IASTED International Conference on Robotics and Applications (RA 2005), Cambridge, U.S.A.

COLLABORATIVE Q(λ) REINFORCEMENT LEARNING ALGORITHM - A PROMISING ROBOT LEARNING FRAMEWORK

Uri Kartoun*, Helman Stern*, Yael Edan*, Craig Feied**, Jonathan Handler**, Mark Smith**, Michael Gillam**
*Department of Industrial Engineering and Management, Ben-Gurion University of the Negev
Be'er-Sheeva, 84105, Israel
{kartoun, helman, yael}@bgu.ac.il
**Institute for Medical Informatics, Washington Hospital Center
110 Irving St., Washington DC, NW, 20010, U.S.A.
{cfeied}@ncemi.org

ABSTRACT
This paper presents the design and implementation of a new reinforcement learning (RL) based algorithm. The proposed algorithm, $CQ(\lambda)$ (collaborative $Q(\lambda)$) allows several learning agents to acquire knowledge from each other. Acquiring knowledge learnt by an agent via collaboration with another agent enables acceleration of the entire learning system; therefore, learning can be utilized more efficiently. By developing collaborative learning algorithms, a learning task solution can be achieved significantly faster if performed by a single agent only, namely the number of learning episodes to solve a task is reduced. The proposed algorithm proved to accelerate learning in navigation robotic problem. The $CQ(\lambda)$ algorithm was applied to autonomous mobile robot navigation where several robot agents serve as learning processes. Robots learned to navigate an 11 x 11 world contains obstacles and boundaries choosing the optimum path to reach a target. Simulated experiments based on 50 learning episodes showed an average improvement of 17.02% while measuring the number of learning steps required reaching definite optimality and an average improvement of 32.98% for convergence to near optimality by using two robots compared with the $Q(\lambda)$ algorithm [1, 2].

KEY WORDS
Robot simulation, reinforcement learning, and navigation

1. Introduction

Reinforcement learning (RL) used in this work is learning through direct experimentation [3, 4]. It does not assume the existence of a teacher that provides training examples. Instead, in RL experience is the only teacher. The learner receives signals (reinforcements) from the process by getting an indication about how well it is performing the required task. These signals are usually associated to some dramatic condition - *e.g.*, accomplishment of a subtask (reward) or complete failure (punishment), and the learner's goal is to optimize its behavior based on some performance measure (maximization of a cost function). The learning agent learns the associations between observed states and chosen actions that lead to rewards or punishments, i.e., it learns how to assign credit to past actions and states by correctly estimating costs associated to these events [5].

Navigation task can be broken down into three parts [6]: (i) localization, the process of figuring out where the robot is; (ii) mapping, the process whereby the robot builds a model of its environment, and (iii) planning, the process of figuring out how the robot can get to other places. Robots are inherently uncertain about the state of their environments. Uncertainty arises from sensor limitations, noise and the fact that real-world environment is unpredictable. Learning to navigate in realistic environments requires novel algorithms for identifying important events and find efficient action policies.

In [7] navigation learning of a miniature mobile robot is described. The robot equipped with vision capabilities learns to navigate a maze using several RL-based algorithms. [8] demonstrate a two mode Q-learning on a humanoid robot at a 17 x 17 maze for improving Q-learning performance. A RL algorithm for accelerating acquisition of new skills by real mobile robot is described in [9]. The algorithm speeds up Q-learning by applying memory-based sweeping [10] and was tested within an image-based visual servoing framework on an ActivMedia PeopleBot mobile robot for a docking task. A solution for robotic docking based on neural and reinforcement is presented in [11]. The solution was achieved partially by training of a value function unit and four motor units via RL. [12] describe a collaborative process enabling a robotic learner to acquire concepts and skills from human examples. During teaching the robot

requires to perform tasks based on human instructions. The robot executes the tasks and by incorporating feedback its hypothesis space is converged. With Q-learning approach, the robot learns a button pushing task.

Although Q-learning and $Q(\lambda)$ were used in many fields of robotics [*e.g.*, 13, 10, 14, 15, 16, 17], the issue of acceleration of learning is still significant. It includes the acceleration of learning toward finding an optimal or close to optimal solution. In order to improve learning, we suggest the new $CQ(\lambda)$ algorithm. The $CQ(\lambda)$ algorithm proved to accelerate learning in a system composed of several similar learning processes.

