards exist. Our main simplifying assumption is to define a restricted venue where the required behavior can be realized with separate RL modules. The primary assumption is that such modules are available by virtue of having many modules with independent state variables so that they do not interfere with each other when activated in subsets (Chang et al. 2004; Russell and Zimdars 2003; Sprague and Ballard 2003). Of course in many interesting problems it may turn out that modules interfere with each other, but our assumption is that for the tasks at hand, independent state representations are available for the considered modules. This assumption shifts the problem of interacting modules to having independent state representations.

We will consider two separate but related cases. In the first setting, a collection of individual modules has separate and independent state spaces as well as separate and independent action spaces. In this case, the individual modules do not interact through state transitions or rewards. As an example, Chang et al. (2004) considered the case of ten individual learners working on ten separate and not interacting tasks in ten separate mazes but observing a single cumulative reward signal. The second case also uses the assumption of separate states with independent state transitions but all modules share the same action space. As an example, Singh and Cohn (1998) considered the case of a single agent moving in a grid world while foraging for different food types and avoiding a predator. While individual modules represented their states with respect to the different food types and the predator independently, the action of moving within the maze had to be coordinated within a single action space of moving in the grid world.

Let's first consider the case of separate action spaces. A module with its own actions can be defined as an MDP, i.e. the i-th module is given by

$$M^{(i)} = \{S^{(i)}, A^{(i)}, T^{(i)}, R^{(i)}\} \tag{5}$$

where the superscripts reflect that the information is referred to the i^{th} MDP. The states of the different modules are assumed to be non-overlapping. In such a case, the optimal value function is readily expressible in terms of the component value functions and the states and actions are fully factored so that there is no overlap and additionally the following two conditions hold:

$$P(s_{t+1}^{(1)}, \dots, s_{t+1}^{(N)} | s_t^{(1)}, \dots, s_t^{(N)}, a_t^{(1)}, \dots, a_t^{(N)}) = \prod_{i=1}^{N} P(s_{t+1}^{(i)} | s_t^{(i)}, a_t^{(i)}) \tag{6}$$

$$P(r_t^{(1)}, \dots, r_t^{(N)} | s_t^{(1)}, \dots, s_t^{(N)}, a_t^{(1)}, \dots, a_t^{(N)}) = \prod_{i=1}^{N} P(r_t^{(i)} | s_t^{(i)}, a_t^{(i)}) \tag{7}$$

Then these two conditions can be used together with Eq. (2) in order to arrive at the result:

$$Q(s_t^{(1)}, \dots, s_t^{(N)}, a_t^{(1)}, \dots, a_t^{(N)}) = \sum_{i=1}^{N} Q^{(i)}(s_t^{(i)}, a_t^{(i)}) \tag{8}$$

If Eqs. (6) and (7) hold and all the rewards are known, the action maximizing Eq. (8) can be selected and is guaranteed to be optimal. In this decomposed formulation, each module can follow its own policy $\pi^{(i)}$, mapping from the local states $s^{(i)}$ to the local actions $a^{(i)}$. This case is appropriate for the case of all modules having separate action spaces when each module can be identified with an agent that may be expected to act independently.

The second case, which is our focus, is that of a single agent pursuing multiple goals that are divided up between multiple independent modules that the agent can activate internally (Humphrys 1996; Karlsson 1997; Russell and Zimdars 2003; Singh and Cohn 1998; Sprague and Ballard 2003). The main problem specification that this constraint introduces is that the action space is shared such that $a^{(i)} = a$, for all i, so the i-th module is now:

$$M^{(i)} = \{S^{(i)}, A, T^{(i)}, R^{(i)}\} \tag{9}$$

An even simpler case arises, when all individual states $s^{(i)}$ correspond to the full state s, as, e.g., in Russell and Zimdars (2003). Although this obviously does not give any advantage in terms of reducing the dimensionality of the state space for individual modules, this case is interesting, because it is straightforward to prove the optimality of individual modules' values when using an on-policy learning algorithm such as SARSA.

How are actions selected when all modules share a single common action space? This case requires some form of action selection in order to mediate the competition between actions proposed by individual modules. While other modular RL solutions commonly employ action selection at the level of the actions selected by the modules (e.g., Daw et al. 2005; Doya et al. 2002) we have the modules interact at the level of their values. This has the distinct advantage that conflicts of modules proposing different actions can be resolved at the level of the expected total future rewards. We use the probabilistic softmax action selection with temperature τ:

$$P(a_t | s_t^{(1)}, \ldots, s_t^{(N)}) = \frac{\exp\left(\sum_{i=1}^{N} Q^{(i)}(s_t^{(i)}, a_t)/\tau\right)}{\sum_b \exp\left(\sum_{i=1}^{N} Q^{(i)}(s_t^{(i)}, b)/\tau\right)} \tag{10}$$

Once the action has been selected it is used for all modules. Note how the above equation combines the Q-values from all N active modules to probabilistically select an action that jointly promises high reward.

Finally we introduce the idea of a common single global reward that is obtained as result of the actions relative to the individual component state spaces of the contributing tasks. As an example, Sprague et al. (2007) considered the task of navigating along a walkway while avoiding obstacles and approaching targets. This task can be modeled as a composition of three individual component tasks, requiring one module for navigation, a second module for obstacle avoidance, and a third module for target approach. In this case, the global reward is the sum of the individual rewards obtained for each of the three component tasks.

Accordingly, the agent now needs to compute the credit for each module. That is, initially the individual rewards due to each module are not known, but only the global reward is observed by the agents at each time step. Here we assume for simplicity that the total, global reward G_t at some time t is the linear sum of all component MDPs' rewards:

$$G_t = \sum_{i \in \mathcal{M}} r_t^{(i)}. \tag{11}$$

Thus we can write:

$$M^{(i)} = \{S^{(i)}, A, T^{(i)}, G_{\mathcal{M}}\} \tag{12}$$

where the subscript $\mathcal{M}$ denotes that at each time step G is a function of the modules that are active.

4 Learning Module Activation

Our central assumption is that an overall complex problem can be factored into a small set of MDPs. Given a large set of modules with separate state representations it is now necessary to learn which modules to select in order to solve a specific task. In the following we assume that tasks are encapsulated within *episodes* of fixed length parameter Δ (See Fig. 1). During each episode, only a subset of the total module set is active. The guiding hypothesis is that in the timecourse of behavior, a certain set of goals is pursued and therefore the corresponding set of modules that are needed to achieve these goals become active and those that correspond to tasks that are not pursued become inactive (Sprague et al. 2007). During an episode the composition of a particular module set is assumed to not change. Given this constraint, the pivotal idea is that, within each episode, each active module can refine its own reward estimates by having access to the sum of the reward estimates of the other active modules. Thus, the set of active modules may change between episodes so that over time the actions taken direct the agent to different parts of the composite state space. The simplifying assumption of episodes allows reducing the problem of selecting modules to the beginning of an episode and also allows for a formal proof that the rewards of individual modules can be estimated correctly with the proposed credit assignment algorithm. As example, consider the previously mentioned problem of collecting food and avoiding a predator. Within an episode, only two food source may be present, requiring only the two associated foraging modules to be active. But within a later episode, the predator and the remaining food source may also be present, requiring all three food modules and the predator module to be active.

