

# Scalable Reinforcement Learning through Hierarchical Decompositions for Weakly-Coupled Problems

Hazem Toutounji\*<sup>†</sup>, Constantin A. Rothkopf\*, and Jochen Triesch\*

\* Frankfurt Institute for Advanced Studies, Frankfurt, Germany.

<sup>†</sup> Institute of Cognitive Science, University of Osnabrueck, Osnabrueck, Germany.

Email: {toutounji, rothkopf, triesch}@fias.uni-frankfurt.de

**Abstract**—Reinforcement Learning, or Reward-Dependent Learning, has been very successful at describing how animals and humans adjust their actions so as to increase their gains and reduce their losses in a wide variety of tasks. Empirical studies have furthermore identified numerous neuronal correlates of quantities necessary for such computations. But, in general it is too expensive for the brain to encode actions and their outcomes with respect to all available dimensions describing the state of the world. This suggests the existence of learning algorithms that are capable of taking advantage of the independencies present in the world and hence reducing the computational costs in terms of representations and learning. A possible solution is to use separate learners for task dimensions with independent dynamics and rewards. But the condition of independence is usually too restrictive. Here, we propose a hierarchical reinforcement learning solution for the more general case in which the dynamics are not independent but weakly coupled and show how to assign credit to the different modules, which solve the task jointly.

## I. INTRODUCTION

Confronted with the complexities of their environment, animals and humans need to learn how to choose among the huge amounts of responses available for each situation. The degree to which a certain action is desirable or rewarding in the long run defines a mechanism to carry out learning known as *Reinforcement Learning* (RL) [1] or *Reward-Dependent Learning*. The learner has to benefit from the environment’s feedback to predict the availability of future rewards, use it to evaluate its situation, and choose its next action accordingly. The observed outcomes can then be used to refine the prediction of future rewards.

Evidence for the brain computing with quantities present in Temporal Difference (TD) algorithms of RL is abundant. Single-unit recordings from dopaminergic neurons in the basal ganglia show how their phasic activity corresponds to errors in predicting future rewards [2] and these findings are confirmed in humans from the BOLD activity using fMRI imaging [3], [4], [5]. Also, neurons have been identified to detect reward, associate it with different outcomes, and control behavior [6]. Further studies illustrated the role of dopamine neurons in decision making by modulating their activity according to the values of future actions [7]. Even in situations where no explicit notion of reward is available, dopamine seems to play a role of an intrinsic reward as is evident, for instance, from self-stimulation experiments [8].

However, only very small state spaces are usually considered in the aforementioned experiments and models, such as classical conditioning [2], sequential instrumental conditioning with two possible actions [9], or decision making in two-armed bandit decision tasks [7]. But learning coordination of multiple limbs or in general multiple goals in natural settings across developmental time scales requires dealing with many dimensions describing the state of the world. Especially during learning, encoding the outcomes of actions with respect to all possible dimensions representing the state of the world is prohibitively expensive in artificial and biological systems alike.

For the particular case when actions in different dimensions are independent, an agent can learn individual modules for separate variables describing the full state space, see e.g. [10], [11], [12]. Evidence from the cognitive science literature for the ability of humans to learn separate task solutions is also abundant and comes from different areas [13], [14], [15]. In fact, embodied cognition research recently suggested how an agent could learn performing multiple tasks that aim at different goals while performing a common action [16]. The agent is capable of learning the contributions of each single module to the overall reward by solving the structural credit assignment problem.

Here we address the more difficult case in which independence conditions between modules don’t hold. Both in motor control and embodied cognition, transitions of one module might affect those of another, i.e. in a weakly-coupled environment. For instance, when learning to grasp objects, it is not necessary to represent the state of all joints simultaneously. Instead, first controllers for individual movement primitives are learned and subsequently their coordination, as required e.g. when jointly grasping an object with both hands. Indeed, the developmental literature has demonstrated the progress of learning to proceed in this way. The solution we propose here is to hierarchically decompose the control problem and to proceed with learning across the levels of hierarchy from the bottom up.

