IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, May 2014.

Bayesian Exploration and Interactive Demonstration in Continuous State MAXQ-Learning

Kathrin Gräve and Sven Behnke

***Abstract*— Deploying robots for service tasks requires learning algorithms that scale to the combinatorial complexity of our daily environment. Inspired by the way humans decompose complex tasks, hierarchical methods for robot learning have attracted significant interest. In this paper, we apply the MAXQ method for hierarchical reinforcement learning to continuous state spaces. By using Gaussian Process Regression for MAXQ value function decomposition, we obtain probabilistic estimates of primitive and completion values for every subtask within the MAXQ hierarchy. From these, we recursively compute probabilistic estimates of state-action values. Based on the expected deviation of these estimates, we devise a Bayesian exploration strategy that balances optimization of expected values and risk from exploring unknown actions. To further reduce risk and to accelerate learning, we complement MAXQ with learning from demonstrations in an interactive way. In every situation and subtask, the system may ask for a demonstration if there is not enough knowledge available to determine a safe action for exploration. We demonstrate the ability of the proposed system to efficiently learn solutions to complex tasks on a box stacking scenario.**

I. Introduction

One of the long-standing goals of robotics research is the eventual deployment of robots to tasks in our everyday life. Given the complexity of our environment and the way it is constantly changing, this requires learning methods that are able to efficiently learn complex task policies in large state spaces. Hierarchical methods are a promising approach to scale robot learning to these challenges. Recognizing the fact that complex tasks usually exhibit hierarchical structure, hierarchical learning methods employ various kinds of abstraction to decompose tasks into smaller units of reduced complexity. For instance, repetitive sequences of actions can often be aggregated to macro operators, allowing complex tasks to be solved in terms of these more abstract actions as illustrated in Fig. 1. Reducing the number of steps and choices on higher levels in this way may greatly accelerate learning. This kind of abstraction is often referred to as *temporal abstraction*. Another way to reduce the complexity of tasks is to identify dimensions of the state space that are irrelevant to some of the subtasks. Subtasks may also be reused from different contexts in order to efficiently exploit existing knowledge. These strategies are known as *state abstraction* and *subtask sharing* [1]. Dietterich proposed the MAXQ value function decomposition [2]—a framework for hierarchical reinforcement learning that successfully leverages these kinds of abstraction in discrete settings.

The authors are with the Autonomous Intelligent Systems group, Department of Computer Science, University of Bonn, Germany. graeve@ais.uni-bonn.de This work was supported by the B-IT Research School.

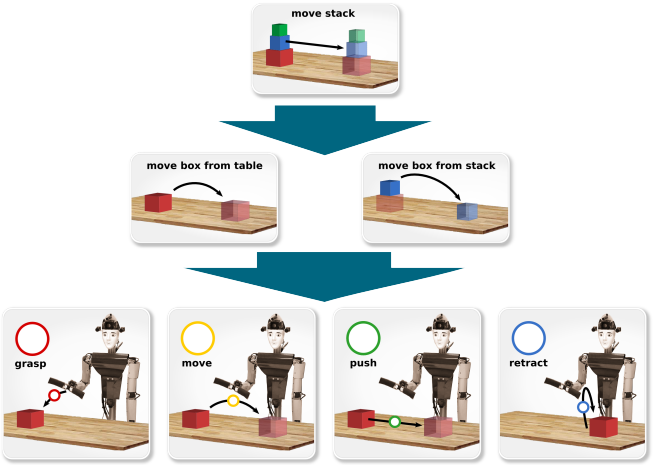

Fig. 1. Hierarchical decomposition of the sample task of moving a stack of boxes from the whole task on the top level via subtasks for moving partial stacks to movement primitives on the bottom level.

In this work, we apply the MAXQ framework to continuous state spaces—a typical characteristic of practical tasks—and complement reinforcement learning with learning from demonstrations in an integrated, interactive approach. We employ *Gaussian Process Regression (GPR)* [3] to obtain approximations of the value function for primitive actions and the completion functions of composite actions of every subtask of a hierarchical task. From these, we compute probabilistic estimates of state-action values by recursively aggregating estimates of completion values of composite tasks and values of primitive tasks throughout a subtask hierarchy. These estimates are central to our approach since their associated uncertainties express the confidence of the system in the outcome of actions given the available knowledge. Based on the estimated values and their uncertainties, we derive a Bayesian exploration criterion that makes informed decisions on promising actions instead of performing arbitrary random exploration. This drastically reduces the number of trials needed to find rewarding actions.

Besides performance with respect to obtained rewards and number of trials, *safety* is another major concern in practical applications. A successful learning algorithm must avoid exploring actions with unpredictable outcomes. We therefore incorporate a trade-off in our exploration strategy that uses *expected deviation* [4] to balance optimization of reward and risk from executing actions with uncertain outcome.

The efficiency of our exploration approach is tied to the availability of prior experience to guide it. Rather than boot-

strapping the system, we propose an interactive approach, allowing the system to request a demonstration of the current subtask from a human expert if the available knowledge is insufficient. Complementing reinforcement learning with learning from demonstrations in this way provides functional priors that accelerate reinforcement learning by narrowing its search space. On the other hand, the human effort of providing demonstrations is reduced by iteratively choosing the most appropriate learning method for every subtask on every level of abstraction of a hierarchical task. Hence, demonstrations are limited to specific subtasks at specific levels of abstraction whereas autonomous exploration of solutions is preferred whenever possible. Finally, in combination with our Bayesian exploration criterion, the set of human demonstrations acts as a behavioral prior that biases the robot's behavior towards actions that are predictable for a human collaborator. This is advantageous to applications where robots are to collaborate closely with humans.