The paper is organized as follows. In Section two, reinforcement learning theory is described. That includes description of the Q, $Q(\lambda)$ and the new $CQ(\lambda)$ algorithms. Section three describes the learning system. Section four demonstrates simulation experiments and results of applying the $CQ(\lambda)$ algorithm on a navigation problem. Concluding remarks follow in Section five.

2. Reinforcement Learning

The basic assumption on the study of RL is that any state s_{t+1} made by the agent must be a function only of its last state and action: $s_{t+1} = f(s_t, a_t)$ where $s_t \in S$ is the state at time-step t and $a_t \in A$ is the action taken. Naturally, if the agent can faithfully observe the states of the process which by definition summarize all the relevant information about the process dynamics at a certain instant of time then its observations are Markovian. On the other hand, if the observations made by the agent are not sufficient to summarize all the information about the process, a non-Markovian condition takes place: $s_{t+1} = f(s_t, s_{t-1}, s_{t-2}, ..., a_t, a_{t-1}, a_{t-2}, ...)$ [5].

It is usually more adequate to express the dynamic of the process through a collection of conditional transition probabilities $P(s_{t+1} \in S \mid s_t \in S, a_t \in A)$. Of particular interest is the discounted infinite horizon formulation of the MDP (Markov Decision Process) problem. Given a finite set of possible actions $a \in A$, a finite set of process states $s \in S$, a stationary discrete-time stochastic process, modeled by transition probabilities $P(s_{t+1} \mid s_t, a_t)$ and a finite set of bounded reinforcements $r(s_t, a_t) \in R$, the agent tries to find out a stationary policy of actions $a_t^* = \pi^*(s_t)$ which maximizes the expected cost function (Eq. 1):

$$V^{\pi}(s_i) = \lim_{M \to \infty} E[\sum_{t=0}^{M} \gamma^t r(s_t, \pi(s_t))], \quad (1)$$

for every state s_i. π indicates the dependency on the followed action policy, via the transition probabilities $P(s_{t+1} \mid s_t, \pi(s_t))$. The discount factor $0 \le \gamma < 1$ forces recent reinforcements to be more important than remote ones. The optimal cost function is presented in Eq. 2 where it is possible that there will be more than a single optimal policy π^* [18].

$$V^*(s_0) = \lim_{M \to \infty} E[\sum_{t=0}^{M} \gamma^t r(s_t, \pi^*(s_t))]. \quad (2)$$

2.1. *Q*-Learning

The RL algorithm Q-learning [1], is modified here. In Q-learning the system estimates the optimal action-value function directly and then uses it to derive a control policy using the local greedy strategy. The advantage of Q-learning is that the update rule is model free as it is a rule that just relates Q values to other Q values. It does not require a mapping from actions to states and it can calculate the Q values directly from the elementary rewards observed. Q is the system's estimate of the optimal action-value function [19]. It is based on the action value measurement $Q(s_t, a_t)$, defined in Eq. 3:

$$\begin{aligned} Q(s_t, a_t) &= E[r(s_t, a_t) + \gamma V^*(s_{t+1})] = \\ &= r(s_t, a_t) + \gamma \sum_{s_{t+1} \in S} P(s_{t+1} \mid s_t, a_t) V^*(s_{t+1})], \end{aligned} \quad (3)$$

which represents the expected discounted cost for taking action a_t when visiting state s_t and following an optimal policy thereafter. From this definition and as a consequence of the Bellman's optimality principle [20], Eq. 4 is derived:

$$\begin{aligned} &Q(s_t, a_t) = r(s_t, a_t) + \\ &\gamma \sum_{s_{t+1} \in S} P(s_{t+1} \mid s_t, a_t) \max_a Q(s_{t+1}, a_t). \end{aligned} \quad (4)$$

