J Intell Robot Syst (2010) 60:217–239
DOI 10.1007/s10846-010-9422-y

A Human-Robot Collaborative Reinforcement Learning Algorithm

Uri Kartoun · Helman Stern · Yael Edan

Received: 29 November 2009 / Accepted: 1 April 2010 / Published online: 5 May 2010

Abstract This paper presents a new reinforcement learning algorithm that enables collaborative learning between a robot and a human. The algorithm which is based on the $Q(\lambda)$ approach expedites the learning process by taking advantage of human intelligence and expertise. The algorithm denoted as $CQ(\lambda)$ provides the robot with self awareness to adaptively switch its collaboration level from autonomous (self performing, the robot decides which actions to take, according to its learning function) to semi-autonomous (a human advisor guides the robot and the robot combines this knowledge into its learning function). This awareness is represented by a self test of its learning performance. The approach of variable autonomy is demonstrated and evaluated using a fixed-arm robot for finding the optimal shaking policy to empty the contents of a plastic bag. A comparison between the $CQ(\lambda)$ and

This work was partially supported by the Paul Ivanier Center for Robotics Research and Production Management, and by the Rabbi W. Gunther Plaut Chair in Manufacturing Engineering, Ben-Gurion University of the Negev.

We would like to thank Dr. Amir Shapiro from the Department of Mechanical Engineering for his suggestions, and comments. We would also like to thank Dr. Sigal Berman and Mr. Amit Gil from the Department of Industrial Engineering and Management for their critical reading of the text.

U. Kartoun (✉)
Microsoft Medical Medial Lab (M3L), Microsoft, Washington DC, USA
e-mail: ukartoun@microsoft.com

H. Stern · Y. Edan
Department of Industrial Engineering and Management, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel

H. Stern
e-mail: helman@bgu.ac.il

Y. Edan
e-mail: yael@bgu.ac.il

the traditional $Q(\lambda)$-reinforcement learning algorithm, resulted in faster convergence for the $CQ(\lambda)$ collaborative reinforcement learning algorithm.

Keywords Robot learning · Reinforcement learning · Human-robot collaboration

1 Introduction

To expand the use of robots in everyday tasks they must be able to perform in unpredictable and continuously changing environments. Since it is impossible to model all environments and task conditions in a rigorous enough manner, robots must learn independently how to respond to the world and how the world responds to actions they take.

One approach to robot learning is reinforcement learning (RL) [1–21]. In RL the robot receives positive/negative rewards from the environment indicating how well it is performing a task. The robot's learning goal is to optimize system responses by maximizing a reward function. This is achieved through gaining experience and through direct interaction with the robot's environment. Under uncertainty, the robot may fail to make the correct associations between the observed states and chosen actions that lead to higher rewards. Moreover, certain problems are often too memory intensive to store the values for each state. Another disadvantage is that RL-based approaches require substantial interaction with the environment to test large numbers of state-action values until an effective policy is determined. One approach to overcome this problem is to avoid storing all state-action pairs, and instead to compute them dynamically as the need arises [22]. A major disadvantage of RL concerns its slow convergence toward satisfactory solutions. This research addresses this problem by developing a collaborative RL algorithm to speed up the learning process.

Previous research indicates that collaboration of a robot with a human is essential to minimize the amount of time required by a robot to accomplish a learning task (e.g., [9, 22–30]). Thomaz and Breazeal [31, 32] describe a new RL-based approach for providing reward signals by a human which depend on both past actions and future rewards. Two ways of providing rewards were analyzed to determine whether people prefer to communicate feedback about particular aspects of a state rather than an entire world state: (1) rewarding a whole state of the world and (2) rewarding a state of a particular object. Results achieved indicate that, in general, people use the reward signal not only to provide feedback about past actions, but also to provide future directed rewards to guide subsequent actions. Lockerd and Breazeal [33] and Breazeal and Thomaz [34] describe a collaborative process enabling a robotic learner to acquire concepts and skills from human examples. During teaching, the robot is required to perform tasks based on human instructions. Convergence is provided through online feedback.

A Confidence-Based Autonomy (CBA) approach which enables an agent to learn a policy through interaction with a human teacher is presented in [29]. CBA consists of two components which take advantage of the complementary abilities of humans and computer agents. The first component, Confident Execution, enables the agent to learn a policy based on demonstrations obtained by regulating its autonomy and requesting help from the teacher. Demonstrations are selected based on

automatically calculated classification confidence thresholds. The second component, Corrective Demonstration, enables the teacher to improve the learned policy by correcting mistakes made by the agent through supplementary demonstrations. The approach was evaluated in a complex simulated driving domain [35] and results show that using Confident Execution the agent requires fewer demonstrations to learn the policy than when demonstrations are selected by a human teacher. For calculating the confidence threshold, two methods are considered: (1) single fixed threshold, and (2) multiple adjustable thresholds (see also [30]). A single fixed confidence threshold value provides a simple mechanism to approximate the high confidence regions of the state-space. However, it causes the agent to request too many demonstrations about things it already knows, and too few demonstrations about unlearned behaviors [29]. A multi-threshold approach in which classification of new points is performed by first selecting the action class with the highest confidence for the query enables one to decide between human demonstration and autonomy. The CBA algorithm was proven highly effective [29] in learning a variety of single-robot [21, 36] and multiple-robot [37] tasks.

The RL-based learning algorithm Q-learning [38], and its variation $Q(\lambda)$ [39], an incremental multi-step Q-learning algorithm that combines one-step Q-learning with eligibility traces, have been used in many robotic applications (e.g., [1, 4, 5, 40]). The learning algorithms Q and $Q(\lambda)$ are not capable of accepting human intervention, and as such the agent is placed in an unknown environment and explores it independently with the objective of finding an optimal policy. The major drawback of these approaches is the large amount of interaction required between the robot and the environment until an effective policy is determined.

To accelerate learning of the $Q(\lambda)$ algorithm, this paper presents a collaborative $Q(\lambda)$ RL algorithm, denoted as $CQ(\lambda)$. The collaborative algorithm integrates the experience of a robot and a human. The contributions of the paper are two-fold: (1) the paper introduces a threshold-based method for the agent (robot) to request advice from a human advisor (HA) and (2) the paper applies this approach in a bag-shaking task, in which a robot must learn how to shake a bag so as to release a knot tying it and thus release the objects within. Two main methods for a human to intervene in the learning process are considered for the shaking task. The first is to provide the reward and the second is to provide guidance on which policy to prefer. It was decided to choose the second method as a more direct way to help the robot learn the best policy. Using "reward intervention" would be possible for the HA, but for the specific task, a more accurate method was available which is automated by using a digital scale as a sensor to count the items that are extracted from the bag. An additional novelty of this work is how the collaboration with a robot is addressed, allowing a human to directly manipulate the Q values through a linguistic-based interface.

Unlike [2, 3], where the rewards are controlled by a human, or as described in [41], where user commands are employed for modifying the robot's reward function, in $CQ(\lambda)$ the human provides guidance to the robot to perform an improved policy. Collaboration here is similar to a learning application as described in [31], where human reward signals can be treated as an "interactive rewards interface" in which humans can give rewards to a learning agent by rewarding a whole world state. In [42], a trainer is asked to intervene and suggest help when two extreme Q-values are sufficiently close. This is done by examining the minimum and maximum values and

comparing the result to a pre-defined width parameter. In lieu of examining specific Q-values, the intervention in $CQ(\lambda)$ is based on examining the performance history of the agent over a pre-defined number of learning episodes and comparing it to a performance threshold. In [29], the evaluation of the level of confidence is made for each state of an episode. This method may not be applicable to tasks that require fast state-action transitions such as the bag-shaking task described in this paper. For such a task, it is not possible to stop the robot arm in a particular state and let the human intervene. Instead, the robot must perform a set of state-action transitions to shake the contents from the bag. At the end of such a learning episode a reward is given, and the decision whether to ask for human advice is determined.