Note that this type of hierarchical RL is distinct from the often considered temporal hierarchy, which can be described as a concatenation of elemental solutions in time. Such hierarchical RL can often be accommodated conceptually in the

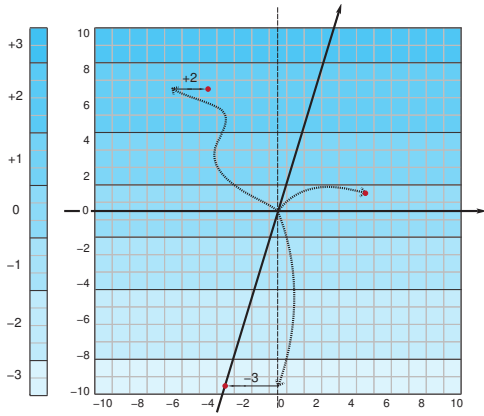


Fig. 1. A schematic illustration of the task. The color bar indicates crosstalk values at each area of the state space. The dotted lines show example actions and the resulting crosstalk of these actions.

semi-Markov-process framework [19]. This field has obtained attention early on in the development of RL as e.g. in [17], [18] and has seen increased recent interest [20]. The work presented here instead can better be described as structural hierarchy, which also was considered early on in the development of RL e.g. [21], [22]. The distinct solution we present here is based on a method for assigning credit to component modules that may be active concurrently and to modules at higher levels in the hierarchy, which learn adjusting the actions of lower level modules and accurately account for their contribution to the total reward.

We show that a solution for such a task, where transition dynamics of one module affect that of another, can be obtained with such a structural hierarchical RL approach. Individual model-free modules learn task solutions for individual dimensions. An additional model-free module furthermore learns to coordinate the independent lower-level modules by compensating for the weak coupling in their dynamics. Importantly, due to the weak coupling, the coordinating module needs only to learn with low resolution over the full state space. This, in addition to the decoupling of dimensions, results in huge saving in storage, computation, and learning time. We compare different structural credit assignment rules and training schedules and show the significant differences in learning times and asymptotic performance through simulations. Thus, we solve a problem relevant for learning agents and make concrete predictions about the required signals under different learning schemes, and we show differences in the resulting learning curves.

## II. BACKGROUND METHODS

### A. The Reinforcement Learning Problem

RL [1] comprises a set of algorithms for learning how to select sequences of actions so as to maximize the total collected reward. The interaction between the controller and environment can be expressed as a Markov Decision Process (MDP) [23], which in turn is defined by a 4-tuple  $(S, A, T, R)$ .  $S$  and  $A(s)$  are respectively the set of states and that of

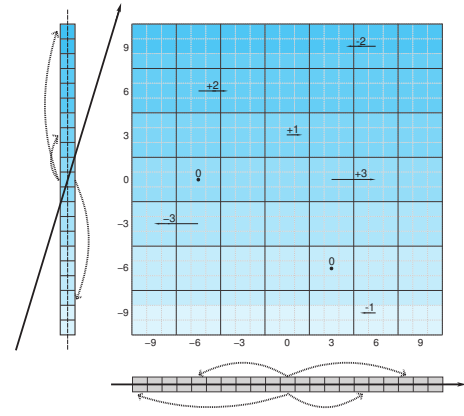


Fig. 2. A schematic illustration of the hierarchy. The vertical module learns vertical actions; the horizontal module learns horizontal actions and is affected by crosstalk. The third module learns horizontal coordinating actions and has a lower resolution over the full state space (the resolution coefficient is  $res = 3$  in this case).

actions to perform at these states.  $T_{ss'}^a = P(s'|s, a)$  defines transition dynamics. i.e. the probability of moving to state  $s'$  when performing action  $a$  at state  $s$ .  $R_{ss'}^a = E(r|s, a, s')$  is a reward model. i.e. the expected value of getting a reward  $r$  upon performing the action  $a$  from  $s$  and getting to  $s'$ .

The goal of a RL algorithm is to find an optimal policy  $\pi$  mapping states to actions that maximizes future discounted reward from experience collected while interacting with the environment. A common approach is to estimate so-called action value functions  $Q^\pi(s, a)$  that describe how useful it is for the agent to perform action  $a$  when in state  $s$  and following policy  $\pi$  thereafter. It can be shown that the Q-value function for optimal policies fullfills the following condition:

$$Q^*(s, a) = \sum_{s'} P_{ss'}^a \left[ R_{ss'}^a + \gamma \max_{a'} Q^*(s', a') \right] \quad (1)$$

where  $0 < \gamma \leq 1$  is a discounting factor that quantifies the importance given to future rewards in relation to more recent ones. This formula is an example of a Bellman optimality equation [24], [25]. Given any policy  $\pi$ , state values can be obtained from action values:

$$V^\pi(s) = \max_a Q^\pi(s, a) \quad (2)$$

A popular method for solving the RL problem is the SARSA learning rule [26]:

$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma Q(s', a') - Q(s, a)] \quad (3)$$

where  $\alpha$  is a learning rate and the term in brackets defines errors in predicting the reward. Convergence is attained when the prediction error converges to zero. In this sense, at each time step through learning with SARSA, the controller is estimating its reward by:

$$\hat{r} = Q(s, a) - \gamma Q(s', a') \quad (4)$$

## B. Learning for Factorized Environments

Let's first consider an environment where states and actions in different dimensions are not overlapping. In this case, each dimension can be expressed with a separate Markov decision process  $(S^{(i)}, A^{(i)}, T^{(i)}, R^{(i)})$ . In other words, each MDP will correspond to one RL module and the environment's model can be factorized as follows [16]:

$$P(s_{t+1}|s_t, a_t) = \prod_{i=1}^M P(s_{t+1}^{(i)}|s_t^{(i)}, a_t^{(i)}) \quad (5)$$

$$P(r_{t+1}|s_t, a_t, s_{t+1}) = \prod_{i=1}^M P(r_{t+1}^{(i)}|s_t^{(i)}, a_t^{(i)}, s_{t+1}^{(i)}) \quad (6)$$

where  $M$  is the number of modules.  $s$  is an  $M$ -dimensional vector that corresponds to the composite state of the  $M$  modules. This also applies to both  $a$  and  $r$ . Given this along with Bellman optimality equation (1), value functions are also factorizable:

$$Q(s_t, a_t) = \sum_{i=1}^M Q(s_t^{(i)}, a_t^{(i)}) \quad (7)$$

$$V(s_t) = \sum_{i=1}^M V(s_t^{(i)}) \quad (8)$$

and a controller learning independent value functions for each single module converges to an optimal solution while saving in storage, computation, and learning time.

## C. Credit Assignment

In a factorized environment, each controller could learn from the total observed reward, which is the result of the combined actions of all learners. In this case each controller only needs to have access to the total reward signal. The problem with this setting is that this composite reward is indeed obtained by the joint actions of all active modules. Therefore, an individual controller may learn an appropriate policy, but it will learn values that are not equal to those obtained from its individual action contributions. Instead, the total reward should be seen as the sum of the actual reward for an individual controller plus the contributions from the other active modules. Thus, each single module contributes only some portion of the full reward, and it should therefore only receive the amount of reward that it did actually contribute:

$$R_t = \sum_{i=1}^M r_t^{(i)} \quad (9)$$

where  $R$  is the global, total reward observed by the system which comprises all the individual reward contributions.

These reward contributions are not known to the separate modules a priori and can therefore be initialized randomly, but they can be bootstrapped during learning depending on the difference between the received global reward and the sum of

previous estimates of individual components [27], [16]:

$$\hat{r}^{(i)}(s^{(i)}, a^{(i)}) \leftarrow \hat{r}^{(i)}(s^{(i)}, a^{(i)}) + \beta \left( R - \sum_{j=1}^M \hat{r}^{(j)}(s^{(j)}, a^{(j)}) \right) \quad (10)$$

Using these estimates, each module can learn its corresponding action values with the SARSA algorithm where prediction errors at each time step are computed by:

$$\delta^{(i)} = \hat{r}^{(i)} + \gamma Q(s^{(i)}, a^{(i)}) - Q(s^{(i)}, a^{(i)}) \quad (11)$$

We will next show how the ideas of section II can be extended in a hierarchical fashion to efficiently solve tasks where the environment is not fully factorized.

## III. WEAKLY-COUPLED ENVIRONMENTS

### A. A Weakly-Coupled Task

For ease of exposition, we will use a very simple task to exemplify the proposed hierarchical learning algorithms. The task is only two-dimensional as we principally aim to show that non-factorizability in weakly-coupled problems is overcome by a hierarchical solution. We plan on applying this framework to more complex problems in the future.

We suggest a grid-world task with 2-dimensional state space of size  $N \times N$ , in which a controller is attempting, starting from uniformly random initial conditions, to reach a goal state where the reward is maximal. Crucially, transitions in the horizontal direction of movement are influenced by the amplitude of the vertical movement in that vertical actions affect horizontal actions with a small additional shift. We call this amount of interference a *Crosstalk*.

Given a maximum possible crosstalk value  $MC$ , a vertical action  $a_v$ , and a positive parameter  $\epsilon \ll 1$ , the horizontal state is shifted by a crosstalk of  $\text{Round}[2*(1+\epsilon)*MC*a_v/(N-1)]$  states.

A state is defined by a 2-dimensional position vector in relation to the central goal state. At each state, the controller performs 2-dimensional actions that keep it within the grid. Due to the crosstalk, we allow for extra  $MC$  actions at each side and each row of the grid so that the controller has the possibility to reach any state with one action only.

The reward decays linearly from the goal state until it reaches a minimum value at which it saturates  $MC + 1$  steps away from the goal state. As such, the reward at each state  $(s_h, s_v)$  is defined by:

$$\max \left\{ \frac{(r_{min} - r_{max}) \cdot (|s_h| + |s_v|)}{MC + 1} + r_{max}, r_{min} \right\} \quad (12)$$

Unless otherwise specified, all simulations are run with a grid of size  $21 \times 21$  and  $MC = 3$ . Fig. 1 shows a schematic illustration of the task. The color bar to the left indicates crosstalk values for different states.