To model the factorization, let us suppose that in a substantial repertoire of N' modules, at the beginning of an episode it can be determined for one reason or

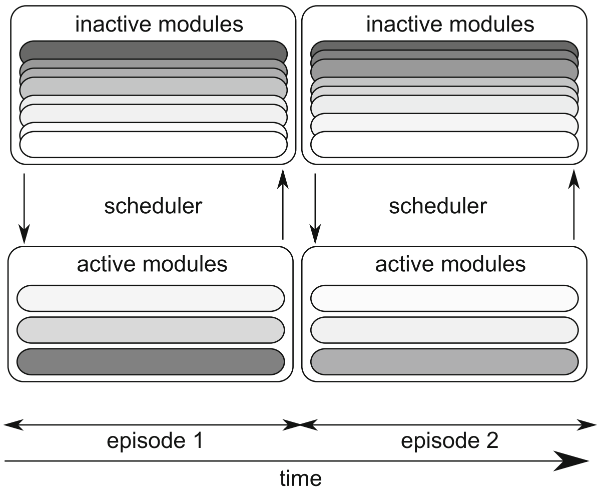

Fig. 1 The credit assignment algorithm exploits the constraint that in any period during behavior there is only a subset of the total module set that is active. We term these periods episodes. In the timecourse of behavior, modules that are needed become active and those that are no longer needed become inactive. The diagram depicts two sequential episodes of three modules each. The different modules are denoted with different shading. The *vertical arrows* denote the scheduler's action in activating and deactivating modules. On the top is the large library of possible modules. Our formal results only depend on each module being chosen sufficiently often and not on the details of the selection strategy. The same module may be selected in sequential episodes

another that only a smaller set N are sufficient for solving the task at hand where typically $N < N'$. So what is the best way to choose N modules from a possible N'? We select modules probabilistically according to the value of the initial state at the beginning of each episode using the softmax function:

$$P(M^{(i)}) = \frac{\mathrm{e}^{\frac{V^{(i)}(s^{(i)})}{\upsilon}}}{Z} \tag{13}$$

where $Z = \sum_{j=1}^{N'} \mathrm{e}^{\frac{V^{(j)}(s^{(j)})}{\upsilon}}$ is a normalizing term over the applicable modules, $V^{(i)}(s^{(i)})$, the value of the sate $s^{(i)}$ for module i is given by $V^{(i)}(s^{(i)}) = \arg\max_a\{Q^{(i)}(s^{(i)}, a)\}$, and υ is the common temperature parameter of the Boltzmann distribution that governs the stochasticity of module selection given the difference in values $V^{(i)}(s^{(i)})$. The idea is that although initially the value estimates of different modules will be unreliable, as the modules learn their respective rewards, the estimates will improve and direct the effort towards modules that can achieve the promised rewards. Note that this sampling strategy does not affect whether or not individual modules' value functions will converge. That is, a given learner of a component MDP $M^{(i)}$ working alone will converge towards its optimal value function as long as the space of states and actions $S^{(i)} \times A^{(i)}$ is sampled infinitely often in the limit. The same criterion applies when it is working with other modules, although whether or not a module follows its optimal policy is a separate question. Independent agents may follow their own policy recommendations, but for modules activated with a single agent, a separate function that must be supplied to adjudicate among the possibly different action recommendations, as in Sprague et al. (2007), where the action selected was a maximum of reward weighted Q values.

In summary,

1. The overall behavior of modules (or agents in the multi-agent venue) is such that they work together in different subsets for a set time Δ so that the reward estimates can be learned;
2. The sum of the current estimates of the reward across all subset is accessible to each individual module in the subset at each moment by assumption;
3. The sampled subsets must span the module space because the reward calculations demand this.

The consequences of a module being activated are that:

1. It has used an associated procedure, such as a visual routine (Ballard et al. 1997; Ullman 1984), to compute the initial state the module is in. In our examples the state is computed at the beginning of an episode for all modules.
2. Its Q-values are included in the sum indicated in Eq. (10) used to select an action.
3. It influences the global reward that is received at every time step.

5 Credit Assignment with Modular Behaviors

Each active module represents some portion of the entire state space and contributes to the composite actions, but without some additional constraint they only have access to a global performance measure, defined according to Eq. (11) as the sum of the individual rewards collected by all of the $\mathcal{M}$ active modules at each time step to a global reward, i.e. $G_t = \sum_{i \in \mathcal{M}} r_t^{(i)}$. The problem that we tackle is how to learn the composite quality values $Q^{(i)}(s^{(i)}, a^{(i)})$ when only global rewards G_t are directly observed, but not the individual values $\{r_t^{(i)}\}$ (See Fig. 2), across many task combinations. The additional constraint that we introduced is that the system can use the sum of reward estimates from the modules that are co-active at any instant. This knowledge leads to the idea to use the different sets to estimate the difference between the total observed reward G_t and the sum of the current estimates of the individual rewards of the concurrently running behaviors. Credit assignment is achieved by bootstrapping these estimates over multiple task combinations, during which different subsets of behaviors are active. Dropping the temporal subscript for convenience, this reasoning can be formalized as requiring the individual behaviors to learn independent reward models $r^{(i)}(s^{(i)}, a^{(i)})$. The current reward estimate $\hat{r}^{(i)}$ for one particular behavior i, is obtained as

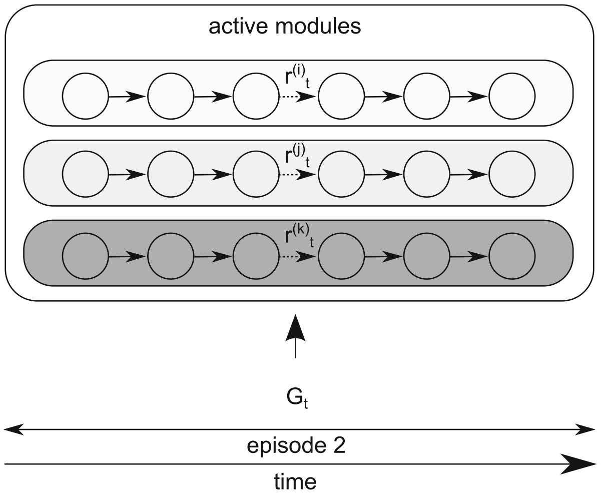

Fig. 2 The fundamental credit assignment problem for a biological agent using a modular architecture. When individual component learners transition between states represented by circles they carry out actions represented by arrows. At a particular instant shown with *dotted arrows*, when multiple modules are active and only a global reward signal G is available, the modules each has to be able to calculate how much of the rewards is due to their activation. Our setting simplifies the problem by assuming that individual reinforcement learning modules are independent and communicate only their estimates of their reward values. The modules can be activated and deactivated asynchronously across episodes