II. Related Work

Over the past decade, many researchers have investigated ways to exploit hierarchical structure to scale reinforcement learning to complex robotics applications [5].

One direction of research explores hierarchical *policy search* methods that optimize the long-term reward of parameterized policies. Daniel et al. [6] proposed a hierarchical extension of the Relative Entropy Policy Search (REPS) [7] method to learn sequences of parameterized movement primitives. The sequence learning problem is formulated as a constrained optimization problem on the parameters of a high-level gating network responsible for action selection and the parameters of low-level motion primitives. Stulp and Schaal [8] extend the PI^2 algorithm [9] to simultaneously optimize the intermediate goals of a sequence of movement primitives and their shape parameters. Both approaches allow for continuous state- and action spaces by using parameterized dynamical systems to represent primitive actions and perform supervised learning from demonstrations to bootstrap parameters. In both cases, though, the length of an action sequence has to be known in advance, limiting the flexibility of the system to select optimal action sequences.

Konidaris et al. [10] developed a more general approach where trajectories are first segmented and then merged to skill trees by considering their statistical similarities in reverse order. Sequences may be trained either by demonstrating viable solutions or by explorative reinforcement learning. Similar to our approach, skill trees encode a deterministic task policy that does not require planning at runtime.

The MAXQ method presents an alternate, value function-based approach that allows reinforcement learning agents to benefit from state abstractions. Several extensions to the original MAXQ method have been proposed to improve scalability and performance by integrating prior knowledge.

One direction of research investigates ways to make more efficient use of experiences gathered by integrating models of the environment and the reward structure into the MAXQ framework. Cao et al. [11] augmented the MAXQ hierarchy with Bayesian priors on distributions over primitive reward and environment models. Learning then proceeds in a mixed model-based/model-free fashion. Every action on the primitive level leads to an update of the posterior over model parameters. At fixed intervals, new primitive model parameters are sampled and the reward and completion values in the MAXQ hierarchy are updated recursively, using the sampled models. Our approach is different in that it keeps the original model-free structure of MAXQ and proposes a Bayesian exploration strategy. Expert knowledge is incorporated in the form of demonstrated task solutions rather than priors on model distributions. Jong and Stone [12] proposed a hierarchical decomposition of reward and transition models for all subtasks of a hierarchy and extended MAXQ with the optimistic exploration of the model-based R-MAX [13] algorithm. The resulting algorithm, fitted R-MAXQ, uses instance-based function approximation to generalize model parameters to unobserved state-action pairs and fitted value iteration [14] to learn policies for subtasks on continuous state and action spaces. Whereas the exploration strategy of R-MAX favors unknown state-action pairs, we argue in this work, that an exploration strategy which balances optimization of reward with risk from executing unknown actions is favorable in real-world applications.

Bai et al. [15] apply MAXQ to continuous state- and action spaces by avoiding an explicit representation of completion- or value functions and, instead, propose an online algorithm that recursively computes values of visited states according to the MAXQ decomposition. To render this process feasible, the proposed MAXQ-OP algorithm maintains goal state distributions for subtasks and approximates the completion function based on a sampled subset of the possible goal states of a subtask. The performance of the approach largely depends on the availability of domain knowledge to specify a prior distribution of goal states for subtasks. In contrast, our approach uses task demonstrations to convey expert knowledge. By using instance-based generalization of information from visited states, our approach also avoids precomputing exhaustive policies, similar to an online approach. However, in contrast to the ad-hoc computations performed by the approach of Bai et al., our system remembers observed instances to improve future decisions.

III. Proposed Method

In this work, we propose an integrated approach to hierarchical robot learning in complex domains, based on the MAXQ framework for hierarchical reinforcement learning. Our first contribution is the application of MAXQ to continuous state spaces using Gaussian Process (GP) approximations of completion- and value functions of composite and primitive tasks, respectively. From these, we compute estimates of state-action values by recursively aggregating GP estimates and associated uncertainties of subtask hierarchies. For reinforcement learning, we derive a Bayesian exploration criterion based on expected deviation that balances optimization of long-term rewards and potential risks to the robot from executing actions with unpredictable outcome for

efficient and safe learning. Secondly, we complement MAXQ with learning from demonstrations by explicitly allowing the proposed system to ask for human expert knowledge if the previously gathered experiences do not suffice to confidently propose an action. The decision for either way of learning is taken incrementally in every encountered situation and for all subtasks within the hierarchy. This leads to a coherent system where the same logic is applied to all subtasks on every level of abstraction. By leveraging the strengths of both, reinforcement learning and learning from demonstrations, our combined approach is able to solve tasks in large state spaces more quickly than either learning method alone and with little involvement on the human expert's side.