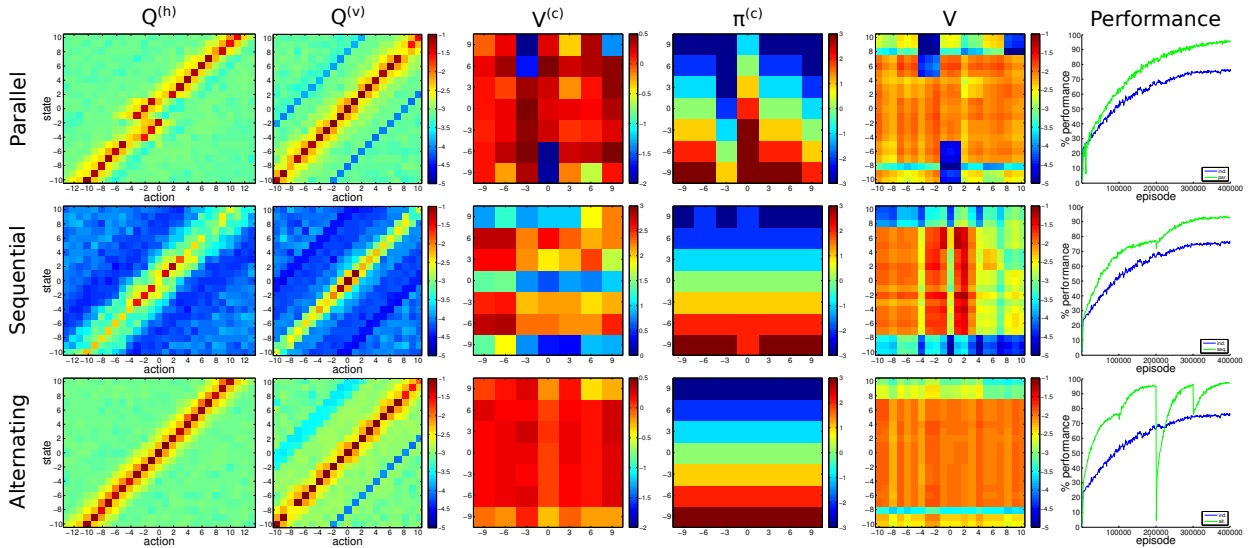


Fig. 3. Simulation results of the first crediting condition. Figures from top to bottom corresponds to *parallel*, *sequential*, and *alternating* schedules. From left to right are the horizontal and vertical action values where each row corresponds to a state (the color bar indicates that values ranges from  $-5$  to  $-1$ ), the state values of the coordinating module (ranging from  $0$  to  $3$  for the sequential schedule and from  $-2$  to  $0.5$  for the other two), the policy of the coordinating module ranging from  $-3$  to  $3$ , the combined state values computed from (16) and ranges from  $-5$  to  $-1$ , and the percentage performance of the hierarchical solution compared to that of a solution where only the lower modules are turned on.

## B. Hierarchical Solution

We start with the above learning problem with two one-dimensional modules, a horizontal and a vertical one. The horizontal module’s transition dynamics is affected by the vertical module’s actions. Given the coupling between the two dimensions of the task, it is not possible to solve it by using two separate modules, since the factorization conditions (5) and (6) don’t hold. Our solution is to introduce a third higher-level module that learns to coordinate the two lower-level ones by learning values of actions on the horizontal direction (the direction affected by crosstalk).

For small values of  $MC$ , the coordinating module needs only to learn values over the whole environment but with lower resolution since close-by states have the same amount of crosstalk. We don’t learn the structure of the coordinating module as we consider that beyond the scope of this paper. Fig. 2 shows a schematic illustration of the hierarchy.

## IV. LEARNING

All modules learn with the SARSA algorithm. It is important to use an on-policy learning rule when multiple modules are learning simultaneously, as the carried out actions leading to a reward may be the conjunction of multiple modules. Accordingly, future actions may not reflect the currently optimal policy, which is a requirement for an off-policy learning algorithm such as Q-learning.

Simulations are divided into epochs. During an epoch learning in a module is either switched on or off. The modules choose actions following an  $\epsilon$ -greedy exploration policy. This implies that a module chooses an action randomly with some probability  $\epsilon$ . Otherwise, it performs the currently optimal action. During an epoch,  $\epsilon$  decreases exponentially with time

so that the module becomes more exploitative. The choice of parameters is as follows. At each epoch,  $\epsilon$  takes values in the range  $0.5 \rightarrow 0.01$ . Learning rate is fixed to  $\alpha = 0.1$  and the discounting factor is  $\gamma = 0.9$ .

