原著論文

Original Paper

Reinforcement Learning of Multi-Party Trading Dialog Policies

Takuya Hiraoka — Nara Institute of Science and Technology*1
takuya-h@is.naist.jp

Kallirroi Georgila — University of Southern California Institute for Creative Technologies
kgeorgila@ict.usc.edu

Elnaz Nouri — (ditto)
nouri@ict.usc.edu

David Traum — (ditto)
traum@ict.usc.edu

Satoshi Nakamura — Nara Institute of Science and Technology
s-nakamura@is.naist.jp

keywords: dialog policy, trading dialog, negotiation dialog, multi-party dialog, reinforcement learning

Summary

Trading dialogs are a kind of negotiation in which an exchange of ownership of items is discussed, and these kinds of dialogs are pervasive in many situations. Recently, there has been an increasing amount of research on applying reinforcement learning (RL) to negotiation dialog domains. However, in previous research, the focus was on negotiation dialog between two participants only, ignoring cases where negotiation takes place between more than two interlocutors. In this paper, as a first study on multi-party negotiation, we apply RL to a multi-party trading scenario where the dialog system (learner) trades with one, two, or three other agents. We experiment with different RL algorithms and reward functions. We use Q-learning with linear function approximation, least-squares policy iteration, and neural fitted Q iteration. In addition, to make the learning process more efficient, we introduce an incremental reward function. The negotiation strategy of the learner is learned through simulated dialog with trader simulators. In our experiments, we evaluate how the performance of the learner varies depending on the RL algorithm used and the number of traders. Furthermore, we compare the learned dialog policies with two strong hand-crafted baseline dialog policies. Our results show that (1) even in simple multi-party trading dialog tasks, learning an effective negotiation policy is not a straightforward task and requires a lot of experimentation; and (2) the use of neural fitted Q iteration combined with an incremental reward function produces negotiation policies as effective or even better than the policies of the two strong hand-crafted baselines.

1. Introduction

Trading dialogs are a kind of interaction in which an exchange of ownership of items is discussed, possibly resulting in an actual exchange. These kinds of dialogs are pervasive in many situations, such as marketplaces, business deals, school lunchrooms, and some kinds of games, like Monopoly or Settlers of Catan [Guhe 12]. Most of these dialogs are non-cooperative [Asher 13, Traum 08], in the sense that mere recognition of the desire for one party to engage in a trade does not provide sufficient inducement for the other party to accept the trade. Usually a trade will only be accepted if it is in the perceived interest of each party. Trading dialogs can be considered as a kind of negotiation, in which participants use various tactics to try to reach an agreement. It is common to have dialogs that may involve multiple offers or even multiple trades. In this way, trading dialogs are different from other sorts of negotiation in which a single decision (possibly about multiple issues) is considered, for example partitioning a set of items [Georgila 14, Nouri 13]. Another difference between trading dialogs and partitioning dialogs is what happens when a deal is not made. In partitioning dialogs, if an agreement is not reached, then participants get nothing, so there is a very strong incentive to reach a deal, which allows pressure and can result in a "chicken game", where people give up value in order to avoid a total loss. By contrast, in trading dialogs, if no deal is made,

*1 Currently the first author (Takuya Hiraoka) is working at the Nippon Electric Company. This work was done while he was a Ph.D. student at the Nara Institute of Science and Technology, and a visiting researcher at the University of Southern California Institute for Creative Technologies.

participants stick with the status quo. Competitive two-party trading dialogs may result in a kind of stasis, where the wealthier party will pass up mutually beneficial deals, in order to maintain primacy. On the other hand, multi-party trading dialogs involving more than two participants changes the dynamic again, because now a single participant cannot necessarily even block another from acquiring a missing resource, because it might be available through trades with a third party. A player who does not engage in deals may lose relative position, if the other participants make mutually beneficial deals.

The main goal of our research is to build dialog systems that can negotiate and trade with humans or other agents in the real or virtual world. Imagine a dialog system performing the role of a human in online auctions or marketplaces. In such situations, the system should be able to negotiate with other participants with different values of items; some of these participants may have goals similar to the goals of the system (collaborative), and some may have goals that compete with the system's goals (competitive). In order for the dialog system to be successful in trading and to decide on the right action in a particular context, it needs to have a strong dialog policy.

In this paper, we present a first approach toward learning dialog policies for multi-party trading dialogs. We introduce a simple, but flexible game-like scenario, where items can have different values for different participants, and also where the value of an item can depend on other items held. We examine a number of strategies for this game, including random, simple, and complex hand-crafted strategies, as well as several reinforcement learning (RL) [Sutton 98] algorithms, and examine performance with different numbers and kinds of opponents.

In most of the previous work on statistical dialog management, RL was applied to cooperative slot-filling dialog domains. For example, RL was used to learn the policies of dialog systems for food ordering [Williams 07a], tourist information [Williams 07b], flight information [Levin 00], appointment scheduling [Georgila 10], and e-mail access [Walker 00]. In these typical slot-filling dialog systems, the reward function depends on whether the user's goal has been accomplished or not. For example, in the food ordering system presented by Williams and Young [Williams 07a], the dialog system earns higher rewards when it succeeds in taking the order from the user.

Recently, there has been an increasing amount of research on applying RL to negotiation dialog domains, which are generally more complex than slot-filling dialog because the system needs to consider its own goal as well as the user's goal, and may need to keep track of more information, e.g., what has been accepted or rejected so far, proposals and arguments on the table, etc. Georgila and Traum [Georgila 11] applied RL to the problem of learning negotiation dialog system policies for different cultural norms (individualists, collectivists, and altruists). The domain was negotiation between a florist and a grocer who had to agree on the temperature of a shared retail space. Georgila [Georgila 13] used RL to learn the dialog system policy in a two-issue negotiation domain where two participants (the user and the system) organize a party, and need to decide on both the day that the party will take place and the type of food that will be served. Then Papangelis and Georgila [Papangelis 15] extended this work and learned dialog policies in a four-issue negotiation scenario. Also, Heeman [Heeman 09] modeled negotiation dialog for a furniture layout task, and Paruchuri et al. [Paruchuri 09] modeled negotiation dialog between a seller and a buyer. Efstathiou and Lemon [Efstathiou 14] focused on non-cooperative aspects of trading dialog, and Georgila et al. [Georgila 14] used multi-agent RL to learn negotiation policies in a resource allocation scenario. Finally, Hiraoka et al. [Hiraoka 14] applied RL to the problem of learning cooperative persuasive policies using framing, and Nouri et al. [Nouri 12] learned models for cultural decision-making in a simple negotiation game (the Ultimatum Game). In contrast to typical slot-filling dialog systems, in these negotiation dialogs, the dialog system is rewarded based on the achievement of its own goals rather than those of its interlocutor. For example, in Georgila [Georgila 13], the dialog system gets a higher reward when its party plan is accepted by the other participant.

Note that in all of the previous work mentioned above, the focus was on negotiation dialog between two participants only, ignoring cases where negotiation takes place between more than two interlocutors. However, in the real world, multi-party negotiation is quite common. In this paper, as a first study on multi-party negotiation, we apply RL to a multi-party trading scenario where the dialog system (learner) trades with one, two, or three other agents. We experiment with different RL algorithms and reward functions. The negotiation strategy of the learner is learned through simulated dialog with trader simulators. In our experiments, we evaluate how the performance of the learner varies depending on the RL algorithm used and the number of traders. To the best of our knowledge this is the first study that applies RL to multi-party (more than two participants) negotiation dialog management. We are not aware of any previous research on dialog using RL to learn the system's policy in multi-party negotiation.*2

Our paper is structured as follows. Chapter 2 provides an introduction to RL. Chapter 3 describes our multi-party trading domain. Chapter 4 describes the dialog state and set of actions for both the learner and the trader simulators, as well as the reward functions of the learner and the hand-crafted policies of the trader simulators. In Chapter 5, we present our evaluation methodology and results. Finally, Chapter 6 summarizes the paper and proposes future work.