A. Continuous MAXQ Learning with Uncertainties

The MAXQ framework has been proposed by Dietterich [2] to learn recursively optimal hierarchical policies for (Semi-) Markov Decision Processes. Its key idea is to recursively decompose the value $V^\pi(i,s)$ of state s in subtask i given a fixed policy π into the sum of the value $V^\pi(a,s)$ of subtask a chosen according to $\pi_i(s)$ and the cumulative reward $C^\pi(i,s,a)$ for completing i after finishing a. In this context, $C^\pi(i,s,a)$ is called the *completion function* of subtask i in situation s for action a. Following the notation of [2], this can be expressed as follows:

$$Q^\pi(i,s,a) = V^\pi(a,s) + C^\pi(i,s,a),$$
$$V^\pi(i,s) = \begin{cases} Q(i,s,\pi_i(s)) & \text{if } i \text{ composite} \\ \sum_{s'} P(s'|s,i)R(s'|s,i) & \text{if } i \text{ primitive,} \end{cases} \tag{1}$$

where R is the single-step reward received if executing a primitive action i in state s leads to a transition to state s', and P is the probability of the transition to s'. One of the major differences between MAXQ and other methods for hierarchical reinforcement learning is that the value functions of subtasks in MAXQ are *context-free*, i.e., they do not depend on the context their subtask was invoked from. This allows to reuse subtask policies in different task contexts and to benefit from state abstraction.

Based on this decomposition, Dietterich et al. extended Q-Learning [16] to the MAXQ hierarchy and proved its converge to a recursively optimal policy. For every step of a primitive task, the value function of the task is updated using the experienced reward. Once a subtask is completed, the completion function of its parent is updated with the estimated value of the new state in the parent task. Computing this estimate involves recursively unfolding Eq. (1), replacing the fixed policy with a $\max$ operator that selects the best action in every step. To avoid the hierarchical credit assignment problem during learning, the designer of a MAXQ hierarchy may assign different kinds of rewards to subtasks on different levels.

Learning a hierarchical policy with the MAXQ framework requires to maintain the completion values of all composite subtasks and value functions for all primitive tasks. In continuous domains, explicitly representing these quantities for all states is infeasible. At the same time, in practical applications, a reinforcement learning agent will only ever visit a small part of the state and action spaces. A common technique is therefore to represent the value function using function approximation techniques. In this paper, we investigate this idea in the context of hierarchical reinforcement learning by applying GPR to the MAXQ learning algorithm.

GPR is a technique for non-parametric Bayesian regression which yields predictions with an associated uncertainty of function values at unseen inputs. For every subtask a on every level i, we propose to maintain a GP approximation of the completion function $C(i,s,a)$ or the value function $V(s)$ in the case of primitive actions. Estimates of completion values and values of primitive actions can then be obtained in the form of predictions based on a prior on the approximated function and a set of gathered training samples.

From these, we compute an estimate of the value $Q(i,s,a)$ by recursively aggregating estimates and their uncertainties throughout a subtask hierarchy, analogous to Eq. (1):

$$Q(i,s,a) \sim \mathcal{N}\begin{pmatrix} \mu_Q \\ \sigma_Q^2 \end{pmatrix} = \begin{pmatrix} \mu_{V(a,s)} + \mu_{C(i,s,a)} \\ \sigma^2_{V(a,s)} + \sigma^2_{C(i,s,a)} \end{pmatrix}. \tag{2}$$

Here, $V(a,s)$ decomposes recursively into the sum of normally distributed GP estimates for completion values and values of primitive actions, as described by Eq. (1). Due to the generalization abilities of the Gaussian Process approximation, the accuracy of estimates increases as the system gathers experience from interacting with the environment.

B. Expected Deviation

In practical applications, obtaining samples for reinforcement learning is often expensive. It is therefore important that learning algorithms exploit information efficiently and attain an effective exploration policy with a minimal number of trials. *Bayesian Reinforcement Learning* [17] allows to make an informed decision for a promising next action by considering the uncertainty associated with Bayesian estimates of achievable performance. While the evolution of total accumulated reward over time is a relevant measure for the theoretical analysis of algorithms and for performance comparisons, practical applications also need to consider safety when choosing actions. Greedy optimization of rewards may easily lead to the exploration of actions with fatal consequences for the robot or its environment.

In previous work, we have introduced an exploration strategy based on *expected deviation* [4]. It allows for safe, yet data-efficient learning by balancing the expected improvement [18] and degradation of candidate actions. The expected improvement $ED^\oplus$ equals the expected value of the predicted improvement in terms of value of a candidate action over the value Q_{best} of the best known action. Similarly, the expected degradation $ED^\ominus$ measures how much the value of a candidate action is expected to fall short of Q_{best}. Both quantities are computed directly from the distribution of the estimated state-action value $Q(x)$ of a candidate input. Incorporating a trade-off with the expected degradation into the optimization of the expected improvement fosters safety

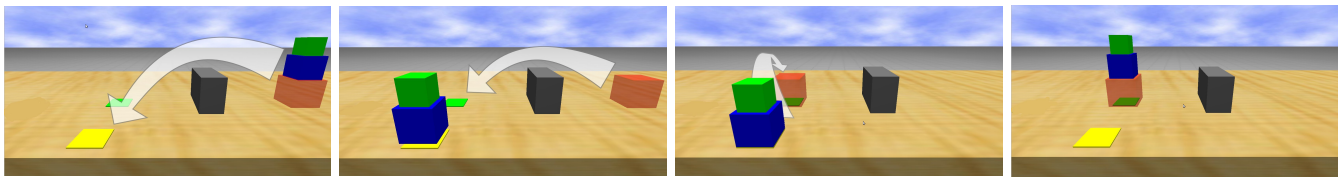

Fig. 2. Strategy the system has to learn in order to relocate a stack of three boxes without collapsing it. It involves splitting the stack by moving the top boxes to a temporary location (yellow). In a second step, the remaining stub may be moved to the target location (green) where it will be joined with the upper boxes in a third step. The task is complicated by a static obstacle (gray box) in the center of the workspace.

since it depreciates actions that seem promising because a large uncertainty in their value estimate leaves an opportunity for large improvement.