These characteristics (max operator inside the expectation term and policy independence) allow an iterative process for calculating an optimal action policy via action values which is the essence of Q-learning. 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, the system's initial control policy, P, is established. This is achieved by assigning to P the action that locally maximizes the action-value. At time-step t, the agent

visits state $s_t \in S$ and selects an action $a_t \in A$, receives from the process the reinforcement $r(s_t, a_t) \in R$ and observes the next state s_{t+1}. Then it updates the action value $Q(s_t, a_t)$ according to Eq. 5 which describes a Q-learning one step:

$$Q_{t+1}(s_t, a_t) = (1-\alpha)Q_t(s_t, a_t) + \alpha[\, r(s_t, a_t) + \gamma \hat{V}_t(s_{t+1})], \quad (5)$$

where $\hat{V}_t(s_{t+1}) = \max_{a_t \in A}[Q_t(s_{t+1}, a_t)]$ is the current estimate of the optimal expected cost $V^*(s_{t+1})$ and α is the learning rate which controls how much weight is given to the reward just experienced, as opposed to the old Q estimate. The process repeats until stopping criterion is met (*e.g.*, robot reached target). $\alpha = 1$ gives full weight to new experiences. As α decreases, the Q-value is built up based on all experiences, and new unusual experience does not disturb the established Q-value much. The greedy action $\hat{V}_t(s_{t+1}) = \max_{a_t \in A}[Q_t(s_{t+1}, a_t)]$ is the best the agent performs when in state s_t, but for the initial stages of the learning process it uses randomized actions that encourages exploration of the state-space. Under some reasonable conditions [21] this is guaranteed to converge to the optimal Q-function [19].

2.2. CQ(λ)-Learning

$Q(\lambda)$ [1, 2] is a generalization of Q-learning that uses eligibility traces, $e(s_t, a_t)$: the one-step Q-learning is a particular case with $\lambda = 0$. The Q-learning algorithm learns quite slowly because only one time-step is traced for each action. To boost convergence of learning, the multi-step tracing mechanism, the eligibility trace, is used, in which the Q values of a sequence of actions can be updated simultaneously according to the respective lengths of the eligibility traces [13]. An outline of the multi-step $Q(\lambda)$ learning algorithm [1, 2], which is based on the tableau version in [22], is shown in Fig. 1.

Initialize $Q(s,a) = 0$ and set eligibility trace $e(s,a) = 0$ for all (s,a)
Repeat (for each learning episode):
  Set 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}
    Choose a_{t+1} for s_{t+1} with a certain policy (*e.g.*, ε-greedy)
    $a^* \leftarrow \arg\max_{b \in A} Q(s_{t+1}, b)$ (if a_{t+1} ties for the max, then $a^* \leftarrow a_{t+1}$)
    $\delta_t \leftarrow r_t + \gamma Q(s_{t+1}, a^*) - Q(s_t, a_t)$
    $e(s_t, a_t) \leftarrow e(s_t, a_t) + 1$
    For all (s_t, a_t):
      $Q(s_{t+1}, a_{t+1}) \leftarrow Q(s_t, a_t) + \alpha \delta_t e(s_t, a_t)$
      If $a_{t+1} = a^*$, then $e(s_t, a_t) \leftarrow \gamma\lambda e(s_t, a_t)$
        else $e(s_t, a_t) \leftarrow 0$
  $s_t \leftarrow s_{t+1}$; $a_t \leftarrow a_{t+1}$
until a stopping condition

Fig. 1. *Q(λ)* learning algorithm [1, 2]

The new $CQ(\lambda)$ (Fig. 2) algorithm objective is to accelerate learning in a system composed of several similar learning processes. It is based on that a state-action value of an agent or learning process is updated according to the maximal value within all other learning processes state-action values exist in a learning system. Alternatively if only two learning agents are involved such as a robot and human, state-action values known to the human are acquired by the robot learning function.

Initialize $Q_i(s,a) = 0$ and set eligibility trace $e_i(s,a) = 0$ for all (s,a) $i \in \{1, 2, \ldots N\}$
where N is the number of learning processes (*e.g.*, robot, human).
Repeat (for each learning process):
  Repeat (for each learning episode):
    Set initial state s_{i_t} and pick initial action a_{i_t}
    Repeat (for each step of episode):
      Take action a_{i_t}, observe reward r_{i_t} and the next state $s_{i_{t+1}}$
      Choose a_{t+1} for s_{t+1} using a certain policy (*e.g.*, softmax)
      $\delta_{i_t} \leftarrow r_{i_t} + \gamma_i Q_i(s_{i_{t+1}}, a_i^*) - Q_i(s_{i_t}, a_{i_t})$
      $e(s_t, a_t) \leftarrow e(s_t, a_t) + 1$
      For all (s_{i_t}, a_{i_t}):
        $Q_i(s_{i_{t+1}}, a_{i_{t+1}}) \leftarrow \max[Q_{i \in N}(s_{i_t}, a_{i_t}) + \alpha_{i_t} \delta_{i_t} e_i(s_{i_t}, a_{i_t})]$
        If $a_{i_{t+1}} = a_i^*$, then $e_i(s_{i_t}, a_{i_t}) \leftarrow \gamma\lambda e_i(s_{i_t}, a_{i_t})$
          else $e_i(s_{i_t}, a_{i_t}) \leftarrow 0$
    $s_{i_t} \leftarrow s_{i_{t+1}}$; $a_{i_t} \leftarrow a_{i_{t+1}}$
until a stopping condition