2. Reinforcement Learning

Reinforcement learning (RL) is a machine learning technique for learning the policy of an agent that takes some action to maximize a reward (not only immediate but also long-term or delayed reward). In this section, we briefly describe RL in the context of dialog management. In dialog, the policy is a mapping function from a dialog state to a particular system action. In RL, the policy's goal is to maximize a reward function, which in traditional task-based dialog systems is user satisfaction or task completion [Jokinen 09]. RL is applied to dialog modeling in the framework of Markov decision processes (MDPs) or partially observable Markov decision processes (POMDPs).

In this paper, we follow an MDP-based approach. An MDP is defined as a tuple $\langle S, A, P, R, \gamma \rangle$ where S is the set of states (representing different contexts) which the system may be in (the system's world), A is the set of actions of the system, $P : S \times A \rightarrow P(S, A)$ is the set of transition probabilities between states after taking an action, $R : S \times A \rightarrow \Re$ is the reward function, and $\gamma \in [0,\ 1]$ a discount factor weighting long-term rewards. At any given time step i the world is in some state $s_i \in S$. When the system performs an action $\alpha_i \in A$ following a policy $\pi : S \rightarrow A$, it receives a reward $r_i(s_i, \alpha_i) \in \Re$ and transitions to state s_{i+1} according to $P(s_{i+1}|s_i, \alpha_i) \in P$. The quality of the policy π followed by the agent is measured by the *expected future reward*, also called Q-function, $Q^\pi : S \times A \rightarrow \Re$.

We experiment with 3 different RL algorithms:

LinQ: This is the basic Q-learning algorithm with linear function approximation [Sutton 98]. The Q-function is a weighted function of state-action features. It is updated whenever the system performs an action and gets a reward for that action (in contrast to batch RL mentioned below).

LSPI: In least-squares policy iteration (LSPI), the Q-function is also approximated by a linear function (similarly to LinQ). However, unlike LinQ, LSPI is a batch learning method. It samples the training data one or more times (batches) using a fixed system policy (the policy that has been learned so far), and the approximated Q-function is updated after each batch. We use LSPI because it has been shown to achieve higher performance than LinQ in tasks where states are represented as multidimensional vectors, such as the problem of riding a bicycle [Lagoudakis 03]. In our domain, the agents have to keep track of information about multiple traders, and this information is encoded into a multidimensional vector that represents the dialog state (see Section 4·1). Therefore we expect LSPI to work better than LinQ in our domain as well.

NFQ: Neural fitted Q iteration (NFQ) uses a multi-layered perceptron as the Q-function approximator. Like LSPI, NFQ is a batch learning method. We introduce NFQ because it has been shown to perform well in some tasks [Riedmiller 05]. One such task is robot soccer [Riedmiller 09], where the state is represented as a multidimensional vector with a size larger than that of the bicycle problem where LSPI is successful. In our domain, as the number of traders increases, the size of the vector that represents the state becomes larger too. Therefore we expect NFQ to achieve better performance in trading domains with many traders.

During training we use ϵ-greedy exploration, i.e., the system randomly selects an action with a probability of ϵ (we used a value of 0.1 for ϵ) otherwise it selects the action which maximizes the Q-function given the current state. During testing there is no exploration and the policy is dictated by the Q-values learned during training.

3. Multi-Party Trading Domain

Our domain is trading, where two or more traders have a number of items that they can keep or exchange with the other traders in order to achieve their goals. The value of each item for a trader is dictated by the trader's payoff matrix. So at the end of the interaction each trader earns a number of points based on the items that it holds and the value of each item. Note that each trader has its own payoff matrix. During the interaction, each trader can

*2 Note that there is some previous work on using RL to learn negotiation policies among more than two participants. For example, Mayya et al. [Mayya 11] and Zou et al. [Zou 14] used multi-agent RL to learn the negotiation policies of sellers and buyers in a marketplace. Moreover, Pfeiffer [Pfeiffer 04] used RL to learn policies for board games where sometimes negotiation takes place among players. However, these works did not focus on negotiation dialog (i.e., exchange of dialog acts, such as offers and responses to offers), but only focused on specific problems of marketing or board games. For example, in Zou et al. [Zou 14]'s work, RL was used to learn policies for setting selling/purchasing prices in order to achieve good payoffs.

trade an item with the other traders (i.e., offer an item in exchange for another item). If the addressee of the offer accepts it, then the items of the traders involved in this exchange are updated. If the offer is not accepted, the dialog proceeds without any changes in the number of items that each trader possesses. To make the search space of possible optimal trading policies more tractable, we assume that (1) each trader can only trade one item at a time, (2) only one trader is allowed to take the turn (decide to trade) at a time, and (3) each offer is addressed only to one other trader. Each trader can take the turn in random order, unless there is a pending offer. That is, if a trader makes an offer to another trader, then the addressee of that offer has priority to take the next turn; the addressee can decide to accept the offer, or to do nothing, or to make a different offer. Note that the traders do not know each other's payoff matrices but they know the items that each trader owns. The dialog is completed after a fixed period of time passes or when all traders decide not to make any offers.

In our experiments, there are three types of items: apple, orange, and grape, and each trader may like, hate, or feel neutral about each type of fruit. At the end of the dialog the trader earns 100 points for each fruit that he likes, 0 points for each fruit that he is neutral to, and –100 points for each fruit that he hates. Payoff matrices are structured such that there is always one fruit that each trader likes, one fruit that he is neutral to, and one fruit that he hates. Furthermore, all traders can get a big payoff for having a fruit salad, i.e., the trader earns 500 additional points if he ends up with one fruit of each type. Thus even hated fruits may sometimes be beneficial, but only if they can be part of a fruit salad. Thus the outcome for a trader o_{tr} is calculated by Equation (1).

$$\begin{aligned} o_{tr} = \; & Pay(\mathrm{apple_{tr}}) * Num(\mathrm{apple_{tr}}) \\ & + Pay(\mathrm{orange_{tr}}) * Num(\mathrm{orange_{tr}}) \\ & + Pay(\mathrm{grape_{tr}}) * Num(\mathrm{grape_{tr}}) \\ & + Pay(\mathrm{salad_{tr}}) \end{aligned} \tag{1}$$

$$Pay(\mathrm{salad_{tr}}) = \begin{cases} 500 & \text{if } Num(\mathrm{apple_{tr}}) \geq 1 \\ & \text{and } Num(\mathrm{orange_{tr}}) \geq 1 \\ & \text{and } Num(\mathrm{grape_{tr}}) \geq 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where Pay is a function which takes as argument a fruit type and returns the value of that fruit type for the trader, and Num shows the number of items of a particular fruit type that the trader possesses. At the beginning of each dialog, the initial conditions (i.e., number of items per fruit type and payoff matrix) of the traders (except for the learner) are randomly assigned. The learner always has the same payoff matrix for all dialogs, i.e., the learner always likes grape, always feels neutral about apple, and always hates orange. Also, the total number of fruits that the learner holds in the beginning of the dialog is always 3. However, the number of each fruit type that the learner holds is randomly initialized for each dialog, e.g., the learner could be initialized with (1 apple, 2 oranges, 0 grapes), or (1 apple, 1 orange, 1 grape), etc. The total number of fruits for each trader is determined based on his role (Rich: 4 items, Middle: 3 items, Poor: 2 items), which is also randomly assigned at the beginning of each dialog. Table 1 shows two example dialogs.

4. Methodology for Learning Multi-Party Negotiation Policies

In this chapter, we present our methodology for training the learner, including how we built our trader simulators. The trader simulators are used as negotiation partners of the learner for both training and evaluating the learner's policy (see Chapter 5).

4·1 Learner's Model

Below we define the reward function, sets of actions, and state of our MDP-based learner's model. Note that we use two kinds of rewards.

