# A Learning Automata based Solution for Optimizing Dialogue Strategy in Spoken Dialogue System

G.Kumaravelan
Department of Computer Science.
Pondicherry University, Karaikal Campus, Karaikal.

R.Sivakumar
Department of Computer Science.
AVVM Sri Pushpam College, Poondi.

## ABSTRACT
Application of reinforcement learning methods in the development of dialogue strategies that support robust and efficient human–computer interaction using spoken language is a growing research area. In spoken dialogue system, Markov Decision Processes (MDPs) provide a formal framework for making dialogue management decisions for planning. This framework enables the system to learn the value of initiating an action from each possible state which in turn facilitates the maximization of the total reward. However, these MDP systems with large state-action spaces lead to intractable solution. The goal of this paper is, thus, to present a novel approximation method with sampling practice to compute an optimal solution to control dialogue strategy based on learning automata. Compared to other baseline reinforcement learning methods the proposed approach exhibits a better performance with regard to the learning speed, good exploration/exploitation in its update and robustness in the presence of uncertainty in the states obtained.

## General Terms
Human-Computer Interaction.

## Keywords
Learning Automata, Reinforcement Learning, Markov Decision Process, Spoken Dialogue System.

## 1. INTRODUCTION
In the recent years, spoken dialogue technology has emerged as a demanding area for researchers in artificial intelligence and human-computer interaction [1]. The Spoken Dialogue System (SDS) receives speech inputs from the user and the system responds with the required action and information. It allows various interactive applications dealing with directory assistance, information providing systems, robot control, planning assistance, troubleshooting etc. These systems have increasingly become competent of supporting multiple tasks and of accessing information from a broad selection of sources and services. The general spoken dialogue systems typically consist of three components as shown in Figure 1. The subsystems for input (conveying information from the user to the system in terms of *Signal Processing*, *Dialogue Act Recognition*, *User Goal Recognition*), control (deciding how to react in terms of *Discourse Analysis*, *Database Query*, *System Action Prediction*) and output (conveying information from the system to the user in terms of *Utterance Realization*).

This paper is primarily concerned with the design of the Dialogue Management (DM), which is the central component within the SDS. DM determines the related communicative actions to be taken for a given goal and a particular set of observations about the dialogue history. In other words, DM is solely responsible for controlling the flow of interaction, which is, referred to as dialogue strategy or policy in an efficient and natural way. This is a challenging task in most of the spoken dialogue systems wherein the dialogue strategy is handcrafted by a human designer which leads to errors, strenuous and non-portable.

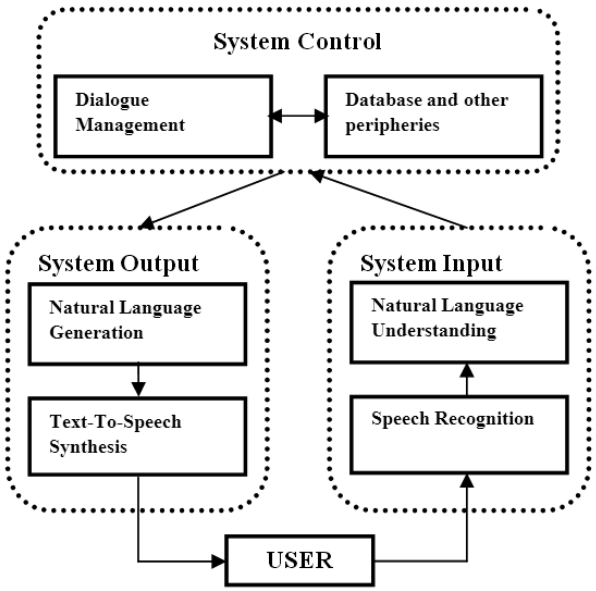


**Fig 1: High level architecture of spoken dialogue system**

Current research trends indicate attempts to find a way to automate the development of dialogue strategy using machine learning techniques. In practice, Reinforcement Learning (RL) techniques show appealing cognitive capabilities since they try to learn the appropriate set of actions to choose in order to maximize a scalar reward by following a trial and error interaction with an environment [2]. In this context, the dialogue strategy is regarded as a sequence of states with a reward for executing an action which in turn inducing a state transition in the conversational environment. The objective for each dialogue state is to choose such an action that leads to the highest expected long-term reward. For SDSs, these reward signals are associated with task completion and dialogue length. Hence, the system model covers the dynamics of Markov Decision Processes (MDPs) with a set of states S, a set of actions A, a state transition function, and a reward for each selected action. In this framework, a reinforcement learning agent aims at optimally mapping states to actions, i.e. "*finding the optimal policy so as to maximize an overall reward*" [3].

In order to use RL to optimize dialogue strategy, a number of technical challenges need to be overcome. These include choosing an appropriate *reward function*, *scalability*, *robustness*, and *portability* [4]. In particular, because of varying levels of confidence measure in speech recognition, the state space of the dialogue often becomes continuous. More recently, there have been several research attempts to

overcome these issues to model uncertainty in the dialogue. The limitations of these contributions indicate that the dilemma in *exploration versus exploitation* type in learning algorithms and *curse of dimensionality* in modeling the state space has to be solved completely. Hence, this paper examines and shows, with the aid of technique based on learning automata called "*Recursive Automata Sampling Algorithm* (RASA)" [5], how to approximate the continuous state-action value function for estimating optimal dialogue policy independent of state space size. The proposed approach adapts a specific implementation of the *Hidden Information State* (HIS) model [6]. In this proposed method, the state estimator maintains the distribution of the intact dialogue states often in multiple stages instead of discrete grid points. It also identifies all possible dialogue paths to choose an action that maximizes the reach of a dialogue towards a successful completion.A major motivation is to improve robustness where uncertainty and planning in relation to application context in different situations of the dialogue exists. In principle, the proposed learning approach has several advantages over grid-based and rule-based approaches in the dialogue systems viz.,

- adaptive sampling mechanism with probabilistic and dynamical aspects
- a data-driven development cycle, and
- reduced computational demands.