Fig. 2. *CQ(λ)*-learning algorithm

where δ_t is the temporal difference (TD) error which specifies how different the new value is from the old prediction and $e(s_t, a_t)$ is the eligibility trace that specifies how much a state-action pair should be updated at each time-step. When a state-action pair is first visited, its eligibility is set to one. Then at each subsequent time-step it is reduced by a factor $\gamma\lambda$. When it is subsequently visited, one is added to its eligibility trace [23].

In [24] it is stated that "the convergence of $Q(\lambda)$ is not assured anymore for $\lambda > 0$, but experiences show that the learning is faster". Several action selection policies are described in literature where the greedy policy [25] is always to choose the best action. Other policies (*e.g.*, "softmax" or "ε-greedy" [22]) are stochastic and based on choosing a suboptimal policy to explore the state-action space. In [22] it is stated that "although ε-greedy action selection is an effective and popular means of balancing exploration and exploitation [*e.g.*, 26], one drawback is that when it explores it chooses equally among all actions. This means that it is as likely to choose the worst-appearing action as it is to choose the next-to-best action. The obvious solution is to vary the action probabilities as a graded function of estimated value". One way to do that is to choose action a_t with probability that depends on

the value of $Q(s, a)$. This is known as a "softmax" action selection. A common method is to use a Gibbs or Boltzmann distribution, where the probability of choosing action a_t in state s_t is proportional to $e^{\beta_t Q(s_t, a_t)}$, i.e., in state s_t the agent chooses action a_t with probability

$$P(a_t \mid s_t) = \frac{e^{\beta_t Q(s_t, a_t)}}{\sum_{a_{t+1}} e^{\beta_t Q(s_t, a_t)}}, \quad (6)$$

where β_t is a positive parameter which specifies how randomly values should be chosen. When β_t is low, the actions are chosen about the same amount each other. As β_t increases, the highest valued actions are more likely to be chosen, and in the limit $\beta_t \to \infty$ the best action is always chosen [23].

In implementing the $CQ(\lambda)$ we adapt the attitude described in [4] for one agent and apply it for the collaborative agent attitude: learning rate of each process $i \in \{1, 2, \ldots N\}$, α_{i_t}, is set relatively high and is reduced adaptively over time. This is done independently for each state-action pair; number of times each state-action pair has been previously updated, $c_{i(s,a)}$, is calculated. The effective learning rate, $\alpha_{i(s,a)_t}$, is then determined from the initial rate by

$$\alpha_{i(s,a)_t} = \frac{\alpha_{i(s,a)_t}}{c_{i(s,a)} + 1}. \quad (7)$$

The principle is that the more and more knowledge about a certain state-action pair is gained, the less it is should be modified in response to any particular experience. Convergence was proved by [21].

3. Learning System

Robots' states include robot locations in a 11 x 11 two dimensional world (Fig. 3). The state s_t is defined by: $s_{(i,j)_t} = \{i, j\}$ where $i \in \{1, 2, ..., 11\}$ and $j \in \{1, 2, ..., 11\}$. Actions can be taken at each state are: traveling north, west, south and east. An action, a_t, is noted as $a_{(k)_t}$ where $k \in \{1, 2, 3, 4\}$. Rewards defined as $r_{(l)_t}$ where $l \in \{-1, 0, +1\}$. If the robot reaches target, the reward is +1. If it passes through an undesirable area such as an obstacle / boundary, the reward is -1. Otherwise, the reward is 0. Learning episode is a description of one session of reaching the target. Performance measures include: (i) $N_{i_{near-optimal}}$ - convergence to near optimality - mean of the last N_t learning step values, and (ii) $N_{i_{optimal}}$ - convergence to optimality - number of learning steps required to achieve the shortest path and repeat it infinite number of times where $i \in \{1, 2, \ldots N\}$ and N is the number of learning processes.