The algorithm was tested with a fixed-arm six degrees of freedom robot manipulating a bag containing objects as a case study. The learning task is to observe the position of the bag located on the platform, grasp it, and learn how to shake out its contents in minimum time by interacting with the environment and by using knowledge acquired from a HA. As opposed to previous research in which human-robot collaboration was included by changing the rewards, in this algorithm the reward function is automatically updated when a falling object event is detected by a digital scale placed under the robot. By taking this approach, the robot is endowed with enough self awareness to detect when the system learning performance is low, at which time the human is asked to intervene. The HA provides new guidance through a linguistic interface which implicitly changes the Q values of the RL algorithm. For a robot to empty the bag, one side of its gripper must slide under the bottom of the bag, grasp it and lift it to a "shake starting position" vertically over the table. The bag is held upright by the robot from its bottom so that its open part, now closed by a knot, is facing down. Several assumptions are made: (1) a plastic bag has already been recognized; (2) the bag has been grasped and moved to its shaking starting position over the table and (3) the knot on the bag is such that it can be opened by shaking.

Most previous work on knot problems refers mainly to tying and untying tasks that involve strings or cords (e.g., [43, 44]) or manipulating deformable objects (e.g., [45, 46]). The task described in this paper is different in that the robot manipulator does not grasp a string or rope, but instead grasps a plastic bag, which is closed by tying the plastic handles of the bag into a knot. The robot manipulator tries to open the bag by shaking loose the knot in lieu of grasping the knot itself.

The paper is organized as follows. After describing the Q and $Q(\lambda)$ algorithm fundamentals in Section 2, the $CQ(\lambda)$ approach is detailed in Section 3. The $CQ(\lambda)$ implementation is presented in Section 4. Section 5 presents the results of applying the $CQ(\lambda)$-learning algorithm on the practical system and comparing its performance to $Q(\lambda)$-learning. Discussion and conclusions are provided in Sections 6 and 7, respectively, including possible future directions of research.

2 Q and $Q(\lambda)$- Learning

The basic assumption in Markov Decision Processes [47] is that any state s_{t+1} occupied by an agent is a function only of its last state and action: $s_{t+1} = f(s_t, a_t)$ where $s_t \in S$ and $a_t \in A$ are the state and action, respectively, at time step t [48]. In Q-learning, an algorithm specific to Markov systems, the system estimates the optimal action-value function directly and then uses it to derive a control policy using

the local greedy strategy [38]. It is stated in [5] "Q-learning can learn a policy without any prior knowledge of the reward structure or a transition model." Q-learning is thus referred to as a "model-free approach" where Q values can be calculated directly from the elementary rewards observed. Q is the system's estimate of the optimal action-value function [49]. It is based on the action value measurement $Q(s_t, a_t)$, defined in (1).

$$Q(s_t, a_t) = E\left[r(s_t, a_t) + \gamma V^*(s_{t+1})\right] = \\ = r(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} P(s_{t+1} | s_t, a_t) V^*(s_{t+1}) \quad (1)$$

where $V^*(s_{t+1})$ is the optimal expected reward and the γ parameter is the discount rate that describes how farsighted the agent is; small values of γ (e.g., close to zero) make the agent give immediate events higher significance. Equation 1 represents the expected discounted reward for taking action a_t when visiting state s_t, and following an optimal policy thereafter. From this definition and as a consequence of Bellman's optimality principle [50], Eq. 2 is derived.

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} P(s_{t+1} | s_t, a_t) \max_a Q(s_{t+1}, a_t) \quad (2)$$

The first step of the algorithm is to initialize the system's action-value function, Q. Since no prior knowledge is available, the initial values can be arbitrary (e.g., uniformly zero). Next, at each time step t, the agent visits state $s_t \in S$ and selects an action $a_t \in A$. Then it receives from the process the reinforcement $r(s_t, a_t) \in R$ and observes the next state s_{t+1}. The procedure continues by updating the action value $Q(s_t, a_t)$ according to Eq. 3 which describes a Q-learning one step.

$$Q_{t+1}(s_t, a_t) = (1 - \alpha) Q_t(s_t, a_t) + \alpha \left[r(s_t, a_t) + \gamma \hat{V}_t(s_{t+1})\right] \quad (3)$$

where $\hat{V}_t(s_{t+1}) = \max_{a \in A} \left[Q_t(s_{t+1}, a)\right]$ is the current estimate of the optimal expected reward $V^*(s_{t+1})$ and α is the learning rate which controls how much weight is given to the immediate reward, as opposed to the old Q estimate. The greater α, the more the state-action value tends toward new information. High values of α make learning faster, but end up receiving slightly lower rewards. The process repeats until a stopping criterion is met. The greedy action $\arg\max_{a_t \in A} \left[Q_t(s_{t+1}, a_t)\right]$ is the best the agent performs when in state s_{t+1}. For the initial stages of the learning process, however, actions are chosen randomly to encourage exploration of the environment. Under some reasonable conditions (the rewards are bounded, the learning rate is in the range of zero to one and decays to zero while summing to infinity) [51], by iteratively applying Eq. 3, convergence to the optimal value function is guaranteed [49].

A generalization of Q-learning, represented by $Q(\lambda)$ [39] uses eligibility traces, $e(s_t, a_t)$: the one step Q-learning is a particular case with $\lambda = 0$ [52]. The Q-learning algorithm learns quite slowly because only one time step is traced for each action [41]. To boost learning, a multi-step tracing mechanism, the eligibility trace, is used in which the Q values of a sequence of actions are updated simultaneously according to the respective lengths of the eligibility traces [1]. λ represents the eligibility decay rate. The greater λ is, the longer the sequence of values of state-action pairs updated.

Although the convergence of $Q(\lambda)$ is not assured for $\lambda > 0$, experience shows that learning is faster [52]. Several action selection policies are described in the literature for RL where the greedy policy (e.g., [53, 54]) is always to choose the best action. Other policies (e.g., "softmax" [6] or "ε-greedy" [55]) are stochastic and based on choosing a suboptimal policy to explore the state-action space.

3 *CQ*(λ)-Learning for Human-Robot Systems

In $CQ(\lambda)$ two collaboration levels are defined: (1) autonomous—the robot decides which actions to take, i.e., the robot updates its state-action values according to the standard $Q(\lambda)$ learning algorithm, and (2) semi-autonomous—the robot requests collaboration with the human advisor (HA). The HA then suggests policies and the robot combines this knowledge and updates its state-action values. Human-robot collaboration is unnecessary as long as the robot learns policies autonomously and adapts to new states. The HA is required to intervene and suggest alternative policies or parameters, if the robot determines that its learning performance is low. At this point the robot switches its learning level from autonomous (self performing) to semi-autonomous (acquiring human knowledge).

To decide if human assistance should be requested, a moving average learning performance measure, L_{ave}, a scalar in the range [0, 1], is calculated over the last N most recent learning episodes (Eq. 4).

$$L_{ave} = \left(\sum_{i=n-N}^{n-1} (S_i) \right) \Big/ N \tag{4}$$

where n is the current learning episode, $i = n - N, n - N + 1, n - N + 2, \ldots n - 1$. A binary value S_i indicates whether a policy was successful for the ith episode or not. The threshold[1] for a successful episode is defined as $\overline{R} = 20$.

$$S_i = \begin{cases} 1 \; if \;\; R_i > \overline{R} \\ 0 \qquad else \end{cases} \tag{5}$$

where R_i is the reward achieved for the ith learning episode. In Eq. 6 Λ is defined as a minimum acceptable performance threshold, which is compared to the average performance, L_{ave}, calculated[2] over the N most recent learning episodes. If the robot performance falls below the threshold, the robot switches between fully autonomous operation and semi-autonomous operation and requests human intervention.

$$L_{ave} = \left(\sum_{i=n-N}^{n-1} (S_i) \right) \Big/ N < \Lambda : \quad 0 \leq L_{ave} \leq 1 \tag{6}$$

Although the performance of the robot could have been evaluated in terms of the criterion to be optimized, the expected total reward, it was decided to let it be

[1] The threshold value for a successful episode was determined empirically in experiments but may be automatically determined by a method described in Section 6.

[2] N was set to five in the experiments and is based on the variability of the process. One can use exponential smoothing in lieu of an average with an adaptive smoothing constant.

evaluated by averaging the success of its last N most recent learning episodes. This was done since it is more intuitive for the human to view this data in average success rate (or percentage), rather than presenting this information in terms of the expected total reward which may not be clear to the human.

The robot learning agent starts performing a learning episode (a task) in a physical environment that contains a real robot and objects (Fig. 1). At the end of the learning episode the robot agent receives a numerical reward. This reward indicates how well the robot performed the task, i.e., the higher it is, the more efficient the current policy is. In addition to this indication, the reward value determines if the learning episode is considered to be successful. Although we consider here learning in a Markov Decision Process where we explicitly give a reward function, an alternative method shall be considered in future experiments such as in [35]. The inverse reinforcement learning method is based on observing an expert demonstrating the task and is expected to be useful in complex learning tasks where it is difficult to manually specify an explicit reward function (e.g., driving a car). Experiments conducted in a grid-world and car driving simulations showed that using the method guarantees that the policy found will have performance comparable to or better than that of an expert.