In this work, we apply Bayesian expected deviation learning to MAXQ hierarchical reinforcement learning. In MAXQ, state-action values are not represented explicitly, but stored implicitly as completion and primitive state values that are approximated by a Gaussian Process in our approach. From these, we recursively compute a probabilistic estimate of the value of candidate state-action pairs using Eq. (2) needed to compute the expected improvement and degradation. The resulting objective function on the expected deviation ED is defined as:

$$\mathbf{E}D(x) := \mathbf{E}D^{\oplus}(x, Q_{\text{best}}) - f(\mathbf{E}D^{\ominus}, Q_{\text{best}}) \quad (3)$$

$$f(\mathbf{E}D^{\ominus}, Q_{\text{best}}) = \Big(Q_{max} - \big(Q_{\text{best}} - \mathbf{E}D^{\ominus}(x, Q_{\text{best}})\big)\Big)^2,$$

where the maximum achievable value Q_{max}—which is usually known at design time—is used as a normalizer.

Depending on the structure of the space of actions, different strategies may be used to optimize this quantity. Note that optimization of the expected deviation does not involve sampling rewards from the environment. Rather, the value of actions is predicted efficiently using the GP approximation.

C. Learning From Demonstration

Informed exploration relies on the availability of prior experience to base decisions on. Therefore, reinforcement learning algorithms are often accelerated by combining them with learning from demonstrations (LfD) [19]. Furthermore, demonstrations are a convenient and familiar way of conveying knowledge for human teachers.

In this work, we apply learning from demonstrations to MAXQ by allowing our system to ask for human assistance for any subtask in the hierarchy. The human expert is then expected to demonstrate a solution for the particular subtask using the lower-level actions available to the subtask. The recorded action sequence is subsequently executed and rewards are collected for the visited state-action pairs. Once the subtask is completed, the completion values are updated in reverse order from the terminal state to the state where the demonstration was originally requested. This way, whenever an update is computed, it benefits from the preceding update of the value of its successor. Since subtasks in the MAXQ hierarchy are context-free and due to the generalization abilities of the GP approximation described in Sec. III-A, demonstrations naturally propagate to similar situations and different tasks sharing the same subpolicy.

D. Learning Method Selection

Learning from demonstrations and learning from experience gathered through interaction with the environment both offer unique strengths but also have weak points. Whereas demonstrations are an effective way of transferring existing expert knowledge and task constraints to robots without expressing them explicitly, they are often expensive to obtain. On the other hand, reinforcement learning allows to adapt existing solutions to improve their effectiveness and to apply them under changing conditions. It can be highly ineffective though, if no prior information is available. Combining both ways of finding task solutions in a flexible way therefore is a promising direction towards the development of more efficient learning methods. Accordingly, the system presented in this work integrates learning from imitation and reinforcement learning as complementary control flows. On all levels of the hierarchy, whenever an action needs to be taken, the system autonomously and independently for every subtask decides for the most appropriate way of finding a solution based on the situation at hand. This way, the human involvement is focused on areas where autonomous learning cannot be performed safely and at the same time the overall effort is reduced by allowing the system learn from autonomous interaction where appropriate.

The individual decisions for learning from demonstrations or reinforcement learning are based on previously gathered experiences that are generalized by GPR. If there is enough knowledge to generate a solution that does not entail the risk of damaging the robot by executing actions with utterly unknown outcome, reinforcement learning is chosen. If the available knowledge is insufficient to propose a solution to the task, e.g. because a similar situation was never observed before or all evaluated solutions turned out to be poor, our system will ask for a demonstration of the (sub-)task.

To determine whether a suitable and safe action can be generated for the current situation s_{curr}, we consider pairs (a, s) of previously taken actions that led to a positive reward and the situation they were taken in. Among them, we search for an optimal trade-off between value and similarity to the current situation s_{curr} by optimizing

$$\hat{x} = \underset{(s,a)\in X}{\operatorname{argmin}}\ \alpha\left(Q_{max} - \mu_Q(s,a)\right) + \|s_{\text{curr}} - s\|\,. \quad (4)$$

Here, $X \subseteq S \times A$ is the set of positive training samples and α expresses the relative preferences for high value or similarity to the current situation. If optimization yields an action promising good performance while being sufficiently similar

to previously executed actions, i.e., its score undercuts a predefined threshold θ, it is taken as a starting point for reinforcement learning. If no such action is found, our system asks for assistance from a human expert. The threshold determines the carefulness of the system. If θ is set to a small value, the system only attempts actions on its own if they exhibit little risk and draws on human expert demonstrations otherwise. Given a large θ, the system will lean on expert knowledge less often which in turn entails an increased risk of failing a task.