The first type of reward is based on Equation (3). In this case, the learner is rewarded based on its outcome only at the end of the dialog. In all other dialog turns i its reward is 0.

$$r_{end} = \begin{cases} o_{tr} & \text{if dialog ends} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

We also introduce an *incremental reward* for training, because rewarding a learning agent only at the end of the dialog makes the learning problem very difficult, thus subgoals can be utilized to reward the learning agent incrementally [McGovern 01]. The incremental reward at turn i is given by Equation (4), where $o_{tr}(i)$ is the outcome for a trader applied at time point i.

$$r_i' = \begin{cases} \gamma * o_{tr}(i) - o_{tr}(i-1) & \text{if } i > 0 \\ 0 & \text{if } i = 0 \end{cases} \tag{4}$$

This equation represents the improvement on the outcome of the learner at turn i compared to its outcome at the previous turn $i-1$. Note that this implementation of the incremental reward function is basically the same as reward

Table 1 Examples of two trading dialogs among traders TR1, TR2, and TR3. In these examples, the payoff matrix of TR1 is (apple: –100, orange: 100, grape: 0), that of TR2 is (apple: –100, orange: 0, grape: 100), and that of TR3 is (apple: 0, orange: –100, grape: 100). Item and Outcome show the number of items per fruit type of each trader and the points that each trader has accumulated after an action. A stands for apple, O for orange, and G for grape.

		Item			Outcome		
Speaker	Utterance	TR1	TR2	TR3	TR1	TR2	TR3
Dialog 1:							
1: TR1	TR2, could you give me an orange? I'll give you a grape. (Offer)	A: 0, O: 0, G: 3	A: 1, O: 1, G: 0	A: 0, O: 1, G: 2	0	–100	100
2: TR2	Okay. (Accept)	A: 0, O: 1, G: 2	A: 1, O: 0, G: 1	A: 0, O: 1, G: 2	100	0	100
Dialog 2:							
1: TR2	TR1, could you give me a grape? I'll give you an apple. (Offer)	A: 0, O: 0, G: 3	A: 1, O: 1, G: 0	A: 0, O: 1, G: 2	0	–100	100
2: TR1	I want to keep my fruits. (Keep)	A: 0, O: 0, G: 3	A: 1, O: 1, G: 0	A: 0, O: 1, G: 2	0	–100	100
3: TR3	TR2, could you give me an apple? I'll give you a grape. (Offer)	A: 0, O: 0, G: 3	A: 1, O: 1, G: 0	A: 0, O: 1, G: 2	0	–100	100
4: TR2	Okay. (Accept)	A: 0, O: 0, G: 3	A: 0, O: 1, G: 1	A: 1, O: 1, G: 1	0	100	500

shaping, and has the following property [Ng 99, El Asri 13]: the policy learned by using Equation (4) maximizes the expectation of the cumulative reward given by Equation (3).

The learner's actions are presented below. By speaker we mean the trader who is performing the action. In this case, the speaker is the learner, but as we will see below this is also the set of actions that a trader simulator can perform.

Offer(A, I_s, I_a): offering addressee A to trade the speaker's item I_s for the addressee's item I_a.

Accept: accepting the most recent offer addressed to the speaker.

Keep: passing the turn without doing anything. If there is a pending offer addressed to the speaker, then this offer is rejected.

The dialog state consists of the *offered table* and the distribution of the items among the negotiators:

Offered table: The offered table consists of all possible tuples (Trading partner, Fruit requested, Fruit offered in return). If another agent makes an offer to the learner then the learner's offered table is updated. The dialog state is represented by binary variables (or features). In Example 1, we can see a dialog state in a 2-party dialog, after the learner receives an offer to give an orange and in return take an apple.

Number of items: The number of items for each fruit type that each negotiator possesses. Once a trade is performed, this part of the dialog state is updated in the dialog states of all agents involved in this trade.

Thus the learner's state is represented as a multidimensional vector that consists of (1) binary values representing the offered table, and (2) integer values representing the number of items. To simplify the model, the state does not include information about previous actions of the traders.

4·2 Trader Simulator

In order to train the learner we need trader simulators to generate a variety of trading episodes, so that in the end the learner learns to follow actions that lead to high rewards and avoid actions that lead to penalties. The trader simulator has the same dialog state and actions as the learner. We have as many trader simulators as traders that the learner negotiates with. Thus in a 3-party negotiation we have 2 trader simulators. The policy of the trader simulator can be either hand-crafted, designed to maximize the reward function given by Equation (3); or random.

The hand-crafted policy is based on planning. More concretely, this policy selects an action based on the following steps:

(1) Pre-compute all possible sets of items (called "hands", by analogy with card games, where each item is represented by a card), given the role of the trader (Rich, Middle, Poor) and how many items there can be in the hand.

(2) Compute the valuation of each of the hands, according to the payoff matrix.

(3) Based on the possible trades with the other agents, compute a set of achievable hands, and order them according to the valuations defined in step 2. A hand is "achievable" if there are enough of the right types of items in the deal. For example, if the hand is 4 apples, and there are only 3 apples in the deal, then this hand is not achievable.

(4) Remove all hands that have a lower valuation than

Example1 Status of the learner's dialog state's features in a 2-party trading dialog (learner vs. Agent 1). Agent 1 has just offered the learner 1 apple for 1 of the learner's 2 oranges (but the learner has not accepted or rejected the offer yet). This is why the (Agent 1, orange, apple) tuple has value 1. Initially the learner has (0 apples, 2 oranges, 1 grape) and Agent 1 has (1 apple, 0 oranges, 1 grape). Note that if we had more negotiators e.g., Agent 2, the dialog state would include features for offer tuples for Agent 2, and the number of items that Agent 2 possessed.

Trading partner	Item requested by partner	Item given by partner to learner	Occurrence binary value (used as feature)
Agent 1	apple	orange	0
	apple	grape	0
	orange	apple	1
	orange	grape	0
	grape	apple	0
	grape	orange	0

Agent who possesses fruits	Fruit type	Number of fruits (used as feature)
learner	apple	0
	orange	2
	grape	1
Agent 1	apple	1
	orange	0
	grape	1

the current hand. The remaining set is the set of achievable goals.

(5) Calculate a set of plans for each achievable goal. A plan is a sequence of trades (one item in hand for one item out of hand) that will lead to the goal. There are many possible plans for each goal. For simplicity, we ignore any plans that involve cycles, where the same hand appears more than once.

(6) Calculate the expected utility (outcome) of each plan. Each plan will have a probability distribution of outcomes, based on the probability that each trade is successful. The outcome will be the hand that results from the end state, or the state before the trade that fails. For example, suppose the simulator's hand is (apple, apple, orange), and the simulator's plan is (apple→orange, orange→grape). The three possible outcomes are:

(apple, orange, grape) (i.e., if the plan succeeds) the probability is calculated as $P(t1) * P(t2)$.

(apple, orange, orange) (i.e., if the first trade succeeds and the second fails) the probability is calculated as $P(t1) * (1 - P(t2))$.

(apple, apple, orange) (i.e., if the first trade fails) the probability is calculated as $1 - P(t1)$.

Therefore, the simulator can calculate the expected utility of each plan, by multiplying the probability of each trade with the valuation of each hand from step 2. We set the probability of success of each trade to 0.5 (i.e., uninformative probability). This value of probability represents the fact that the simulator does not know a priori whether the trade will succeed or not.

(7) Select the plan which has the highest expected utility as the plan that the policy will follow.

(8) Select an action implementing the plan that was chosen in the previous step, as follows: if the plan is completed (i.e., the simulator reached the goal), the policy will select Keep as an action. If the plan is not completed and there is a pending offer which will allow the plan to move forward, the policy will select Accept as an action. Otherwise, the policy will select Offer as an action. The addressee of the offer is randomly selected from the traders holding the item which is required for moving the plan forward. The pseudocode of this step is shown in Algorithm 1.

In addition to the above hand-crafted trader simulator's policy, we also use a random policy.