$$\hat{r}^{(i)} \leftarrow \hat{r}^{(i)} + \beta \delta_{r^{(i)}} \tag{14}$$

where the error on the reward estimates δ_r is calculated as the difference between the global reward and the sum of the component estimates:

$$\delta_{r^{(i)}} = G - \sum_{j \in \mathcal{M}} \hat{r}^{(j)} \tag{15}$$

so that Eq. (14) becomes:

$$\begin{aligned} \hat{r}^{(i)} &\leftarrow \hat{r}^{(i)} + \beta \Big(G - \sum_{j \in \mathcal{M}} \hat{r}^{(j)} \Big) \\ &= (1-\beta)\hat{r}^{(i)} + \beta \Big(G - \sum_{j \in \mathcal{M}, j \neq i} \hat{r}^{(j)} \Big) \end{aligned} \tag{16}$$

Together with the module activation protocol, described by Eq. (13) and Δ, Eq. (16) represents the core of our solution to the credit assignment problem. When one particular subset of tasks is pursued, each active behavior adjusts the current reward estimates $\hat{r}_i$ in the individual reward functions according to Eq. (16) at each time step. Over time, the set of tasks that have to be solved will change, resulting in a different set of behaviors being active, so that a new adjustment is applied to the reward functions according to Eq. (16). This bootstrapping process therefore relies on the assertion that subsets of active behaviors visit all component behaviors. Intuitively, the global reward, which is the sum of all obtained rewards, should equal the sum of the reward estimates of all active modules. Each individual module therefore adjusts its reward estimate by combining its current reward estimate, i.e. what it believes it is contributing, and what the other active modules believe it is contributing, i.e. the difference between the total global reward and the sum of all other modules' reward estimates.

The component Q values for the state-action pairs of the individual behaviors are learned using the above estimates of the individual reward functions. Given the current reward estimates obtained through repeated application of Eq. (16), the SARSA algorithm is used to learn the component Q-functions:

$$Q_i(s_t^{(i)}, a_t^{(i)}) \leftarrow Q_i(s_t^{(i)}, a_t^{(i)}) + \alpha\delta_{Qi} \tag{17}$$

where δ_{Qi} now contains the estimates $\hat{r}_t^{(i)}$ and is given by:

$$\delta_{Q_i} = \hat{r}_t^{(i)} + \gamma Q_i(s_{t+1}^{(i)}, a_{t+1}^{(i)}) - Q_i(s_t^{(i)}, a_t^{(i)}) \tag{18}$$

Note that the usage of an on-policy learning rule such as SARSA is essential as noted in Sprague and Ballard (2003), because the arbitration process specified by Eq. (10) may select actions that are suboptimal for one or more of the modules. This has to be kept in mind when considering the optimality of the learnt solutions (cf. Russell and Zimdars 2003). A feature of the SARSA algorithm is that it makes use of suboptimal policy decisions during learning.

To investigate the dynamics of Eq. (16) we ask: in such a setting, given only global reward together with an arbitrary initial estimate of the rewards, will our algorithm for computing rewards converge?

5.1 *Proof of Convergence*

First of all, note that the equations for $\hat{r}$ are independent of the Q values and only depend on the sampling strategy. Next, note that, once the rewards have been determined, the MDPs will converge to their correct policies (Russell and Zimdars 2003) starting from any initial condition. While the rewards are being determined, the MDPs may or may not be converging but that is of no consequence to the proof. So it suffices to show that the reward calculation converges. To do this note that Eq. (16) can be rewritten as:

$$\hat{r}^{(i)} = (1-\beta)\hat{r}^{(i)} - \beta\sum_{j\neq i}\hat{r}^{(j)} + \beta G. \tag{19}$$

This in turn can be written in vector form for a Jacobi iteration k by rewriting the reward estimates $\hat{r}^{(i)}$ at iteration k for all states and actions as a column vector $\mathbf{r}_k$, defining the column vector $\mathbf{G}$ with all entries being equal to the global reward G, and defining the error matrix B according to Eq. (19):

$$\mathbf{r}_{k+1} = B\mathbf{r}_k + \beta\mathbf{G}$$

Now write $\mathbf{r}_{k+1}$ as $\mathbf{r}_o + \mathbf{e}_{k+1}$ where $\mathbf{r}_o$ represents the correct values. Then

$$\mathbf{r}_o + \mathbf{e}_{k+1} = B(\mathbf{r}_o + \mathbf{e}_k) + \beta\mathbf{G}$$

but since $\mathbf{r}_o = B\mathbf{r}_o + \beta\mathbf{G}$ then

$$\mathbf{e}_{k+1} = B\mathbf{e}_k$$

This system will converge to 0 if the spectral radius of the matrix B forming any spanning set of these equations has a value $\rho < 1$. This is satisfied because its diagonal terms are all $1 - \beta$ and the off diagonal terms are either zero or $-\beta$. Thus the determinant is dominated by the trace for small β as the other factors all contain products of at least β^2. Convergence of such a system is therefore geometric provided (a) all the potential variables are included in the equations (b) any two variables do not always appear together, and (c) the diagonal terms in the error matrix B dominate the rest. The first two conditions are satisfied by the assumption in the module activation policy. The last requirement can be assured provided β is chosen appropriately.

5.2 *Incorporating Uncertainty in Reward Model*

One way to handle uncertainty in reward estimates is to model the sum of the other modules' rewards as noise as in Chang et al. (2004). This approach learns correct policies as long as the set of agents (or modules in the single-agent setting) remains constant, but it can introduce severe biases into the Q values. This makes it unusable for our setting where modules are used in different subsets in different episodes must have correct Q values for the equations to work.

Our module activation strategy of using modules in different combinations overcomes the Q value bias problem. When one particular subset of goals is pursued in any particular episode, the corresponding behaviors are active and the estimates of the respective rewards is updated according to Eq. (16) for all component behaviors. The sampling strategy assures that the equation set for the rewards has full rank.

However one can do better. During the computation, the modules' MDPs are typically in different states of completion and consequently have different levels of uncertainty in their reward estimates. This means that on a particular task combination, all component behaviors weight reward estimates in the same way, independent of how well component behaviors have already estimated their share. Thus a drawback of any updating scheme that uses a fixed β value is that it is possible for a behavior to unlearn good reward estimates if it is combined with other behaviors whose reward estimates are far from their true values. Learning can be made much more efficient by considering the respective uncertainties in the estimates of the respective rewards separately. Thus one can have individual β_i values for each module reflect their corresponding reward estimates of uncertainty values.