This paper is organized as follows. Section II reviews the existing techniques and approaches developed in the design of DM. Section III is devoted to useful definitions and essential requirements in dialogue management framework, whereas Section IV presents a detailed description of the learning automata based algorithm to learn optimal dialogue strategy under uncertainty in rich and complex interactive settings. Sections V and VI present the effectiveness of the proposed approach through experiments and quantitative evaluation of the behaviours, respectively. The concluding section presents the summarization of the analysis.

## 2. RELATED WORK

This section reviews the divergent ways of implementing a dialogue strategy. First, the finite state-based approach represents the dialogue structure in the form of a state transition network, where transitions between dialogue states specify all legal paths through the network, which is suitable for system initiative interactions [7]. The design of such systems is relatively straightforward and their behaviour is predictable. One of the most popular methodologies is the Rapid Application Developer of CSLU Toolkit, which allows the designer to specify the dialogue as a finite state model using a drag-and-drop interface. Second, the frame-based approach, also known as slot-based method, represents the dialogue in the form of attribute-value structure that can be seen as a form for which the user should provide values for each field (attribute, slot) of the form. In this approach, the user has the freedom to take the initiative in the dialogue [8]. Finally, Plan-based systems view the communication as a planning process motivated by the achievement of certain goals [9]. However, the designers of a dialogue strategy may need to spend a great deal of time anticipating how potential users will interact with the system through repeated testing and refining so as to deploy dialogue systems with practical performance.

To address this problem, the research community for DM has exploited the benefits of data-driven approaches in the development of stochastic dialogue modeling using RL based on MDPs [10]. This framework follows statistically data-driven development cycle, a precise mathematical model, possibilities for generalization to unseen states, and theoretically principled dialogue modeling to dynamically allow changes to the dialogue strategy. Furthermore, small changes in the environment's dynamics require recomputing the full policy with less time and effort. Hence to automate the design of dialogue strategy a number of on-line and off-line RL methods have been proposed in recent years [11]. The goal of these methods is to learn the value of initiating an action from each potential dialogue state to maximize the long-term reward.

However, when the state-action spaces are small enough to represent in tabular form most of the state-of-the-art, RL algorithms like *Q-learning* and *SARSA* have been used to optimize the dialogue policy by incrementally updating the expected Q-values for each state-action pair via the Bellmann optimality equation. In addition, increase in the size of the state space for these algorithms could lead to the learning problem becoming intractable referred to as "*curse of dimensionality*". Another setback of the above baseline RL algorithm is that it requires an update of the value function over the entire state space that is purely based on value iteration. In the large state space, one may stick to one-iteration for a long-time before any improvement in performance is made. Hence, tabular RL algorithms are designed to operate on individual state-action pairs, and there is a practical limit to the size of the state-action table that can be implemented. This proves to be a major constraint for dialogue strategy developers.

Several approaches to deal with the problem of large state-action spaces have been proposed in recent years. One of the approaches is based on the idea that not all state variables are relevant for learning a dialogue strategy and the state-action space is reduced by carefully selecting a subset of the available state variables by *function approximation* and *hierarchical decomposition*. If the relevant variables are chosen, useful dialogue strategies can be learnt. This technique has been applied successfully in several recent studies [12][13]. In addition, eXtended Classifier System (XCS) model has been applied in dialogue strategy optimization to evolve and evaluate a population of rules/ and RL algorithm is applied to assign rewards to the rules [14]. However, it mitigates the curse of dimensionality problem by using a more compact representation with regions of state-action, but it finds less optimal solutions compared to tabular value functions.

A possible advantage of RASA compared to grid-based approaches is that it does not maintain expensive interpolation between grid points and it scale well to the large state spaces. In addition, RASA does not iteratively generate a new value function over sampled information-states from a previously known value function. However, at each dialogue turn, this approach builds sampled tree of most likely information-states and directly estimates the value of the state at the root of the tree in a bottom- up fashion, which follows a distinct path representing the true state of the dialogue with an adaptive sampling scheme of the action space.

## 3. BACKGROUND

Reinforcement learning is a sub-area of Artificial Intelligence (AI) which considers how an autonomous agent acts through trial-and-error interaction with a dynamic unknown environment. Here, the agent refers to an entity that can perceive the state of the environment, and take actions to

affect the environment's state. In turn, it receives a numerical signal called reinforcement from the environment for every action it takes. Its goal is to maximize the total reinforcements it receives over time. In reinforcement learning, an environment is often modeled as MDP, where the history of the environment can be summarized in a sufficient statistic called state to solve sequential decision making problems.

## 3.1 MDP Basics

An MDP is formally a tuple$\{X, A, P, R, \gamma\}$where $X$ denotes the state space, $A$ the action space, $A(x) \in A$ denote the set of permissible actions in state $x$, $p(x,a)(y)$ the probability of transition from state $x \in X$ to state $y \in X$ when action $a \in A(x)$ is initiated, a reward function $R: K \rightarrow \Re$ where $K = \{ (x,a) \mid x \in X, a \in A(x)\}$, and $\gamma$ the discounting factor $(0 \leq \gamma \leq 1)$. According to this formalism, let $x_t$ denote the state at time (*stage*) $t \in \{0,1,\ldots\}$ and $a_t$ indicates the action chosen at that time. If $x_t = x \in X$ and $a_t = a \in A(x)$, the system transitions from state $x$ to $x_{t+1} = y \in X$ with probability $p(x,a)(y)$, and a immediate reward of $R(x,a)$ is obtained. Once the transition to the next state has occurred, a new action is chosen, and the process is repeated.