5. Evaluation

In this chapter, we evaluate the learner's policies learned with (1) different algorithms i.e., LinQ, LSPI, and NFQ (see Chapter 2), (2) different reward functions i.e., Equations (3) and (4) (see Section 4·1), and (3) different numbers of traders.[*3]

The evaluation is performed in trading dialogs with different numbers of participants (from 2 players to 4 players), and different trader simulator's policies (hand-crafted policy or random policy as presented in Section 4·2). More specifically, there are 9 different setups:

H: 2-party dialog, where the trader simulator follows a hand-crafted policy.

R: 2-party dialog, where the trader simulator follows a random policy.

HxH: 3-party dialog, where both trader simulators follow hand-crafted policies.

∗3 Source files to replicate these experiments are available at: `https://github.com/TakuyaHiraoka/Reinforcement-Learning-in-Multi-Party-Trading-Dialog`

Algorithm 1 Action selection of the hand-crafted policy

Require: The following variables are given:

(1) Index of the trader i.

(2) Set of other traders T^{-i}.

(3) Trader's plan $P = [[I_g^0, I_r^0], ..., [I_g^j, I_r^j], ...]$, where I_g^j represents the item that the trader will give in the jth trade, and I_r^j represents the item that the trader will receive in return in the jth trade.

(4) Offered table O that returns a binary value according to the given tuple (Trading partner, Fruit requested, Fruit offered in return).

(5) Number of items N that returns an integer value according to the given tuple (Trader, Fruit).

(6) Set of possible addressees A for the first trade of the plan.

```
if P = φ then
    return Keep
end if
for t ∈ T^{-i} do
    if O[(t, I_g^0, I_r^0)] == 1 then
        return Accept
    end if
end for
A ← {}
for t ∈ T^{-i} do
    if N[(t, I_r^0)] >= 1 then
        A ← t ∪ A
    end if
end for
randomly select an addressee of an offer a from A
return Offer(a, I_g^0, I_r^0)
```

HxR: 3-party dialog, where one trader simulator follows a hand-crafted policy and the other one follows a random policy.

RxR: 3-party dialog, where both trader simulators follow random policies.

HxHxH: 4-party dialog, where all three trader simulators follow hand-crafted policies.

HxHxR: 4-party dialog, where two trader simulators follow hand-crafted policies and the other one follows a random policy.

HxRxR: 4-party dialog, where one trader simulator follows a hand-crafted policy and the other ones follow random policies.

RxRxR: 4-party dialog, where all three trader simulators follow random policies.

There are also 9 different learner policies:

AlwaysKeep: weak baseline which always passes the turn.

Random: weak baseline which randomly selects one action from all possible valid actions.

LinQ-End: learned policy using LinQ and reward given at the end of the dialog.

LSPI-End: learned policy using LSPI and reward given at the end of the dialog.

NFQ-End: learned policy using NFQ and reward given at the end of the dialog.

LinQ-Incr: learned policy using LinQ and an incremental reward.

LSPI-Incr: learned policy using LSPI and an incremental reward.

NFQ-Incr: learned policy using NFQ and an incremental reward.

Handcraft1: strong baseline following the hand-crafted policy presented in Section 4·2.

Handcraft2: strong baseline similar to Handcraft1 except the plan is randomly selected from the set of plans produced by step 6, rather than picking only the highest utility one (see Section 4·2).

We use the Pybrain library [Schaul 10] for the RL algorithms LinQ, LSPI, and NFQ. The learning parameters follow the default Pybrain settings except for the discount factor γ; we set the discount factor γ to 1. We consider 2000 dialogs as one epoch, and learning is finished when the number of epochs becomes 200 (400000 dialogs). The policy at the epoch where the average reward reaches its highest value is used in the evaluation.

We evaluate the learner's policy against trader simulators. We calculate the average reward of the learner's policy in 20000 dialogs. Furthermore, we show how fast the learned policies converge as a function of the number of epochs in training.

In terms of comparing the average rewards of policies (see Figure 1), NFQ-Incr achieves the best performance in almost every situation. In 2-party trading, the performance of NFQ-Incr is almost the same as that of Handcraft2 which achieves the best score, and better than the performance of Handcraft1. In both 3-party and 4-party trading, the performance of NFQ-Incr is better than that of the two strong baselines, and achieves the best score. In contrast to NFQ-Incr, the performance of the other learned policies is much worse than that of the two strong baselines. As the number of trader simulators who follow a random policy increases, the difference in performance between NFQ-Incr and the other learned policies tends to also increase. One reason is that, as the number of trader simulators who follow a random policy increases, the variability of dialog flow also increases. Trader simulators that follow a hand-crafted policy behave more strictly than trader simulators that follow a random policy. For example, if the trader simulator following a hand-crafted policy reaches its goal, then there is nothing else to do except for Keep. In contrast, if a trader simulator following a random policy reaches its goal, there is still a chance that it will make an offer which will be beneficial to the learner. As a result there are more chances for the learner to gain better outcomes, when the complexity of the dialog is higher. In summary, our results show that combining NFQ with an incremental reward produces the best results.

Moreover, the learning curve in 2-party trading (Figure 2) indicates that, basically, only the NFQ-Incr achieves stable learning. NFQ-Incr reaches its best performance from epoch 140 to epoch 190. On the other hand, LSPI somehow converges fast, but its performance is not so high. Moreover, LinQ converges in the first epoch, but it performs the worst.

Note that the above results depend on the experimental setup, especially the properties of the trader simulators. If different types of simulators had been used then the experimental results might have been different.

Table 2 shows the average number (per dialog) of each one of the learner's actions (Offer, Accept, Keep). It also shows the average number of the offers of the learner that are accepted by the other traders, the average number of all actions performed by the learner, and the average number of all turns (all actions performed by all traders), per dialog, over 20000 dialogs in the 3-party situation. First, for each policy, we focus on the distribution of each one of its actions in HxH, and we can see that the average number of NFQ-Incr's "Accepted offers" is 0.350, which is the highest among all policies (except for the random policy). This result indicates that when NFQ-Incr trades with opponents (trader simulators) that follow hand-crafted policies, it makes appropriate offers that consider its opponents' needs (or plans). The hand-crafted trader simulators always accept reasonable offers that are required for proceeding with their plans. NFQ-Incr has learned to exploit this property of the hand-crafted trader simulators in order to achieve better outcomes. Next, we focus on the distribution of each one of NFQ-Incr's actions in dialogs with trader simulators that follow a random policy. From Table 2, we can see that, as the number of trader simulators that follow a random policy increases, the average number of NFQ-Incr's "Accept" actions also increases. The average number of NFQ-Incr's "Accept" actions in HxH is 0.001, but it becomes 0.329 in HxR and 0.775 in RxR. This shows that trader simulators that follow random policies tend to offer trades profitable to the learner, and NFQ-Incr learns to exploit this property in order to achieve better outcomes.

It is interesting that, even in this simple trading domain, applying RL does not always produce good policies. Without the incremental reward of Equation (4), all the RL algorithms that we use fail to learn policies comparable to or better than the two strong baselines (Figure 1). Even when the incremental reward is used, our RL algorithms (except for NFQ) fail to learn successful policies. This shows that our domain is quite complex, and certainly not easier than other domains to which RL has been successfully applied so far, e.g., slot-filling.

To measure the validity and portability of our learned policies, we perform additional experiments in which the trader simulators follow different policies in the training and evaluation phases:

H'xH': the learner's policy is learned in HxH, but the learned policy is evaluated in 3-party dialog where both trader simulators follow Handcraft2.

H'xR: the learner's policy is learned in HxR, but the learned policy is evaluated in 3-party dialog where one trader simulator follows Handcraft2 and the other one follows a random policy.

Below we discuss our evaluation results (Figure 3, Table 3, and Figure 4), with regard to the validity and portability of our learned policies (especially NFQ-Incr).

