Single-Agent vs. Multi-Agent Techniques for Concurrent Reinforcement Learning of Negotiation Dialogue Policies

Kallirroi Georgila, Claire Nelson, David Traum
University of Southern California Institute for Creative Technologies
12015 Waterfront Drive, Playa Vista, CA 90094, USA
{kgeorgila,traum}@ict.usc.edu

Abstract

We use single-agent and multi-agent Reinforcement Learning (RL) for learning dialogue policies in a resource allocation negotiation scenario. Two agents learn concurrently by interacting with each other without any need for simulated users (SUs) to train against or corpora to learn from. In particular, we compare the Q-learning, Policy Hill-Climbing (PHC) and Win or Learn Fast Policy Hill-Climbing (PHC-WoLF) algorithms, varying the scenario complexity (state space size), the number of training episodes, the learning rate, and the exploration rate. Our results show that generally Q-learning fails to converge whereas PHC and PHC-WoLF always converge and perform similarly. We also show that very high gradually decreasing exploration rates are required for convergence. We conclude that multi-agent RL of dialogue policies is a promising alternative to using single-agent RL and SUs or learning directly from corpora.

1 Introduction

The dialogue policy of a dialogue system decides on which actions the system should perform given a particular dialogue state (i.e., dialogue context). Building a dialogue policy can be a challenging task especially for complex applications. For this reason, recently much attention has been drawn to machine learning approaches to dialogue management and in particular Reinforcement Learning (RL) of dialogue policies (Williams and Young, 2007; Rieser et al., 2011; Jurčíček et al., 2012).

Typically there are three main approaches to the problem of learning dialogue policies using RL: (1) learn against a simulated user (SU), i.e., a model that simulates the behavior of a real user (Georgila et al., 2006; Schatzmann et al., 2006); (2) learn directly from a corpus (Henderson et al., 2008; Li et al., 2009); or (3) learn via live interaction with human users (Singh et al., 2002; Gašić et al., 2011; Gašić et al., 2013).

We propose a fourth approach: concurrent learning of the system policy and the SU policy using multi-agent RL techniques. Both agents are trained simultaneously and there is no need for building a SU separately or having access to a corpus.[1] As we discuss below, concurrent learning could potentially be used for learning via live interaction with human users. Moreover, for negotiation in particular there is one more reason in favor of concurrent learning as opposed to learning against a SU. Unlike slot-filling domains, in negotiation the behaviors of the system and the user are symmetric. They are both negotiators, thus building a good SU is as difficult as building a good system policy.

So far research on using RL for dialogue policy learning has focused on single-agent RL techniques. Single-agent RL methods make the assumption that the system learns by interacting with a stationary environment, i.e., an environment that does not change over time. Here the environment is the user. Generally the assumption that users do not significantly change their behavior over time holds for simple information providing tasks (e.g., reserving a flight). But this is not necessarily the case for other genres of dialogue, including negotiation. Imagine a situation where a negotiator is so uncooperative and arrogant that the other negotiators decide to completely change their negotiation strategy in order to punish her. Therefore it is important to investigate RL approaches that do not make such assumptions about the user/environment.

[1] Though corpora or SUs may still be useful for bootstrapping the policies and encoding real user behavior (see section 6).

Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pages 500–510,
Baltimore, Maryland, USA, June 23-25 2014.

Multi-agent RL is designed to work for non-stationary environments. In this case the environment of a learning agent is one or more other agents that can also be learning at the same time. Therefore, unlike single-agent RL, multi-agent RL can handle changes in user behavior or in the behavior of other agents participating in the interaction, and thus potentially lead to more realistic dialogue policies in complex dialogue scenarios. This ability of multi-agent RL can also have important implications for learning via live interaction with human users. Imagine a system that learns to change its strategy as it realizes that a particular user is no longer a novice user, or that a user no longer cares about five star restaurants.

We apply multi-agent RL to a resource allocation negotiation scenario. Two agents with different preferences negotiate about how to share resources. We compare Q-learning (a single-agent RL algorithm) with two multi-agent RL algorithms: Policy Hill-Climbing (PHC) and Win or Learn Fast Policy Hill-Climbing (PHC-WoLF) (Bowling and Veloso, 2002). We vary the scenario complexity (i.e., the quantity of resources to be shared and consequently the state space size), the number of training episodes, the learning rate, and the exploration rate.

Our research contributions are as follows: (1) we propose concurrent learning using multi-agent RL as a way to deal with some of the issues of current approaches to dialogue policy learning (i.e., the need for SUs and corpora), which may also potentially prove useful for learning via live interaction with human users; (2) we show that concurrent learning can address changes in user behavior over time, and requires multi-agent RL techniques and variable exploration rates; (3) to our knowledge this is the first time that PHC and PHC-WoLF are used for learning dialogue policies; (4) for the first time, the above techniques are applied to a negotiation domain; and (5) this is the first study that compares Q-learning, PHC, and PHC-WoLF in such a variety of situations (varying a large number of parameters).

The paper is structured as follows. Section 2 presents related work. Section 3 provides a brief introduction to single-agent RL and multi-agent RL. Section 4 describes our negotiation domain and experimental setup. In section 5 we present our results. Finally, section 6 concludes and provides some ideas for future work.

2 Related Work

Most research in RL for dialogue management has been done in the framework of slot-filling applications such as restaurant recommendations (Lemon et al., 2006; Thomson and Young, 2010; Gašić et al., 2012; Daubigney et al., 2012), flight reservations (Henderson et al., 2008), sightseeing recommendations (Misu et al., 2010), appointment scheduling (Georgila et al., 2010), etc. RL has also been applied to question-answering (Misu et al., 2012), tutoring domains (Tetreault and Litman, 2008; Chi et al., 2011), and learning negotiation dialogue policies (Heeman, 2009; Georgila and Traum, 2011; Georgila, 2013).