Let $\prod$ be the set of all Markovian policies $\pi = \{\pi_i \mid \pi_i : X \rightarrow A, i = 0,\ldots,\infty\}$. For simulation model the goal is to find the optimal reward-to-go values function for state $x$ in stage $i$ given by

$$V_i^*(x) = E_{\pi \in \Pi}[\sum_{t=i}^{\infty} \gamma^t \; R(x_t, \pi_t(x_t), w_t) | x_i = x] \qquad (1)$$

where $x \in X$, $w$ the reward bounded with the reward function $R(x,a)$ and $x_t = f(x_{t-1}, \pi_{t-1}(x_{t-1}), w_{t-1})$ a random variable denoting the state at stage $t$ following policy $\pi$. It is assumed that every action in $A$ is admissible at every state and the same amount of defined random number is associated with the reward and transition functions.

Alternatively, it is well known that $V_i^*(x)$ can be written recursively as follows:

$$V_i^*(x) = max_{a \in A} \, Q_i^*(x,a) \qquad (2)$$

$$Q_i^*(x,a) = E\left[R(x,a,w) + V_{i+1}^*\big(f(x,a,w)\big)\right] \qquad (3)$$

If the transition probabilities and the reward function are known in advance, an optimal policy can be learnt by dynamic programming principles. Otherwise the agent needs to interact with its environment to learn these probabilities.

## 3.2 Dialogue as a MDP

Within the RL framework to dialogue management, dialogue strategies are represented as MDP, which serves as a formal representation of human-machine dialogue. One of the key advantages of statistical optimization methods for dialogue strategy is that "*the problem can be formulated as a precise mathematical model which can be trained on real data*" [15]. Every MDP is formally described by a finite state space representing all the currently available information regarding internal and external processes controlled by the dialogue system i.e., the knowledge of the concerned domain. For slot filling dialogue system, a common approach is to specify the number of state variables on the number of slots that need to be filled and grounded. For example, a travel information system might include departure city, destination city, date and time of travel. Thus, the total number of possible states is determined by the number of states and its corresponding slot in focus such as 'unknown', 'known' and 'confirmed'. Similarly, the system action set is often narrowed to a small number of actions such as 'request all', 'request n slots', 'verify n slots', 'verify all', or 'quit' by Dialogue Acts (DA). The state transition function describes the dynamics of the environment during the dialogue history. The reward function plays a critical role to notify what was a good dialogue since it would actually necessitate comparing dialogues with one another. In this framework, a DM is a system aiming at optimally mapping states to action that finds the best strategy $\pi$ to maximize an overall reward over time, i.e., the policy that selects those actions, which yield the highest reward over the course of the dialogue.

## 3.3 Learning Automata

Learning Automata (LA) are adaptive decision-making devices operating on unknown random environment, and are associated with a finite set of actions and each action has a certain probability (unknown to the automaton) of getting rewarded by the environment of the automaton [16]. The aim is to learn the ways to choose the optimal action (i.e. the action with the highest probability of being rewarded) through repeated interaction on the system. With regard to applications, LA have been used in game playing, parameter optimization, statistical decision making, distribution approximation, training hidden markov model, congestion avoidance in wireless networks, and modeling of student's behaviour.

Formally the finite learning automaton is a quadruple $<\alpha, \beta, p, T(\alpha, \beta, p)>$, where $\alpha$, $\beta$, and $p$ constitute an action set with $r$ actions, an environment response set, and the probability set $p$ containing $r$ probabilities, each being the probability of performing every action in the current internal automaton state, respectively. The function of $T$ is the reinforcement algorithm, which modifies the action probability vector $p$ with respect to the performed action and received response. If the response of the environment takes binary values, LA model is *P-model*, and if it takes finite output set with more than two-elements that take values in the interval [0,1], such a model is referred to as *Q-model*, and when the output of the environment is a continuous variable in the interval [0,1], it is referred to as *S-model*. Several designs for T have been proposed in the literature, reader is referred to [17] for an extensive treatment. In this paper we target the Pursuit learning algorithm, a special class of estimator algorithm in the context of solving MDPs [18] for optimizing dialogue strategy in spoken dialogue system.

In the Pursuit learning algorithm , "the automaton pursues the current optimal action by increasing the probability of selecting the chosen action while decreasing the probabilities of all other actions" using sample average rewards. At iteration $t$, the automaton selects an action $\alpha(t) \in A$ with respective probabilities $p(t) = \{p_1(t), \ldots, p_r(t)\}$. Whether the automaton is rewarded or penalized, the next step is to increase the component of $p(t)$ whose reward estimate is maximal (the current optimal action), and to decrease the probability of all the other action. From a vector perspective, the probability of updating rules and reward estimate vector can be expressed as follows:

$$p(t+1) = (1-\mu)p(t) + \mu em \qquad (4)$$

$$\hat{d}_i = E\{\beta(t) : \alpha(t) = a_i, \; i = 1, \ldots, r\} \qquad (5)$$

As the algorithm proceeds and the different actions are explored, the automaton acquires more and more information about the reward probabilities $d$'s indirectly through the environment feedback $\beta$'s. In other words, estimates $\hat{d}(t)$ of $d$ at time $t$ can be used to update the sampling probabilities $p(t)$ in such a way that those actions with large $\hat{d}(t)$ are more

likely to be chosen again in the next iteration. In this case, *em* is the unit vector representing the currently estimated optimal action, namely, the action with the maximal reward estimate and the internal parameter $\mu$ controls the size of steps that can be made in moving from $p(t\text{-}1)$ to $p(t)$. In general, small values of $\mu$ correspond to slower rates of convergence, and vice versa.

