Proc. 23rd IEEE Int. Symposium on Robot and Human Interactive Communication (RO-MAN 2014), Edinburgh, UK, pp. 692-697, IEEE 2014

Online Adaptation of Dialog Strategies based on Probabilistic Planning*

Steffen Müller, Sina Sprenger, Horst-Michael Gross

Abstract— **In this paper, a dialog modeling approach for long-term interaction between a service robot and a single user is presented, which enables a user-adaptive interaction behavior of the robot. Central element of the dialog system is a probabilistic model of the user's reactions to the robot's behavior, which is learned online and used for a probabilistic planning process based on message passing in a dynamic factor graph. The suggested approach has been applied to implement a complex application on a mobile service robot, which has been tested in a 10 day evaluation study with 16 users in order to get a feedback on usability of the interaction design, adaptation skills, and feasibility of a rapid application development. Results and findings of that study are presented here briefly.**

I. Introduction

In our current research project SERROGA (SERvice RObotics for health (Gesundheits) Assistance) [1], we focus on developing demonstrators for robotic applications in the context of accident prevention and assistance for elderly people living alone in their home environment. In that context, a long-term interaction behavior of an autonomous service robot and the human user is subject of our studies, since this is an essential factor for acceptance of such a system in the very heterogeneous target group. As a vision, we see a robot, that is living together with the humans in their home environment. Furthermore, we expect the development of an emotional binding of the user to his or her personal robot over the time, which is reinforced by the ability of the system to adapt to the user's needs and preferences.

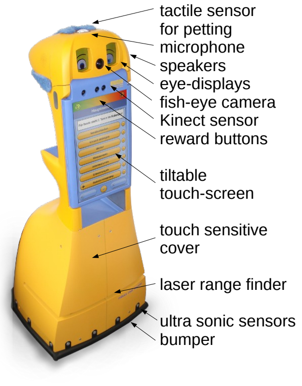


Fig. 1. Scitos-G3 robot called Max

The robot platform for our experiments in the SERROGA project is a Scitos-G3. (see Fig. 1) That robot was designed as a hands-off assistant in our previous project CompanionAble [2]. Meanwhile, it has been improved by additional interaction modalities enabling an intuitive communication by a multi-modal user interface consisting of a touchscreen, touch sensitive cover, and a touch sensitive patch of fur used for petting the robot. Current work in progress intends for inclusion of speech recognition as additional input channel. For output, the robot can use synthesized voice, its tiltable screen, an artificial face consisting of two eye displays, as well as its motion capabilities. A laser range finder and a Kinect sensor, together with a differential drive enable autonomous localization and navigation skills. A fish-eye camera is used for person detection and tracking, which is a key functionality for a successful interaction.

Based on that hardware, we implemented a layered software architecture using the robot middleware MIRA [3] in order to enable a modular application concept allowing for an extensible set of services that are realizable in a rapid development process. Furthermore, the robot has to appear as a single conversational subject managing the various services executed in parallel. Details on our software architecture can be found in [2] as well. Additionally, adaptation skills in the dialog behavior which are a prerequisite for long-term interaction were considered in the implementation. In this paper, we focus on the dialog manager and dialog configurations in the individual independent services.

In the remaining part of the paper, the dialog modeling concept is introduced based on an example from our evaluation application. For that, first, some basic ideas and related work on adaptivity in the human-machine dialog are discussed, before the details of our evaluation study and the concept of dialog modeling as well as the probabilistic planning approach are presented. In the final part, the findings of our user study and the achieved results are discussed.

II. Adaptivity in the Dialog Behavior

During a long-term scenario, we expect a prototypical evolution of an interaction behavior over time. In the first phase, the user needs to get to know the robot. The system has to give advice how to use it and has to introduce its capabilities. The dialog initiative is primarily at the side of the robot. Later, in a second phase, when the user is more familiar with the capabilities of the robot, the initiative will be in the user's hand, and the robot should learn the user's preferences and needs. Specifically, the robot has to learn, which services are used in which situation and what are the user's attitudes towards the various options the robot has in its dialog behavior. Also preferred selections are learned by the robot in order to apply that knowledge later. After a while, a stable phase is reached. The robot now can make use of the observations in interaction with that specific user. Thus, it is able to act proactively depending on the current situation. Then, a mixed initiative dialog should emerge within the limited domain of the robot's services. Nevertheless, the ability to learn and change interaction behavior should not be limited to the initial phases. Changes of user's attitudes have to be tracked continuously and life-long. Thus, it is essential,