IV. Experiments

To evaluate the performance of our approach, we choose a simplified version of a typical manipulation task from our everyday life that involves choosing among manipulation strategies based on the situation at hand. The task is to relocate a stack of small boxes on a table safely to a designated position. On this task, we conducted several experiments using the physics-based simulator Gazebo [20]. They demonstrate that the proposed system successfully and safely learns to solve the task from various situations.

A. Task Description

In our experiments, each episode consists of a stack of boxes that is placed in the robot's workspace and a designated target position where the stack—or parts of it—have to be placed. To indicate what part of the stack needs to be moved, the lowest affected box is highlighted. At the start of each episode, the real-valued coordinates of the initial and target positions in the robot's workspace, as well as the total number of boxes on the stack and the number of boxes to be displaced are chosen at random. This allows the system to learn the task under various conditions. The task is complicated by a static obstacle placed at the center of the robot's workspace, preventing a displacement on a straight line between certain locations of the stack and some goal locations. On the other hand, actions are rewarded according to the work they consume—so the system benefits from pushing instead of lifting a box if possible. A second complication arises from the physical interactions of the stacked boxes modelled by our simulation, causing stacks to become instable and to collapse if more than two boxes are relocated at the same time. Here, the system has to learn to lift part of the stack off to a temporary location and to reassemble the partial stacks at the destination to limit the number of boxes that are moved at the same time. This strategy is depicted in Fig. 2.

B. Setup

In order to apply the proposed algorithm to this task, we define a MAXQ hierarchy with subtasks at three levels of abstraction (see Fig. 3).

1) Action spaces: On the topmost, most abstract level, there is only a single action that solves the entire task of moving a stack of boxes. To achieve this goal, the system has to learn to decide for a manipulation strategy and to combine intermediate level actions accordingly. On the intermediate level, there are two tasks for the top-level action to choose from, corresponding to the possible positions of a box in a stack: move a box that is located directly on the table and move a box that is located at a higher position in the stack. Every iteration involves moving a single box—including all boxes on top of it—to a designated target position. Sometimes, the whole top-level task may be completed with a single intermediate action that moves an entire stack. In other situations, several intermediate actions on partial stacks may need to be combined to solve the whole task. Besides learning to choose an appropriate action in every situation, the system has to select the source location of the box and a desired target location. Locations have to be chosen from a predefined set of *reference points*. Reference points represent task-specific locations independent from their coordinates in a particular instance of the task. They are predetermined by the designer of the MAXQ hierarchy. For our experiments, the initial location of the stack, the desired goal location of the stack, and the location of a temporary depository have been identified as relevant for the top-level task. For each of them, there are reference points at different heights corresponding to the positions within the stack.

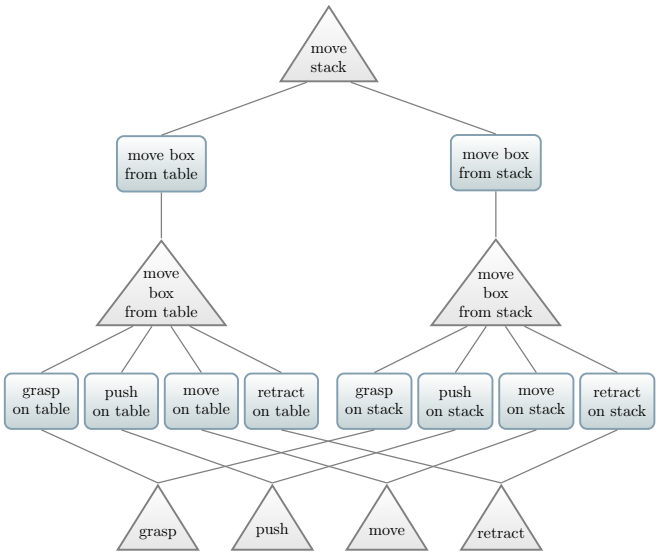

Fig. 3. MAXQ graph [2] illustrating the sample task used in our experiments. MAX nodes represent subtasks and are depicted as triangles, Q nodes correspond to actions available to subtasks and are displayed as rectangles. Every Q node maintains the value of completing its parent task after executing its child action.

In order to complete an intermediate task, there are four low-level motion primitives the system may combine: grasping an object, lifting an object to another position, pushing an object, and retracting the hand to a resting position. For any situation, the system has to choose the correct action as well as two reference point IDs determining the start and the goal of the desired movement. On the intermediate level, the set of available reference points includes the start and target locations determined by the action parameters from the top level and a predefined resting position the manipulator may assume between actions.

TABLE I

ACTION AND STATE REPRESENTATIONS IN OUR EXPERIMENTS

level	**action variables**	
top	action	$\in$ {move from stack, move from table}
	start	$\in$ {initial location, goal, depository} $\times$ 3
	goal	$\in$ {initial location, goal, depository} $\times$ 3
inter-mediate	action	$\in$ {grasp, push, move, retract}
	start	$\in$ {start, goal, resting position}
	goal	$\in$ {start, goal, resting position}
bottom	$-$	

level	**state variables**	
top	box location	$\in$ {initial location, goal, depository, absent} (per box)
	selected box	$\in \{0, \ldots,$ number of boxes$\}$
inter-mediate	box location	$\subset \mathbb{R}^2$
	goal	$\subset \mathbb{R}^2$
	holding object	$\in \{0, 1\}$
	current location	$\in$ {start, goal, resting position} $\times$ 3
bottom	movement start	$\subset \mathbb{R}^2$
	movement goal	$\subset \mathbb{R}^2$

Actions on the lowest level of the MAXQ hierarchy correspond to movement primitives that are treated as atomic actions in our experiments. For the purpose of our experiments, the parameters of the movement primitives were trained beforehand using the approach described in our earlier work [21]. Given the ID of a movement primitive and the required reference point IDs, our system generates a trajectory using a controller that computes the effects of the movement and updates the simulated environment.