# 4. METHODOLOGY

In light of the previous discussion, this section details how PLA sampling class of algorithm aimed at accurately and efficiently estimating the optimal state-value function for modelling the inherent uncertainty in spoken dialogue strategy. At each sampled state in the dialogue history, taking an action is adaptive at each stage that defines the union of the slots specified in the active dialogue goals. The state-action space is explored by selecting actions randomly which leads to the estimation of value function that converge to the true value asymptotically in the total number of samples. At the start of the dialogue, PLA sampling algorithm generates a sampled tree in a recursive manner to estimate the optimal value at an initial state. Consequently, an adaptive sampling mechanism is incorporated for selecting relevant action to sample at each branch in the tree. The branching process construction goes on until the last stage is reached. At each simulated (next) state at a stage, a fixed intermediate reward is allocated among feasible actions and the reward is used with the current probability estimate for the optimal action. In this case, a simulated state corresponds to an automaton viewed as a dialogue manager which in turn updates its value function along with the probability distribution over the action space at each iteration in a recursive manner to the context of solving MDP.

Figure 2 presents PLA sampling algorithm for estimating $V_i^*(x)$ for a given state *x*. The inputs to the algorithm are charaterised as follows:

- Stage *i*.
- State $x \in X$.
- Sampling parameter $N_i$ ($\sum_{a \in A(x)} N_a^i(x)$) is the number of times action a has been sampled from state *x* in stage *i*.
- $\mu_i \in [0,1]$ is the learning rate.

Finally, the output of the algorithm is $V_i^{Ni}(x)$, the estimate of $V_i^*(x)$.

As the dialogue progresses, the algorithm builds a sampled tree of depth *H* which defines the number stages to track the status of the slots and its values mentioned by both the user and the system in terms of dialogue acts in various contexts. Initially, a root node is associated to the first decision stage and a branching factor of $N_i$ at each level *i* initializes the probability distribution over the action space $p_{xo}$ as the uniform distribution. At each dialogue cycle, the previous system act and each input user act is get compared and an action which is either primitive or abstract is sampled from the probability distribution $p_{xo}(k)$ and reward rk is assigned independently (an action and a reward resultant to an edge in the tree). For the sampled action $a(k) \in A(x_0)$, the Q-function estimate is updated using the expected reward $R'(x_0, a(k), r_k)$ and next state ($f(x_0, a(k), r_k)$, and the count variable $N_{a(k)}^0(x_0)$ is incremented, where a recursive call is made to estimate $\hat{V}_1^{N1}$ at the simulated next state by executing dialogue repeatedly. At the end of each dialogue, this is followed by updating the estimate of the optimal action with the discounted reward and its probability distribution $P_{x0}(k)$ respectively in the direction of current estimate of the optimal action by adding $\mu_i$ to its probability mass and subtracting a proportional amount from all other actions. In this case, $E(.)$ denotes the expectation corresponding to the estimated optimal action with the highest reward. After $N_0$, iterations, the PLA algorithm estimates the optimal value $V_0^*(x_0)$ by the *Q*-function value at the currently estimated optimal action of the $x_0$-automaton by setting

$$\hat{Q}_0^{N0}(x_0, a) = \frac{1}{N_a^0(x_0)} \sum_{\mathrm{j:a(j)=a}} \mathrm{R}'\left(\mathrm{x_0}, \mathrm{a}, \mathrm{r_j}\right) + \hat{V}_1^{N1}\left(f(x_0, a, r_j)\right) \quad (6)$$

**Fig 2: PLA based policy optimization algorithm**

**Input**: i, x, $N_i$, $\mu_i$.
**Output**: $\hat{V}_i^{Ni}(x) = \hat{Q}_i^{Ni}(x, \hat{a})$
**Initialization**:
Set $P_x(0)(a) = \frac{1}{|A(x)|}$ ; initial random order of actions;
Set $N_a^i(x) = 0$ ; number of times action *a* is sampled from the state *x*;
Set $M_i(x, a) = 0 \; \forall a \in A(x)$ ; average reward on taking action *a* from the state *x*;
Set *k* = 0 ; iteration count;
// Generate dialogue using $\epsilon$-greedy policy
**repeat**
// Random action selection with probability $\epsilon$ along with reward $r_k$ from user simulator
Sample $a(k) \sim P_x(k), r_k \sim U(0,1)$;
Update *Q*-function estimate for $a = a(k) only$:
$M_i(x, a(k)) \leftarrow M_i(x, a(k)) + R'(x_0, a(k), r_k) + \hat{V}_{i+1}^{Ni+1}(f(x, a(k), r_k))$;
$N_{a(k)}^i(x) \leftarrow N_{a(k)}^i(x) + 1$;
$\hat{Q}_i^{Ni}(x, a(k)) \leftarrow M_i^{Ni}(x, a(k)) / N_{a(k)}^i(x)$;
Update optimal action estimate policy
$\hat{a} \in \text{argmax } \hat{Q}_i^{Ni}(x, a), \forall a \in A(x)$.
Update probability distribution over action space with respect to pursuit scheme defined in equation (4)
$P_x(k+1)(a) \leftarrow (1 - \mu_i P_i(k)(a) + \mu_i E\{\hat{a} = a\} \; \forall a \in A(x))$;
$k \leftarrow k + 1$;
**until** *k* = $N_i$; converged

task-oriented, slot-filling dialogue system in the travel planning domain. In slot-filling dialogues, an effective strategy is the one that interacts with the user in a satisfactory way while trying to minimize the length of the dialogue. The goal of the experimental system is to give information about flights based on specific user preference in a simulated learning environment.