*This work has received funding from the Federal State of Thuringia and the European Social Fund (OP 2007-2013) under grant agreement N501/2009 to the project SERROGA (project number 2011FGR0107).

that the interaction policy is adapted online based on passive observations on user's reactions to specific behaviors, but also based on explicit rewards given by the user, e.g. by means of petting the robot to praise a desirable behavior. Unfortunately, such a reinforcement-based approach comes along with the exploration-exploitation dilemma. The system may behave in the way the user commended, but it also has to try other possibilities not necessarily liked by the user in this moment in order to find a better policy and learn the most favored interaction behaviors.

III. Related Work

In literature, several approaches for dialog modeling can be found, mainly in the field of natural language generation and speech-based dialog systems, which also can have capabilities of adaptation, but are not exactly compatible with the service robot domain.

A common approach is using Reinforcement Learning (RL) techniques for optimization of a dialog flow, even when inputs are uncertain. The Partially Observable Markov Decision Process (POMDP) is used here, but this comes along with high computational effort, which is handled by several approaches for simplification [4], [5]. These approaches try to find a policy in order to maximize the discounted future reward, that is generated according to a system internal reward function. Therefore, the system designer has to define in advance the long-term goals during upcoming dialogs. A drawback of many Reinforcement Learning approaches is the effort for incorporation of reward given by a user online. In order to consider these observations immediately, the complete interaction policy has to be optimized again. Therefore, in the domain of dialog learning, a complicated application development scheme has been established, where in a training phase interaction data is acquired (often with a mock-up or Wizard of Oz application), and afterwards the optimization run is applied offline in order to generate the (non-adaptive) productive dialog system. This is a clear drawback we want to overcome in our solution.

Pineau et al. [6] applied a POMDP model for learning an optimal dialog behavior for a service robot called Pearl. An interesting aspect of that approach is a hierarchical task decomposition, which is similar to our intended modularity of services. Although they applied that decomposition of the general assistance task, the complexity of the realizable functionality was still very limited. That approach also relies on a hand-crafted reward function, and the policy was computed offline before application. Thus, the robot could not consider individual user's characteristics discovered at runtime.

An alternative model for learning a behavior online from direct user rewards is called TAMER [7]. This approach explicitly models which feedback a user gives for a certain state-action pair, and then acts greedily in order to get the maximum reward for the next action. The argumentation for this λ_0 strategy is the idea, that the human supervisor estimates the utility value of a state-action combination and represents this in the reward signal already. This might be correct to a certain degree, but has a clear disadvantage. The system can only act in order to achieve the user's goals, but is not able to incorporate system-internal concurring goals. One interesting aspect of the TAMER system is the reward model. This allows to predict the user rewards and apply them internally, even if the user is not giving feedback for each action. Thus, this model allows that the user has to get active by giving feedback only if s/he wants to modify the behavior, not if s/he is pleased with it.

An alternative dialog system applying probabilistic inference is presented in [8] and later in [9]. Inference techniques have been applied to a statistical model of the dialog in order to reason the goals the user might have in mind and, therefore, decide which information needs to be asked or given in the next dialog steps. Unfortunately, this idea is not directly transferable to our scenario, where the goal of the dialog is not only determined by the user but also by the system itself (e.g. a health assistant robot wants to engage the user in communication or physical activities). Additionally, the direct reward by the user, which is used to modify the way things are communicated, is hard to introduce in that approach.

In our approach, we wanted to combine the idea of the robot system's own goals, which e.g. can be a success in encouraging a user to perform some exercises or only a fast task completion, with explicit and implicit rewards given directly by the user during the interaction. To achieve the capability of online learning/adaptation, we suggest the substitution of the RL-typical representation of the optimal policy for the dialog by an online planning mechanism. This also allows for changing optimization goals as well as discovering new states of the dialog at runtime, which would be difficult or even impossible for implicit planning methods.