Validity: In Figure 3, we can see that, in contrast to the experimental result shown in Figure 1, NFQ-Incr performs slightly worse than the two strong baselines. One reason for this is that NFQ-Incr is learned against trader simulators following Handcraft1, and thus these learned policies could be overfitted to these trader simulators. NFQ-Incr is very conservative while interacting with other trader simulators in HxH. NFQ-Incr does not accept most of the offers of the trader simulators (Table 2), but it works better than the two strong baselines in this setting (Figure 1). In contrast, in H'xH' the two strong baselines that accept more offers from the other simulators than NFQ-Incr (Table 3), achieve better performance than NFQ-Incr (Figure 3). This shows that a conservative strategy works better against Handcraft1 than Handcraft2. To avoid such overfitting and produce policies that can perform well in various trading domains, learning should be performed against trader simulators that follow a variety of policies.

Portability: Although NQF-Incr does not outperform the two strong baselines in H'xH' and H'xR, it performs much better than the weak baselines (Figure 3). Therefore, if the learner is required to adapt to a new trading situation, we can potentially use the policy learned

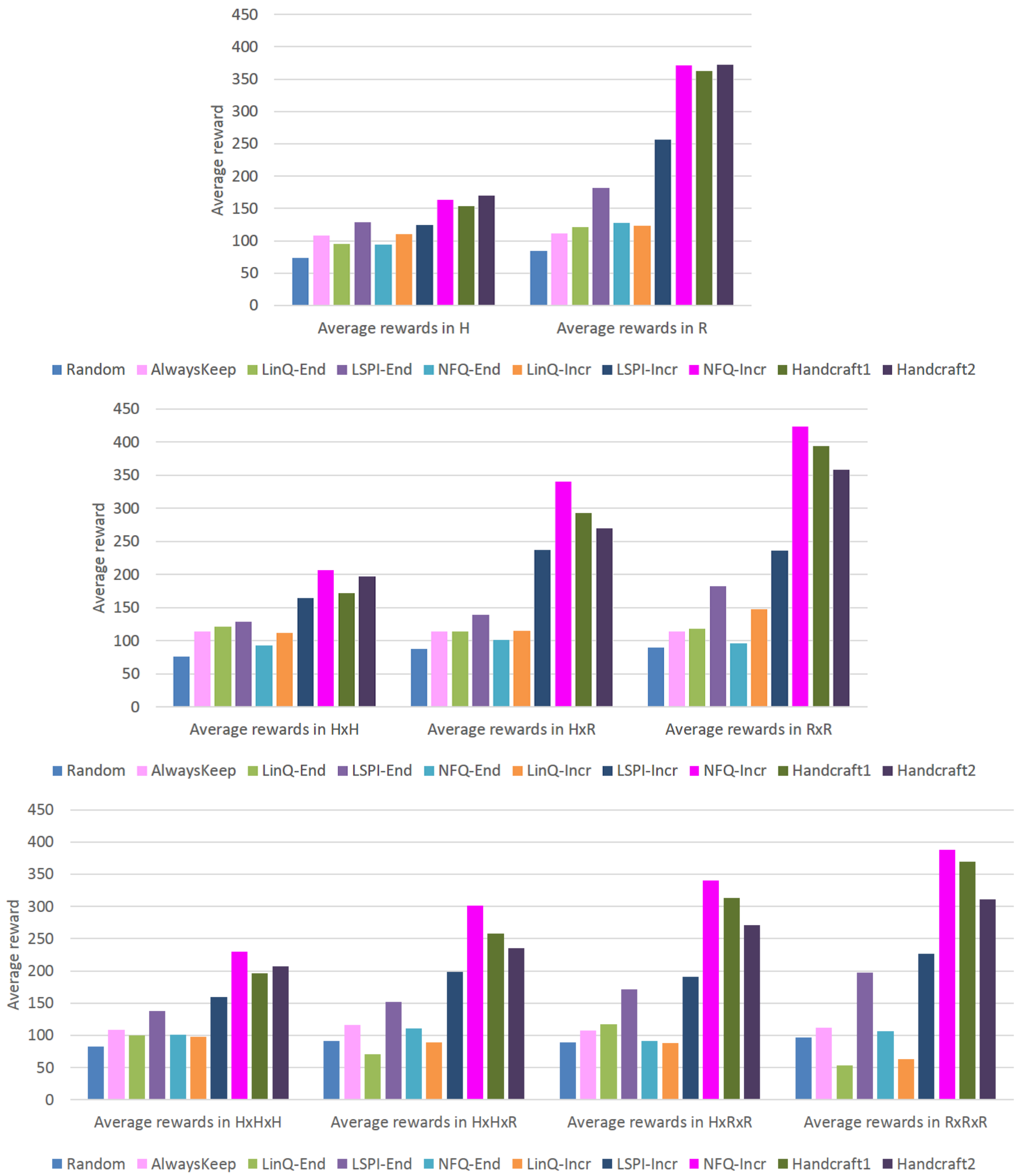

Fig. 1 Comparison of RL algorithms and types of reward functions. The upper figure corresponds to 2-party dialog, the middle figure to 3-party dialog, and the lower figure to 4-party dialog. In these figures, the performances of the policies are evaluated by using the reward function given by Equation (3).

with NFQ-Incr as an initial policy of the learner. We perform a preliminary experiment to evaluate the effect of transferring a learned policy to a new domain, which results in faster convergence during learning and policies that are as good or even better than Handcraft1 (Figure 4).

In our experiments, we focus on trading dialogs among a small number of traders, and thus we do not need to apply scaling approaches to the state/action space of each trader. However, as the number of traders increases, the state/action space of each trader can become large and intractable. It is possible that a large state space includes redundant information. In such a case, to scale the state space to a tractable size, it makes sense to consider only important information. For example, in our trading domain, if we know that a trader can perform well based only on information about (1) the current offer (i.e., what fruit is requested and what fruit is offered in return), and