# 5. EXPERIMENTS

The dialogue management problem studied in this paper is a

## 5.1 Task Representation

The state space representation of the dialogue problem has four slots representing the confidence of slot values of 1) departure city, 2) destination city, 3) departure date, and 4) departure time of the air travel domain. Each of these slot-values is associated with filling and confirming the confidence of the slot relevant to the dialogue history information which defines the level of stages in the learning algorithm. For example, a particular slot has not yet been stated, stated but not grounded, or grounded. The action space of the human-machine spoken dialogue includes the following system

action model available for exploration in every state and is given as follows:

- Greet
- Ask a slot (REQUEST_INFO())
- Explicit confirm (EXPCONF())
- Implicit confirm and ask a slot (REQUEST_INFO_WITH_IMPCONF())
- Close and present information

Similarly the user action model is the following:

- Yes – answer
- No – answer
- Provide-asked
- Re-provide-asked
- Remain silent

**Table 1. The principal dialogue acts used in flight booking system**

| System dialogue acts | User dialogue acts |
|---|---|
| GREETING() | command (bye) |
| REQUEST_INFO(dep_value) | provide_ Info(dep_value)* |
| REQUEST_INFO(dest_value) | provide_ Info(dest_value)* |
| REQUEST_INFO(date_value) | provide_ Info(date_value)* |
| REQUEST_INFO(time_value) | provide_ Info(time_value)* |
| REQUEST_INFO_WITH_IMP CONF(dep_value) | answer(yes) |
| REQUEST_INFO_WITH_IMP CONF(dest_value) | answer(yes) |
| REQUEST_INFO_WITH_IMP CONF(date_value) | silence() |
| REQUEST_INFO_WITH_IMP CONF(time_value) | |
| EXPCONF(low+medconfs) | |
| DATABASE-RESULT | |
| * Multiple slot values can be provided in a single utterance | |

The most common dialogue acts for task oriented human-machine simulated dialogue are listed in table 1 and a simple dialogue illustrating their use is shown in table 2.

## 5.2 User Simulation

Initially, a baseline system is built using hand-crafted policy. This uses a large number of internal system variables and advanced strategies such as variable confidence thresholds that try to fill the available slots one after the other and terminate the dialogue episode if all the slots are filled. In order to keep the design problem tractable, the hand-crafted policy has been kept as simple as possible instead of taking much dialogue context into account. To assert the impact of the sample-efficiency of the proposed method, different sizes of datasets (dialogue corpora) which represent the problem space are required. However, collecting large amounts of data may take a lot of time. Therefore, a user simulation technique has been used to generate different datasets [19]. The task of the user simulator is to provide examples of how a human would behave while interacting with the system. It provides user actions at a dialogue act level and uses two main components *user goal* and *user agenda*. At the start of each dialogue, the goal describes the full set of constraints that the user requires to satisfy such as departure city, destination city, date and time. The agenda stores an ordered list of dialogue acts that the user is planning to use in stack like structure in order to omplete its task. At the end of every dialogue the system receives a -1 penalty for every action it takes, a final reward of +20 in case of successful dialogue when all the necessary information has been obtained. In the case where no flight detail matches the attribute-slot values, the dialogue is deemed successful and a suitable alternative is offered. Since a characteristic dialogue will require about six or seven turns to complete, this implies that the achievable cumulative reward has an upper bound of 14.

**Table 2. An example dialogue and its representation at the dialogue act level**

| Speaker | Utterance | Dialogue act |
|---|---|---|
| SYS | Welcome to the travel planning system. How can I help you today? Please give the name of the city that you are starting from. | [GREETING] [REQUEST_INFO dep_val)] |
| USR | I would like to fly from Chennai | [provide_info dep_val='chennai')] |
| SYS | So you want to travel from Chennai. Which city are you travelling to? | [REQUEST_INFO_ WITH_IMPCONF(dest_val)] |
| USR | Mumbai | [provide_info (dest_val='mumbai')] |
| SYS | Ok, please give me the date you wish to leave. | [REQUEST_INFO (date_value)] |
| USR | April 14th | [provide_info (date_val='14/04/2011')] |
| SYS | At what time do you prefer to travel? | [REQUEST_INFO (time_val)] |
| USR | In the morning | [provide_info (time_val='morning')] |
| SYS | Ok, accessing the database. I found 2 flights from Chennai to Mumbai. Choice 1: departing at six thirty a.m. and arriving at eight fifteen a.m. The price is 2500 rupees. Choice 2: departing at nine a.m. and arriving at ten twenty a.m. The price is 3700 rupees. | [DATABASE_RESULT] |
| USR | Ok, thank you goodbye | bye() |

## 5.3 Behaviour of Value Function Approximation

In each experiment, a dialogue strategy was allowed to evolve over a fixed number of dialogues with a simulated user. For each of these training sessions, the resulting estimated value function is used to compute the estimated optimal policy. The stopping criterion is based on the number of changes made to the policy. For simplicity, the number of samples ($N$) at each stage is taken to be the same and thus the input parameter $\mu$ (*learning rate*) is chosen to be $1 - 2^{-1/N}$ , independent of the stages. The estimates of value function obtained by the policies trained using varied number of samples is shown in table 3. Figure 3 plots the (averaged) estimates of value

function as a function of the number of samples used for training during the dialogue episodes. Our experimental results show that the dialogue policy learned using PLA performs significantly better than the hand-coded policy and needs only few samples to learn fairly good policies. Most importantly, it is revealed from the table 3 that a higher value of learning rate ($\mu$) leads to faster convergence but results in a suboptimal action due to the fact that the pursuit scheme is guaranteed to be $\epsilon$-optimal. Hence, generally the computational efficiency at worst case is determined by $O(N^H)$, where $N$ is fixed with some predefined maximum value according to the experimental settings.