Today, a variety of highly sophisticated dialog modeling techniques exists, mostly related to a very complex application development process. Our aim is to provide a framework for rapid application development, which is realized by combining simple frame-based multi-modal dialog with the capabilities of optional adaptation without introducing additional configuration effort. A key to a manageable design effort is the possibility for problem decomposition. Similar to hierarchical abstract machines [10], that are also used in the Reinforcement Learning domain for restriction of the action space, in our approach individual sub-dialogs are defined as independent modules each restricting the policy to a reasonable subset and having the ability of calling other sub-dialogs on demand.

IV. The OfficeMate Evaluative Scenario

Before the dialog model and the planning mechanism are presented, the scenario of our evaluation study is introduced to serve as an example for the following explanations. To study the interaction design to be used for our health assistant robot, we had to find a setup for conducting several long-term interactions in parallel. Caused by restrictions on access to the robot as well as the number of test users needed, an evaluation scenario had been chosen, that involved 16 members of our lab at ten consecutive workdays in parallel. The robot acted as an "Office Mate" by visiting each trial participant once a day beginning at 9:00 a.m. and offering its services. Further points of interest for our study were the feasibility of the application development process and the correct operation of the user adaptation skills.

The evaluation involved two questionnaires the participants had to answer, which were intended to get a baseline of the subjects' prior expectations regarding the robots capabilities, skills and services and to ask for the user experience afterwards.

Among the participants, there were 4 female and 12 male, while half of the group had an age between 18 and 29 and the other half from 30 to 45 years. All participants were familiar with IT but were not directly involved in the development of the robot.

From the questionnaires, a set of services had been extracted that the participants identified as possible functionality an office companion robot should have. These services comprise a *greeting* with some smalltalk regarding the well-being of the user, providing the *menu for the refectory*, providing *weather information and forecast*, *entertaining* with a set of websites (comics, jokes), *informing* via a set of news websites, and *reminding of appointments* as well as showing the *current date and time*. These services were implemented on our test system. Also the content of the news and entertainment services was selected based on the questionnaires and likewise the preferred way of talking to the user either by formal of informal speech.

The interaction sessions conducted with each participant once per day had the following structure: First the robot autonomously moved to the office of the participant. Once a person was detected at the target point, the robot asked to confirm the identity and started the greeting dialog. At that point, the user-specific dialog models were loaded from file and allowed for the adaptivity in the remaining dialog. After this, the robot activated a main menu dialog, where the user could start services or got suggestions on which services might be useful in that situation. During the dialog, the user had the possibility to give explicit reward to the robot at any time by means of a "like" and "dislike" button on the screen. Also petting the robot served as positive reward during the interaction. In the dialog session, a turn-based scheme of interaction was used. When the robot was on turn, it expressed a message, asked for some information or simply provided a screen with the requested content to the user. Each robot action consisted of a spoken text and a respective screen page offering the input options needed. Then followed the user's turn, and the robot waited for some input, which was communicated by the touch screen. Also internal events or timeouts could trigger a new turn for the system. The session was finished when the user did not react anymore or actively selected the "leave me now" entry on the main menu.

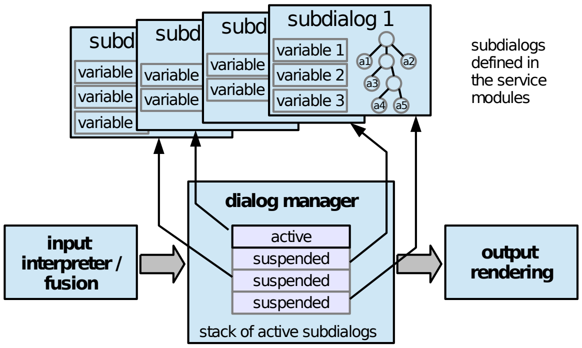


Fig. 2. Concept of the dialog manager: First, the variables in sub-dialogs are filled by a multi-modal input interpreter. Then, the top of the stack of active sub-dialogs in the dialog manager is considered for action selection where the planning inference is applied to select an action, which is expressed by output renderer afterwards.