### A. Scheduling

Learning across the different levels of the hierarchy can progress differently depending on whether an individual layer adjusts its parameters. To explore the consequences of different learning schedules, we compare three different sequences of adjusting the parameters of the respective level of the hierarchy. In a *Parallel Schedule*, the two levels of the hierarchy learn simultaneously. With a *Sequential Schedule*, learning in lower-level modules is turned on only in the beginning, and when the performance has become relatively stable, only the coordinating module learns. Finally, the *Alternating Schedule* repeats the two epochs of a sequential schedule twice.

### B. Credit Assignment of the Hierarchy

We run our simulation with two different crediting conditions described below, which require the representation of different quantities for the learning ensemble. In the first, the two lower-level modules don’t know their contribution to the global reward  $R$ . They both learn on the basis of the observed reward, so that the estimated reward across all modules is actually  $2R$ . The coordinating module, however, receives what remains of these two units of reward that depends on the lower-level modules’ reward estimates computed with (4). The coordinating module’s reward estimate is thus given by:

$$\hat{r}^{(c)} = 2R - \hat{r}^{(h)} - \hat{r}^{(v)} \quad (13)$$

and its prediction error is:

$$\delta^{(c)} = \hat{r}^{(c)} + \gamma Q(s'^{(c)}, a'^{(c)}) - Q(s^{(c)}, a^{(c)}) \quad (14)$$

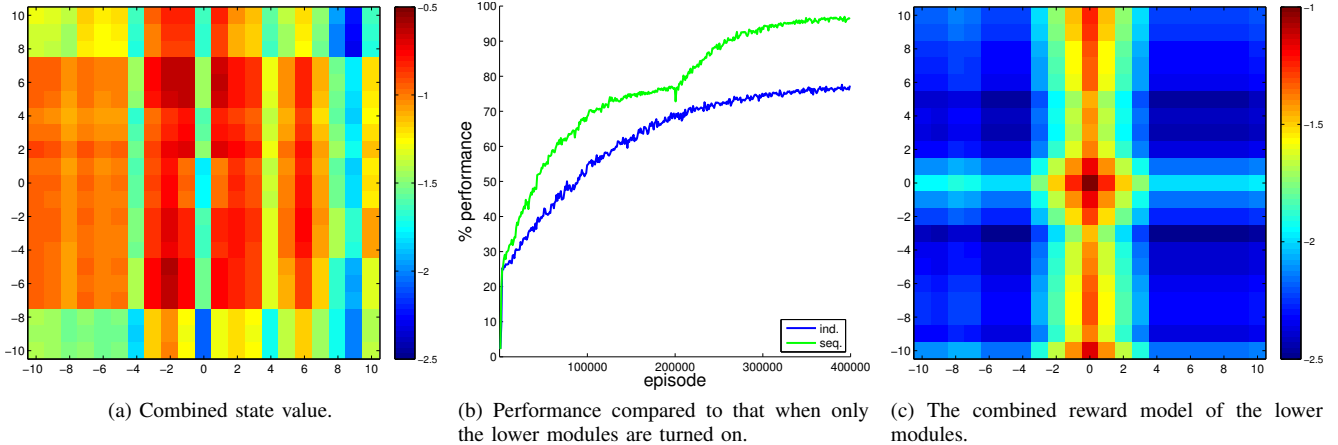


Fig. 4. Sample simulation results of the second crediting condition with a sequential schedule.

where the superscripts  $(h)$ ,  $(v)$ , and  $(c)$  refers to the horizontal, vertical, and coordinating modules respectively. A state of the coordinating module  $s^{(c)}$  is obtained by applying a function  $f_c(s^{(h)}, s^{(v)})$  that maps a full state to the low resolution state space of the coordinating module.

Since it is a single controller that runs the three modules to achieve a single task, an alternative is to split the received reward among the individual components. To this end, we implement a second crediting condition where the low-level modules learn their contribution to a single unit of global reward  $R$  according to the credit assignment algorithm (9) and (10) with  $\beta = 0.1$ . The only change in the coordinating module is that its reward estimate becomes:

$$\hat{r}^{(c)} = R - \hat{r}^{(h)} - \hat{r}^{(v)} \quad (15)$$

where  $\hat{r}^{(x)}$  is the estimate of reward at the next state  $s^{(x)}$  of a lower module.

By learning how to compensate for the weak coupling in the dynamics of the two lower-level modules, and by depending on what remains of the global reward to trigger its learning, the coordinating module cooperates with the lower-level modules to approximately achieve the factorization conditions (5) and (6). We assert our intuition through the following simulations.