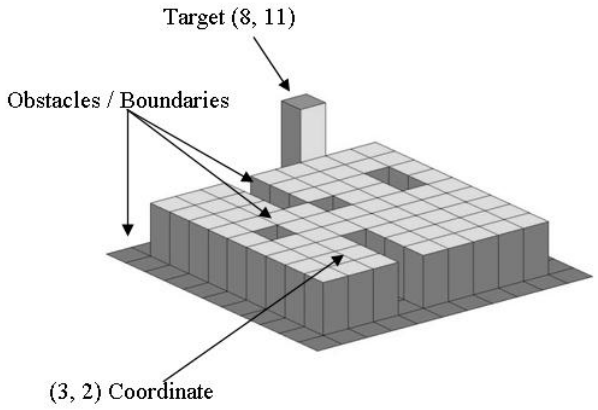

Fig. 3. An 11 x 11 two dimensional world

The world consists of three layers: (i) environmental cells - areas where the robot should navigate in its way to target; (ii) obstacles / boundaries - reduced cells and (iii) target - a higher cell (Fig. 3). Robots can move from a cell to any one of its four close neighbors with the restriction that they can not move out of the world. The task of the learning agents is to learn navigating the world by choosing optimal routes to a known target through environmental learning and knowledge sharing.

4. Experimental Results

The simulated system was evaluated using two experimental model sets and significance levels of mean steps to converge values were examined. In both models robot agents learn the world simultaneously and share knowledge. The first model contains two robot agents where the first learns according to $CQ(\lambda)$ and the second learns according to the $Q(\lambda)$ algorithms. The second model contains three robot agents where the first learns according to $CQ(\lambda)$ and the second and third learn according to the $Q(\lambda)$ algorithms.

At the first experimental model contains two robot agents. The first robotic agent noted as $i = 1$ learns according to the $CQ(\lambda)$ algorithm, i.e., gathers knowledge from the other robot whereas the second agent noted as $i = 2$ learns according to the $Q(\lambda)$ algorithm and does not gain knowledge from the first robot. The following parameters were set: $N = 2$, $N_t = 10$, $\alpha_1 = \alpha_2 = 0.95$ (initial values),

$\gamma_1 = \gamma_2 = 0.99$, and $\lambda_1 = \lambda_2 = 0.5$ (Fig. 2). At the second experimental model contains three robot agents, the first robot noted as $i = 1$ learns according to the $CQ(\lambda)$ algorithm, i.e., gathers knowledge from the other two robots whereas the second and third robots noted as $i = 2$ and $i = 3$ learn according to the $Q(\lambda)$ algorithm and do not gain knowledge from the first robot. The model was set with the following parameters: $N = 3$, $N_t = 10$, $\alpha_1 = \alpha_2 = \alpha_3 = 0.95$ (initial values), $\gamma_1 = \gamma_2 = \gamma_3 = 0.99$, and $\lambda_1 = \lambda_2 = \lambda_3 = 0.5$ (Fig. 2).

Experiments based on 50 simulation runs were conducted for both models. An example for state-action values of one simulation run is given in Fig. 4.In both models number of learning episodes was set to 100, i.e., the algorithm stops after an agent navigates from starting point to target 100 times. For evaluating system performance, a specific state with coordinates (3, 2) was chosen. For this state, the optimal route length traveling to the target at coordinates (8, 11) is 14 steps (Fig. 3). Based on the results shown in Table 1, ten hypotheses were evaluated (Table 2).

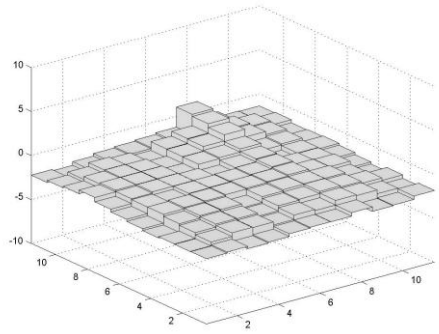

Fig. 4. 11 x 11 world state-action value map after 100 learning episodes

Null hypotheses H_{10} and H_{20} were rejected with P-values equal to $2.58 \cdot 10^{-5}$ and 0 respectively; namely; the mean coefficients of learning agents are not equal. Null hypothesis H_{30} was not rejected with P-value equals to 0.643 which results an equal means of the two learning agents. Null hypotheses H_{40} and H_{50} were rejected with P-values equals to $4.52 \cdot 10^{-6}$ and $6.59 \cdot 10^{-7}$ respectively concluding a difference between the means of the learning agents. The null hypothesis H_{60} was not rejected with P-value equals to 0.824; namely; the means of these two learning agents are equal. Null hypotheses H_{70} and H_{80} were rejected with P-values equals to $1.33 \cdot 10^{-15}$ and 0 respectively; namely; there is a difference between the means of the learning agents. Fig. 5 shows an example of one of the 50 simulation runs using two robots to converge (Table 1).