V. ADAPTIVE MODULAR DIALOG SYSTEMS

The implementation of the dialog system is realized in the control layer of our software architecture [2]. It consists of a central Dialog Manager which consumes all kinds of inputs from the graphical interface (GUI) and other input modalities, updates the dialog state and selects the adequate action depending on that state. Other robot software components for navigation and perception skills realize an interface to the robot infrastructure for the individual services in the task layer. In this way, the software realizes a modular design, where each sub-dialog (greeting, weather info, news, entertainment, etc.) is an independent module defined by a service in the task layer. Thus, it is easy to add new functionality and refer to, or combine existing dialog capabilities in new dialogs, such that the borders of the modules get blurry for the user.

The modules implement a content specific back-end functionality and define the required sub-dialogs. The configuration for a sub-dialog is mainly a definition of **the state space** S (input variables and internal context) and a **set of output actions** available $A = \{a_1, \ldots, a_K\}$ as well as a **default policy**. To get an impression of that principle, the main menu sub-dialog of our Office Mate is described in more detail. Aims of the main menu dialog corresponding to respective system actions are a_1 asking for the next service a user wants to use, a_2 suggesting new services not used so far, and offering services proactively in situations, where the system has observed that the user often started a specific service function. For that last function, the system is able to predict user's selections (see Sec. V-C) but it has to ask for confirmation of the choice, if the prediction is ambiguous, which is a further possible action a_3. In order to introduce the user to the robot's way to interact, all these actions $a_1 - a_3$ are available in a second version $a_4 - a_6$ with a supplementary verbal and written advice. Besides these user centered actions, also system internal actions are available, which comprise the activation of a sub-dialog for the selected service a_7 or leaving the main menu dialog, when the user is not responding anymore a_8. Therefore, expected user inputs are the selected service or the positive or negative confirmation of the the robot's suggestions. These inputs directly correspond to two of the state variables v_1 and v_2 of the sub-dialog. Further state variables are representing the situational context and the history of the conversation. In detail we have a variable v_3 holding the number of appointments scheduled for today and a further variable v_4 counting the services executed during the session, which is necessary to predict a sequence of selections correctly. All variables can have either real-valued or discrete domains. Additionally, the state vector S also contains a certainty value $C_1, \ldots, C_n$ for each variable representing the reliability of the acquired knowledge. This is necessary to represent uncertainty of inputs, e.g. from speech recognition. Also predictions of user's inputs are uncertain and can be handled that way (see Sec. V-C). As a third part of the dialog's state space, we hold a counter H_k for each action a_k available, keeping track of the history of executed system actions. By means of these counters, it is possible to change system's reactions after a couple of repetitions of certain actions.

Hence, the state representation for one sub-dialog is a vector:

$$S = (V_1, \ldots, V_N, C_1, \ldots, C_N, H_1, \ldots, H_K) \qquad (1)$$

Based on the current value of that state S, the dialog manager has to select an action each time the system is on turn. In order to reduce the necessity for exploration in an unordered set of actions and in order to prevent from selection of irrational action sequences that might frustrate the user, it is necessary to limit the set of selectable actions in each state S. In our system, this is done by a manually designed decision tree over the state variables (see Fig. 2 upper part), which is more flexible compared to finite state machines regarding the aim for a mixed initiative dialog. Each node in the decision tree decides between two alternative branches in the tree based on a condition on the state S, while in each leave of the decision tree there is one or a set of possible actions. These sets of actions allow to define options the dialog manager has to chose from, based on the rewards and observed transitions. System internal goals are realized by means of labeling states as desirable goal states when a respective action is selected. For instance, in the main menu example the "leave me now" selection followed by the action that finishes the sub-dialog is the goal. Other services like the *entertaining website provider* define a goal when the user has left to the main menu and at least one website has been visited. In the *entertain* as well as in the *news* service, the user first has to select from a list of available sites and then gets shown the respective content. Again the system has the ability to suggest websites the user did not select so far or start a website directly based on an internal prediction of the selection to be expected. It also may ask for confirmation, if this prediction yields ambiguous results. Like in the main menu, for all services, actions communicating to the user are available in two versions: with and without additional explanation.