Assuming that the between-module fluctuations are uncorrelated and follow a Gaussian distribution one can express the gain for each reward estimate in terms of the individual uncertainties in the respective reward estimates $(\sigma^{(i)})^2$:

$$\begin{aligned}\beta_i &= \frac{(\sigma^{(i)})^2}{\sum_{j=1}^{N} (\sigma^{(j)})^2} \\ &= \frac{(\sigma^{(i)})^2}{\sum_{j\neq i}^{N} (\sigma^{(j)})^2 + (\sigma^{(i)})^2} \end{aligned} \tag{20}$$

where the last term in the denominator is the variance in the observation noise. This can be seen as a straightforward approximation of the respective measurement uncertainties as in a cue integration setting. Since the factor for weighting the current estimate of a module's reward (See Eq. (16)) is $1 - \beta_i$, the effect is that relatively high-variance reward estimates will be discounted with respect to those of co-active modules. Thus, reward estimates for states that have been visited often and modules that have been used will tend to have lower uncertainties than reward estimates of states that have been visited rarely or modules that have not yet been used often.

6 Simulation Results

We demonstrate the algorithm on three separate problems that are chosen to illustrate different aspects of the proposed solution. The first problem uses ten individual modules that live in separate, non-interacting state spaces and carry out independent, non-interfering actions but observe a single common reward, which is the sum of all individual learners' contributions. The example is from Chang et al. (2004) and utilizes component tasks originally introduced by Sutton and Barto (1998). The second example is a predator–prey type problem and considers the case in which all component modules are within the same agent, i.e. where there is only a single action space which is shared among all individual learners. This problem was considered by Singh and Cohn (1998) and has been expanded here from its original version to include 15 different food sources and 5 predators to illustrate how to learn module selection over episodes. The third problem is a multi-tasking problem of an agent walking on a sidewalk while avoiding obstacles and picking up litter (Sprague and Ballard 2003). This problem illustrates the calculation of reward

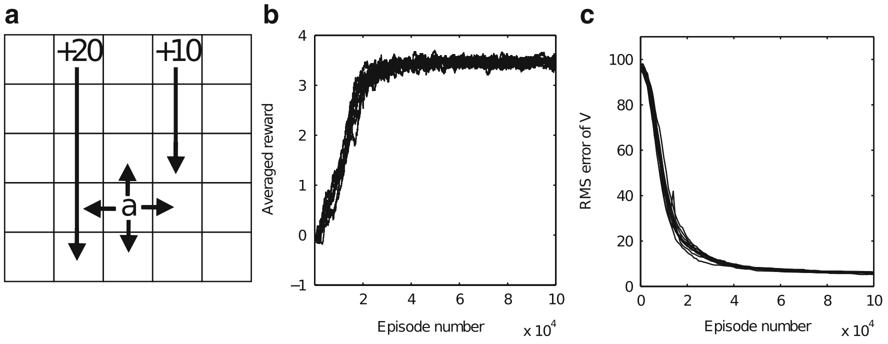

Fig. 3 Learning progress for the separate action spaces case following Chang et al. (2004). (**a**) Each agent is able to move in the four cardinal directions. Each transition results in a reward of 0 with the exception of the two states from where the *arrows* start. (**b**) The plot shows that each of the ten modules obtains the same average reward per iteration after learning of all tasks. (**c**) The plot shows that the RMS error between the optimal value functions and learned value functions decreases with learning

in a simulated three-dimensional world. For all these simulations, the RL learning parameter α was 0.1. The first set of experiments uses both constant β values from the set $\{0.01, 0.1, 0.2, 0.4, 0.8\}$ and the variance weighted β computed according to Eq. (20), i.e. it weights the reward estimates by their respective uncertainties. The remaining experiments use constant β values between 0.01 and 0.5 as indicated on the respective figures.

6.1 The Computation of Accurate Q-Values with Separate Actions

The first problem, drawn from Sutton and Barto (1998), allows testing the basic credit computation algorithm by considering multiple modules with independent action spaces, a case that is closely related to multi-agent problems. This results in a setting without the complicating factor of action selection that is required when modeling multiple active modules forming a single agent. With multiple modules, each learner is placed on a 5×5 grid and is able to move in four directions, i.e. North, East, South, and West (See Fig. 3).

A transition between positions results in a reward of 0. If a movement is made toward the walls of the grid, the agent obtains a reward of -1 and stays at the same location. There are two locations on the grid, which result in a transition to a new state for all selected actions. On one of these a reward of 20 units is obtained while in the second case a reward of 10 units is obtained. The problem is set up in such a way that the optimal policy successively collects the reward of 20 units, given a discount factor of 0.9. For the simulation, after each Δ of 30 iterations, a new subset of agents

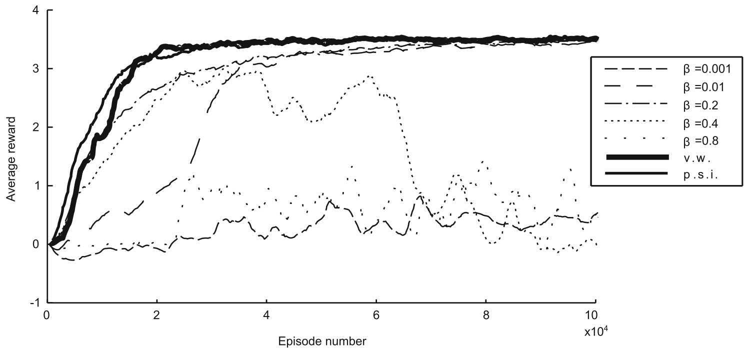

Fig. 4 Comparison of learning rate settings. Effect of different learning rates and the variance-weighted (v.w.) learning on the accumulated reward averaged over all modules in the Sutton and Barto problem in comparison with a Kalman Filter that has perfect state information (p.s.i.) about all component modules

of sizes between 3 and 7 were chosen randomly. The scheduler chose these subsets according to a uniform distribution over all modules. These modules were then run on the grid world and learned their respective Q-values with reward estimates that were updated according to Eq. (16). Part (c) of Fig. 3 shows that the agents learn to divide up the global reward and are therefore able to learn their respective policies. These results show that the basic algorithm for computing Q values works in this multi-agent setting.

While one can choose the value for β guided by the convergence theorem, a more informed possibility is to use Eq. (20) to weight the module's reward estimates by their inverse variances. To test this formula, we ran experiments to compare the weighting of reward estimates by their variance estimates with weightings using hand chosen β values. Figure 4 shows different learning rates compared to the variance-weighted result as well as a Kalman filter simulation with perfect information about the entire state space. The graph clearly shows the superiority of the variance-weighted estimation process compared to the fixed β values as well as the small loss incurred compared to the filtering agent in the considered problem.

6.2 *Module Selection for Single Agent with Multiple Reward Sources*

This problem is described in Singh and Cohn (1998) where the authors explore the use of multiple modules for a single task in a grid-world problem. These single-agent problems come closer to representing a problem that would have to be addressed by a biological agent since, unlike in the multi-agent problem of the previous section, the action space is shared by the modules.