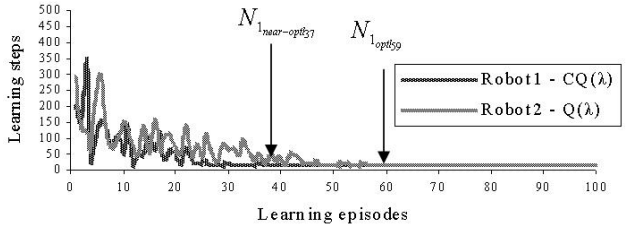

Fig. 5. Two learning agent convergence example of one simulation run

Fig. 6 shows an example of one of the 50 simulation runs using three robots to converge to optimal route (Fig. 3).

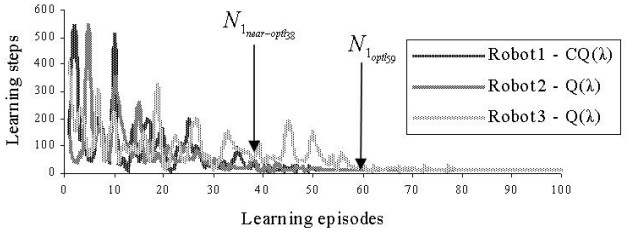

Fig. 6. Three learning agent convergence example of one simulation run

It is clearly seen both from hypotheses H_{10} through H_{80} evaluation and from Figs. 5 and 6 that the robot that uses the $CQ(\lambda)$ algorithm has superior learning performance while converging either to near optimality or to optimality with comparison to robots that use the $Q(\lambda)$ algorithm. It is also seen from Figs. 5 and 6 that initially a large number of steps is required to achieve learning because at that time the agents do not have enough knowledge about the world. As learning proceeds and the agents gain more knowledge number of steps drops dramatically.

For evaluating whether adding a third learning agent improves learning, hypotheses nine and ten were tested (Table 2). Null hypotheses H_{90} and H_{100} were not rejected with P-value equals to 0.81 and 0.436 respectively; namely; the means are equal. This means that an additional learning agent does not improve system learning significantly.

5. Discussion and Conclusions

A new type of a collaborative reinforcement learning algorithm has been developed in traditional navigation task. We demonstrated in simulation an acceleration of learning and the convergence to an optimum using

collaboration between several learning agents. We presented the design and implementation via simulation of the $CQ(\lambda)$ algorithm applied on autonomous mobile robots for navigation in an 11 x 11 two dimensional world contains obstacles and boundaries choosing the optimum path to reach a target. 50 simulation runs showed an average improvement of 17.02% while measuring the number of learning steps required reaching definite optimality and an average improvement of 32.98% for convergence to near optimality by using two robots compared with the $Q(\lambda)$ algorithm. Significant statistical difference was indicated for both convergence to optimality and convergence to near optimality while comparing two robots; the first uses $CQ(\lambda)$ and the second uses $Q(\lambda)$. While using three robots; the first uses the $CQ(\lambda)$ and the second and third use $Q(\lambda)$, we found that there is no statistical significant differences in both convergence to optimality and convergence to near optimality while comparing the $Q(\lambda)$-based robots. We found statistical significant differences in both convergence to optimality and convergence to near optimality while comparing the $CQ(\lambda)$-based robot to the other two. In addition we found that there is no statistical significant differences in both convergence to optimality and convergence to near optimality using $CQ(\lambda)$-based robot learns in a two robots environment or an $CQ(\lambda)$-based robot learns in three robots environment, i.e., we conclude that adding another robot in addition to two collaborative robots does not achieve significant improvement. We demonstrated: (i) superiority of the $CQ(\lambda)$ algorithm over the $Q(\lambda)$ and (ii) proved statistically that three learning processes have no significant advantage over two learning processes, thereby when $CQ(\lambda)$ will be tested on a real robot, only one additional learning process (*e.g.*, another robot or a human) will be applied.