Tab. I summarizes the variables describing the action spaces on the different levels of the task hierarchy. Throughout our experiments, we assume that the effects of actions are deterministic.

2) State spaces: Using a MAXQ decomposition of our task and its value function allows us to benefit from state abstraction by factoring the state space in order to work with smaller subspaces on different levels of the task hierarchy. This reduces the complexity of the individual subtasks and accelerates learning. On the top level, a state encompasses an ID for each box indicating whether the box is located at the random initial position of the stack, the random target location, the temporary depository, or whether it is located outside the workspace. A further ID value designates the height of the box that should be displaced. On the intermediate level, the state space is reduced to contain only the information relevant for the subtask on this level, i.e. parameters referring to the box to be displaced. These are the current coordinates of the box to be displaced, the coordinates of the goal position, whether the robot is currently holding the object and a reference point ID indicating the current position of the manipulator. Although we are not concerned with learning motion primitive parameters in this work, rewards received on the lowest level are still represented with a distinct GP model per motion primitive so appropriate actions can be selected on higher levels. A situation on this level is defined by the start and goal coordinates of executed primitives. The state variables on all levels are summarized in Tab. I.

3) Rewards: In the proposed approach, rewards are assigned individually to subtasks on different levels of a hierarchical task. This allows us to reward or penalize specific subtasks without influencing their parents that may not have contributed to the subtask's performance. This is also referred to as the *hierarchical credit assignment problem* [2].

To assess the performance of a low-level task, we measure the amount of physical work involved in executing a motion primitive of length T for a given pair of start and goal parameters. It is defined as:

$$W = \sum_{t=1}^{T} |E_k(t) - E_k(t-1)|\,; \quad E_k(t) = \frac{1}{2} m v^2(t), \quad (5)$$

where E_k is the kinetic energy, computed from the mass m and the velocity v at time t of the moved objects.

On the intermediate level, we detect whether there was a collision between the manipulated stack and other elements of the world. Action sequences leading to a collision are penalized with a negative reward. Instead, if all unconcerned objects are still at their original position and no collision is detected, a reward of zero is assigned. On the top level, we consider the degree to which the original task was completed to assign a reward. If the task was completed without altering the environment in unintended ways, a positive reward is assigned. If the environment was altered while executing the task, the chosen action is penalized with a negative reward. In all other cases, a neutral reward of zero is assigned and further iterations are taken until the task terminates. In summary, rewards are assigned according to

$$R(a,s) = \begin{cases} & \underline{\text{TOP LEVEL}} \\ -1 & \text{terminated with side effects,} \\ 1 & \text{success,} \\ 0 & \text{else,} \\ & \underline{\text{INTERMEDIATE LEVEL}} \\ -1 & \text{terminated with collision or side effects,} \\ 0 & \text{else,} \\ & \underline{\text{BOTTOM LEVEL}} \\ -W & \text{every action.} \end{cases}$$

Rather than approximating the Q function with a single GP using a fixed kernel width, we model each of the possible kinds of rewards with a separate GP and combine the predictions in a Gaussian mixture model. This allows us to assign different weights to the components. Throughout the experiments described in this section, we set the kernel widths to 0.2 on the top level and to 0.35 on the lower levels, respectively, for the successful and 0.055 for the unsuccessful cases. Hence, unsuccessful cases have a more local influence. The threshold θ used to make a decision between reinforcement and imitation learning was set to θ=0.3 on the top level and 0.25 on the lower levels, respectively.

V. Results

To demonstrate how the proposed approach benefits from both, the hierarchical structure of the task and the combination of reinforcement learning with learning from demonstrations, we ran 300 random episodes of the task.

Fig. 4 depicts the sequence of decisions for a learning method made by the system over the course of the random episodes. It clearly shows that the system successfully learned the task from as few as 18 demonstrations across all levels. On the top level, human assistance was requested in a total of six episodes, corresponding to each of the possible configurations of the stack. All requests for a demonstration were made during the first 16 episodes. Three of six were even made during the first three episodes of our experiment. This is in line with the expectation that the system does not perform random exploration if no prior knowledge is available and on the other hand no longer requests demonstrations if sufficient experiences have been gathered. Compared to approaches that require the teacher to bootstrap the learning process with a number of carefully selected training samples, the proposed system requests demonstrations on-demand whenever an unknown situation is encountered. Bootstrapping therefore is not necessary in our approach although it can be applied to incorporate available domain knowledge. The occasional late demonstrations are accounted for by the random generation of task parameters that sometimes produce novel task configurations late in the process. For instance, in episode 16 of our experiments, the task of displacing the whole stack with three boxes was drawn for the first time and led our system to request a demonstration. A similar effect can be observed on the intermediate level where five of seven demonstrations of moving a box that is located directly on the table are requested during the first 27 episodes. Still, the final demonstration is not requested until episode 66 where a box has to be put on top of another box. Again, no similar configuration in this corner of the workspace had been observed before. The ability to incorporate demonstrations late in the process is advantageous in case the conditions of a task change and additional knowledge is needed to solve the task. In this respect, approaches that merely use demonstrations to obtain an initial policy for reinforcement learning are limited compared to our approach. On the intermediate level, the task of moving a box that is located on top of another box is simpler than the task of moving a box located directly on the table since for the latter, the system needs to consider the obstacle on the table in its decision for a learning method. Consequently, only five demonstrations are required throughout the first 29 episodes in order to learn this task.