## V. RESULTS

### A. Simulation

Fig. 3 illustrates results of learning for the crediting condition (13) with the three proposed schedules. Performance is measured by reward-per-step, averaged over 1000 episodes, and normalized to a percentage. The performance of the three schedules is 95, 93, and 97% respectively while it saturates at around 75% when the coordinating module is turned off. These results are robust across different trials. The performance doesn't reach 100% due to the low resolution of the coordinating module and the possibility of its states having more than one value of crosstalk as one can see in Fig. 2.

Crosstalk overlap can also be seen by looking at the combined state value function adopted from the factorization

of the full value function (8) and computed by:

$$V(s^{(h)}, s^{(v)}) = V^{(h)}(s^{(h)}) + V^{(v)}(s^{(v)}) + V^{(c)}(s^{(c)}) \quad (16)$$

where  $s^{(c)} = f_c(s^{(h)}, s^{(v)})$ . The combined value function doesn't converge to the expected value of  $2R$  at states on the bottom and top low resolution states where there is a crosstalk overlap.

With the parallel schedule, where the three modules are learning during the entire period, the controller needs to explore all the possible combinations of actions of its three modules, and it ends up converging to a policy where the horizontal module creates triplets of states. It also fails to converge on some of the states due to the complexity of the exploration dynamics. This effect can be observed in the fluctuations of performance at the beginning of learning when exploration rates are high.

When running a sequential schedule, each lower module learns a stochastic policy over its allotted state space during the first epoch of simulation without the coordination from the upper module. This explains the low action values of the lower modules and the high contribution of the upper module to the combined state values after the end of the second epoch. On the other hand, at the third and fourth epochs of the alternating schedule, both levels of the hierarchy learn to take the other level contributions into account which results in a more convergent combined state value function and a higher performance. The big dip in performance at the beginning of the third epoch is due to the high exploration rate at the two lower modules which transiently reduces the already learned values.

Learning with the crediting condition (15) gives results that are similar qualitatively to those obtained from the crediting condition (13) (Fig. 4), but differs in that the controller converges to the true state value function (Fig. 4a) since each module learns its true contribution to the total observed reward  $R$ . This is an important result as it makes sure that each module learns the true total discounted sum of future reward for a particular state that it is contributing, which is

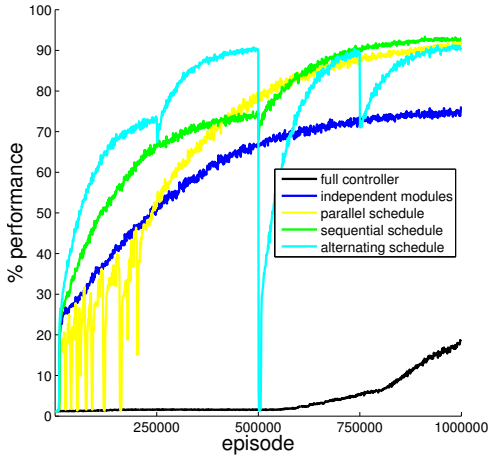


Fig. 5. Comparison of performance in solving the task with  $49 \times 49$  states and a  $MC = 6$  between a single controller, two independent modules, and the three possible hierarchical schedules. The resolution parameter of the coordinating module in the hierarchical cases is set to  $res = 7$ .

consistent with the reward contributions of the other learners. Also, by learning their contribution to the reward, the two lower modules capture the dynamics of the environment more closely. This allows the coordinating module to learn a better value function that results in carrying out a better policy, and as a consequence, reaching a maximum performance even while running the sequential schedule (Fig. 4b).

To illustrate how assigning credit to the lower modules results in a better correspondence to the environment, we show the combined reward model of the two models learned by the lower modules in Fig. 4c where:

$$\hat{r}(s^{(h)}, s^{(v)}) = \hat{r}^{(h)}(s^{(h)}) + \hat{r}^{(v)}(s^{(v)}) \quad (17)$$

We see that the combined reward model  $\hat{R}$  is similar to the actual reward  $R$ , especially near the center of the grid where the goal is.

### B. Computational Complexity

For the problem size we presented so far, a single controller is capable of reaching an optimal performance in the simulation times used in the experiments. However, we show in Fig. 5 that for a much bigger task with  $49 \times 49$  states, a single controller performs very poorly while the hierarchical solution exceeds 90% performance.

In fact, since at each one of the  $N \times N$  states there are  $N \times (N + 2MC)$  possible actions, the number of action values to be stored and learned in the uni-modular case scales as  $O(N^4)$ .