As mentioned in section 1, there are three main approaches to the problem of learning dialogue policies using RL.

In the first approach, a SU is hand-crafted or learned from a small corpus of human-human or human-machine dialogues. Then the dialogue policy can be learned by having the system interact with the SU for a large number of dialogues (usually thousands of dialogues). Depending on the application, building a realistic SU can be just as difficult as building a good dialogue policy. Furthermore, it is not clear what constitutes a good SU for dialogue policy learning. Should the SU resemble real user behavior as closely as possible, or should it exhibit some degree of randomness to explore a variety of interaction patterns? Despite much research on the issue, these are still open questions (Schatzmann et al., 2006; Ai and Litman, 2008; Pietquin and Hastie, 2013).

In the second approach, no SUs are required. Instead the dialogue policy is learned directly from a corpus of human-human or human-machine dialogues. For example, Henderson et al. (2008) used a combination of RL and supervised learning to learn a dialogue policy in a flight reservation domain, whereas Li et al. (2009) used Least-Squares Policy Iteration (Lagoudakis and Parr, 2003), an RL-based technique that can learn directly from corpora, in a voice dialer application. However, collecting such corpora is not trivial, especially in new domains. Typically, data are collected in a Wizard-of-Oz setup where human users think that they interact with a system while in fact they interact with a human pretending to be the system, or by having human users interact with a preliminary version of the dialogue system. In both cases the resulting interactions are expected to be quite dif-

ferent from the interactions of human users with the final system. In practice this means that dialogue policies learned from such data could be far from optimal.

The first experiment on learning via live interaction with human users (third approach) was reported by Singh et al. (2002). They used RL to help the system with two choices: how much initiative it should allow the user, and whether or not to confirm information provided by the user. Recently, learning of "full" dialogue policies (not just choices at specific points in the dialogue) via live interaction with human users has become possible with the use of Gaussian processes (Engel et al., 2005; Rasmussen and Williams, 2006). Typically learning a dialogue policy is a slow process requiring thousands of dialogues, hence the need for SUs. Gaussian processes have been shown to speed up learning. This fact together with easy access to a large number of human users through crowd-sourcing has allowed dialogue policy learning via live interaction with human users (Gašić et al., 2011; Gašić et al., 2013).

Space constraints prevent us from providing an exhaustive list of previous work on using RL for dialogue management. Thus below we focus only on research that is directly related to our work, specifically research on concurrent learning of the policies of multiple agents, and the application of RL to negotiation domains.

So far research on RL in the dialogue community has focused on using single-agent RL techniques where the stationary environment is the user. Most approaches assume that the user goal is fixed and that the behavior of the user is rational. Other approaches account for changes in user goals (Ma, 2013). In either case, one can build a user simulation model that is the average of different user behaviors or learn a policy from a corpus that contains a variety of interaction patterns, and thus safely assume that single-agent RL techniques will work. However, in the latter case if the behavior of the user changes significantly over time then the assumption that the environment is stationary will no longer hold.

There has been a lot of research on multi-agent RL in the optimal control and robotics communities (Littman, 1994; Hu and Wellman, 1998; Busoniu et al., 2008). Here two or more agents learn simultaneously. Thus the environment of an agent is one or more other agents that continuously change their behavior because they are also learning at the same time. Therefore the environment is no longer stationary and single-agent RL techniques do not work well or do not work at all. We are particularly interested in the work of Bowling and Veloso (2002) who proposed the PHC and PHC-WoLF algorithms that we use in this paper. We chose these two algorithms because, unlike other multi-agent RL methods (Littman, 1994; Hu and Wellman, 1998), they do not make assumptions that do not always hold and do not require quadratic or linear programming that does not always scale.

English and Heeman (2005) were the first in the dialogue community to explore the idea of concurrent learning of dialogue policies. However, English and Heeman (2005) did not use multi-agent RL but only standard single-agent RL, in particular an on-policy Monte Carlo method (Sutton and Barto, 1998). But single-agent RL techniques are not well suited for concurrent learning where each agent is trained against a continuously changing environment. Indeed, English and Heeman (2005) reported problems with convergence. Chandramohan et al. (2012) proposed a framework for co-adaptation of the dialogue policy and the SU using single-agent RL. They applied Inverse Reinforcement Learning (IRL) (Abbeel and Ng, 2004) to a corpus in order to learn the reward functions of both the system and the SU. Furthermore, Cuayáhuitl and Dethlefs (2012) used hierarchical multi-agent RL for co-ordinating the verbal and non-verbal actions of a robot. Cuayáhuitl and Dethlefs (2012) did not use PHC or PHC-WoLF and did not compare against single-agent RL methods.

With regard to using RL for learning negotiation policies, the amount of research that has been performed is very limited compared to slot-filling. English and Heeman (2005) learned negotiation policies for a furniture layout task. Then Heeman (2009) extended this work by experimenting with different representations of the RL state in the same domain (this time learning against a hand-crafted SU). In both cases, to reduce the search space, the RL state included only information about e.g., whether there was a pending proposal rather than the actual value of this proposal. Paruchuri et al. (2009) performed a theoretical study on how Partially Observable Markov Decision Processes (POMDPs) can be applied to negotiation domains.