The robot measures its learning performance, in a moving average manner, by averaging the success of its last N most recent learning episodes and comparing this

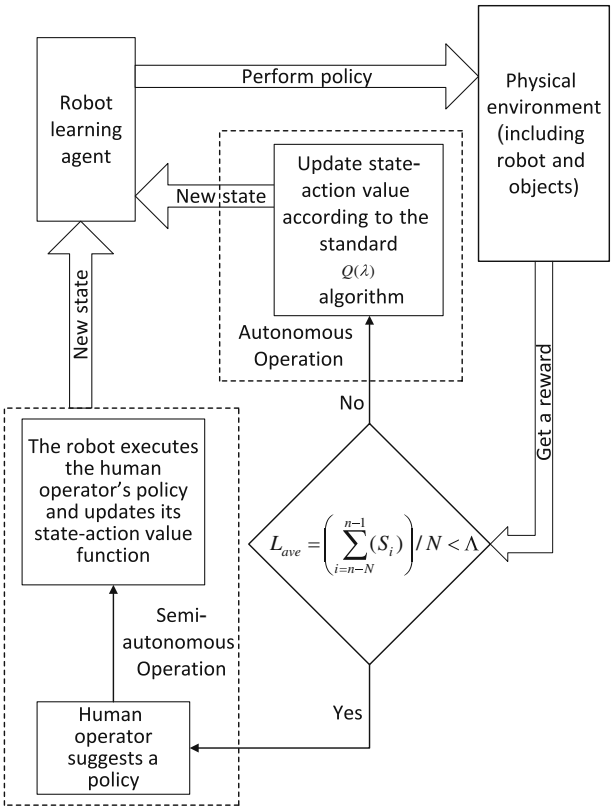


Fig. 1 Flowchart for robot and a human advisor $CQ(\lambda)$-learning

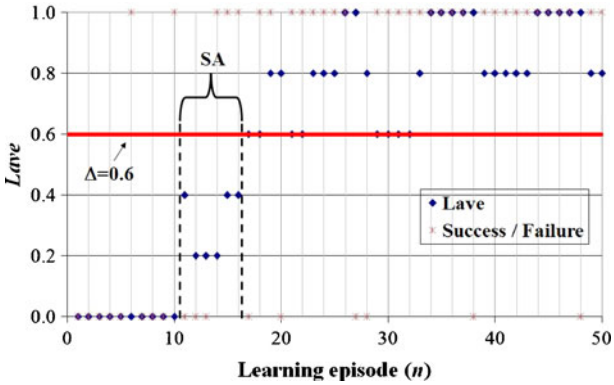


Fig. 2 The moving average learning performance measure, L_{ave} during $CQ(\lambda)$-learning. *SA* represents a semi-autonomous mode

value, L_{ave}, to Λ. At this stage, if performance is considered to be high, the robot will perform the next learning episode autonomously. Otherwise, human intervention and guidance is triggered. According to the human advice, the robot updates its Q table. The procedure is repeated M times where M (set a priori) is the maximal number of learning episodes.

An example of variable system autonomy is shown in Fig. 2 and consists of two system modes: (1) full autonomy (A), and (2) semi-autonomy (SA)—acquiring human guidance. In the example, N was set to five, and the acceptable learning performance threshold was set to $\Lambda = 0.6$; where L_{ave} greater than Λ indicates that the robot's learning performance is high, otherwise, human intervention is allowed, i.e., semi-autonomous learning is performed. The pseudo code for $CQ(\lambda)$-learning for a human and a robot is given in Fig. 3.

4 Bag Shaking Experiment with a Fixed-Arm Robot

4.1 Overview

The performance of the $CQ(\lambda)$-learning algorithm was tested in a bag shaking task. The objective is to find the best shaking trajectory of the robot to discharge the contents of the bag. It is assumed that a plastic bag containing a number of identical objects with known weights (such as screws) is placed on a table with its opening initially closed by the knot. For a robot to empty the contents of a bag, one side of its gripper must slide under the bottom of the bag (the part opposite the knot, grasp it and lift it to a "shake starting position" vertically over the table. It is assumed that the type of bag and the location of the opening are known.

The type of bag grasp operation was determined by automating bag classification, achieved using support vector machines (SVMs) [56]. By finding a set of optimal features representing a bag, a classification rate of 96.25% was obtained for a polynomial kernel of degree nine. In this paper it is assumed that a plastic bag has already been recognized, and the $CQ(\lambda)$ algorithm performance is based on shaking out all its contents.

Fig. 3 $CQ(\lambda)$-learning pseudo code for a human and a robot

Initialize $L_{ave} = 0$, $Q(s,a) = 0$ and eligibility trace $e(s,a) = 0$ for matrices of size $|S| x |A|$ for the robot learning process.
Set $n = 1$ for the first learning episode.
Repeat (for each learning episode):
If $L_{ave} < \Lambda$, let the system be in a *Semi-Autonomous* mode,
Else, let the system be in an *Autonomous* mode.
Set learning process to initial state s_t and pick initial action a_t.
Repeat (for each step of episode):
Take action a_t, observe reward r_t and the next state s_{t+1}.
If the system is in *Semi-Autonomous* mode: utilize HA's linguistic suggestions to change the $Q(s,a)$ values,
Else, choose a_{t+1} for s_{t+1} using a greedy action selection method (or ε-greedy).

$$a^*_{t+1} \leftarrow \arg\max_{a_t} Q(s_{t+1}, a_{t+1})$$

$$\delta_t \leftarrow r_t + \gamma Q(s_{t+1}, a^*_{t+1}) - Q(s_t, a_t)$$

$$e_t(s_t, a_t) \leftarrow e_t(s_t, a_t) + 1$$

For all (s_t, a_t):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \delta_t e_t(s_t, a_t)$$

If $a_{t+1} = a^*_{t+1}$, **Then**

$$e_t(s_t, a_t) \leftarrow \gamma\lambda e_t(s_t, a_t)$$ **Else**

$$e_t(s_t, a_t) \leftarrow 0$$

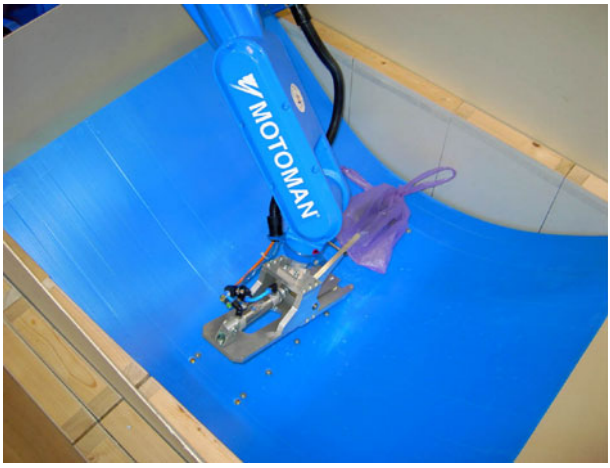
$$s_t \leftarrow s_{t+1}; a_t \leftarrow a_{t+1}$$

Until $i = M$ where M is the maximal number of learning episodes.
where |X| = cardinality of X.

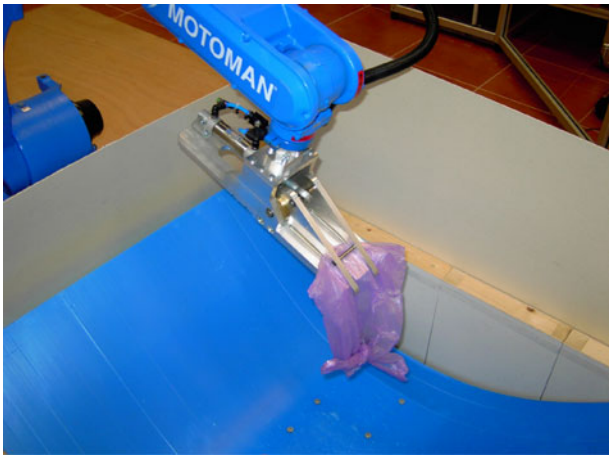
4.2 Task Definition

The experiment utilizes a Motoman UP-6 fixed-arm robot positioned over a table surface. A dedicated gripper was specifically designed for grasping a plastic bag (Fig. 4). The learning task is to observe the position of a plastic bag located on a table surface, grasp it with a fixed-arm robot, and learn how to shake out its contents in minimum time. This is done via both interaction with the environment and acquiring knowledge from a human advisor (HA). A digital scale placed under the table surface was used to automatically measure rewards.