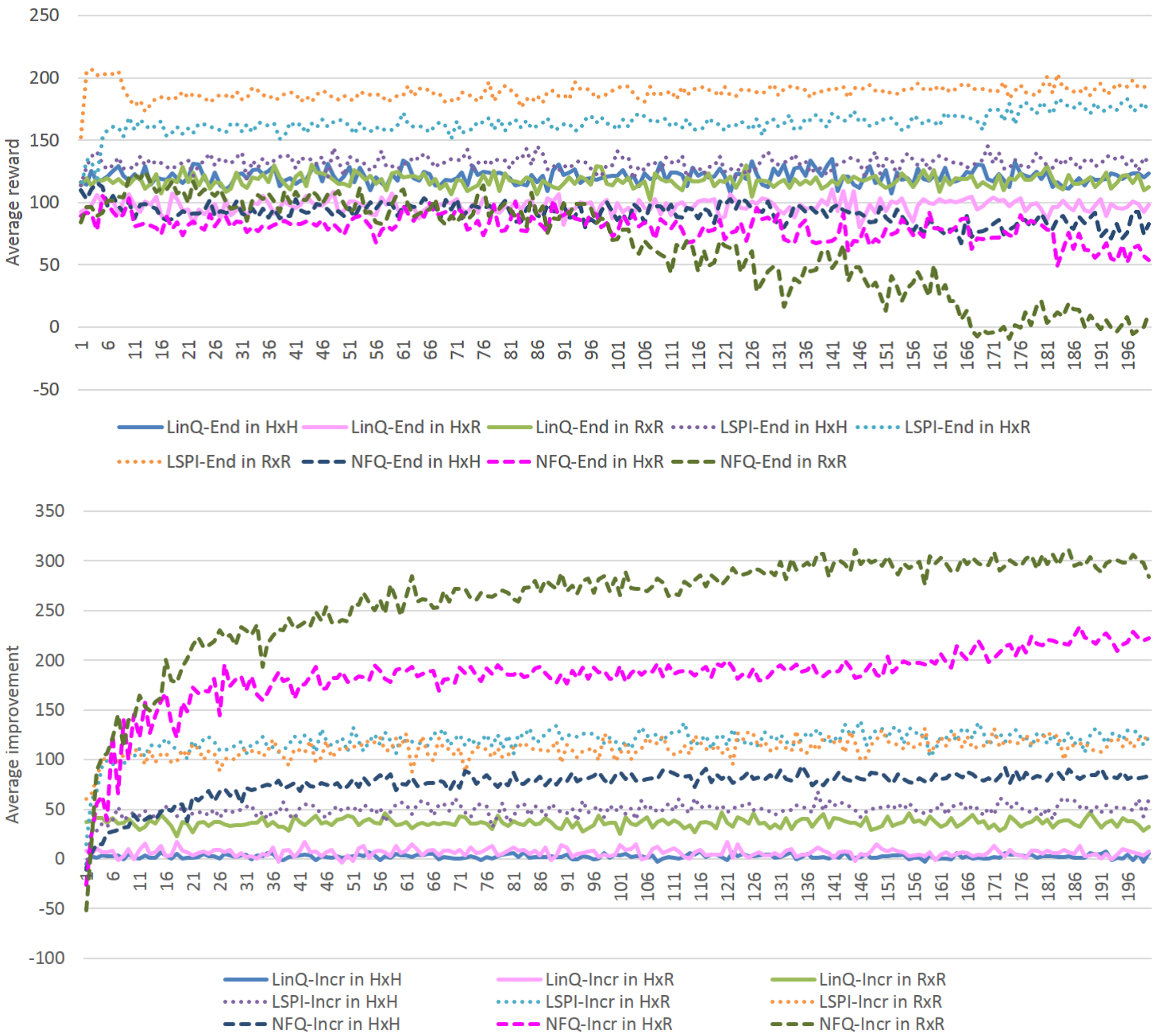

Fig. 2 Number of epochs vs. performance of learned policies in 3-party trading. The upper figure shows the performance when the reward is given by Equation (3). The lower figure shows the performance when the reward is given by Equation (4).

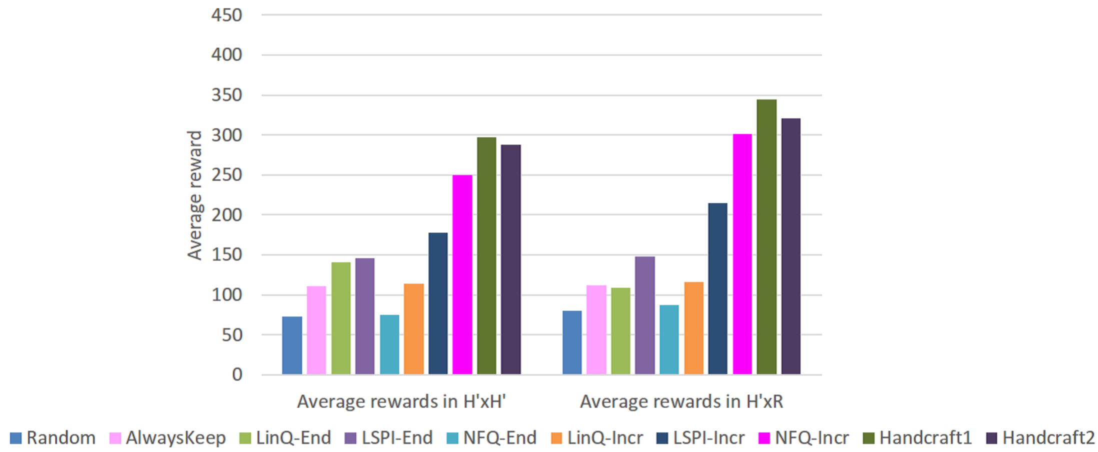

Fig. 3 Comparison of RL algorithms and types of reward functions in 3-party dialog. In these figures, the performances of the policies are evaluated by using the reward function given by Equation (3).

(2) the number of fruits that the trader possesses, we can reconstruct a new state, which consists of this information only, and its size will be independent of the number of traders. We can scale the traders' action spaces in the same manner. This type of scaling is similar to the summary state [Williams 07b] and summary action [Thomson 10]

Table2 Averages of the number of each one of the learner's actions per dialog, of the number of total turns, and of achievements of fruit salad per dialog, over 20000 dialogs in 3-party trading. Note that "Accepted offer" is an offer made by the learner that is accepted by the other traders.

HxH	Offer	Accepted offer	Accept	Keep	Total actions	Total turns	Fruit salad
Random	3.577	0.463	0.191	1.169	4.937	14.534	0.156
AlwaysKeep	0.000	0.000	0.000	4.231	4.231	11.003	0.227
LinQ-End	6.296	0.076	0.000	0.002	6.298	17.876	0.210
LSPI-End	1.546	0.115	0.245	2.337	4.128	11.001	0.165
NFQ-End	5.266	0.172	0.094	0.447	5.807	16.785	0.197
LinQ-Incr	6.312	0.078	0.000	0.000	6.312	17.911	0.206
LSPI-Incr	5.462	0.170	0.128	0.306	5.896	16.835	0.270
NFQ-Incr	4.520	0.350	0.001	0.914	5.435	15.462	0.355
Handcraft1	3.078	0.128	0.024	2.030	5.131	14.091	0.329
Handcraft2	3.370	0.197	0.137	1.585	5.092	14.386	0.343

HxR	Offer	Accepted offer	Accept	Keep	Total actions	Total turns	Fruit salad
Random	4.204	0.465	0.311	1.254	5.769	16.084	0.168
AlwaysKeep	0.000	0.000	0.000	4.518	4.518	11.804	0.222
LinQ-End	0.017	0.000	0.000	4.552	4.569	11.927	0.219
LSPI-End	2.018	0.050	0.393	2.407	4.818	12.562	0.219
NFQ-End	5.386	0.251	0.187	0.829	6.401	17.732	0.170
LinQ-Incr	6.632	0.082	0.001	0.001	6.633	18.651	0.214
LSPI-Incr	5.826	0.229	0.143	0.785	6.754	18.866	0.219
NFQ-Incr	5.069	0.190	0.329	0.933	6.331	17.213	0.626
Handcraft1	2.529	0.189	0.213	2.863	5.605	14.847	0.532
Handcraft2	3.326	0.273	0.343	2.134	5.803	15.769	0.459

RxR	Offer	Accepted offer	Accept	Keep	Total actions	Total turns	Fruit salad
Random	4.656	0.600	0.531	1.247	6.434	17.941	0.178
AlwaysKeep	0.000	0.000	0.000	5.106	5.106	14.183	0.218
LinQ-End	7.071	0.302	0.000	0.000	7.071	19.889	0.177
LSPI-End	6.334	0.312	0.685	0.049	7.067	19.864	0.101
NFQ-End	5.123	0.304	0.351	1.170	6.644	18.556	0.177
LinQ-Incr	7.056	0.304	0.000	0.000	7.056	19.896	0.171
LSPI-Incr	2.499	0.156	0.539	2.963	6.001	16.626	0.364
NFQ-Incr	4.081	0.256	0.775	1.977	6.833	18.379	0.801
Handcraft1	1.810	0.244	0.370	3.830	6.010	16.197	0.701
Handcraft2	2.742	0.365	0.581	2.930	6.253	17.082	0.591

approaches. However, to apply these scaling approaches, we need to know what features are really important for successful trading. This is not always easy to do manually, but we could instead use unsupervised learning or feature selection techniques to automatically construct the state/action space.

Our current setup assumes that each trader knows the number of items that the other traders possess, but in a real life setting this may not be possible. In that case, it would be more difficult for the trader to perform well, and he may need to keep track of recent trades. Our current model does not take into account the trade history (e.g., what trades have succeeded or failed so far). Therefore if the trader does not know the number of items that the other traders have, he may ask his trading partner to provide an item that the partner does not possess. Even if the trader can keep track of the trade history, this additional information may not necessarily help him make an appropriate offer. The reason is that a trade that failed in a previous turn, may succeed after a redistribution of items resulting from trades between other parties.

Table3 Averages of the number of each one of the learner's actions per dialog, of the number of total turns, and of achievements of fruit salad per dialog, over 20000 dialogs in 3-party trading. Note that "Accepted offer" is an offer made by the learner that is accepted by the other traders.