Looking at all three plots side by side reveals that the requests for demonstrations were often made at different episodes on each level, i.e. a lack of information to generate a safe solution on one level does not necessarily imply that subtasks on lower levels also cannot be solved without a demonstration and vice versa. Taking the decision for a learning method individually on each level therefore saves human effort for unnecessarily complex demonstrations. For instance, a demonstration is requested for moving the green box that is located on top of two other boxes. The demonstration in this case involves choosing an appropriate subtask and its parameters—in this case "move from stack". The system is then able to apply the primitive actions involved in this subtask autonomously, drawing on experiences from past episodes and different task configurations. The same is true for the strategy needed to displace an entire stack of three boxes, depicted in Fig. 2. Again, the strategy on the top level has to be demonstrated to the system but the second intermediate level action can be executed autonomously. This is facilitated by the subtask sharing and generalization abilities our system permitting reuse of acquired knowledge in different contexts, independent of the parent task. In the same way, both subtasks on the intermediate level draw on the same four primitive actions. There are also instances where a top level task may be completed without assistance but a demonstration is needed to solve a subtask on a lower level. For example, in episode five, a demonstration is requested on the intermediate level because little information is available for that part of the continuous state space of that subtask, while the parent task is solved autonomously.

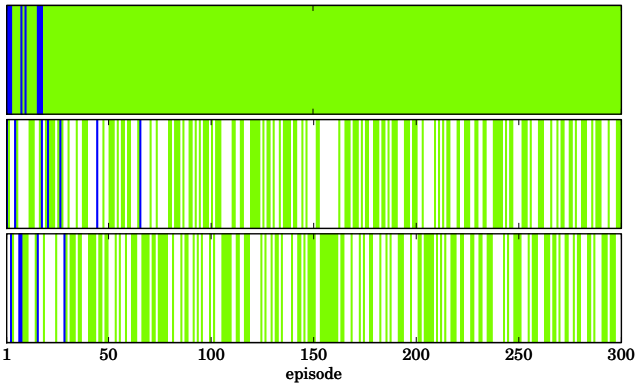

Fig. 4. Choices made by our system during an experiment with 300 episodes of the full task with up to three boxes. Each bar depicts the choices made for a particular subtask and every episode is represented by a segment. The top bar represents decisions made for the top-level task, the lower bars represent the alternate tasks on the intermediate level. Consequently each segment is marked in only one of them. Blue (dark) segments denote episodes where the system asked for a demonstration. Episodes where the system selected an action autonomously are marked in green (light).

In summary, our experiments confirm that the system quickly learns to apply different strategies depending on the number of boxes in a stack and to split and reassemble large stacks to prevent them from collapsing. Similarly, it quickly apprehends that moving objects across the obstacle requires lifting them over whereas pushing objects is the more energy efficient solution in all other cases where the object is located directly on the table. While the system initially clings to the demonstrated solutions, it starts to apply solutions from different contexts as more samples become available and the certainty about the value of the demonstrated solutions increases. It does not, however, attempt arbitrary actions with unknown outcome but instead recombines previously

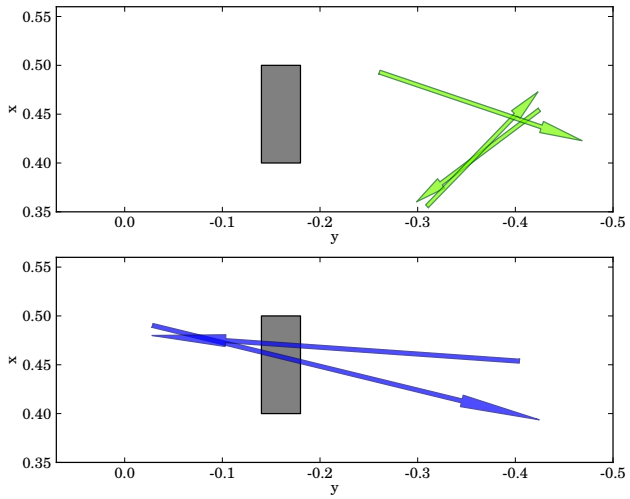

Fig. 5. Randomly generated tasks where the system chose suboptimal actions for the "move box from table" subtask. The top graph depicts settings where a box was lifted to a destination although simply pushing it would have been the more efficient solution. Arrows in the bottom graph represent settings where the system caused a collision by trying to push the object.

seen actions. For instance, objects are unnecessarily lifted in episodes 22, 24 and 51. In episodes 200 and 221, the system even tries to push a box even though the obstacle obstructs the way as depicted in Fig. 5. Collisions result in punishing rewards and are abandoned in favor of the correct and energy-efficient solution quickly. Collisions were only attempted once for every direction across the obstacle.