In the original formulation, an agent moves on a 5×5 grid. Possible actions move the agent in the eight compass directions. Moves at the edge of the grid-world which would result in the agent leaving the grid result in the agent staying in the current position. The grid is populated by three food items and one predator. The picking up of a food item results in a reward of one unit and the repositioning of the food item to a new and empty location. The world is also populated by a predator, which moves every other time unit towards the current position of the agent. The agent obtains a reward of 0.5 units for every time step during which it does not collide with the predator. Each learner represents the position of the respective food item or predator, i.e. there are 625 states for each of the three food modules and for the predator module, where in the original problem a total of four learners were always active in order to solve a single task with four component goals.

Previously Singh and Cohn (1998) and Sprague and Ballard (2003) used this task in multi-goal learning but both studies used individual rewards that were delivered for each task as separate reward signals. In the current study the problem was made harder by making the reward that each behavior sees be the global sum of the individual rewards G. Furthermore, instead of using three food sources and one predator there are a total of 15 types of food sources and 5 types of predators. At the beginning of each episode, three food sources are selected randomly according to a uniform distribution over the total of 15 different food sources. Similarly, one predator is selected randomly from the pool of 5 different predators according to a uniform distribution, so that during every episode a total of three food sources and one predator are present, as in the original problem. But now the set of necessary modules may change for each episode, as each food source and each predator requires the corresponding module to be active. So now the agent has to learn which modules to select for an episode and each module has to learn its respective reward model by solving the credit assignment problem. The probabilistic module activation according to Eq. (13) was used to select four different modules at the beginning of each episode. The behaviors therefore also have to learn when they are best activated.

Simulations were run for different values of β and compared to a learner that instead of choosing the set of modules according to Eq. (13) selected learners at random with equal probability at the beginning of each episode. The learning rate for the reward model of this learner was set to an intermediate value of $\beta = 0.2$. The temperature T in Eq. (13) was changed from 3 to 0.01 over the course of learning. The rewards for all foods and predators were set to the values of the original problem by Singh and Cohn (1998). Figure 5 shows the average reward earned at each time step and the root mean square error between the true and learnt value functions. For intermediate learning rates of β between 0.01 and 0.2, the obtained reward approaches the maximum obtainable reward and the corresponding value functions approach the correct ones.

The learners are able to learn when they are best scheduled to run and the reward estimates become more accurate over trials according to Fig. 6, which demonstrates that for intermediate learning rates for β between 0.01 and 0.2, the reward estimates approach the true reward values as shown by the RMS error between all reward

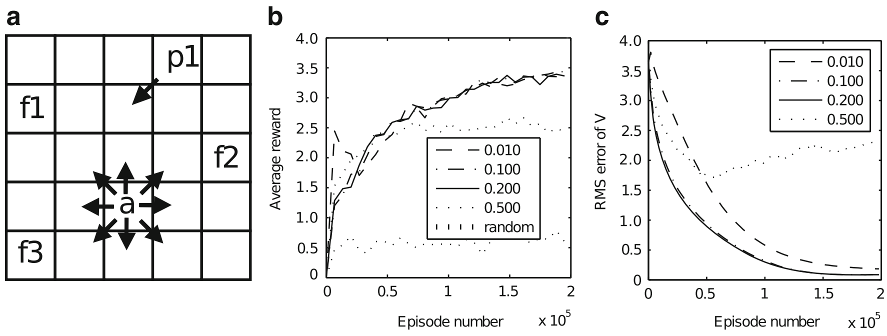

Fig. 5 Learning progress for the common action space case following Singh and Cohn (1998). (**a**) An agent is located on a 5 × 5 grid and searches to find three different food sources f1 to f3 and tries to avoid the predator p, which moves every other time step towards the agent. Simulations were run in which three food sources from a set of 15 and one predator from a set of 5 were selected. (**b**) Average reward collected using consumable rewards with different learning rates β in Eq. 16 and a randomly scheduled agent. (**c**) Root mean squared error between the true value functions and the learned value functions of all behaviors over trials

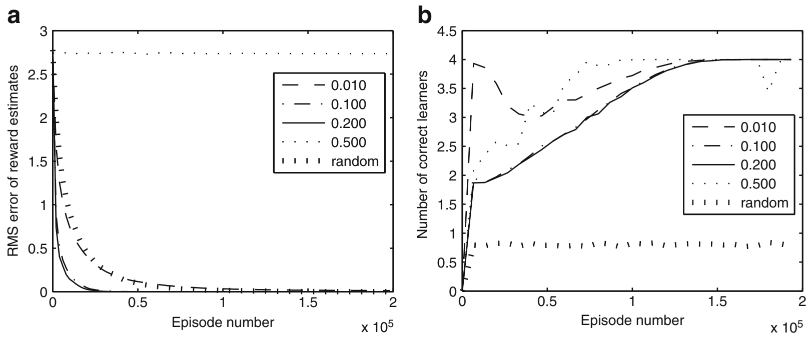

Fig. 6 Learning progress for the foraging task. (**a**) RMS error between the true rewards and the reward estimates of all behaviors over trials. The three curves correspond to different learning rates β in Eq. 16. (**b**) Number of correctly chosen learners on each episode for different learning rates β. As they are chosen in groups of four, choosing the right four modules is the best possible result

estimates and the true rewards. By contrast, a learner with a learning rate of $\beta = 0.5$ does not converge on the correct reward model over the course of the simulations. Accordingly, this learner does neither approach the correct value function nor approach the average reward collected by the other learners as shown in Fig. 5. The random learner's reward estimates improve over time as shown in Fig. 6, because the reward model can be learned using samples from the randomly selected states. But the average reward obtained per iteration cannot approach the level of the other learners, as inappropriate modules continue to be scheduled. Finally, Fig. 7 is a graphical depiction of the active set of modules over the total number of episodes. Black dots represent modules selected by the agent which are not appropriate for

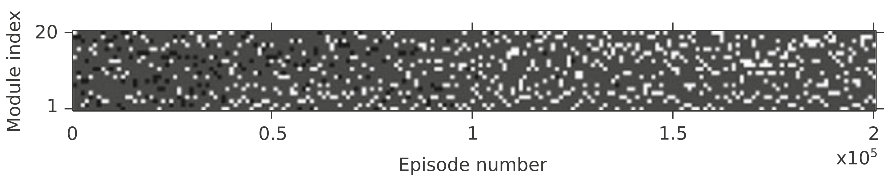

Fig. 7 Learning progress of module selection. Every 10,000 iterations the four modules selected by the evaluation function are plotted using a key of *Black* = inappropriate, *White* = appropriate, and *Gray* = not activated. As the computations progress, the reward values $V(s)$ become increasingly accurate and cause the correct modules to be always selected

the respective episode's task combination, while white dots represent correctly selected modules and gray dots are inactive modules. The plot shows that the learner improves selection of modules right from the first episode and has learnt to schedule the correct subset of modules for the task at hand after 15,000 episodes.