On the other hand, the first level modules of the hierarchy need only to learn  $2N(N + MC)$  action values which increases as  $O(N^2)$ , and the coordinating module needs to learn  $2MC \times (N/res)^2$  action values. By assuming weak coupling and binding the growth of the resolution parameter to  $O(\sqrt{N})$ , the number of action values the coordinating module learns grows with  $O(MC \times N)$ . As such, the storage and computational demands of the hierarchy are bounded by

$O(N^2)$  in comparison to the  $O(N^4)$  action values required by the uni-modular solution. This doesn't change when learning how to assign credit to the first level of the hierarchy since this only increases the number of parameters to be learned by  $2N$  values.

## VI. CONCLUSION

Reward mediated learning has been shown to be one of the fundamental ways in which animals and humans adjust the actions they carry out in order to increase their gains and decrease their losses. Theoretical work has developed a variety of learning algorithms in the model-based and model-free RL setting and empirical evidence has identified numerous biological correlates with quantities necessary for computations in these models.

But it is still an unsolved problem how to handle the representational and computational burden imposed by the large number of possible dimensions describing the state of the world together with the action outcomes. For naturalistic settings this is a particularly severe problem. This curse of dimensionality in terms of state representations has led to the suggestion that computations should take advantage of the factorization underlying the dynamics and rewards in the world.

Previous work on such factorizations has considered a large variety of interdependences of world dynamics and reward functions. It also considered a wide variety of solutions in terms of the type of modularity, the level of prior knowledge, and the communication between individual modules [21], [10], [11], [12].

Here we propose a model free RL solution in which a hierarchy with two levels of modules learn to jointly solve a task with interacting dynamics. The modules on the first level represent only single dimensions of the task, which are not sufficient to find an optimal solution. Accordingly, these modules learn to carry out actions for their respective states which jointly are far from optimal. An additional module on the next level of the hierarchy then learns to coordinate the overall behavior, by learning when to correct the joint actions of the lower level modules.

We propose different credit assignment computations and training schedules and compare these methods through simulations and show that the structural credit assignment problem can be solved by the hierarchical system. The representational burden and therefore the learning difficulties of the system are significantly lower than for the full system needing to represent the full joint state space. Importantly, the credit assignment computations lead to correct estimates of the relative contributions of the individual modules to the overall reward. Thus, the system is composed of individual modules, which individually over time learn to act on the basis of their limited individual representations, but jointly can solve the full task through the coordinating control of a module on a higher level of the hierarchy by estimating all respective contributions to the achievement of the goal.

Finally, we show that the training schedule in this system is crucial in terms of the performance during learning, the rate of learning, and the final level of performance. The best results are obtained by first learning the control on the lower level of the hierarchy and then proceeding to learn to coordinate the lower level modules. It is tempting to try to relate this finding to the protracted development of synaptic connectivity in infants, where an initial exuberance of synaptic connections in the neocortex is followed by later pruning [28], [29], [30], [31]. Interestingly, this rise and fall of synapse numbers does not occur simultaneously in all areas of cortex. Rather, it happens early for lower level cortical areas such as primary visual cortex, where synapse numbers peak between 1 and 2 years of age, while it is later for higher cortical areas such as prefrontal cortex, where synapse numbers reach their maximum at around 4 years of age [32]. This may reflect the need to learn complex higher-level behaviors on top of sufficiently stable lower-level building blocks.

While our results demonstrate the potential of hierarchical decompositions for a simple example task, future work needs to tackle more complex problems and address how such hierarchies can be learned autonomously.

#### ACKNOWLEDGMENT

This work was supported by the German Federal Ministry of Education and Research within the BMBF Project Bernstein Fokus: Neurotechnologie Frankfurt, FKZ 01GQ0840 and the EU-Project IM-CLeVeR, FP7-ICT-IP-231722.