The robot has no a-priori knowledge regarding the most efficient shaking policy for any given plastic bag, but it learns this information by interaction with the environment and from human guidance. The HA provides guidance in terms of linguistic suggestions which are in turn used to modify the system's Q table (a detailed explanation is found under Section 4.3). For the experiment, robot states



(a)



(b)

Fig. 4 Experimental setup. **a** Plastic bag grasp operation. **b** Robot grasped the plastic bag and moved it to a central point above a table surface. The bag is held upright by the robot gripper and is initially closed by the knot

are described in Table 1 and pertain to its gripper location in a three-dimensional grid (Fig. 5). The performance of the task is a function of a set of actions, $a_t \in A$, for each physical state of the system.

Table 1 States description of the three-dimensional grid

State(s)	Description	Number of states
$S_{(Center)}$	State center	1
$s_{(X_{3-})}$, $s_{(X_{2-})}$, $s_{(X_{1-})}$ $s_{(X_{1+})}$, $s_{(X_{2+})}$, $s_{(X_{3+})}$	States where the robot can move over its X axis	6
$s_{(Y_{3-})}$, $s_{(Y_{2-})}$, $s_{(Y_{1-})}$, $s_{(Y_{1+})}$, $s_{(Y_{2+})}$, $s_{(Y_{3+})}$	States where the robot can move over its Y axis	6
$s_{(Z_{3-})}$, $s_{(Z_{2-})}$, $s_{(Z_{1-})}$ $s_{(Z_{1+})}$, $s_{(Z_{2+})}$, $s_{(Z_{3+})}$	States where the robot can move over its Z axis	6

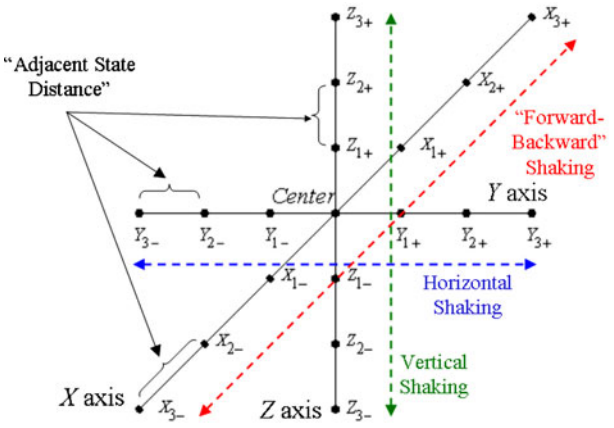


Fig. 5 Motoman UP-6 fixed-arm robot state-space

An action, a_t, consists of a robot movement from a state point, s_t, along a single coordinate direction (X, Y, Z). The Y axis is defined as the horizontal shaking axis, i.e., actions are performed in parallel to the horizon (left to right). At the X axis (forward to backward), actions are performed in parallel to a horizon perpendicular to the Y axis. Over the Z axis, actions are performed vertically (up to down). It is assumed that the bag has been grasped and moved to a central point centered 50 cm above a table surface. The robot starts a shaking policy from this $s_{(Center)_t}$ state. From $s_{(Center)_t}$ it can move in the direction of any of the three coordinates reaching any of the other 18 states. The adjacent state distances in any axis are 30 mm. From any robot state other than $s_{(Center)_t}$, the robot is limited to either symmetrical actions or returning to the center position (e.g., from state $s_{(X_{2-})t}$ the robot can move to $s_{(Center)_t}$ or to $s_{(X_{2+})t}$).

System performance was evaluated using two measures calculated after each learning episode: (1) the time it took the robot to empty the contents of the bag, or the time limit whichever is smaller (see T below), and (2) the reward.

4.3 Human Adviser's Interface

The interface (Fig. 6) allows the HA to *implicitly control* the Q table through two types of input: (1) *Center Control*, and (2) *Swing Control* (Fig. 6a). Performance is displayed graphically and numerically as (1) policy times, and (2) cumulative success rate (Fig. 6b). The HA is not aware of the Q values as such, but uses linguistic controls to guide the robot to emphasize changes in the coordinate direction of movement when the robot is at the origin of the 3D space, $s_{(Center)_t}$, and to modify the amplitude swings along single coordinate directions. The two input types are as follows:

1. *Center Control*—The *Center Control* options are relevant when the robot is located at the center of the coordinate system. The HA can give guidance to change the robot's likelihood to move from the center position to a particular axis's set of states (all six possible states for a specific axis; see Fig. 5). If during shaking the robot passes through the center position while performing a shaking

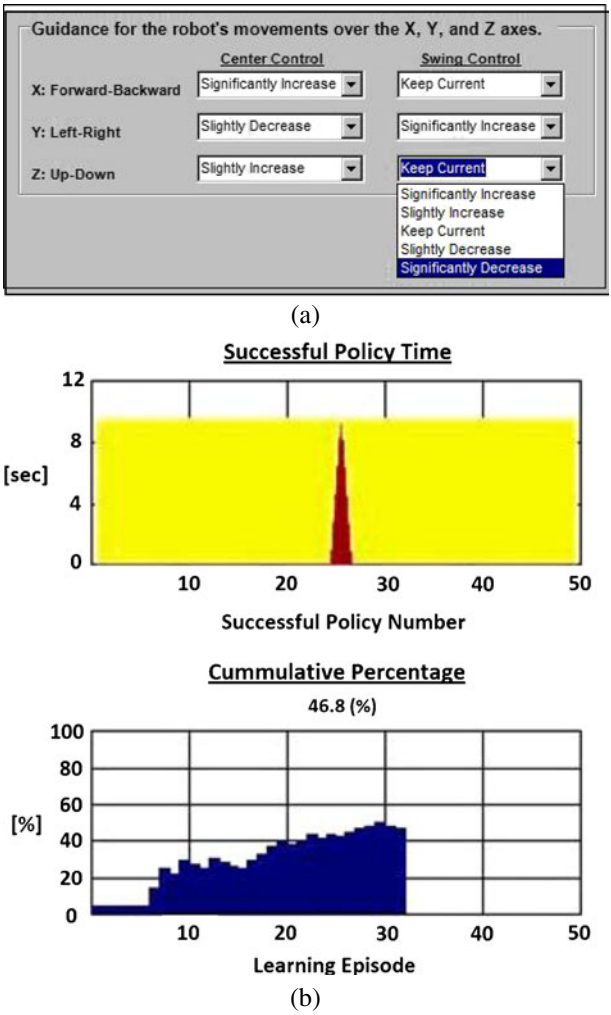


Fig. 6 Human interface. **a** Linquistic controls. **b** An example for system performance

operation over a specific axis, based on the Q table, the robot might switch its shaking actions to a different axis.

2. *Swing Control*—The *Swing Control* options, change the likelihood of a move to a state's mirror position[3] or a move to the center position (Fig. 5). Using this option allows the HA to guide the robot to perform a longer sequence of actions over a specific axis by decreasing the likelihood of moving to the center position and switch to a different axis.

The guidance options in the control interface are human centered providing five options (see Fig. 6a) for center and swing control interventions on each of the coordinate axis as defined below.

[3]A mirror position of a state is the physical symmetrical state over the same axis. This state and its mirror are at the same distance from the center state.

Center Control (for each axis—X, Y, or Z) options are:

- *Significantly increase or slightly increase*—increases all Q values of state-action pairs from the center position to any of the possible six states located over the chosen axis by 50% and 10%, respectively.
- *Keep current*—no change in the Q table.
- *Significantly decrease or slightly decrease*—decreases all Q values of state-action pairs from the center position to any of the possible six states located over the chosen axis by 50% and 10%, respectively.