H'xH'	Offer	Accepted offer	Accept	Keep	Total actions	Total turns	Fruit salad
Random	4.427	0.808	0.364	1.346	6.137	17.655	0.144
AlwaysKeep	0.000	0.000	0.000	5.763	5.763	14.748	0.222
LinQ-End	6.970	0.272	0.000	0.002	6.973	19.365	0.175
LSPI-End	2.511	0.249	0.525	2.667	5.702	15.268	0.106
NFQ-End	5.892	0.443	0.217	0.521	6.630	18.779	0.166
LinQ-Incr	6.954	0.260	0.000	0.001	6.955	19.309	0.169
LSPI-Incr	6.068	0.525	0.283	0.367	6.717	18.890	0.262
NFQ-Incr	5.286	0.635	0.003	1.269	6.558	18.076	0.429
Handcraft1	2.406	0.332	0.071	3.818	6.294	16.540	0.545
Handcraft2	3.045	0.424	0.237	2.994	6.276	16.957	0.486

H'xR	Offer	Accepted offer	Accept	Keep	Total actions	Total turns	Fruit salad
Random	4.545	0.705	0.452	1.324	6.321	17.626	0.161
AlwaysKeep	0.000	0.000	0.000	5.631	5.631	15.015	0.222
LinQ-End	0.019	0.001	0.000	5.616	5.636	15.064	0.219
LSPI-End	2.674	0.139	0.610	2.511	5.796	15.619	0.134
NFQ-End	5.538	0.374	0.244	0.941	6.723	18.703	0.164
LinQ-Incr	6.901	0.271	0.001	0.000	6.902	19.191	0.172
LSPI-Incr	5.643	0.433	0.176	1.134	6.953	19.346	0.312
NFQ-Incr	5.035	0.401	0.436	1.260	6.730	18.193	0.524
Handcraft1	2.146	0.283	0.223	3.952	6.320	16.667	0.625
Handcraft2	2.943	0.391	0.388	3.047	6.379	17.156	0.541

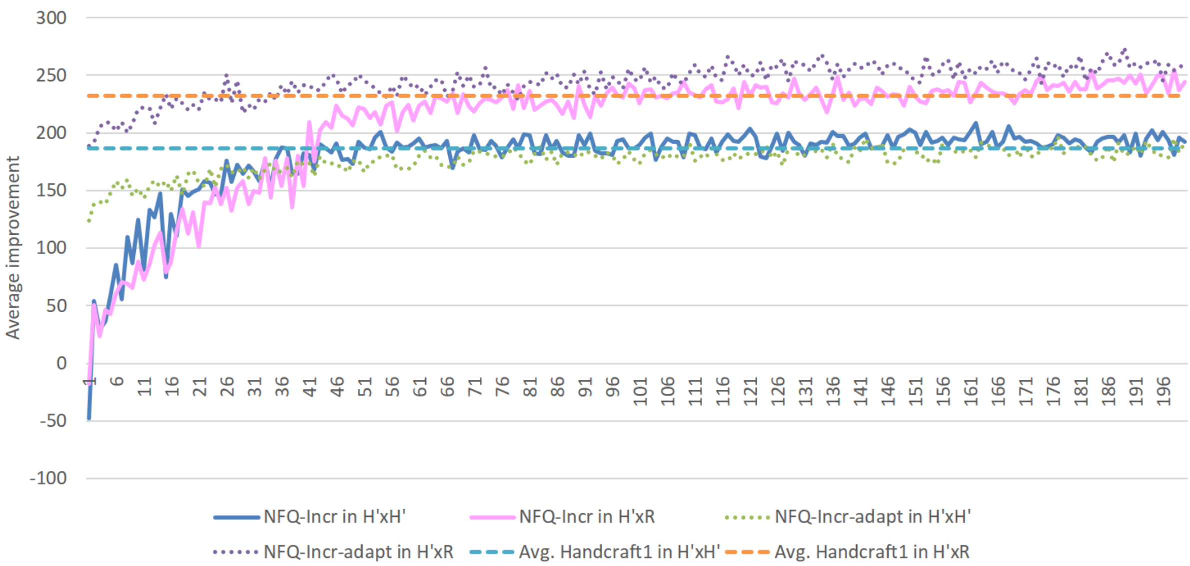

Fig. 4 Number of epochs vs. performance of learned policies in 3-party trading. The figure shows the performance when the reward is given by Equation (4). **NFQ-Incr-adapt** uses policies learned in different trading domains as initial policies. In this experiment, the policy learned in HxH is used as an initial policy in H'xH', and the policy learned in HxR is used as an initial policy in H'xR. **Avg. Handcraft1** represents the average cumulative reward of Handcraft1 over 20000 dialogs.

6. Conclusion

In this paper, we used RL to learn the dialog system's (learner's) policy in a multi-party trading scenario. We experimented with different RL algorithms and reward functions. The negotiation policies of the learner were learned and evaluated through simulated dialog with trader simulators. We presented results for different numbers of traders.

Our results showed that (1) even in simple multi-party trading dialog domains, learning an effective negotiation policy is not a straightforward task and requires a lot of experimentation; and (2) the use of neural fitted Q iteration combined with an incremental reward function produces as effective or even better negotiation policies than the policies of two strong hand-crafted baselines. These experimental results depend on the properties of the trader simulators.

For future work we will expand the dialog model to augment the dialog state with information about the estimated payoff matrix of other traders. This means expanding from an MDP-based dialog model to a POMDP-based model. We will also apply multi-agent RL [Georgila 14] to multi-party trading dialog. Furthermore, we will perform evaluation with human traders. Finally, we will collect and analyze data from human trading dialogs in order to improve our models and make them more realistic.

Acknowledgments

This research was partially supported by the 2014 Global Initiatives Program, JSPS KAKENHI Grant Number 24240032, and the Commissioned Research of the National Institute of Information and Communications Technology (NICT), Japan. This material was also based in part upon work supported by the National Science Foundation under Grant Number IIS-1450656, and the U.S. Army. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or the United States Government, and no official endorsement should be inferred.

♢ References ♢

[Asher 13] Asher, N., and Lascarides, A.: Strategic conversation, Semantics and Pragmatics, Vol. 6, pp. 1–62 (2013)

[Efstathiou 14] Efstathiou, I., and Lemon, O.: Learning non-cooperative dialogue behaviours, In Proc. of SIGDIAL, pp. 60–68 (2014)

[El Asri 13] El Asri, L., Laroche, R., and Pietquin, O.: Reward shaping for statistical optimisation of dialogue management, In Proc. of SLSP, pp. 93–101 (2013)

[Georgila 10] Georgila, K., Wolters, M. K., and Moore, J. D.: Learning dialogue strategies from older and younger simulated users, In Proc. of SIGDIAL, pp. 103–106 (2010)

[Georgila 11] Georgila, K., and Traum, D.: Reinforcement learning of argumentation dialogue policies in negotiation, In Proc. of INTERSPEECH, pp. 2073–2076 (2011)

[Georgila 13] Georgila, K.: Reinforcement learning of two-issue negotiation dialogue policies, In Proc. of SIGDIAL, pp. 112–116 (2013)

[Georgila 14] Georgila, K., Nelson, C., and Traum, D.: Single-agent vs. multi-agent techniques for concurrent reinforcement learning of negotiation dialogue policies, In Proc. of ACL, pp. 500–510 (2014)

[Guhe 12] Guhe, M., and Lascarides, A.: Trading in a multiplayer board game: Towards an analysis of non-cooperative dialogue, In Proc. of CogSci., pp. 1626–1631 (2012)

[Heeman 09] Heeman, P. A.: Representing the reinforcement learning state in a negotiation dialogue, In Proc. of ASRU, pp. 450–455 (2009)

[Hiraoka 14] Hiraoka, T., Neubig, G., Sakti, S., Toda, T., and Nakamura, S.: Reinforcement learning of cooperative persuasive dialogue policies using framing, In Proc. of COLING, pp. 1706–1717 (2014)

[Jokinen 09] Jokinen, K., and McTear, M.: Spoken dialogue systems, Synthesis Lectures on Human Language Technologies, Morgan & Claypool (2009)