6.3 Learning Walkway Navigation in a Virtual 3D Environment

This problem uses a humanoid agent navigating in a realistic three-dimensional virtual reality environment. The agent must use vision to extract features from the environment that define each module's state space. Also the agent's discrete state spaces must guide it successfully through the fine-grained environment. The walkway navigation task was first considered by Sprague et al. (2007) where a factorized solution was presented. However, that solution was obtained by delivering each of the individual learners their respective reward; that is, the agent received three separate rewards, one for the walkway following module, one for the obstacle avoidance module, and one for the litter picking up module. This problem was modified here with the additional constraint of only global reward being observed by all modules in each task combination. The global reward was always the sum of the rewards obtained by the individual modules according to Eq. (11).

The parameterization of the statespace is shown in Fig. 8. Each module represents the states with a two-dimensional vector consisting of a distance and an angle. For the picking up and the avoidance behaviors, these are the distance to the closest litter object and obstacle, respectively, and the signed angle between the current heading direction and the direction towards the object. The distance is scaled logarithmically similarly to the original setup by Sprague et al. (2007) and the resulting distance d_i is then discretized into $n = 21$ possible values between 0 and infinite distance. The angles within the field of view, i.e. with a magnitude smaller than 50 degrees are similarly discretized to 21 values. The exponential function was chosen such that the edge of an obstacle or target coincides with the edge between the first and second state and state 21 corresponds to all distances greater or equal to $4m$.

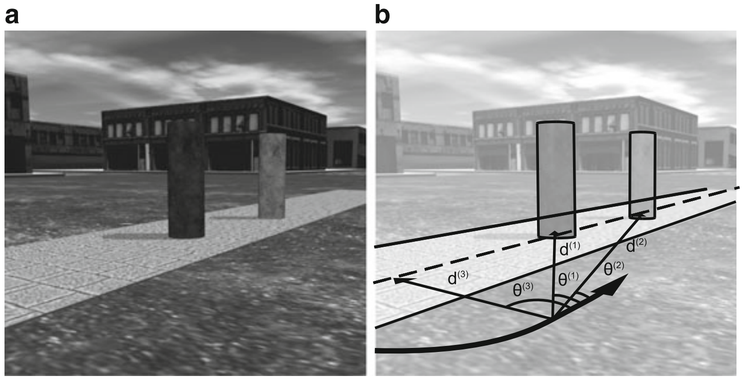

Fig. 8 The walkway navigation tasks. (**a**): typical view from the agent while navigating in the virtual environment. The three possible tasks are following the walkway, avoiding obstacles, which are the *dark cylinders*, and picking up litter, corresponding to the *light cylinders*. (**b**): Schematic representation of the statespace parameterization for the learners. See text for details

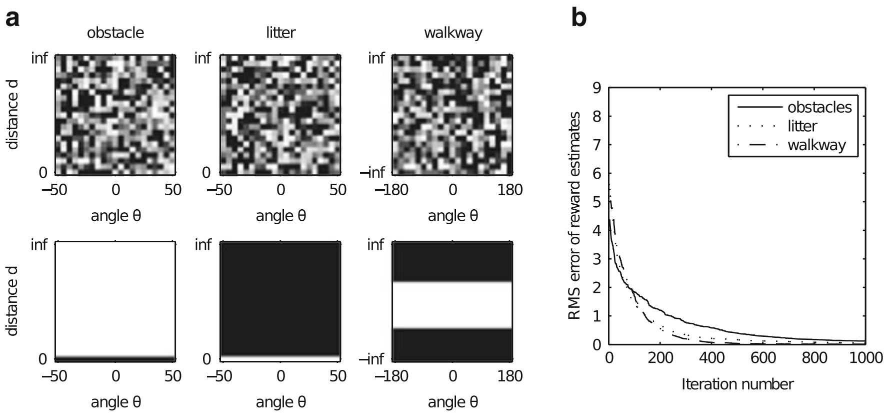

Fig. 9 Modular Reward estimates across learning. (**a**) Reward calculations for the walkway navigation task for the three component behaviors. *Top row*: Initial values. *Bottom row*: Final reward estimates. (**b**) Time course of learning reward for each of the three component behaviors. RMS error between true and calculated reward as a function of iteration number

The walkway statespace is also slightly different from Sprague et al. (2007) in that it represents all positions of the agent relative to the walkway for all possible walking directions. Finally, instead of three possible actions as in Sprague et al. (2007) the current simulations use five actions corresponding to steering at one of the five angles $\{-15, -7.5, 0, 7.5, 15\}$ with additive Gaussian noise of variance $\sigma = 1$. The reward values displayed as a function of the state space locations are shown in Fig. 9. Staring from random values and receiving only global reward at each step, the agent's modules are able to arrive at the true reward values.

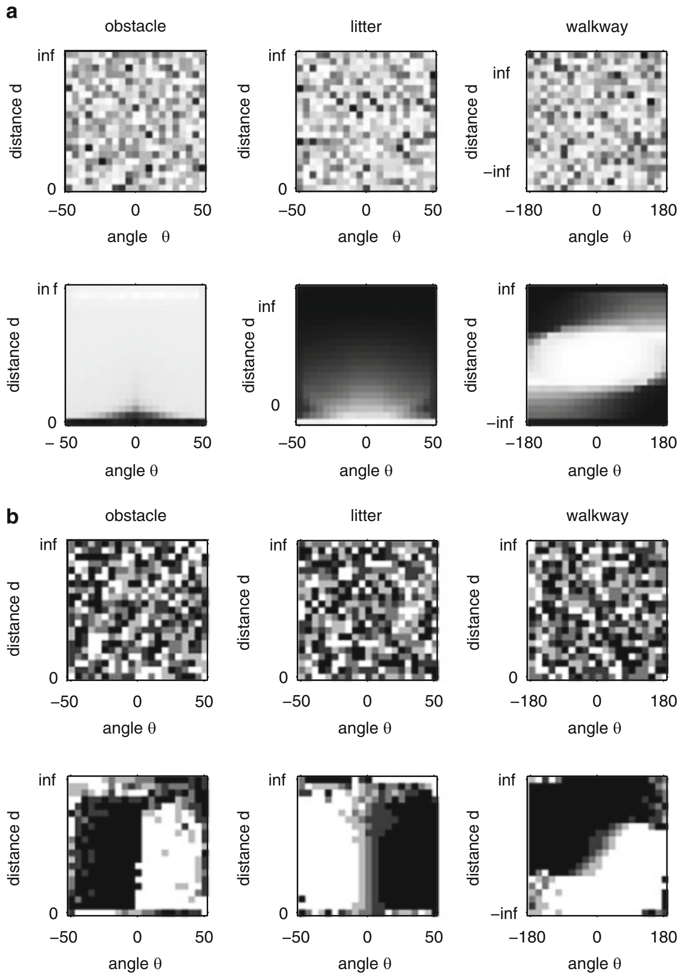

Fig. 10 Values and policies for navigation modules across learning. (**a**) Representations of value functions in the walkway navigation task for the three component behaviors across learning. *Top row*: initial value functions $V^{(i)}(s)$ at iteration 1. *Bottom row*: value functions $V(s)$ after 20,000 episodes. (**b**) Representations of policies in the walkway navigation task for the three component behaviors across learning. *Top row*: policies $\pi^{(i)}$ in episode 1. *Bottom row*: policy estimates. Key: For Value functions, light values are high numbers; for policies light numbers are left turns and darker numbers are right turns

The accuracy of these estimates is shown in Fig. 9. Thus the individual learners are able to learn their correct reward model for their respective tasks.