Swing Control (for each axis—X, Y, or Z) options are:

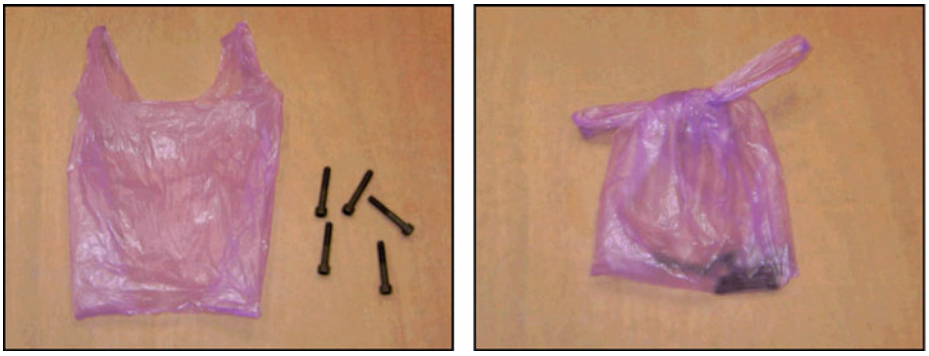
- *Significantly increase or slightly increase*—increases all six Q state-action value pairs of moving back to the center for the chosen axis by 50% and 10%, respectively, while keeping current the state-action values of moving back to the center.
- *Keep current*—no change in the Q table.
- *Significantly decrease or slightly decrease*—decreases all six Q state-action value pairs of moving back to the center for the chosen axis by 50% and 10%, respectively, while keeping current the state-action values of moving back to the center.

The HA must make a decision based on his expertise and from observation of the rewards obtained by the robot. To assist the HA, information on the performance of the robot is presented in the interface as shown in Fig. 6b. Reporting on system performance is shown by using two graphs: (1) *Successful policy time*—a red arrow shows the time the robot performed the last successful shaking compared with the average of all policies (the yellow surface), and (2) *Cumulative success rate*—a measure of the percent of all successful learning episodes performed. The real-time information presented in these graphs helps the HA to give better advice. This is due to the clear presentation that allows him to intuitively compare between the current system performance vs. the outcome of his recent advices.

4.4 Experiments

The initial system values were set at $\gamma = 0.9$, $\lambda = 0.5$, and $\alpha = 0.05$. The relatively low learning rate ($\alpha = 0.05$) was chosen to regard past experience as important [20, 57].[4] Similarly to [42], the discount rate (γ) was set to 0.9. This makes the robot give higher significance to future rewards. The eligibility trace (λ) was set to 0.5 as in [58]. This value was chosen to let a long sequence of values of state-action pairs to be updated while maintaining a reasonable computational time. The learning performance threshold was empirically determined as $\Lambda = 0.6$ by an experiment under which Λ was varied over a range of values (an automated method for dynamically adjusting Λ is described in Section 6). For the experiment, five identical screws are inserted into the plastic bag while resting on the table (Fig. 7). Screws falling from the bag were measured by their weight with a digital scale. In

[4]During the first ten learning episodes the robot is forced to learn autonomously. This is to consider past experience as more significant during the crucial early stages of learning.



(a) (b)

Fig. 7 Plastic bag with objects. **a** Plastic bag and five screws. **b** Closed plastic bag with screws

this paper it is assumed identical items are placed in the bag, but in the case of nonidentical items (different sizes and weights), one can pre-weigh the bag and base the reward on the total weight extracted. The performances of $Q(\lambda)$ and $CQ(\lambda)$ were compared for $M = 50$ learning episodes. To balance between exploration and exploitation (e.g., [59, 60]), an ε-greedy action selection with $\varepsilon = 0.1$ was used[5] for the first 45 learning episodes to encourage exploration and decreased to $\varepsilon = 0$ for the last five episodes to allow the system to exploit the best policy.

An event-based weight history was obtained from the digital scale and was used for automatic measurements of the rewards. The reward function for learning episode n is denoted as R_n as shown in Eq. 7.

$$R_n = \sum_{j=0}^{T} \frac{(W_j - W_{j-1})/w}{t_j} \tag{7}$$

Here W_j is the current weight of all items that were shaken from the bag, measured at time t_j (increments of 0.25 s). If at time t_j one or more objects fall from the bag, dividing the weight differences by t_j effectively increases the reward for items that fall early. Here w is the weight of one object (a constant value), and W_{j-1} is the weight measured by the scale during the previous time. $T = \min\{$*Fixed Horizon Time, Amount of Time when all Objects Fell*$\}$ is the time of shaking (one learning episode), where the *Fixed Horizon Time* is measured as the time it takes the robot to perform a pre-defined number of actions and was set to 100. If no objects fall after 100 actions, it is assumed that the bag is closed. This implies that T is calculated as the amount of time it took the robot to perform 100 actions (this time varies from one learning episode to another because different policies may be taken in different learning episodes). The *Amount of Time when all Objects Fell* is the amount of time till the bag is empty. The value $(W_j - W_{j-1})/w$ represents the number of objects that fell at time t_j.

[5]Setting $\varepsilon = 0.1$ allows the robot to select the best action out of the set of all actions with probability 0.9, and to select one of the non best actions uniformly with probability 0.1.

The robot starts its first learning episode by performing a random shaking policy over the X, Y, Z and axes. The default speeds and adjacent state distance were set to 1,000 mm/s and 30 mm respectively for all axes.[6] In the initial ten episodes the robot was forced to autonomously learn the environment, and thus no human collaboration was allowed until episode 11. When robot learning performance is low (6), the HA is asked to intervene to provide guidance for the robot's movements over the X, Y, Z and axes. The HA uses the interface shown in Fig. 6 to guide the robot to prefer some actions over others without a demand to understand what Q value levels are.

5 Experimental Results

The average time to empty the contents of the bag using $CQ(\lambda)$ and $Q(\lambda)$-learning over 50 learning episodes was 9.78 and 10.4 s, respectively, resulting in an improvement of 6%.[7] From a reward perspective, the average reward achieved over the 50 episodes was measured as 28.81 for $CQ(\lambda)$ and 20.11 for $Q(\lambda)$, i.e., an improvement of 43.3%. The human intervention rate measured for the $CQ(\lambda)$-learning experiment was 12%.

Since robot learning performance is improved as the number of learning episodes grows, it was decided to compare between $CQ(\lambda)$ and $Q(\lambda)$ over the last ten learning episodes. Comparing $CQ(\lambda)$ with $Q(\lambda)$-learning the average time to empty the contents of a bag was 5.95 seconds and 8.49 seconds, respectively, i.e., an improvement of 30%. From a reward perspective, the average reward achieved was measured as 47.17 for $CQ(\lambda)$ and 27.8 for $Q(\lambda)$, i.e., an improvement of 69.7%.

Based on the results achieved, the following hypotheses were evaluated.[8]

H_{10}: There is a significant difference between *average time* to complete emptying the contents of a bag while comparing $Q(\lambda)$ with $CQ(\lambda)$-learning.

H_{20}: There is a significant difference between *average reward* while comparing $Q(\lambda)$ with $CQ(\lambda)$-learning.

H_{10} and H_{20} were accepted with P-values equal to $1.66{\cdot}10^{-6}$ and 0.01, respectively. This means that the average time and average reward to complete emptying the contents of the bag measured over the last ten learning episodes are not equal while comparing $Q(\lambda)$ with $CQ(\lambda)$ exhibiting lower average time to extract objects at a higher reward.

Performance times for each learning episode are shown in the scatter plot of Fig. 8 for both the $CQ(\lambda)$ and $Q(\lambda)$ runs. Exponential smoothing of the data, using a damping factor of 0.8 for the measurements, are shown as solid and dotted lines for $CQ(\lambda)$ and $Q(\lambda)$, respectively. Considering all 50 learning episodes, the $CQ(\lambda)$ learning curve resulted an average of 10.84 s with a standard deviation of 4.42 as compared to $Q(\lambda)$, which resulted in an average time of 11.32 s with a standard deviation of 2.69. Considering the last ten learning episodes, the $CQ(\lambda)$ learning

[6] Moderate magnitudes of speeds and adjacent state distances were chosen not to harm the robot.

[7] Excluding grasping and lifting.

[8] The evaluation is for the last ten learning episodes in which exploration was complete.