Throughout our experiments, the instances mentioned above were the only instances where the system did not correctly solve a subtask. On the top level, there was no failure at all since the safety term in our exploration strategy prevented the system from choosing actions it hadn't observed before.

VI. Conclusion

In this paper, we presented a new system for hierarchical robot learning that combines reinforcement learning and learning from demonstrations to teach a robot complex tasks from human everyday life. Our method applies the well-known MAXQ method for hierarchical reinforcement learning to continuous state spaces by approximating completion values on compound levels and values of primitive states for all subtasks with GPs. This allows us to compute predictions of the value of unknown state-action pairs with an associated uncertainty. From these, we derive a Bayesian exploration criterion that safely optimizes the value of actions by trading off the expected improvement and degradation of GP estimates, thereby protecting the robot from actions with unpredictable outcome. To reduce the number of trials needed to find a good policy, we complement our reinforcement learning approach with learning from human expert demonstrations. Rather than limiting the use of demonstrations to the initialization of reinforcement learning, we integrate both ways of learning as alternate control flows in our system. For every situation and subtask, the system autonomously decides for either of them based on previously gathered experiences. This way, we efficiently exploit the complementary strengths of both approaches without inheriting their individual shortcomings. We evaluated our approach on a challenging box-stacking task involving a hierarchy of subtasks and different manipulation strategies that need to be chosen appropriately. Our results demonstrate that the proposed system is able to successfully learn the task and benefits from its hierarchical structure and the independent decision for a learning method for every subtask. We would like to dedicate future work to the development of a coherent system with the ability to simultaneously acquire low-level motion primitives and complex task skills on a real robot.

References

[1] B. Hengst, "Hierarchical Approaches," in *Reinforcement Learning: State of the Art*. Springer Berlin Heidelberg, 2012, ch. 9, pp. 293–323.

[2] T. G. Dietterich, "Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition," *Journal of Artificial Intelligence Research*, vol. 13, pp. 227–303, 2000.

[3] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[4] K. Gräve, J. Stückler, and S. Behnke, "Improving Imitated Grasping Motions through Interactive Expected Deviation Learning," in *Proc. IEEE-RAS Int. Conf. on Humanoid Robots*, 2010, pp. 397–404.

[5] J. Kober, D. Bagnell, and J. Peters, "Reinforcement Learning in Robotics: A Survey," *Int. Journal of Robotics Research*, 2013.

[6] C. Daniel, G. Neumann, O. Kroemer, and J. Peters, "Learning Sequential Motor Tasks," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2013.

[7] J. Peters, K. Muelling, and Y. Altun, "Relative Entropy Policy Search," in *Proc. Nat. Conf. on Artificial Intelligence*, 2010.

[8] F. Stulp and S. Schaal, "Hierarchical Reinforcement Learning with Movement Primitives," in *Proc. IEEE-RAS Int. Conf. on Humanoid Robots*, 2011, pp. 231–238.

[9] E. Theodorou, J. Buchli, and S. Schaal, "A Generalized Path Integral Control Approach to Reinforcement Learning," *Journal of Machine Learning Research*, vol. 11, pp. 3137–3181, Dec. 2010.

[10] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot Learning from Demonstration by Constructing Skill Trees," *Int. Journal of Robotics Research*, vol. 31, no. 3, pp. 360–375, 2012.

[11] F. Cao and S. Ray, "Bayesian Hierarchical Reinforcement Learning," in *Proc. Neural Information Processing Systems*, 2012, pp. 73–81.

[12] N. K. Jong and P. Stone, "Compositional Models for Reinforcement Learning," in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2009, pp. 644–659.

[13] R. I. Brafman and M. Tennenholtz, "R-MAX - A General Polynomial Time Algorithm for Near-Optimal Reinforcement Learning," *The Journal of Machine Learning Research*, vol. 3, pp. 213–231, 2003.

[14] G. J. Gordon, "Stable Function Approximation in Dynamic Programming," Carnegie-Mellon University, Pittsburgh, PA, Tech. Rep., 1995.

[15] A. Bai, F. Wu, and X. Chen, "Online Planning for Large MDPs with MAXQ Decomposition," in *Proc. Int. Conf. on Autonomous Agents and Multiagent Systems*, vol. 3, Richland, SC, 2012, pp. 1215–1216.

[16] C. Watkins and P. Dayan, "Q-Learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[17] N. Vlassis, M. Ghavamzadeh, S. Mannor, and P. Poupart, "Bayesian Reinforcement Learning," in *Reinforcement Learning*. Springer Berlin Heidelberg, 2012, ch. 11, pp. 359–386.

[18] D. R. Jones, "A Taxonomy of Global Optimization Methods Based on Response Surfaces," *Journal of Global Optimization*, vol. 21, no. 4, pp. 345–383, 2001.

[19] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A Survey of Robot Learning from Demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[20] N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, vol. 3, 2004, pp. 2149–2154.

[21] K. Gräve and S. Behnke, "Incremental Action Recognition and Generalizing Motion Generation based on Goal-Directed Features," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012, pp. 751–757.