Georgila and Traum (2011) built argumentation dialogue policies for negotiation against users of different cultural norms in a one-issue negotiation scenario. To learn these policies they trained SUs on a spoken dialogue corpus in a florist-grocer negotiation domain, and then tweaked these SUs towards a particular cultural norm using hand-crafted rules. Georgila (2013) learned argumentation dialogue policies from a simulated corpus in a two-issue negotiation scenario (organizing a party). Finally, Nouri et al. (2012) used IRL to learn a model for cultural decision-making in a simple negotiation game (the Ultimatum Game).

3 Single-Agent vs. Multi-Agent Reinforcement Learning

Reinforcement Learning (RL) is a machine learning technique used to learn the policy of an agent, i.e., which action the agent should perform given its current state (Sutton and Barto, 1998). The goal of an RL-based agent is to maximize the reward it gets during an interaction. Because it is very difficult for the agent to know what will happen in the rest of the interaction, the agent must select an action based on the average reward it has previously observed after having performed that action in similar contexts. This average reward is called *expected future reward*. Single-agent RL is used in the framework of Markov Decision Processes (MDPs) (Sutton and Barto, 1998) or Partially Observable Markov Decision Processes (POMDPs) (Williams and Young, 2007). Here we focus on MDPs.

An MDP is defined as a tuple (S, A, T, R, γ) where S is the set of states (representing different contexts) which the agent may be in, A is the set of actions of the agent, T is the transition function $S \times A \times S \rightarrow$ [0, 1] which defines a set of transition probabilities between states after taking an action, R is the reward function $S \times A \rightarrow \Re$ which defines the reward received when taking an action from the given state, and γ is a factor that discounts future rewards. Solving the MDP means finding a policy $\pi : S \rightarrow A$. The quality of the policy π is measured by the expected discounted (with discount factor γ) future reward also called Q-value, $Q^{\pi} : S \times A \rightarrow \Re$.

A *stochastic game* is defined as a tuple (n, S, $A_{1...n}$, T, $R_{1...n}$, γ) where n is the number of agents, S is the set of states, A_i is the set of actions available for agent i (and A is the joint action space $A_1 \times A_2 \times ... \times A_n$), T is the transition function $S \times A \times S \rightarrow$ [0, 1] which defines a set of transition probabilities between states after taking a joint action, R_i is the reward function for the ith agent $S \times A \rightarrow \Re$, and γ is a factor that discounts future rewards. The goal is for each agent i to learn a mixed policy $\pi_i : S \times A_i \rightarrow$ [0, 1] that maps states to mixed strategies, which are probability distributions over the agent's actions, so that the agent's expected discounted (with discount factor γ) future reward is maximized.

Stochastic games are a generalization of MDPs for multi-agent RL. In stochastic games there are many agents that select actions and the next state and rewards depend on the joint action of all these agents. The agents can have different reward functions. Partially Observable Stochastic Games (POSGs) are the equivalent of POMDPs for multi-agent RL. In POSGs, the agents have different observations, and uncertainty about the state they are in and the beliefs of their interlocutors. POSGs are very hard to solve but new algorithms continuously emerge in the literature.

In this paper we use three algorithms: Q-learning, Policy Hill-Climbing (PHC), and Win or Learn Fast Policy Hill-Climbing (PHC-WoLF). PHC is an extension of Q-learning. For all three algorithms, Q-values are updated as follows:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a)+\alpha\left(r+\gamma \text{max}_{a'}Q(s^{'},a^{'})\right) \quad (1)$$

In Q-learning, for a given state s, the agent performs the action with the highest Q-value for that state. In addition to Q-values, PHC and PHC-WoLF also maintain the current mixed policy $\pi(s,a)$. In each step the mixed policy is updated by increasing the probability of selecting the highest valued action according to a learning rate δ (see equations (2), (3), and (4) below).

$$\pi(s,a) \leftarrow \pi(s,a) + \Delta_{sa} \quad (2)$$

$$\Delta_{sa} = \begin{cases} -\delta_{sa} & \text{if } a \neq \text{argmax}_{a'}Q(s,a^{'}) \\ \Sigma_{a' \neq a}\delta_{sa'} & \text{otherwise} \end{cases} \quad (3)$$

$$\delta_{sa} = \min\left(\pi(s,a), \frac{\delta}{|A_i|-1}\right) \quad (4)$$

The difference between PHC and PHC-WoLF is that PHC uses a constant learning rate δ whereas

PHC-WoLF uses a variable learning rate (see equation (5) below). The main idea is that when the agent is "winning" the learning rate δ_W should be low so that the opponents have more time to adapt to the agent's policy, which helps with convergence. On the other hand when the agent is "losing" the learning rate δ_{LF} should be high so that the agent has more time to adapt to the other agents' policies, which also facilitates convergence. Thus PHC-WoLF uses two learning rates δ_W and δ_{LF}. PHC-WoLF determines whether the agent is "winning" or "losing" by comparing the current policy's $\pi(s,a)$ expected payoff with that of the average policy $\tilde{\pi}(s,a)$ over time. If the current policy's expected payoff is greater then the agent is "winning", otherwise it is "losing".

$$\delta = \begin{cases} \delta_W & \text{if } \begin{cases} \Sigma_{\alpha'} \pi(s,\alpha') Q(s,\alpha') > \\ \Sigma_{\alpha'} \tilde{\pi}(s,\alpha') Q(s,\alpha') \end{cases} \\ \delta_{LF} & \text{otherwise} \end{cases} \quad (5)$$

More details about Q-learning, PHC, and PHC-WoLF can be found in (Sutton and Barto, 1998; Bowling and Veloso, 2002).