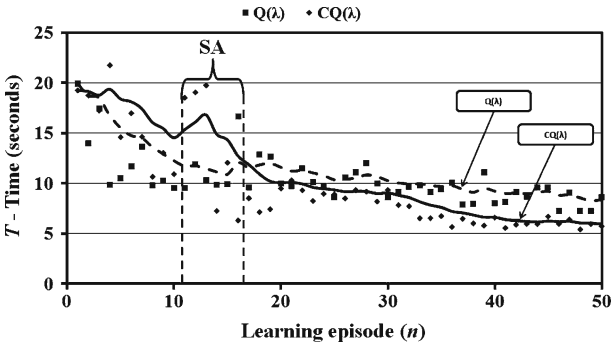


Fig. 8 Performance times for events-based reward. *SA* represents a semi-autonomous mode

curve resulted in an average of 6.21 s with a standard deviation of 0.13 while for $Q(\lambda)$, the average time was 8.83 s with a standard deviation of 0.31.

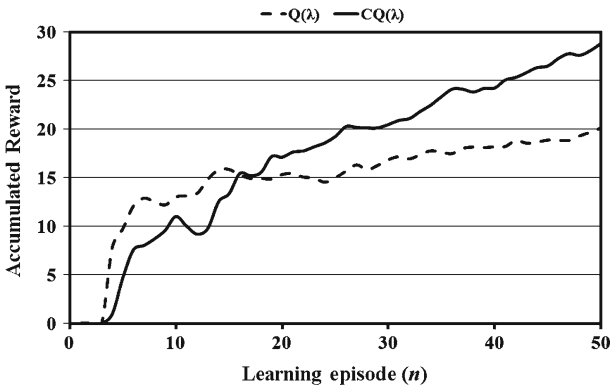
It is seen from smoothed time performance data that $CQ(\lambda)$ is superior, having approached convergence at about episode 45. Reward performance superiority of $CQ(\lambda)$ over $Q(\lambda)$ is shown in Fig. 9.[9] Theoretically, system performance should be identical for both $CQ(\lambda)$ and Q during the first ten learning episodes; however rewards and shaking times may slightly be different from one another even if the sequence of robot actions taken is identical due to the stochastic nature of the problem. For the $CQ(\lambda)$ test run, Fig. 8 shows that after human intervention was first allowed (right after episode 10), the robot started and continued to request assistance from the HA for six episodes. One sees that at the 11th, 12th and 13th learning episodes the shaking performance times are significantly high. This can be explained due to the explorative strategies the HA tried over these episodes when first asked to intervene at this stage. The HA quickly recovered providing a good policy over the next three episodes. This was enough guidance so that the robot continued to operate autonomously and never requested advice again.

The human's intelligence and the experience gathered while conducting previous experiments described in [7, 8] taught him that the robot should be guided to choose actions that will shake the bag mostly over the Y axis and with a small number of actions over the X axis to be more effective. Therefore, when asked to intervene, the human picked the highest possible value for the number of swings over the Y axis, an intermediate value for the number of swings over the X axis, and reduced the likelihood of swings over the Z axis.

Intuitively, vertical shaking should work best, but experimental results showed that for both $CQ(\lambda)$ and $Q(\lambda)$ the best policies showed shaking most of the time over the Y axis with very little activity over the X axis. One possible explanation for favoring the Y axis may be the type of knot holding the plastic bag closed; pulling it sideways loosens the knot faster. During the experiments performed when the HA was asked to intervene, the decision was to prefer a strategy that will cause the robot to continuously perform actions that will mostly occur at locations as far as possible over the Y axis (left–right) and small amount of actions over the

[9] In the first four learning episodes no items fell from the bag, resulting in a zero reward.

Fig. 9 Reward performance



X(forward–backward) axis, while eliminating any action over the vertical axis (up-down). Specifically, the chosen strategy is described as follows:

1. *Left-Right* strategy—for *Center Control* keep choosing *Slightly Decrease* and for *Swing Control*, keep choosing *Significantly Increase Swings*. This strategy will make the robot move mostly over the left-right axis; and, only occasionally, move over the forward-backward axis.
2. *Forward-Backward* strategy—for *Center Control* keep choosing *Slightly Increase* and for *Swing Control*, keep choosing *Keep Current*. This strategy allows the robot to perform a few shakings over the forward-backward axis, but causes it to "prefer" moving back to the center position.
3. *Up-Down* strategy—for *Center Control*, keep choosing *Significantly Increase*. That is because it is desired to prevent the robot from being shaken over the vertical axis. Moving back to the center rather than moving to one of the mirror vertical states will allow the robot to "escape" from the set of possible vertical states, go back to the center position and possibly to move to a horizontal or to a forward-backward shaking. That is why it is better also to keep choosing *Significantly Decrease Swings* which will contribute for moving back to the center.

To summarize, the results indicated that learning was faster when the HA was asked to intervene in the robot's activity. These results are consistent with previous experiments described in [7, 8, 61]. Although other alternatives are available for solving the proposed problem, such as cutting open the bag and sliding the objects out of the bag, the shaking application was selected to serve as a test-bed for the $CQ(\lambda)$ algorithm.

6 Discussion

Unlike [2, 3], where the rewards are controlled by a human, or as described in [41] where user commands are employed for modifying the robot's reward function, in $CQ(\lambda)$ the human provides guidance to the robot to perform an improved policy. The

method for improving learning performance described in this paper is to *implicitly control* the Q table of a robot learning problem using a linguistic-based human interface. Collaboration here is similar to a learning application described in [31], where human reward signals can be treated as an "interactive rewards interface" in which humans can give rewards to a learning agent by rewarding a whole world state. In [42], a trainer is asked to intervene and suggest help when two extreme Q-values are sufficiently close. This is achieved by examining the minimum and maximum values and comparing the result to a pre-defined width parameter. In lieu of examining specific Q-values, the intervention in $CQ(\lambda)$ is based on examining the performance history of the agent over a pre-defined number of learning episodes and comparing it to a performance threshold.

To divide the state-space into regions of high confidence (autonomous execution) and low confidence (demonstration), [29] compared between two thresholds that were calculated dynamically. The agent as in [29] performed a Confidence Execution procedure after it reaches a new state. The procedure (in which system's thresholds and classifier are updated) determines whether to request a human demonstration or to execute an action by taking account previous corrective demonstrations. This is different from the $CQ(\lambda)$ method suggested here in which the current agent's learning performance is evaluated based on its history experience and then this level of performance (which dynamically changed as more learning episodes are accomplished) is compared to a pre-defined threshold. This threshold value is pre-determined to satisfy the system's designer for an acceptable and subjective level of learning performance. $CQ(\lambda)$ allows the agent to perform a high number of state-action transitions and only then evaluates its performance and reward. Then, based on its level of learning performance it "asks" for human assistance or continues performing the next learning episode autonomously. For the bag shaking task, the Confidence Execution approach of requesting human help in situations of unfamiliar\ambiguous states would not be feasible since the state-action transitions are performed very fast. This means that for this physical-oriented task, a human can only intervene after a sequence of state-action transitions, i.e., after accomplishing a complete learning policy.

At first look, the robotic task described in this paper looks simple; to only allow movement on one of three axes of a relatively small state-space, to only allow symmetrical movements in those axes, and to only allow switching between axes at one point. The grasping location and the bag opening orientation are known as well. However, while the location of the bag opening is known in advance, learning to find the directions to shake the bag optimally within minimal shaking actions using reinforcement learning by a robot is not a trivial task. Relaxing the assumption on knowing the location of the knot could end up with a different shaking policy. For example if the knot is on the side after grasping instead of on the bottom, one would suspect the shaking would be more on the vertical direction to loosen the knot and more in the horizontal direction in order to cause the items to "jump" out of the bag. In this case the physical model could be extended to take this into account and accelerate the items to the correct direction. Additionally, since the focus of the system was on developing a collaborative part and showing it is better than the system without a HA, the relatively small size of the state-space was not a concern. The state-space was kept small to avoid excessive computations and it was felt that using a method for handling a high dimension space would not provide added value

to prove the main hypothesis. The collaborative method can be applied to expanded state-spaces.

Future research should address the interface design which could be critical for performance. The current interface was tailor designed to the specific application to provide "hints" at the beginning of the learning episode rather than guiding the robot during the whole episode. This was assumed critical due to the bag-shaking task which requires performing very fast physical robot transitions in a short time (seconds). Hence, it was not possible to let the user "drive" the robot during the whole episode right after the robot determines at the beginning of the episode that it needs human assistance. In other tasks that may use $CQ(\lambda)$-learning, such guidance is definitely possible and should be considered. Additionally, the interface was designed to address a 3D state-space problem. As such, if the coordinate increments are made finer to provide a larger state-space, the interface is not affected as it is based on linguistic terms. For example, the user can suggest higher or lower robot arm swing amplitudes without actually mentioning there values.