The value functions and policies of these simulations are shown in Fig. 10, at both the first iteration with random initial values and after learning, when the agent has walked the walkway for 1,000 episodes. As can be seen from the representation of the reward estimates, the individual behaviors have learned the true rewards of their respective tasks, where not intersecting with an obstacle results in a reward of one unit, intersecting a litter object gives four units of reward, and staying on the walkway results in a reward of 0.8 units.

7 Discussion and Conclusions

Natural behavior consists of extended sequential perception and action with multiple concurrent and changing goals. Solving such tasks can be accomplished by some form of a decomposition such that elemental task solutions from a repertoire of behaviors can be reused and used in combinations. Here a specific modular architecture was considered in which separate independent state representations are assumed to be available for each of a large number of behavioral modules. These modules can be combined to solve tasks and task combinations but they only observe a single global reward value. The goal of the system is to learn about each individual component task's rewards and values and to learn how to schedule combinations of these elemental task solutions in order to achieve good performance on the composite tasks.

The primary contribution of this chapter is to describe a way of learning to activate learners with independent state representations so that they can jointly solve a composite control task. The presented solution is based on a credit assignment computation that enables individual modules to learn their correct reward functions and correct value functions from executing different task combinations while observing only global instead of individual reward. The key constraint, motivated by the need for a system that would potentially scale to a large library of behaviors, is to assume that the overall system can be structured such that it could achieve its goals by using only a subset of its behavioral repertoire at any instant and that the reward gained by this subset is the total of that earned by its component behaviors. The use of modules allows the rewards obtained by reinforcement to be estimated online.

By using an on-policy learning method, it is ensured that the learnt action-value functions are correct estimates of the total discounted future reward of each individual module so that the actions selected by an individual module involved in solving only its respective tasks will be optimal. In task combinations, if the summation of Q-values holds, then the composite tasks will be solvable by some subset of the provided learners and these will learn their respective contributions to the sum of rewards. As action selection is based on combining the estimated action-values and not the action themselves, the selected actions will be optimal with respect to the composite task. If additional learners are available, whose actions, given their state representations, do not contribute to any reward, these will be selected less over the time course of learning. Thus, the system can learn that only a small subset of learners may actually contribute to reward in different task combinations.

The present work is related to other approaches that start out with compositional solutions to individual problems and then devise methods in order to combine a large number of such elemental solutions, e.g. Singh and Cohn (1998) and Meuleau et al. (1998). Both approaches are concerned with learning solutions to large MDPs by utilizing solutions or partial solutions to smaller component MDPs. In Meuleau et al. (1998) the solutions to such components are heuristically combined to find an approximate solution to the composite MDP by exploiting assumptions on the structure of the joint action space. A way of learning a composite MDP from individual component MDPs by merging has been described in Singh and Cohn (1998). First the policies for elemental component MDPs are learned by separate modules. In their formulation, when combined, the different modules share a common action space, i.e. one single action is chosen in the composite problem. This is often used as the criterion to distinguish between single agent modular and multiagent problems. Because of this selection of a single action, each individual action is not necessarily optimal for all the component MDPs. Given that the component value function is not necessarily obtained as a linear sum of the component policies, which are each representing different reward types without a common currency, the authors propose a way of initializing the value iteration process of the composite problem using bounds on the state values derived from the state values of the individual component MDPs. In our venue small numbers of behaviors that are appropriate for the current situation are selected in an online fashion. In this situation it is essential to get the Q-values right. An algorithm that models other modules' contribution as pure noise (Chang et al. 2004) will compute the correct policy when all the behaviors or agents are active but this result will not extend to active subsets of modules and behaviors, as incorrect Q values, when used in subsets, will cause suboptimal behavior.

Many open problems remain to be considered at the computational and algorithmic levels but also in relating the proposed computations to biological systems. It would be desirable to learn the separate state representations that are required by the individual modules, i.e. to learn the underlying factorizations. Which state variables are independent and how can sensory variables be learned for individual modules? Here we considered having a large collection of individual learners with different state representations from the start and presented a way of learning to select the appropriate ones for episodes with the respective task combinations. But the scheduling algorithm may quickly get overwhelmed if the number of available modules is very large. Another central issue relates to the factorization assumption itself. There may be problems that do not allow for such factorizations simply because of the underlying interactions of state variables. Recently, Toutounji et al. (2011) showed how the credit assignment algorithm used in this chapter can be used also for the case of interacting modules. In this case, a hierarchy of modules can learn, adjusting the contributions from low level modules that treat a problem as factorizable although the state transitions interact. By learning to correctly divide up the observed total reward, higher level modules adjust the contributions from the lower level modules to achieve the overall task.

Another central open problem in the context of multiple modules used in the present chapter relates to the balancing of exploration and exploitation. As individual modules have different state representations and may have different interaction histories because they can be scheduled independently, a particular action that may be exploiting for one module may be exploratory for another module. In general, in a model-based RL framework, in which the uncertainties about the rewards and transitions are made explicit, optimal balance between exploration and exploitation can be achieved naturally by acting optimally with respect to the expected total discounted future rewards, i.e. the current belief over the underlying MDPs. Thus, as these modules may be reused during later episodes, the trade-off between exploration and exploitation can be understood as a balance between attaining the objectives in the current episode and the reduction of uncertainties about the component tasks, as they may become valuable in later episodes. Albeit computationally intractable in the general case, this is a principled way of expressing the resulting tendency of an agent to carry out exploratory actions, if these promise a higher reward in the long run, a behavior that can be equated with curiosity.

Acknowledgements The research reported herein was supported by NIH Grants RR009283 and MH060624. C. R. was additionally supported by EC MEXT-project PLICON and by the EU-Project IM-CLeVeR, FP7- ICT-IP-231722.

References

Ballard, D. H., Hayhoe, M. M., Pelz, J. (1995). Memory representations in natural tasks. *Journal of Cognitive Neuroscience, 7*(1), 68–82.

Ballard, D. H., Hayhoe, M. M., Pook, P., Rao, R. P. N. R. (1997). Deictic codes for the embodiment of cognition. *Behavioral and Brain Sciences, 20*, 723–767.