As discussed in sections 1 and 2, single-agent RL techniques, such as Q-learning, are not suitable for multi-agent RL. Nevertheless, despite its shortcomings Q-learning has been used successfully for multi-agent RL (Claus and Boutilier, 1998). Indeed, as we see in section 5, Q-learning can converge to the optimal policy for small state spaces. However, as the state space size increases the performance of Q-learning drops (compared to PHC and PHC-WoLF).

4 Domain and Experimental Setup

Our domain is a resource allocation negotiation scenario. Two agents negotiate about how to share resources. For the sake of readability from now on we will refer to apples and oranges.

The two agents have different goals. Also, they have human-like constraints of imperfect information about each other; they do not know each other's reward function or degree of rationality (during learning our agents can be irrational). Thus a Nash equilibrium (if there exists one) cannot be computed in advance. Agent 1 cares more about apples and Agent 2 cares more about oranges. Table 1 shows the points that Agents 1 and 2 earn for each apple and each orange that they have at the end of the negotiation.

	Agent 1	**Agent 2**
apple	300	200
orange	200	300

Table 1: Points earned by Agents 1 and 2 for each apple and each orange that they have at the end of the negotiation.

Agent 1: offer-2-2 (I offer you 2 A and 2 O)
Agent 2: offer-3-0 (I offer you 3 A and 0 O)
Agent 1: offer-0-3 (I offer you 0 A and 3 O)
Agent 2: offer-4-0 (I offer you 4 A and 0 O)
Agent 1: accept (I accept your offer)

Figure 1: Example interaction between Agents 1 and 2 (A: apples, O: oranges).

We use a simplified dialogue model with two types of speech acts: offers and acceptances. The dialogue proceeds as follows: one agent makes an offer, e.g., "I give you 3 apples and 1 orange", and the other agent may choose to accept it or make a new offer. The negotiation finishes when one of the agents accepts the other agent's offer or time runs out.

We compare Q-learning with PHC and PHC-WoLF. For all algorithms and experiments each agent is rewarded only at the end of the dialogue based on the negotiation outcome (see Table 1). Thus the two agents have different reward functions. There is also a penalty of -10 for each agent action to ensure that dialogues are not too long. Also, to avoid long dialogues, if none of the agents accepts the other agent's offers, the negotiation finishes after 20 pairs of exchanges between the two agents (20 offers from Agent 1 and 20 offers from Agent 2).

An example interaction between the two agents is shown in Figure 1. As we can see, each agent can offer any combination of apples and oranges. So if we have X apples and Y oranges for sharing, there can be $(X+1) \times (Y+1)$ possible offers. For example if we have 2 apples and 2 oranges for sharing, there can be 9 possible offers: "offer-0-0", "offer-0-1", ..., "offer-2-2". For our experiments we vary the number of fruits to be shared and choose to keep X equal to Y.

Table 2 shows our state representation, i.e., the state variables that we keep track of with all the possible values they can take, where X is the num-

Current offer: $(X + 1) \times (Y + 1)$ possible values
How many times the current offer has already been rejected: (0, 1, 2, 3, or 4)
Is the current offer accepted: yes, no

Table 2: State variables.

ber of apples and Y is the number of oranges to be shared. The third variable is always set to "no" until one of the agents accepts the other agent's offer.

Table 3 shows the state and action space sizes for different numbers of apples and oranges to be shared used in our experiments below. The number of actions includes the acceptance of an offer. Table 3 also shows the number of state-action pairs (Q-values). As we will see in section 5, even though the number of states for each agent is not large, it takes many iterations and high exploration rates for convergence due to the fact that both agents are learning at the same time and the assumption of interacting with a stationary environment no longer holds. For comparison, in (English and Heeman, 2005) the state specification for each agent included 5 binary variables resulting in 32 possible states. English and Heeman (2005) kept track of whether there was an offer on the table but not of the actual value of the offer. For our task it is essential to keep track of the offer values, which of course results in much larger state spaces. Also, in (English and Heeman, 2005) there were 5 possible actions resulting in 160 state-action pairs. Our state and action spaces are much larger and furthermore we explore the effect of different state and action space sizes on convergence.

During learning the two agents interact for 5 epochs. Each epoch contains N number of episodes. We vary N from 25,000 up to 400,000 with a step of 25,000 episodes. English and Heeman (2005) trained their agents for 200 epochs, where each epoch contained 200 episodes.

We also vary the exploration rate per epoch. In particular, in the experiments reported in section 5.1 the exploration rate is set as follows: 0.95 for epoch 1, 0.8 for epoch 2, 0.5 for epoch 3, 0.3 for epoch 4, and 0.1 for epoch 5. Section 5.2 reports results again with 5 epochs of training but a constant exploration rate per epoch set to 0.3. An exploration rate of 0.3 means that 30% of the time the agent will select an action randomly.

Finally, we vary the learning rate. For PHC-WoLF we set δ_W = 0.05 and δ_{LF} = 0.2 (see section 3). These values were chosen with experimentation and the basic idea is that the agent should learn faster when "losing" and slower when "winning". For PHC we explore two cases. In the first case which from now on will be referred to as PHC-W, we set δ to be equal to δ_W (also used for PHC-WoLF). In the second case which from now on will be referred to as PHC-LF, we set δ to be equal to δ_{LF} (also used for PHC-WoLF). So unlike PHC-WoLF, PHC-W and PHC-LF do not use a variable learning rate. PHC-W always learns slowly and PHC-LF always learns fast.