**Table 3. Value function estimates of the PLA method**

| Number of samples | PLA | Learning parameter | Estimated optimal |
|---|---|---|---|
| 4 | 11.33 (0.27) | 0.1591 | 13.602 |
| 10 | 12.87 (0.25) | 0.0670 | |
| 15 | 13.31 (0.16) | 0.0408 | |
| 25 | 13.56 (0.13) | 0.0273 | |
| Each entry represents the mean based on 100 test episodes with standard error in parenthesis | | | |

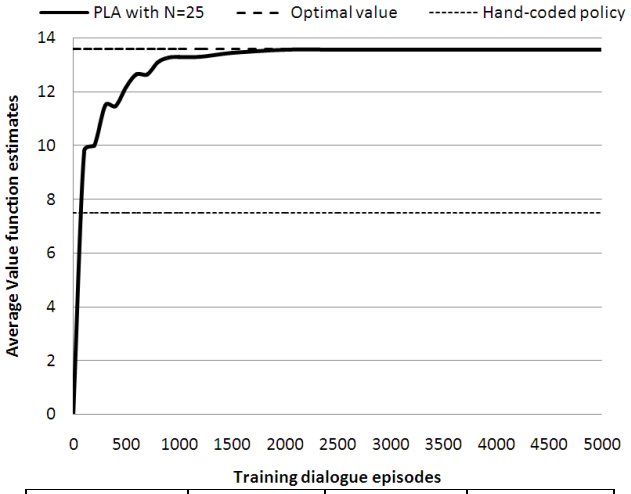


**Fig 3. Convergence of value function estimate tested on a user simulator**

Figure 4 shows the efficiency of the dialogue policies learned using PLA is marginally superior when compared to the policies learned using Reinforcement Learning with Grid Points (RLGP). Both these algorithms have been proven to converge towards the optimal policy. The PLA is converged very quickly with almost 500 steps. This speed of convergence is one of the advantages of the PLA method when developing a new system or adapting it to changes in the tendency in the data. In contrast, RLGP policies require a large number of iterations irrespective of the number of samples used for training. Since the policies learned using PLA have relatively less variations, these policies are more stable and reliable when compared to RLGP.

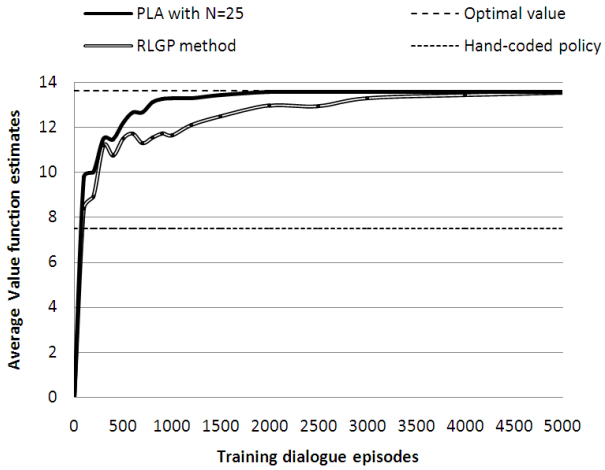


**Fig 4. Comparison of value function estimate in different approaches**

# 6. EVALUATION

To evaluate the effectiveness of a learnt strategy with real users, an end-to-end dialogue system with complete speech and language processing modules becomes inevitable. This section describes one such trial of learnt strategy with hand-coded baseline strategy. In both the cases, the performance of the trained policy and the hand-crafted policy were tested with the PARAdigm for DIalogue System Evaluation framework [20]. The primary objective is to maximize *user satisfaction*, and it derives a combined performance metric for a dialogue system as a weighted linear combination of *task-success measures* and *dialogue costs*.

## 6.1 System Configuration and Trial Setup

The system integrates existing modules for speech recognition and synthesis with hand-coded modules implementing language parsing, language generation and database connectivity. The recognizer used was the application toolkit for HTK [21] with a vocabulary of about 1,800 words. The speech synthesis was based on the festival text-to-speech system [22]. The natural language parser consisted of a hand-coded set of pattern matching rules to extract the dialogue act type with a list of slot/value pairs from each utterance and some code to parse dates. Similarly, the language generator was a relatively simple template-based system, comprising a set of rules for combining texts with slot values. The system dialogue acts allows the system to request the user for the slot values or to confirm these values, either explicitly or implicitly and to restart or end the dialogue. Finally, the system presents the results of a user's database query. The user dialogue acts allow the user to provide slot information and to terminate the dialogue.

To evaluate the learnt strategies, 20 postgraduate students (eight-female, 12-male) with an average age of 21 have tested our system. Each participant was asked to complete six pre-defined dialogue tasks where each task involved finding the details of the flights from one city to another city on a specific date and time. It is important to mention that since our testers had no previous experience with a dialogue system, our experiments were performed by novice users.

## 6.2 Subjective Trial Results

To evaluate user satisfaction, each participant was given the user-satisfaction survey Table 4 used within PARADISE framework, which measures the opinion of the respondents

with several metrics about the behaviour or the performance of the system (TTS performance, ASR performance, task ease, interaction pace, user expertise, system response, Expected behaviour, and Future use). The answers to the questions were based on a five-class ranking 1, indicating strongly disagree, to 5, indicating strongly agree. For our experiment, the mean user satisfaction value was 28.70 as shown in Table 5. These results indicate that the performance of the PLA learnt strategy (system A) is perceived to have a comparable quality to baseline RLGP (system B) in terms of task ease-of-use, user expertise, expected behaviour and future use.