Barrett, H., & Kurzban, R. (2006). Modularity in cognition: framing the debate. *Psychological Review; Psychological Review, 113*(3), 628.

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation, 2*(1).

Chang, Y.-H., Ho, T., Kaelbling, L. P. (2004). All learning is local: multi-agent learning in global reward games. In S. Thrun, L. Saul, B. Schölkopf (Eds.), *Advances in neural information processing systems 16*. Cambridge: MIT.

Daw, N., & Doya, K. (2006). The computational neurobiology of learning and reward. *Current opinion in Neurobiology, 16*(2), 199–204.

Daw, N. D., Niv, Y., Dayan, P. (2005). Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience, 8*(12), 1704–1711.

Dayan, P., & Hinton, G. E. (1992). Feudal reinforcement learning. In *Advances in neural information processing systems 5* (pp. 271–271). Los Altos: Morgan Kaufmann Publishers, Inc.

Doya, K., Samejima, K., Katagiri, K.-I., Kawato, M. (2002). Multiple model-based reinforcement learning. *Neural Computation, 14*(6), 1347–1369.

Fodor, J. A. (1983). *Modularity of Mind.* Cambridge: MIT.

Gábor, Z., Kalmár, Z., Szepesvári, C. (1998). Multi-criteria reinforcement learning. In *Proceedings of the fifteenth international conference on machine learning* (pp. 197–205). Los Altos: Morgan Kaufmann Publishers Inc.

Gershman, S., Pesaran, B., Daw, N. (2009). Human reinforcement learning subdivides structured action spaces by learning effector-specific values. *The Journal of Neuroscience, 29*(43), 13524–13531.

Guestrin, C., Koller, D., Parr, R., Venkataraman, S. (2003). Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research, 19*, 399–468.

Humphrys, M. (1996). Action selection methods using reinforcement learning. In P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, S. W. Wilson (Eds.), *From animals to animats 4: proceedings of the fourth international conference on simulation of adaptive behavior* (pp. 135–144). Cambridge: MIT, Bradford Books.

Jacobs, R., Jordan, M., Nowlan, S., Hinton, G. (1991). Adaptive mixtures of local experts. *Neural Computation, 3*(1), 79–87.

Kable, J., & Glimcher, P. (2009). The neurobiology of decision: consensus and controversy. *Neuron, 63*(6), 733–745.

Kaelbling, L. P. (1993). Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of the tenth international conference on machine learning* (vol. 951, pp. 167–173). Los Altos: Morgan Kaufmann.

Karlsson, J. (1997). *Learning to solve multiple goals*. PhD thesis, University of Rochester.

Kok, J. R., & Vlassis, N. (2004). Sparse cooperative q-learning. In *Proceedings of the international conference on machine learning* (pp. 481–488). New York: ACM.

Land, M. F., & McLeod, P. (2000). From eye movements to actions: how batsmen hit the ball. *Nature Neuroscience, 3*, 1340–1345.

Mannor, S., & Shimkin, N. (2004). A geometric approach to multi-criterion reinforcement learning. *The Journal of Machine Learning Research, 5*, 325–360.

Meuleau, N., Hauskrecht, M., Kim, K.-E., Peshkin, L., Kaelbling, L., Dean, T., Boutilier, C. (1998). Solving very large weakly coupled markov decision processes. In *AAAI/IAAI* (pp. 165–172). Menlo Park: AAAI Press.

Minsky, M. (1988). *The society of mind*. New York: Simon and Schuster.

Morris, G., Nevet, A., Arkadir, D., Vaadia, E., Bergman, H. (2006). Midbrain dopamine neurons encode decisions for future action. *Nature Neuroscience, 9*(8), 1057–1063.

Natarajan, S., & Tadepalli, P. (2005). Dynamic preferences in multi-criteria reinforcement learning. In *Proceedings of the 22nd international conference on machine learning* (pp. 601–608). New York: ACM.

Pinker, S. (1999). How the mind works. *Annals of the New York Academy of Sciences, 882*(1), 119–127.

Ring, M. B. (1994). *Continual learning in reinforcement environments*. PhD thesis, University of Texas at Austin.

Rothkopf, C. A. (2008). *Modular models of task based visually guided behavior*. PhD thesis, Department of Brain and Cognitive Sciences, Department of Computer Science, University of Rochester.

Rothkopf, C. A., & Ballard, D. H. (2010). Credit assignment in multiple goal embodied visuomotor behavior. *Frontiers in Psychology, 1*, Special Issue on Embodied Cognition(00173).

Rummery, G. A., & Niranjan, M. (1994). On-line Q-learning using connectionist systems. Technical Report CUED/F-INFENG/TR 166, Cambridge University Engineering Department.

Russell, S., & Zimdars, A. L. (2003). Q-decomposition for reinforcement learning agents. In *Proceedings of the international conference on machine learning* (vol. 20, p. 656). Menlo Park: AAAI Press.

Sallans, B., & Hinton, G. E. (2004). Reinforcement learning with factored states and actions. *Journal of Machine Learning Research, 5*, 1063–1088.

Samejima, K., Ueda, Y., Doya, K., Kimura, M. (2005). Representation of action-specific reward values in the striatum. *Science, 310*(5752), 1337.

Schneider, J., Wong, W.-K., Moore, A., Riedmiller, M. (1999). Distributed value functions. In *Proceedings of the 16th international conference on machine learning* (pp. 371–378). San Francisco: Morgan Kaufmann.

Schultz, W., Dayan, P., Montague, P. (1997). A neural substrate of prediction and reward. *Science, 275*, 1593–1599.

Singh, S., & Cohn, D. (1998). How to dynamically merge markov decision processes. In *Neural information processing systems 10* (pp. 1057–1063). Cambridge: The MIT Press.

Sprague, N., & Ballard, D. (2003). Multiple-goal reinforcement learning with modular sarsa(0). In *International joint conference on artificial intelligence* (pp. 1445–1447). Morgan Kaufmann: Acapulco.

Sprague, N., Ballard, D., Robinson, A. (2007). Modeling embodied visual behaviors. *ACM Transactions on Applied Perception, 4*(2), 11.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: an introduction*. Cambridge: MIT.

Toutounji, H., Rothkopf, C. A., Triesch, J. (2011). Scalable reinforcement learning through hierarchical decompositions for weakly-coupled problems. In *2011 IEEE 10th international conference on development and learning (ICDL)* (Vol. 2, pp. 1–7). New York: IEEE.

Ullman, S. (1984). Visual routines. *Cognition, 18*, 97–157.

Von Neumann, J., Morgenstern, O., Rubinstein, A., Kuhn, H. (1947). *Theory of games and economic behavior*. Princeton: Princeton University Press.

Watkins, C. J. (1989). *Learning from delayed rewards*. PhD thesis, University of Cambridge.

Yarbus, A. (1967). *Eye movements and vision*. New York: Plenum Press.