	#States	**#Actions**	**#State-Action Pairs**
1 A & O	40	5	200
2 A & O	90	10	900
3 A & O	160	17	2720
4 A & O	250	26	6500
5 A & O	360	37	13320
6 A & O	490	50	24500
7 A & O	640	65	41600

Table 3: State space, action space, and state-action space sizes for different numbers of apples and oranges to be shared (A: apples, O: oranges).

In all the above cases, training stops after 5 epochs. Then we test the learned policies against each other for one more epoch the size of which is the same as the size of the epochs used for training. For example, if the policies were learned for 5 epochs with each epoch containing 25,000 episodes, then for testing the two policies will interact for another 25,000 episodes. For comparison, English and Heeman (2005) had their agents interact for 5,000 dialogues during testing. To ensure that the policies do not converge by chance, we run the training and test sessions 20 times each and we report averages. Thus all results presented in section 5 are averages of 20 runs.

5 Results

Given that Agent 1 is more interested in apples and Agent 2 cares more about oranges, the maximum total utility solution would be the case where each agent offers to get all the fruits it cares about and to give its interlocutor all the fruits it does not care about, and the other agent accepts this offer. Thus, when converging to the maximum total utility solution, in the case of 4 fruits (4 ap-

ples and 4 oranges), the average reward of the two agents should be 1200 minus 10 for making or accepting an offer. For 5 fruits the average reward should be 1500 minus 10, and so forth. We call 1200 (or 1500) the *convergence reward*, i.e., the reward after converging to the maximum total utility solution if we do not take into account the action penalty. For example, in the case of 4 fruits, if Agent 1 starts the negotiation, after converging to the maximum total utility solution the optimal interaction should be: Agent 1 makes an offer to Agent 2, namely 0 apples and 4 oranges, and Agent 2 accepts. Thus the reward for Agent 1 is 1190, the reward for Agent 2 is 1190, and the average reward of the two agents is also 1190. Also, the convergence reward for Agent 1 is 1200 and the convergence reward for Agent 2 is also 1200.

Below, in all the graphs that we provide, we show the average distance from the convergence reward. This is to make all graphs comparable because in all cases the optimal average distance from the convergence reward of the two agents should be equal to 10 (make the optimal offer or accept the optimal offer that the other agent makes). The formulas for calculating the average distance from the convergence reward are:

$$AD_1 = \frac{\sum_{j=1}^{n_r} |CR_1 - R_{1j}|}{n_r} \quad (6)$$

$$AD_2 = \frac{\sum_{j=1}^{n_r} |CR_2 - R_{2j}|}{n_r} \quad (7)$$

$$AD = \frac{AD_1 + AD_2}{2} \quad (8)$$

where CR_1 is the convergence reward for Agent 1, R_{1j} is the reward of Agent 1 for run j, CR_2 is the convergence reward for Agent 2, and R_{2j} is the reward of Agent 2 for run j. Moreover, AD_1 is the average distance from the convergence reward for Agent 1, AD_2 is the average distance from the convergence reward for Agent 2, and AD is the average of AD_1 and AD_2. All graphs of section 5 show AD values. Also, n_r is the number of runs (in our case always equal to 20). Thus in the case of 4 fruits, we will have CR_1=CR_2=1200, and if for all runs R_{1j}=R_{2j}=1190, then AD=10.

5.1 Variable Exploration Rate

In this section we report results with different exploration rates per training epoch (see section 4).

	Q-learning	**PHC-LF**	**PHC-W**	**PHC-WoLF**
1 A & O	10.5	10	10	10
2 A & O	10.3	10.3	10	10
3 A & O	11.7	10	10	10
4 A & O	15	11.8	11.7	11.7
5 A & O	45.4	29.5	26.5	22.9
6 A & O	60.8	33.4	46.1	33.9
7 A & O	95	56	187.8	88.6

Table 4: Average distance from convergence reward over 20 runs for 100,000 episodes per epoch and for different numbers of fruits to be shared (A: apples, O: oranges). The best possible value is 10.

Table 4 shows the average distance from the convergence reward over 20 runs for 100,000 episodes per epoch, for different numbers of fruits, and for all four methods (Q-learning, PHC-LF, PHC-W, and PHC-WoLF). It is clear that as the state space becomes larger 100,000 training episodes per epoch are not enough for convergence. Also, for 1, 2, and 3 fruits all algorithms converge and perform comparably. As the number of fruits increases, Q-learning starts performing worse than the multi-agent RL algorithms. For 7 fruits PHC-W appears to perform worse than Q-learning but this is because, as we can see in Figure 5, in this case more than 400,000 episodes per epoch are required for convergence. Thus after only 100,000 episodes per epoch all policies still behave somewhat randomly.

Figures 2, 3, 4, and 5 show the average distance from the convergence reward as a function of the number of episodes per epoch during training, for 4, 5, 6, and 7 fruits respectively. For 4 fruits it takes about 125,000 episodes per epoch and for 5 fruits it takes about 225,000 episodes per epoch for the policies to converge. This number rises to approximately 350,000 for 6 fruits and becomes even higher for 7 fruits. Q-learning consistently performs worse than the rest of the algorithms. The differences between PHC-LF, PHC-W, and PHC-WoLF are insignificant, which is a bit surprising given that Bowling and Veloso (2002) showed that PHC-WoLF performed better than PHC in a series of benchmark tasks. In Figures 2 and 3, PHC-LF appears to be reaching convergence slightly faster than PHC-W and PHC-WoLF but this is not statistically significant.