To prove that the $CQ(\lambda)$-learning approach can easily be extended to general robotic tasks, the algorithm was applied to a mobile robot task in which the robot must navigate toward a target location in a two-dimensional world that contains obstacles [61]. An ER-1 mobile robot switches between fully autonomous operation and semi-autonomous navigation using human intervention. Different levels of human intervention corresponding to several robot learning threshold values were defined and the robot switched its activity from semi-autonomous navigation to full autonomy. The required modifications are related to the task and its environment (and not to the algorithm itself) and included customizing a different state-space, and defining different thresholds and rewards. Such modifications shall be applied for any RL-based application. These experiments demonstrated that $CQ(\lambda)$-learning was preferable to Q-learning.

Another example of how to use $CQ(\lambda)$-learning in a different domain can be provided by slightly modifying Sophie's Kitchen task [31, 32]. One way to apply $CQ(\lambda)$-learning here is to define a reward threshold value (similarly to $\overline{R}$) which above a completion of the baking task will not only be considered as successful but also considered as "learned efficiently" (using Eq. 5). Instead of allowing the human to provide a reward at any point in the operation, rewards will only be given by the human when the system is switched from the autonomous mode (learn to accomplish the task independently) and semi-autonomous operation (asking for human advise). It is expected that the agent will be less dependent on the human, as more and more episodes are performed with gradually increasing rewards (and thereby using fewer actions).

Another important aspect that must be addressed in designing a $CQ(\lambda)$-based system is to let the system calculate the necessary thresholds such as the average reward threshold ($\overline{R}$) and the performance threshold Λ, automatically. In our experiments the desired threshold values were determined empirically. The general impact of the threshold on the performance of the algorithm is that it affects: (1) the balance between autonomous and semi-autonomous modes, and (2) the amount of human involvement and improvement of performance. The higher $\overline{R}$, the more "difficult" it is for the system to be considered as performing the learning task well, thereby the system's designer would expect more requests for human intervention. This applies also on picking Λ the threshold value. Picking low values of $\overline{R}$ and

Fig. 10 An adaptive Λ method. Note convergence to a new level or to a local minimum is determined if the variability of L_{ave} falls below some epsilon value

1. Start with an initialization phase in *Autonomous* mode. At the end of this phase calculate an initial L_{ave} for system performance.
2. Set $\Lambda = (1 - L_{ave})/2$ so the system immediately converts to *Semi-Autonomous*.
3. Operate in *Semi-Autonomous* until the performance of L_{ave} (a) is driven above the current threshold $L_{ave} \geq \Lambda$ then go to step 4, or (b) if L_{ave} converges to a new level, then go to Step 5.
4. Operate in *Autonomous* mode until L_{ave} converges to a local minimum or $L_{ave} \leq \Lambda$. Then return to Step 2.
5. Stop.

Note convergence to a new level or to a local minimum is determined if the variability of L_{ave} falls below some epsilon value.

Λ is expected to result the same system performance as picking high values of $\overline{R}$ and Λ. Picking low $\overline{R}$ and high Λ will result less calls for human collaboration but is expected to reduce system performance and vice versa. To conclude, the ability to balance between the two thresholds gives the designer the flexibility to control the amount of human collaboration on the expense of performance. However, the empirical determination of these thresholds involves lots of time and effort, and it is thus advisable to automate this aspect by dynamically changing the thresholds during the operation of the $CQ(\lambda)$ algorithm. A method to automatically adjust the Λ threshold value (a similar method may be used for automatic adjustment of the $\overline{R}$ threshold) is described in Fig. 10.

7 Conclusions

A collaborative reinforcement learning algorithm $CQ(\lambda)$ which expedites the learning process by taking advantage of human intelligence and expertise is developed. By allowing automatic switching between autonomous and semi-autonomous modes, the $CQ(\lambda)$-learning algorithm performs surprisingly well to learn the optimal control policy, given the notoriously slow convergence of RL algorithms. One advantage of our approach is that the human advisor is free to attend to other matters while the system is in autonomous learning mode, and only returns when a request for advice is rendered. The contributions of the paper are two-fold: (1) the paper introduces a threshold-based method for the agent (robot) to request advice from a human advisor (HA), and (2) the paper applies this approach in a bag-shaking task, in which a robot must learn how to shake a bag so as to release a knot tying it and thus release the objects within. An additional novelty of this work is how the collaboration with a robot is addressed, allowing a human to directly manipulate the Q values through a linguistic-based interface.

A comparative evaluation of $CQ(\lambda)$ with the classical $Q(\lambda)$-learning algorithm was made. It was found that there is significant difference between *average time* to complete emptying the contents of a bag as well as the *average reward*. The comparison showed that the $CQ(\lambda)$ algorithm exhibited lower average time to

extract objects and at a higher reward level than that of the $Q(\lambda)$ algorithm. This result suggests that the $CQ(\lambda)$ learning algorithm developed in this research looks promising to overcome the slow learning times of the classical RL method. However, there are some issues that still require attention. How can the robot be sure that all HA suggestions are beneficial? It may be the case that some human advice does not contribute to the robot learning process. One way to overcome this issue is to have the robot assess the quality of the human's advice and in the case of ineffective (unhelpful) learning episodes the robot will reject the human suggestions and revert to an autonomous mode. Under such circumstances, the robot can perform a "back up procedure", erase the inadequate human suggestion, and continue from there. Then it will compare its performance to that of the human and will choose the best policy, i.e., the robot decides whether to use its performance or that induced by human guidance.

References

1. Zhu, W., Levinson, S.: Vision-based reinforcement learning for robot navigation. In: Proceedings of the International Joint Conference on Neural Networks, Washington DC, vol. 2, pp. 1025–1030 (2001)
2. Papudesi, V.N., Huber, M.: Learning from reinforcement and advice using composite reward functions. In: Proceedings of the 16th International FLAIRS Conference, pp. 361–365, St. Augustine, FL (2003)
3. Papudesi, V.N., Wang, Y., Huber, M., Cook, D.J.: Integrating user commands and autonomous task performance in a reinforcement learning framework. In: AAAI Spring Symposium on Human Interaction with Autonomous Systems in Complex Environments, pp. 160–165. Stanford University, CA (2003)
4. Kui-Hong, P., Jun, J., Jong-Hwan, K.: Stabilization of biped robot based on two mode Q-learning. In: Proceedings of the 2nd International Conference on Autonomous Robots and Agents, pp. 446–451. New Zealand (2004)
5. Broadbent, R., Peterson, T.: Robot learning in partially observable, noisy, continuous worlds. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp. 4386–4393. Barcelona, Spain (2005)
6. Bakker, B., Zhumatiy, V., Gruener, G., Schmidhuber, J.: Quasi-online reinforcement learning for robots. In: Proceedings of the 2006 IEEE International Conference on Robotics and Automation, pp. 2997–3002 (2006)
7. Kartoun, U., Stern, H., Edan, Y.: Human–robot collaborative learning of a bag shaking trajectory. In: The Israel Conference on Robotics (ICR 2006), Faculty of Engineering, Tel Aviv University, June (2006)
8. Kartoun, U., Stern, H., Edan, Y.: Human–robot collaborative learning system for inspection. In: IEEE International Conference on Systems, Man, and Cybernetics, Taipei, Taiwan, October, pp. 4249–4255 (2006)
9. Mihalkova, L., Mooney, R.: Using active relocation to aid reinforcement. In: Proceedings of the 19th International FLAIRS Conference (FLAIRS-2006), Melbourne Beach, Florida, pp. 580–585 (2006)
10. Fernández, F., Borrajo, D., Parker, L.E.: A Reinforcement learning algorithm in cooperative multi-robot domains. J. Intell. Robot. Syst. 4(2–4), 161–174 (2005)
11. Kartoun, U., Shapiro, A., Stern, H., Edan, Y.: Physical modeling of a bag knot in a robot learning system. IEEE Trans. Automat. Sci. Eng. 7(1), 172–177 (2010)
12. Katić, D.M., Rodić, A.D., Vukobratović, M.K.: Hybrid dynamic control algorithm for humanoid robots based on reinforcement learning. J. Intell. Robot. Syst. 51(1), 3–30 (2008)
13. Anderson, G.T., Yang, Y., Cheng, G.: An adaptable oscillator-based controller for autonomous robots. J. Intell. Robot. Syst. 54(5), 755–767 (2009)
14. Peters, J., Schaal, S.: Learning to control in operational space. Int. J. Rob. Res. 27, 197–212 (2008)
15. Ribeiro, C.: Embedding a priori knowledge in reinforcement learning. J. Intell. Robot. Syst. 21(1), 51–71 (1998)