**Table 4. User satisfaction metrics**

| Metric | Statement |
|---|---|
| TTS quality | I found the system easy to understand. |
| Task ease-of-use | It was easy to get the information I wanted. |
| Interaction pace | Was the pace of interaction with the system appropriate in this conversation? |
| User experience | I knew what I could say or do at each point in the dialogue. |
| System response | How often was the system sluggish and slow to reply? |
| Expected behavior | The system worked the way I expected it to. |
| Future use | Based on this experience, I would use this system regularly. |

**Table 5. User satisfaction survey results**

| Criteria | System A | System B |
|---|---|---|
| TTS quality | $4.3 \pm 0.7$ | $4.1 \pm 0.6$ |
| Task ease-of-use | $\mathbf{4.2 \pm 0.8}$ | $\mathbf{3.8 \pm 0.7}$ |
| Interaction pace | $3.7 \pm 0.4$ | $3.5 \pm 0.5$ |
| User experience | $\mathbf{4.3 \pm 0.7}$ | $\mathbf{3.9 \pm 0.7}$ |
| System response | $3.7 \pm 0.6$ | $3.5 \pm 0.5$ |
| Expected behavior | $\mathbf{4.1 \pm 0.8}$ | $\mathbf{3.6 \pm 0.7}$ |
| Future use | $\mathbf{4.1 \pm 0.9}$ | $\mathbf{3.5 \pm 0.7}$ |
| User satisfaction | **28.7** | **26.3** |

## 6.3 Objective Trial Results

The task success of the whole dialogue is measured by how well the system and participant accomplish the information requirements of the task by the end of the dialogue. In our experimental setup we have used the well known *Graded Task Success* (GTS) measure [23] that penalizes with different values defined as follows:

$$\text{GTS} = \begin{cases} 1 & \text{for flights details without problems} \\ 2/3 & \text{for flights details with small problems} \\ 1/3 & \text{for flights details with severe problems} \\ 0 & \text{otherwise} \end{cases}$$

Table 6 presents the summary of task success measures and dialogue quality costs which are intended to capture the user's perception of the system's performance. Similarly, table 7 summarizes the correlation analysis between task success measures and user satisfaction. From the result it was observed that all metrics defined in table 4 correlate moderately with overall user satisfaction, the metrics taking task easy and expected behavior having higher degree of correlation. According to PARADISE framework to compute the system performance we performed multiple regression analysis on our data with user satisfaction as the dependent variable, task-success measures and dialogue costs as the independent variables by the following equation.

$$performance = \left(\alpha * \mathcal{N}(k)\right) - \sum_{i=1}^{n} w_i * \mathcal{N}(c_i) \qquad (7)$$

where $\alpha$ is a weight on the task success metric $k$, and $w_i$ is a weight on the dialogue cost function $c_i$ computed from table 6. $\mathcal{N}$ is a Z-score normalization function $\mathcal{N}(x) = \frac{x-\bar{x}}{\sigma_x}$ defined over task success and cost values to account for the fact that they can be measured on different scales (sum, seconds, percentages, etc.) where $\sigma_x$ corresponds to the standard deviation of $x$. From several regression analysis on the normalized dialogue quality metrics (*system turns, user turns, system words per turn, user words per turn and interaction time*) user turns and GTS were chosen to be the predictors (independent variables) of user satisfaction (dependent variable) at $p < 0.05$ significance and yields the predictive equation:

$$Performance = .40\mathcal{N}(GTS) - 0.81\mathcal{N}(user\ turn) \qquad (8)$$

Hence for the 120 test dialogues we found the mean performance of the learnt policy perform better than the hand-coded policy at $p < 0.05$.

**Table 6. Mean values of task success and dialogue quality metrics based on 120 dialogues**

| Metric | System A | System B |
|---|---|---|
| GTS (%) | $76.09 \pm 7.2$ | $64.37 \pm 6.5$ |
| System turns | $6.15 \pm 0.7$ | $7.02 \pm 1.2$ |
| User turns | $4.75 \pm 0.5$ | $5.10 \pm 0.6$ |
| System words per turn | $32.30 \pm 5.2$ | $34.24 \pm 6.1$ |
| User words per turn | $5.79 \pm 1.5$ | $5.34 \pm 2.1$ |
| Interaction time | $22.10 \pm 9.6$ | $25.28 \pm 11.27$ |

**Table 7. Correlation coefficient between task success and user satisfaction metric (significance at $\rho < 0.05$)**

| Metric | GTS | |
|---|---|---|
| | System A | System B |
| TTS quality | .61 | .52 |
| Task ease-of-use | .82 | .69 |
| Interaction pace | .42 | .38 |
| User experience | .35 | .22 |
| System response | NS* | NS* |
| Expected behavior | .59 | .51 |
| Future use | .56 | .46 |
| * NS denotes Not significant | | |