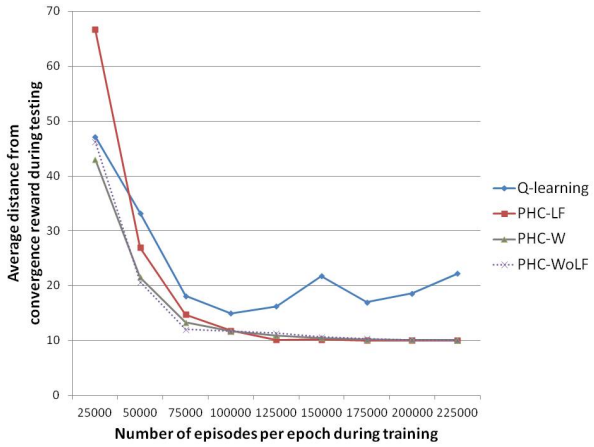

Figure 2: 4 fruits and variable exploration rate: Average distance from convergence reward during testing (20 runs). The best possible value is 10.

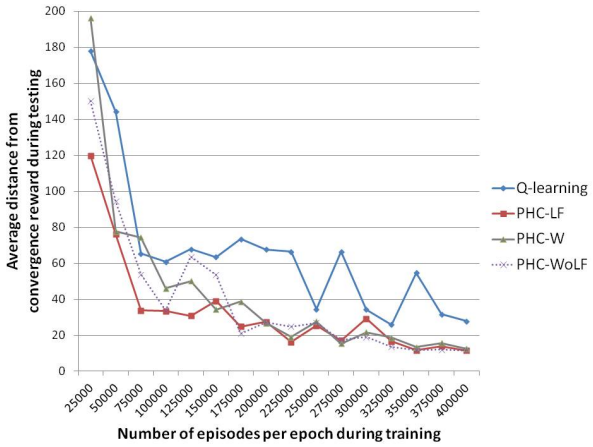

Figure 4: 6 fruits and variable exploration rate: Average distance from convergence reward during testing (20 runs). The best possible value is 10.

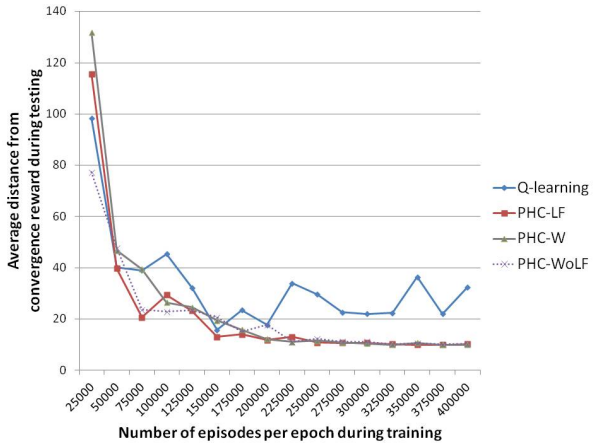

Figure 3: 5 fruits and variable exploration rate: Average distance from convergence reward during testing (20 runs). The best possible value is 10.

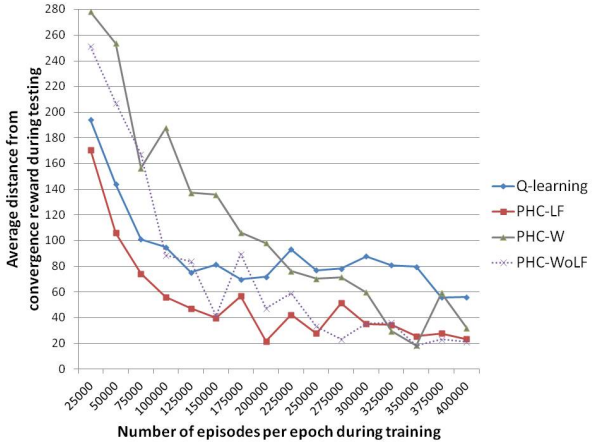

Figure 5: 7 fruits and variable exploration rate: Average distance from convergence reward during testing (20 runs). The best possible value is 10.

5.2 Constant Exploration Rate

In this section we report results with a constant exploration rate for all training epochs (see section 4). Figures 6 and 7 show the average distance from the convergence reward as a function of the number of episodes per epoch during training, for 4 and 5 fruits respectively. Clearly having a constant exploration rate in all epochs is problematic. For 4 fruits, after 225,000 episodes per epoch there is still no convergence. For comparison, with a variable exploration rate it took about 125,000 episodes per epoch for the policies to converge. Likewise for 5 fruits. After 400,000 episodes per epoch there is still no convergence. For comparison, with a variable exploration rate it took about 225,000 episodes per epoch for convergence.

The above results show that, unlike single-agent RL where having a constant exploration rate is perfectly acceptable, here a constant exploration rate does not work.

6 Conclusion and Future Work

We used single-agent RL and multi-agent RL for learning dialogue policies in a resource allocation negotiation scenario. Two agents interacted with each other and both learned at the same time. The advantage of this approach is that it does not require SUs to train against or corpora to learn from.

We compared a traditional single-agent RL algorithm (Q-learning) against two multi-agent RL algorithms (PHC and PHC-WoLF) varying the scenario complexity (state space size), the number

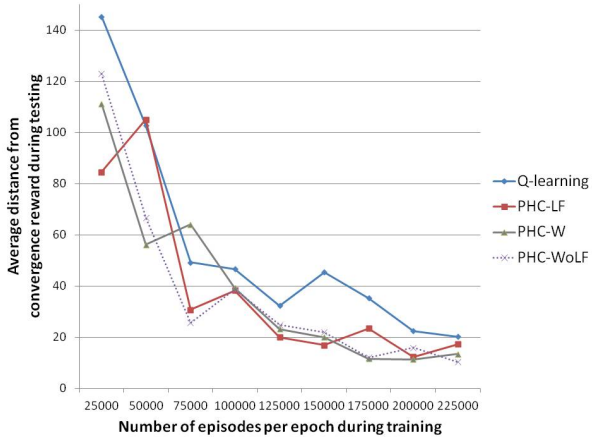

Figure 6: 4 fruits and constant exploration rate: Average distance from convergence reward during testing (20 runs). The best possible value is 10.