Basically there is always a gap between simulated and real robot tests; the sensors and actuators are never ideal and are usually very difficult to model accurately and the operation environment is usually dynamic, etc. Future work includes a verification of the $CQ(\lambda)$ algorithm with real robots for: (i) learning the efficient lifting and shaking policies for emptying the contents of suspicious bags using a fixed arm robot [27, 28, 29, 30], and (ii) learning a navigation task using an Evolution Robotics ER-1 robot [31] when a human serves as an additional learning process.

6. Acknowledgements

This project was partially supported by the Paul Ivanier Center for Robotics Research and Production Management, Ben-Gurion University of the Negev.

References:

[1] C.J.C.H. Watkins, Learning from delayed rewards, *Ph.D. Dissertation*, Cambridge University, 1989.
[2] J. Peng, & R. Williams, Incremental multi-step Q-learning, *Machine Learning, 22*(1-3), 1996, 283-290.
[3] L.P. Kaelbling, M.L. Littman, & A.W. Moore, Reinforcement learning: a survey. *Journal of Artificial Intelligence Research, 4*, 1996. 237-285.
[4] W.D. Smart, Making reinforcement learning work on real robots, *Ph.D. Dissertation*, Brown University, 2002.
[5] C. Ribeiro, Reinforcement learning agents, *Artificial Intelligence Review, 17*(3), 2002. 223-250.
[6] A. Howard, Probabilistic navigation: coping with uncertainty in robot navigation tasks, *Ph.D. Dissertation*, Department of Computer Science and Software Engineering, University of Melbourne, Australia, 1999.
[7] B. Bhanu, P. Leang, C. Cowden, Y. Lin, & M. Patterson, Real-time robot learning. *Proc. of the 2001 IEEE International Conf. on Robotics and Automation*, Seoul, Korea, 2001, 491-498.
[8] P. Kui-Hong, J. Jun, & K. Jong-Hwan. Stabilization of biped robot based on two mode Q-learning, *Proc. of the 2nd International Conf. on Autonomous Robots and Agents*, New Zealand, 2004.
[9] T. Martínez-Marín, & T. Duckett, Fast reinforcement learning for vision-guided mobile robots. *Proc. of the 2005 IEEE International Conf. on Robotics and Automation*, Barcelona, Spain, 2005.
[10] C. Touzet, *Q-learning for robots, the handbook of brain theory and neural networks*, (Cambridge, MA: M. Arbib editor, MIT Press, 2003). 934-937.
[11] C. Weber, S. Wermter, & A. Zochios, Robot docking with neural vision and reinforcement, *Knowledge-based systems, 17*(2-4), 2004. 165-72.
[12] A. Lockerd, & C. Breazeal. Tutelage and socially guided robot learning, *Proc. of IEEE/RSJ International Conf. on Intelligent Robots and Systems*, Sendai, Japan, 2004.
[13] W. Zhu, & S. Levinson, Vision-based reinforcement learning for robot navigation, *Proc. of the international joint conf. on neural network, 2*, Washington DC., 2001, 1025-30.
[14] A.F. Massoud, & L. Caro, Fuzzy neural network implementation of Q(λ) for mobile robots. *WSEAS Transaction on Systems, 3*(1), 2004.
[15] E. Menegatti, G. Cicirelli, C. Simionato, A. Distante, & E. Pagello. Reinforcement learning based omnidirectional vision agent for mobile robot navigation, *Workshop Robotica del IX Convegno della Associazione Italiana Intelligenza Artificiale (AI*IA04)*, 2004.
[16] Y. Dahmani, & A. Benyettou, Seek of an optimal way by Q-learning, *Journal of Computer Science, 1*(1), 2005, 28-30.
[17] R. Broadbent, & T. Peterson. Robot learning in partially observable, noisy, continuous worlds, *Proc. of the 2005 IEEE International Conf. on Robotics and Automation*, Barcelona, Spain, 2005.