[Lagoudakis 03] Lagoudakis, M. G., and Parr, R.: Least-squares policy iteration, Journal of Machine Learning Research, Vol. 4, pp. 1107–1149 (2003)

[Levin 00] Levin, E., Pieraccini, R., and Eckert, W.: A stochastic model of human-machine interaction for learning dialog strategies, IEEE Transactions on Speech and Audio Processing, Vol. 8, No. 1, pp. 11–23 (2000)

[Mayya 11] Mayya, Y., Kyung, L. T., and Seok, K. I.: Negotiation and persuasion approach using reinforcement learning technique on broker's board agent system, In Proc. of NCM, pp. 30–34 (2011)

[McGovern 01] McGovern, A., and Barto, A. G.: Automatic discovery of subgoals in reinforcement learning using diverse density, Computer Science Department Faculty Publication Series, 8 pages (2001)

[Ng 99] Ng, A. Y., Harada, D., and Russell, S.: Policy invariance under reward transformations: Theory and application to reward shaping, In Proc. of ICML, pp. 278–287 (1999)

[Nouri 12] Nouri, E., Georgila, K., and Traum, D.: A cultural decision-making model for negotiation based on inverse reinforcement learning. In Proc. of CogSci., pp. 2097–2102 (2012)

[Nouri 13] Nouri, E., Park, S., Scherer, S., Gratch, J., Carnevale, P., Morency, L. P., and Traum, D.: Prediction of strategy and outcome as negotiation unfolds by using basic verbal and behavioral features, In Proc. of INTERSPEECH, pp. 1458–1461 (2013)

[Papangelis 15] Papangelis, A., and Georgila, K.: Reinforcement learning of multi-issue negotiation dialogue policies, In Proc. of SIGDIAL, pp. 154–158 (2015)

[Paruchuri 09] Paruchuri, P., Chakraborty, N., Zivan, R., Sycara, K., Dudik, M., and Gordon, G.: POMDP based negotiation modeling, In Proc. of MICON, pp. 66–78 (2009)

[Pfeiffer 04] Pfeiffer, M.: Reinforcement learning of strategies for Settlers of Catan, In Proc. of the International Conference on Computer Games: Artificial Intelligence, Design and Education (2004)

[Riedmiller 05] Riedmiller, M.: Neural fitted Q iteration - first experiences with a data efficient neural reinforcement learning method, In Proc. of ECML, pp. 317–328 (2005)

[Riedmiller 09] Riedmiller, M., Gabel, T., Hafner, R., and Lange, S.: Reinforcement learning for robot soccer. Autonomous Robots, Vol. 27, No. 1, pp. 55–73 (2009)

[Schaul 10] Schaul, T., Bayer, J., Wierstra, D., Sun, Y., Felder, M., Sehnke, F., Rückstieß T., and Schmidhuber, J.: Pybrain, Journal of Machine Learning Research, Vol. 11, pp. 743–746 (2010)

[Sutton 98] Sutton, R. S., and Barto, A. G.: Reinforcement Learning: An Introduction, MIT Press Cambridge (1998)

[Thomson 10] Thomson, B., and Young, S.: Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems, Computer Speech and Language, Vol. 24, No. 4, pp. 562-588 (2010)

[Traum 08] Traum, D.: Extended abstract: Computational models of non-cooperative dialogue, In Proc. of SEMDIAL-LONDIAL (2008)

[Walker 00] Walker, M. A.: An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email, Journal of Artificial Intelligence Research, Vol. 12, pp. 387–416 (2000)

[Williams 07a] Williams, J. D., and Young, S.: Partially observable Markov decision processes for spoken dialog systems, Computer Speech and Language, Vol. 21, No. 2, pp. 393–422 (2007)

[Williams 07b] Williams, J. D., and Young, S.: Scaling POMDPs for spoken dialog management, IEEE Transactions on Audio Speech and Language Processing, Vol. 15, No. 7, pp. 2116–2129 (2007)

[Zou 14] Zou, Y., Zhan, W., and Shao, Y.: Evolution with reinforcement learning in negotiation, PLoS One, Vol. 9, No. 7 (2014)

〔担当委員：Danushka Bollegala〕

Received December 25, 2015.

Author's Profile

Hiraoka, Takuya

Takuya Hiraoka graduated from the Department of Informatics, Faculty of Science and Engineering, Kinki University in Japan in 2011. He completed the Ph.D. program at the Graduate School of Information Science, Nara Institute of Science and Technology (NAIST) in Japan in 2016. Currently he is a researcher at the Nippon Electric Company Central Research Laboratories.

Georgila, Kallirroi

Kallirroi Georgila is a Research Assistant Professor at the Institute for Creative Technologies and at the Computer Science Department of the University of Southern California (USC). Before joining USC in 2009 she was a Research Scientist at the Educational Testing Service (ETS) in Princeton, USA, and before that a Research Fellow at the School of Informatics of the University of Edinburgh, in the United Kingdom. Her research interests include all aspects of spoken dialogue processing with a focus on reinforcement learning of dialogue policies, expressive conversational speech synthesis, and speech recognition. Georgila has published over 80 journal articles, conference papers, and technical reports on various topics in spoken dialogue processing. She has served on the organizing, senior, and program committees of many conferences and workshops, including being General Co-Chair for SIGDIAL 2014, Mentoring Chair for SIGDIAL 2012 and 2013, Associate Chair for ICMI 2013, Area Chair for EACL 2012, and Program Co-Chair for SemDial 2011.

Nouri, Elnaz

Elnaz Nouri is a Ph.D. candidate at the Computer Science Department of the University of Southern California. She is a Provost Fellowship recipient and a member of the Natural Dialogue group at the Institute for Creative Technologies where she works under the supervision of Dr. Traum. Her Ph.D. thesis proposes a framework for developing culture sensitive models for decision making in social interactions.

Traum, David

David Traum is the Director of Natural Language Research at the Institute for Creative Technologies (ICT) and a Research Faculty member of the Department of Computer Science at the University of Southern California (USC). He leads the Natural Language Dialogue Group at ICT. More information about the group can be found here: `http://nld.ict.usc.edu/group/` Traum's research focuses on Dialogue Communication between Human and Artificial Agents. He has engaged in theoretical, implementational and empirical approaches to the problem, studying human-human natural language and multi-modal dialogue, as well as building a number of dialogue systems to communicate with human users. Traum has authored over 200 refereed technical articles, is a founding editor of the Journal Dialogue and Discourse, has chaired and served on many conference program committees, and is a past President of SIGDIAL, the international special interest group in discourse and dialogue. Traum earned his Ph.D. in Computer Science at the University of Rochester in 1994.

Nakamura, Satoshi (Member)

Satoshi Nakamura is Professor of Graduate School of Information Science, Nara Institute of Science and Technology, Japan, Honorar professor of Karlsruhe Institute of Technology, Germany, and ATR Fellow. He received his B.S. from Kyoto Institute of Technology in 1981 and Ph.D. from Kyoto University in 1992. He was Director of ATR Spoken Language Communication Research Laboratories in 2000-2008 and Vice president of ATR in 2007-2008. He was Director General of Keihanna Research Laboratories, National Institute of Information and Communications Technology, Japan in 2009-2010. He is currently Director of Augmented Human Communication laboratory and a full professor of Graduate School of Information Science at Nara Institute of Science and Technology. He is interested in modeling and systems of speech-to-speech translation, spoken dialog, and speech recognition. He has been serving for various speech-to-speech translation research projects in the world including C-STAR, IWSLT, and A-STAR. He received many domestic academic awards including Kiyasu Award from the Information Processing Society of Japan, ASJ Award for Distinguished Achievements in Acoustics. He also received the Commendation for Science and Technology by the Minister of Education, Science and Technology, and the Minister of Internal Affair and Communications. He also received LREC Antonio Zampoli Award 2012. He is an Elected Board Member of International Speech Communication Association, ISCA, since June 2011, IEEE Speech and Language Technology Committee member since April 2012, and IEEE Fellow from 2016.