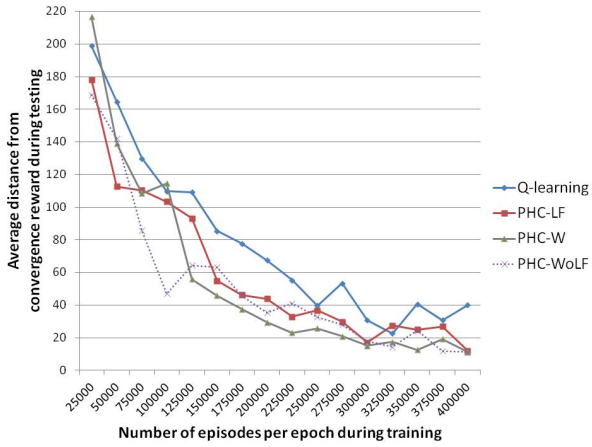

Figure 7: 5 fruits and constant exploration rate: Average distance from convergence reward during testing (20 runs). The best possible value is 10.

of training episodes, and the learning and exploration rates. Our results showed that Q-learning is not suitable for concurrent learning given that it is designed for learning against a stationary environment. Q-learning failed to converge in all cases, except for very small state space sizes. On the other hand, both PHC and PHC-WoLF always converged (or in the case of 7 fruits they needed more training episodes) and performed similarly. We also showed that in concurrent learning very high gradually decreasing exploration rates are required for convergence. We conclude that multi-agent RL of dialogue policies is a promising alternative to using single-agent RL and SUs or learning directly from corpora.

The focus of this paper is on comparing single-agent RL and multi-agent RL for concurrent learning, and studying the implications for convergence and exploration/learning rates. Our next step is testing with human users. We are particularly interested in users whose behavior changes during the interaction and continuous testing against expert repeat users, which has never been done before. Another interesting question is whether corpora or SUs may still be required for designing the state and action spaces and the reward functions of the interlocutors, bootstrapping the policies, and ensuring that information about the behavior of human users is encoded in the resulting learned policies. Gašić et al. (2013) showed that it is possible to learn "full" dialogue policies just via interaction with human users (without any bootstrapping using corpora or SUs). Similarly, concurrent learning could be used in an on-line fashion via live interaction with human users. Or alternatively concurrent learning could be used off-line to bootstrap the policies and then these policies could be improved via live interaction with human users (again using concurrent learning to address possible changes in user behavior). These are open research questions for future work.

Furthermore, we intend to apply multi-agent RL to more complex negotiation domains, e.g., experiment with more than two types of resources (not just apples and oranges) and more types of actions (not just offers and acceptances). We would also like to compare policies learned with multi-agent RL techniques with policies learned with SUs or from corpora both in simulation and with human users. Finally, we aim to experiment with different feature-based representations of the state and action spaces. Currently all possible deal combinations are listed as possible actions and as elements of the state, which can quickly lead to very large state and action spaces as the application becomes more complex (in our case as the number of fruits increases). However, abstraction is not trivial because the agents have no guarantee that the value of a deal is a simple function of the value of its parts, and values may differ for different agents.

Acknowledgments

Claire Nelson sadly died in May 2013. We continued and completed this work after her passing away. She is greatly missed. This work was funded by the NSF grant #1117313.

References

Pieter Abbeel and Andrew Y. Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proc. of the International Conference on Machine Learning*, Bannf, Alberta, Canada.

Hua Ai and Diane Litman. 2008. Assessing dialog system user simulation evaluation measures using human judges. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*, Columbus, Ohio, USA.

Michael Bowling and Manuela Veloso. 2002. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250.

L. Busoniu, R. Babuska, and B. De Schutter. 2008. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 38(2):156–172.

Senthilkumar Chandramohan, Matthieu Geist, Fabrice Lefèvre, and Olivier Pietquin. 2012. Co-adaptation in spoken dialogue systems. In *Proc. of the International Workshop on Spoken Dialogue Systems*, Paris, France.

Min Chi, Kurt VanLehn, Diane Litman, and Pamela Jordan. 2011. Empirically evaluating the application of reinforcement learning to the induction of effective and adaptive pedagogical strategies. *User Modeling and User-Adapted Interaction*, 21(1-2):137–180.

Caroline Claus and Craig Boutilier. 1998. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proc. of the National Conference on Artificial Intelligence*.

Heriberto Cuayáhuitl and Nina Dethlefs. 2012. Hierarchical multiagent reinforcement learning for coordinating verbal and nonverbal actions in robots. In *Proc. of the ECAI Workshop on Machine Learning for Interactive Systems*, Montpellier, France.

Lucie Daubigney, Matthieu Geist, Senthilkumar Chandramohan, and Olivier Pietquin. 2012. A comprehensive reinforcement learning framework for dialogue management optimization. *IEEE Journal of Selected Topics in Signal Processing*, 6(8):891–902.

Yaakov Engel, Shie Mannor, and Ron Meir. 2005. Reinforcement learning with Gaussian processes. In *Proc. of the International Conference on Machine Learning*, Bonn, Germany.