[18] S.M. Ross, *Introduction to stochastic dynamic programming* (New York: Academic Press, 1983).
[19] W.D. Smart, & L. Kaelbling, Practical reinforcement learning in continuous spaces, *Proc. of the 17th International Conf. on Machine Learning*, 2002.
[20] R. Bellman, & R. Kalaba, *Dynamic programming and modern control theory* (New York: Academic Press Inc., 1965).
[21] C.J.C.H. Watkins, & P. Dayan, Q-learning. *Machine Learning, 8*, 1992, 279-292.
[22] R.S. Sutton, & A.G. Barto, *Reinforcement learning: an introduction* (Cambridge, MA: MIT Press, 1998).
[23] A.K. MackWorth, D. Poole, & R.G. Goebel, *Computational intelligence: a logical approach* (Oxford University Press, 1998).
[24] P.Y. Glorennec. Reinforcement Learning: an overview. *European Symposium on Intelligent Techniques*, Aachen, Germany, 2000.
[25] S. Nason, & J.E. Laird, Soar-RL: integrating reinforcement learning with soar, *Proc. of the International Conf. on Cognitive Modeling*, 2004, 51-59.
[26] M. Guo, Y. Liu, & J. Malec, A new Q-learning algorithm based on the metropolis criterion, *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, 34*(5), 2004, 2140-2143.
[27] Y. Edan, U. Kartoun, & H. Stern. Cooperative human-robot learning system using a virtual reality telerobotic interface, *Conf. on Advances in Internet Technologies and Applications*, Purdue University, West Lafayette, Indiana, U.S.A., 2004.
[28] U. Kartoun, H. Stern, & Y. Edan, Virtual reality telerobotic system. *Proc. of the 4th International Conf. on e-Engineering and Digital Enterprise Technology*, Leeds Metropolitan University, Yorkshire, U.K., 2004, 445-454.
[29] U. Kartoun, H. Stern, & Y. Edan, Bag classification using support vector machines. *The 9th On-line World Conf. on Soft Computing in Industrial Applications*, 2004.
[30] http://www.motoman.co.uk/UP20.htm/
[31] http://www.evolution.com/er1/

Table 1. Simulation results

	Experimental model set I		Experimental model set II		
	Robot1	Robot2	Robot1	Robot2	Robot3
Learning strategy	$CQ(\lambda)$	$Q(\lambda)$	$CQ(\lambda)$	$Q(\lambda)$	$Q(\lambda)$
Mean / standard deviation of steps to converge to optimality	56.8 / 13.6	68.4 / 14.0	55.2 / 14.1	68.9 / 13.9	69.4 / 14.4
Mean / standard deviation of steps to converge to near optimality	37.7 / 7.2	56.2 / 10.1	36.8 / 6.5	56.2 / 15.6	56.6 / 11.2

Table 2. Evaluation hypotheses

Evaluation within experimental model set I learning agents	H_{10}: There is no difference between convergence to optimality of the two learning agents. H_{11}: There is a significant difference between the two learning agents.
	H_{20}: There is no difference between convergence to near optimality of the two learning agents. H_{21}: There is a significant difference between the two learning agents.
Evaluation within experimental model set II learning agents	H_{30}: There is no difference between convergence to optimality of $Q(\lambda)$ learning (Robot2) and $Q(\lambda)$ learning (Robot3). H_{31}: There is a significant difference between the two learning agents.
	H_{40}: There is no difference between convergence to optimality of $CQ(\lambda)$ learning (Robot1) and $Q(\lambda)$ learning (Robot2). H_{41}: There is a significant difference between the two learning agents.
	H_{50}: There is no difference between convergence to optimality of $CQ(\lambda)$ learning (Robot1) and $Q(\lambda)$ learning (Robot3). H_{51}: There is a significant difference between the two learning agents.
	H_{60}: There is no difference between convergence to near optimality of $Q(\lambda)$ learning (Robot2) and $Q(\lambda)$ learning (Robot3). H_{61}: There is a significant difference between the two learning agents.
	H_{70}: There is no difference between convergence to near optimality of $CQ(\lambda)$ learning (Robot1) and $Q(\lambda)$ learning (Robot2). H_{71}: There is a significant difference between the two learning agents.
	H_{80}: There is no difference between convergence to near optimality of $CQ(\lambda)$ learning (Robot1) and $Q(\lambda)$ learning (Robot3). H_{81}: There is a significant difference between the two learning agents.
Evaluation between experimental sets I and II	H_{90}: There is no difference between convergence to optimality of using two or three $CQ(\lambda)$ robots. H_{91}: There is a significant difference between the two learning models.
	H_{100}: There is no difference between convergence to optimality of using two or three $CQ(\lambda)$ robots. H_{101}: There is a significant difference between the two learning models.