16. Hoffmann, H., Theodorou, E., Schaal, S.: Human optimization strategies under reward feedback. Abstracts of Neural Control of Movement Conference (NCM 2009) (2009)
17. Schmidhuber, J.: Developmental robotics, optimal artificial curiosity, creativity, music, and the fine arts. Connect. Sci. 18(2), 173–187 (2006)
18. Matarić, M.J.: Reinforcement learning in the multi-robot domain. Auton. Robots 4(1), 73–83 (1997)
19. Dahl, T.S., Matarić, M.J., Sukhatme, G.S.: Multi-robot task allocation through vacancy chain scheduling. J. Robot. Auton. Syst. 57(6), 674–687 (2009)
20. Fukuda, T., Funato, D., Arai, F.: Recognizing environmental change through multiplex reinforcement learning in group robot system. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 2, pp. 972–977 (1999)
21. Chernova, S., Veloso, M.: Confidence-based policy learning from demonstration using Gaussian mixture models. In: International Conference on Autonomous Agents and Multi-agent Systems (AAMAS'07), 2007
22. Touzet, C.F.: Q-Learning for Robots. The Handbook of Brain Theory and Neural Networks, pp. 934–937. MIT Press, Cambridge (2003)
23. Inamura, T., Inaba, M., Inoue, H.: Integration model of learning mechanism and dialogue strategy based on stochastic experience representation using Bayesian network. In: Proceedings of the 9th IEEE International Workshop on Robot and Human Interactive Communication, RO-MAN 2000, pp. 247–252 (2000)
24. Inamura, T., Inaba, M., Inoue, H.: User adaptation of human-robot interaction model based on Bayesian network and introspection of interaction experience. In: International Conference on Intelligent Robots and Systems (IROS 2000), vol. 3, pp. 2139–2144 (2000)
25. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robot. Auton. Syst. 57(5), 469–483 (2009)
26. Katagami, D., Yamada, S.: Interactive classifier system for real robot learning. In: Proceedings of the 9th IEEE International Workshop on Robot and Human Interactive Communication, RO-MAN 2000, pp. 258–263 (2000)
27. Atkeson, C., Schaal, S.: Robot learning from demonstration. In: Proceedings of the International Conference Machine Learning, pp. 12–20 (1997)
28. Price, B., Boutilier, C.: Accelerating reinforcement learning through implicit imitation. J. Artif. Intell. Res. 19, 569–629 (2003)
29. Chernova, S., Veloso, M.: Interactive policy learning through confidence-based autonomy. J. Artif. Intell. Res. 34, 1–25 (2009)
30. Chernova, S., Veloso, M.: Multi-thresholded approach to demonstration selection for interactive robot learning. In: The 3rd ACM/IEEE International Conference on Human–Robot Interaction (HRI'08), pp. 225–232 (2008)
31. Thomaz, A.L., Breazeal, C.: Reinforcement learning with human teachers: evidence of feedback and guidance with implications for learning performance. In: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI), 2006
32. Thomaz, A.L., Breazeal, C.: Teachable robots: understanding human teaching behavior to build more effective robot learners. Artif. Intell. 172, 716–737 (2008)
33. Lockerd, A.L., Breazeal, C.: Tutelage and socially guided robot learning. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, Sendai, Japan (2004)
34. Breazeal, C., Thomaz, A.L.: Learning from human teachers with socially guided exploration. In: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pp. 3539–3544 (2008)
35. Abbeel, P., Ng., A.: Apprenticeship learning via inverse reinforcement learning. In: Proceedings of the International Conference Machine Learning, vol. 69, 2004
36. Chernova, S., Veloso, M.: Learning equivalent action choices from demonstration. In: The International Conference on Intelligent Robots and Systems (IROS 2008), pp. 1216–1221 (2008)
37. Chernova, S., Veloso, M.: Teaching collaborative multi-robot tasks through demonstration. In: IEEE-RAS International Conference on Humanoid Robots, pp. 385–390 (2008)
38. Watkins, C.J.C.H.: Learning from delayed rewards. Ph.D. dissertation, Psychology Department, Cambridge University (1989)
39. Peng, J., Williams, R.: Incremental multi-step Q-learning. Mach. Learn. 22(1–3), 283–290 (1996)
40. Dahmani, Y., Benyettou, A.: Seek of an optimal way by Q-learning. J. Comput. Sci. 1(1), 28–30 (2005)

41. Wang, Y., Huber, M., Papudesi, V.N., Cook, D.J.: User-guided reinforcement learning of robot assistive tasks for an intelligent environment. In: Proceedings of the IEEE/RJS International Conference on Intelligent Robots and Systems, vol. 1, pp. 424–429 (2003)
42. Clouse, J.A.: An Introspection Approach to Querying a Trainer. Technical Report: UM-CS-1996-013. University of Massachusetts, Amherst (1996)
43. Takamatsu, J., Morita, T., Ogawara, K., Kimura, H., Ikeuchi, K.: Representation for knot-tying tasks. IEEE Trans. Robot. 22(1), 65–78 (2006)
44. Wakamatsu, H., Eiji, A., Shinichi, H.: Knotting/unknotting manipulation of deformable linear objects. Int. J. Rob. Res. 25(4), 371–395 (2006)
45. Matsuno, T., Fukuda, T.: Manipulation of flexible rope using topological model based on sensor information. International Conference on Intelligent Robots and Systems, pp. 2638–2643 (2006)
46. Saha, M., Isto, P.: Motion planning for robotic manipulation of deformable linear objects. In: International Conference on Intelligent Robots and Systems, vol. 23(6), pp. 1141–1150 (2007)
47. Bellman, R.: A Markovian decision process. Journal of Mathematics and Mechanics 6, 679–684 (1957)
48. Ribeiro, C.: Reinforcement learning agents. Artif. Intell. Rev. 17(3), 223–250 (2002)
49. Smart, W.D., Kaelbling, L.: Practical reinforcement learning in continuous spaces. In: Proceedings of the 17th International Conference on Machine Learning, pp. 903–910 (2000)
50. Bellman, R., Kalaba, R.: Dynamic Programming and Modern Control Theory. Academic Press, New York (1965)
51. Watkins, C.J.C.H., Dayan, P.: Q-learning. Mach. Learn. 8, 279–292 (1992)
52. Glorennec, P.Y.: Reinforcement learning: an overview. European Symposium on Intelligent Techniques. Aachen, Germany, pp. 17–35 (2000)
53. S., Nason, Laird, J.E.: Soar-RL: integrating reinforcement learning with soar. In: Proceedings of the International Conference on Cognitive Modeling, pp. 51–59 (2004)
54. Natarajan, S., Tadepalli, P.: Dynamic preferences in multi-criteria reinforcement learning. In: Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), Bonn, Germany (2005)
55. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
56. Kartoun, U., Stern, H., Edan, Y.: Bag Classification Using Support Vector Machines. Applied Soft Computing Technologies: The Challenge of Complexity Series: Advances in Soft Computing, pp. 665–674. Springer, Berlin (2006)
57. Frank, M.J., Moustafa, A.A., Haughey, H.M., Curran, T., Hutchison, K.E.: Genetic triple dissociation reveals multiple roles for dopamine in reinforcement learning. In: Proceedings of the National Academy of Sciences, vol. 104(41), pp. 16311–16316 (2007)
58. Abramson, M., Wechsler, H.: Tabu search exploration for on-policy reinforcement learning. In: Proceedings of the International Joint Conference on Neural Networks 4(20–24), 2910–2915 (2003)
59. Guo, M., Liu, Y., Malec, J.: A new Q-learning algorithm based on the metropolis criterion. IEEE Trans. Syst. Man Cybern., Part B, Cybern. 34(5), 2140–2143 (2004)
60. Meng, X., Chen, Y., Pi, Y., Yuan, Q.: A novel multi-agent reinforcement learning algorithm combination with quantum computation. The 6th World Congress on Intelligent Control and Automation, vol. 1, pp. 2613–2617 (2006)
61. Kartoun, U.: Human-Robot Collaborative Learning Methods. Ph.D. dissertation, Department of Industrial Engineering and Management, Ben-Gurion University of the Negev (2007)