# 7. CONCLUSION

This paper has presented a sample efficient approximation method for dialogue manager to generate an optimal response based on MDP framework. The proposed method computes both the optimal action from a given dialogue state and the corresponding optimal value on a sampled tree recursively through sampling and simulations. The performance of the system has been evaluated by interaction with a simulated user and through a live user trial with respect to PARADAISE framework. To generate an action, the sampled tree is traversed recursively in a bottom-up fashion with an adaptive sampling scheme which defines the stages in the problem space. At each level, the automaton queries the local policy based on the immediate reward, and the action proposed by the policy is either primitive or abstract, which depends on the

confidence measure of the slot in focus. If the action is primitive, it is directly executed by the automaton. Otherwise, the automaton queries the policy through its corresponding levels until the so-called slot values are grounded with higher confidence. It has been experimentally demonstrated that the optimal dialogue strategy can be obtained with a small number of training samples and has the potential for significant improvements in robustness compared to hand-crafted and grid-based approaches. Evaluation by real user using PARADAISE framework shows satisfactory results of the learnt policy compared to the baseline reinforcement learning with gird points. For future work, we believe that our approach can also be extended in the direct context of Partially Observable MDPs which account for approximately linear running time of learning algorithms when the state space is continuous and directly incorporates uncertainty imposed to a noisy channel. This suggests that system beliefs need to be modeled at different levels of granularity. In addition, while the results of RASA show faster convergence in finding optimal dialogue policy compared to the grid-based Monte Carlo algorithms, another matter for future investigation is to apply leaning automata based actor-critic approach into the proposed method to significantly speed up policy optimization. Finally, some supplementary informative interaction parameters with respect to PARADISE model are to be investigated to increase the validity of the resulting quality prediction.

## 8. REFERENCES

[1] McTear, M. 2004. Spoken Dialog Technology: Toward the Conversational User Interface, Springer-Verlag.

[2] Sutton, R. S., and Barto, A. G. 1998. Reinforcement Learning an Introduction, MIT press, Cambridge, MA.

[3] Singh, S., Litman, D., and Walker, M. 2002. Optimizing dialogue management with reinforcement leaning: Experiments with the NJFun system. Journal of Artificial Intelligence, vol. 16, 105– 133..

[4] Paek, T., and Pieraccini, R. 2008. Automating spoken dialogue management design using machine learning: an industry perspective. Speech Communication, vol. 50, no.8-9, 716–729.

[5] Chang, H. S., Fu, M., Hu, J., and Marcus, S. I. 2007. Recursive learning automata approach to markov decision processes. IEEE Trans Automat Contr, Vol. 52, no.7, 1349-1355.

[6] Young, S., Gasic, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., and Yu, K. 2010. The hidden information state model: A practical framework for POMDP-based spoken dialogue management. Computer Speech & Language, vol. 25, 150-174.

[7] McTear, M. 1998. Modelling spoken dialogues with state transition diagrams: experience with the CSLU toolkit. In Proc. ICSLP, 1223-1226.

[8] Goddeau, D., Meng, H., Polifroni, J., Seneff, S., and Busayapongchai, S. 1996. A form-based dialogue manager for spoken language applications. In Proc. ICSLP, 701–704.

[9] Rich, C., and Sidner, C. 1998. Collagen: A collaboration manager for software interface agents. User Modeling and User-Adapted Interaction, vol. 8, no. 3/4, 315–350, 1998.

[10] Pietquin, O., and Dutoit, T. 2006. A probabilistic framework for dialog simulation and optimal strategy learning. IEEE Trans Audio Speech Lang Process, vol. 14(2), 589–599.

[11] Frampton, M., and Lemon, O. Recent research advances in Reinforcement Learning in Spoken Dialogue Systems. 2009. The Knowledge Engineering Review, vol. 24, no. 04, 375–408.

[12] Henderson, J., Lemon, O., and Georgila, K. 2008. Hybrid Reinforcement/Supervised Learning of Dialogue Policies from Fixed Data Sets. Computational Linguistics, vol. 34 (4), 487–512.

[13] Cuayáhuitl, H., Renals, S., Lemon, O., and Shimodaira, H. 2010. Evaluation of a hierarchical reinforcement learning spoken dialogue system. Computer Speech & Language, vol. 24, no. 2, 395–429.

[14] Toney, D., Moore, J., and Lemon, O. 2006. Evolving optimal inspectable strategies for spoken dialogue systems. In Proc HLT, 173–176.

[15] Rieser, V. 2008. Bootstrapping Reinforcement Learning-based Dialogue Strategies fromWizard-of-Oz data. PhD dissertation, Saarbruecken Dissertations in Computational Linguistics and Language Technology, Vol. 28.

[16] Thathachar, M. A. L., and Sastry, P. S. 2004. Networks of Learning Automata: Techniques for Online Stochastic Optimization, Kluwer.

[17] Oommen B. J., and Misra, S. 2009. Cybernetics and learning automata. Springer Handbook of Automation, pp. 221-235.

[18] Chang, H. S., Fu, M., Hu, J. , and Marcus, S. I. 2007. An adaptive sampling algorithm for solving Markov decision processes. Operations Research, vol. 53(1), 126-139.

[19] Schatzmann, J., Weilhammer, K., Stuttle, M. N., and Young, S. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. The Knowledge Engineering Review, vol. 21, no. 02, 97–126.

[20] Walker, M., Litman, D., Kamm, C., and Abella, A. 1998. PARADISE: a framework for evaluating spoken dialogue agents. In Proc., Assoc. Comput. Linguist. (ACL), 271–280.

[21] Young, S. ATK: an application toolkit for HTK. Available:http://mi.eng.cam.ac.uk/research/dialogue/atk_home

[22] Yamagishi, J., Zen, H., Toda, T., and Tokuda, K. 2007. Speaker-independent HMM-based speech synthesis system – HTS-2007 system for the Blizzard challenge 2007. In Proc The Blizzard Challenge, http://www.cstr.ed.ac.uk/projects/festival/

[23] Dethlefs, N., Cuayahuitl, H., Richter, K., Andonova, E., and Bateman, J. Evaluating task success in a dialogue system for indoor navigation. In Proc. 14th Workshop on the Semantics and Pragmatics of Dialogue, 143-146.