#### REFERENCES

- [1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, 1998.
- [2] W. Schultz, P. Dayan, and P. R. Montague, "A Neural Substrate of Prediction and Reward," *Science*, vol. 275, no. 5306, pp. 1593–1599, 1997.
- [3] J. A. Gottfried, J. O'Doherty, and R. J. Dolan, "Encoding predictive reward value in human amygdala and orbitofrontal cortex." *Science*, vol. 301, no. 5636, pp. 1104–7, Aug. 2003.
- [4] M. Haruno and M. Kawato, "Different neural correlates of reward expectation and reward expectation error in the putamen and caudate nucleus during stimulus-action-reward association learning." *Journal of neurophysiology*, vol. 95, no. 2, pp. 948–59, Feb. 2006.
- [5] M. Pessiglione, B. Seymour, G. Flandin, R. J. Dolan, and C. D. Frith, "Dopamine-dependent prediction errors underpin reward-seeking behaviour in humans." *Nature*, vol. 442, no. 7106, pp. 1042–5, Aug. 2006.
- [6] W. Schultz, "Multiple reward signals in the brain." *Nature reviews. Neuroscience*, vol. 1, no. 3, pp. 199–207, Dec. 2000.
- [7] G. Morris, A. Nevet, D. Arkadir, E. Vaadia, and H. Bergman, "Midbrain dopamine neurons encode decisions for future action." *Nature neuroscience*, vol. 9, no. 8, pp. 1057–63, Aug. 2006.
- [8] F. Mora and R. Myers, "Brain self-stimulation: direct evidence for the involvement of dopamine in the prefrontal cortex," *Science*, vol. 197, no. 4311, pp. 1387–1389, Sep. 1977.
- [9] Y. Niv, D. Joel, and P. Dayan, "A normative perspective on motivation." *Trends in cognitive sciences*, vol. 10, no. 8, pp. 375–81, 2006.
- [10] M. Humphrys, "Action selection methods using reinforcement learning," in *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*. Cambridge, MA: MIT Press, Bradford Books, 1996, pp. 135–144.
- [11] K. Doya, K. Samejima, K. Katagiri, and M. Kawato, "Multiple model-based reinforcement learning," *Neural computation*, vol. 14, no. 6, pp. 1347–1369, 2002.
- [12] S. Russell and A. Zimdars, "Q-decomposition for reinforcement learning agents." in *In Proceedings of the International Conference on Machine Learning*, 2003.
- [13] H. Shinoda, M. M. Hayhoe, and A. Shrivastava, "What controls attention in natural environments?" *Vision research*, vol. 41, no. 25–26, pp. 3535–45, Jan. 2001.
- [14] C. A. Rothkopf and D. H. Ballard, "Image statistics at the point of gaze during human navigation." *Visual neuroscience*, vol. 26, no. 1, pp. 81–92, 2009.
- [15] A. M. Franco-Watkins, T. C. Rickard, and H. Pashler, "Taxing executive processes does not necessarily increase impulsive decision making." *Experimental psychology*, vol. 57, no. 3, pp. 193–201, Jan. 2010.
- [16] C. A. Rothkopf and D. H. Ballard, "Credit Assignment in Multiple Goal Embodied Visuomotor Behavior," *Frontiers in Psychology*, vol. 1, no. November, pp. 1–13, 2010.
- [17] M. Wiering and J. Schmidhuber, "Hq-learning," *ADAPTIVE BEHAVIOR*, vol. 6, no. 2, pp. 219–246, 1997.
- [18] J. Schmidhuber and R. Wahnsiedler, "Planning simple trajectories using neural subgoal generators," in *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, 1992.
- [19] R. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1, pp. 181–211, 1999.
- [20] M. Botvinick, Y. Niv, and A. Barto, "Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective," *Cognition*, vol. 113, no. 3, pp. 262–280, 2009.
- [21] P. Dayan and G. Hinton, "Feudal reinforcement learning," in *Neural Information Processing Systems 5*, 1992.
- [22] N. Meuleau, M. Hauskrecht, K. Kim, L. Peshkin, L. Pack Kaelbling, T. Dean, and C. Boutilier, "Solving very large weakly coupled Markov decision processes," in *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*. JOHN WILEY & SONS LTD, 1998, pp. 165–172.
- [23] M. Puterman, *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc. New York, NY, USA, 1994.
- [24] R. Bellman, *Dynamic Programming*, 1957.
- [25] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, 1996.
- [26] G. A. Rummery and M. Niranjan, "Online Q-learning using connectionist systems (Tech. Rep. No. CUED/F-INFENG/TR 166)," *Cambridge University Engineering Department, Cambridge*, 1994.
- [27] C. A. Rothkopf, "Modular models of task based visually guided behavior," Ph.D. dissertation, Department of Brain and Cognitive Sciences, Department of Computer Science, University of Rochester, 2008.
- [28] P. R. Huttenlocher, C. de Courten, L. J. Garey, and H. Van der Loos, "Synaptogenesis in human visual cortex—evidence for synapse elimination during normal development." *Neuroscience letters*, vol. 33, no. 3, pp. 247–52, Dec. 1982.
- [29] P. R. Huttenlocher, "Synapse elimination and plasticity in developing human cerebral cortex." *American journal of mental deficiency*, vol. 88, no. 5, pp. 488–96, Mar. 1984.
- [30] P. R. Huttenlocher and C. de Courten, "The development of synapses in striate cortex of man." *Human neurobiology*, vol. 6, no. 1, pp. 1–9, Jan. 1987.
- [31] P. R. Huttenlocher, "Morphometric study of human cerebral cortex development." *Neuropsychologia*, vol. 28, no. 6, pp. 517–27, Jan. 1990.
- [32] P. R. Huttenlocher and A. S. Dabholkar, "Regional differences in synaptogenesis in human cerebral cortex." *The Journal of comparative neurology*, vol. 387, no. 2, pp. 167–78, Oct. 1997.