Michael S. English and Peter A. Heeman. 2005. Learning mixed initiative dialogue strategies by using reinforcement learning on both conversants. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, Vancouver, Canada.

M. Gašić, Filip Jurčíček, Blaise Thomson, Kai Yu, and Steve Young. 2011. On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In *Proc. of the IEEE Automatic Speech Recognition and Understanding Workshop*, Big Island, Hawaii, USA.

Milica Gašić, Matthew Henderson, Blaise Thomson, Pirros Tsiakoulis, and Steve Young. 2012. Policy optimisation of POMDP-based dialogue systems without state space compression. In *Proc. of the IEEE Workshop on Spoken Language Technology*, Miami, Florida, USA.

M. Gašić, C. Breslin, M. Henderson, D. Kim, M. Szummer, B. Thomson, P. Tsiakoulis, and S. Young. 2013. On-line policy optimisation of Bayesian spoken dialogue systems via human interaction. In *Proc. of the International Conference on Acoustics, Speech and Signal Processing*, Vancouver, Canada.

Kallirroi Georgila and David Traum. 2011. Reinforcement learning of argumentation dialogue policies in negotiation. In *Proc. of Interspeech*, Florence, Italy.

Kallirroi Georgila, James Henderson, and Oliver Lemon. 2006. User simulation for spoken dialogue systems: Learning and evaluation. In *Proc. of Interspeech*, Pittsburgh, Pennsylvania, USA.

Kallirroi Georgila, Maria K. Wolters, and Johanna D. Moore. 2010. Learning dialogue strategies from older and younger simulated users. In *Proc. of the Annual SIGdial Meeting on Discourse and Dialogue*, Tokyo, Japan.

Kallirroi Georgila. 2013. Reinforcement learning of two-issue negotiation dialogue policies. In *Proc. of the Annual SIGdial Meeting on Discourse and Dialogue*, Metz, France.

Peter A. Heeman. 2009. Representing the reinforcement learning state in a negotiation dialogue. In *Proc. of the IEEE Automatic Speech Recognition and Understanding Workshop*, Merano, Italy.

James Henderson, Oliver Lemon, and Kallirroi Georgila. 2008. Hybrid reinforcement/supervised learning of dialogue policies from fixed datasets. *Computational Linguistics*, 34(4):487–511.

Junling Hu and Michael P. Wellman. 1998. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proc. of the International Conference on Machine Learning*, Madison, Wisconsin, USA.

Filip Jurčíček, Blaise Thomson, and Steve Young. 2012. Reinforcement learning for parameter estimation in statistical spoken dialogue systems. *Computer Speech and Language*, 26(3):168–192.

Michail G. Lagoudakis and Ronald Parr. 2003. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149.

Oliver Lemon, Kallirroi Georgila, and James Henderson. 2006. Evaluating effectiveness and portability of reinforcement learned dialogue strategies with real users: The TALK TownInfo evaluation. In *Proc. of the IEEE Workshop on Spoken Language Technology*, Palm Beach, Aruba.

Lihong Li, Jason D. Williams, and Suhrid Balakrishnan. 2009. Reinforcement learning for dialog management using least-squares policy iteration and fast feature selection. In *Proc. of Interspeech*, Brighton, United Kingdom.

Michael L. Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Proc. of the International Conference on Machine Learning*, New Brunswick, New Jersey, USA.

Yi Ma. 2013. User goal change model for spoken dialog state tracking. In *Proc. of the NAACL-HLT Student Research Workshop*, Atlanta, Georgia, USA.

Teruhisa Misu, Komei Sugiura, Kiyonori Ohtake, Chiori Hori, Hideki Kashioka, Hisashi Kawai, and Satoshi Nakamura. 2010. Modeling spoken decision making dialogue and optimization of its dialogue strategy. In *Proc. of the Annual SIGdial Meeting on Discourse and Dialogue*, Tokyo, Japan.

Teruhisa Misu, Kallirroi Georgila, Anton Leuski, and David Traum. 2012. Reinforcement learning of question-answering dialogue policies for virtual museum guides. In *Proc. of the Annual SIGdial Meeting on Discourse and Dialogue*, Seoul, South Korea.

Elnaz Nouri, Kallirroi Georgila, and David Traum. 2012. A cultural decision-making model for negotiation based on inverse reinforcement learning. In *Proc. of the Cognitive Science Conference*, Sapporo, Japan.

P. Paruchuri, N. Chakraborty, R. Zivan, K. Sycara, M. Dudik, and G. Gordon. 2009. POMDP based negotiation modeling. In *IJCAI Workshop on Modeling Intercultural Collaboration and Negotiation*, Pasadena, California, USA.

Olivier Pietquin and Helen Hastie. 2013. A survey on metrics for the evaluation of user simulations. *Knowledge Engineering Review*, 28(1):59–73.

Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian Processes for Machine Learning*. MIT Press.

Verena Rieser, Simon Keizer, Xingkun Liu, and Oliver Lemon. 2011. Adaptive information presentation for spoken dialogue systems: Evaluation with human subjects. In *Proc. of the European Workshop on Natural Language Generation*, Nancy, France.

Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *Knowledge Engineering Review*, 21(2):97–126.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research*, 16:105–133.

Richard S. Sutton and Andrew G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

Joel R. Tetreault and Diane J. Litman. 2008. A reinforcement learning approach to evaluating state representations in spoken dialogue systems. *Speech Communication*, 50(8-9):683–696.

Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech and Language*, 24(4):562–588.

Jason D. Williams and Steve Young. 2007. Scaling POMDPs for spoken dialog management. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(7):2116–2129.