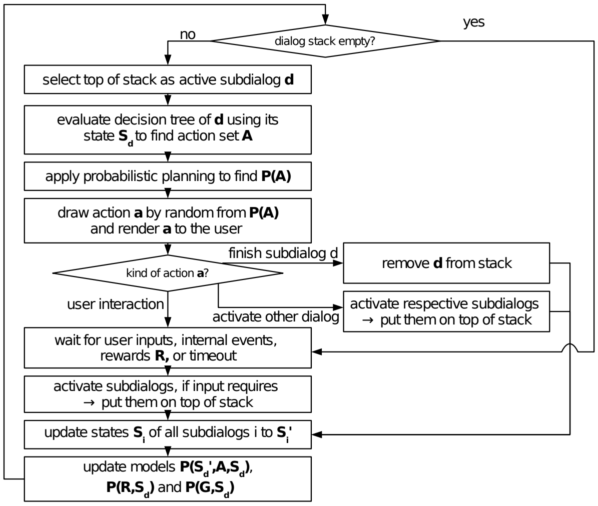

no
dialog stack empty?
yes
select top of stack as active subdialog d
evaluate decision tree of d using its state S_d to find action set A
apply probabilistic planning to find P(A)
draw action a by random from P(A) and render a to the user
finish subdialog d
remove d from stack
kind of action a?
user interaction
activate other dialog
activate respective subdialogs → put them on top of stack
wait for user inputs, internal events, rewards R, or timeout
activate subdialogs, if input requires → put them on top of stack
update states S_i of all subdialogs i to S_i'
update models P(S_d',A,S_d), P(R,S_d) and P(G,S_d)

Fig. 3. Workflow in the dialog manager

A. Control flow in the dialog

Having defined the structure of a sub-dialog, now the coordination of user inputs, multiple active sub-dialogs, and the system output generation are explained. In general, a turn-based control flow is realized, where user input turns and system turns alternate.

The dialog manager holds a stack of active sub-dialogs, where the top most is that one, which is evaluated each time a system action is necessary (start of a system turn). The dialog manager evaluates the decision tree of the top most sub-dialog in the stack using its current state $S = s_0$ in order to get the possible action set for the current situation (see Fig. 3). If only one action is available, this one is executed by the output renderer directly. Here, mainly screen and speech outputs are generated which may refer to values of the variables in the sub-dialogs and communicate content suitable for the current situation (asking questions, confirming inputs or giving answers). Also special actions, like activation of other sub-dialogs or canceling an active sub-dialog, are possible. If a new sub-dialog is activated, the former top most in the stack gets suspended. When the interrupting sub-dialog is finished, the suspended one returns to the top of the stack and gets resumed.

If multiple actions are allowed by the decision tree, the probabilistic planning process is triggered. This planning yields a probability distribution on the actions $P_{plan}(A)$, representing the maximum probability of reaching a system internal goal state while maximizing the expected user rewards on the intermediate states on the way for a given action A.

Since the system does not know neither the user's rewards and the possible transitions in the dialog states nor the goal states that are defined by the actions in these states, there is a need for exploration additionally to the aim for exploitation of the knowledge already acquired (known as exploration-exploitation dilemma in Reinforcement Learning). Furthermore, the progress in the phase model of the long-time interaction, introduced in section II, also has to be considered during action selection. Thus, two additional probability distributions are combined with the planning result before the executed action is drawn from the resulting distribution. The first one represents the number of executions of each available action $P_{count}(A)$ to enforce that all possible actions are tried out equally often during exploration. The second distribution $P_{prio}(A)$ allows for consideration of a priority that is depending on the progress in the long-time interaction. In this way, in the beginning phase, more actions with explanations for the user are selected, and in the stable phase only straight actions without additional help messages as well as proactive actions are recommended.

After the system action is executed, the dialog manager is in the user's turn and waits for any state change by an internal event, by user inputs and rewards, or just by a timeout. All multi-modal user inputs are asynchronously processed in parallel by the input interpreter and will update the dialog state of the respective sub-dialog they belong to. Special inputs or internal events may activate a sub-dialog, which is raised on the stack of active sub-dialogs suspending the old top element. Before starting the new system turn, the probabilistic models used for planning are updated at runtime with the observed state transition and rewards induced by the user's reactions.

B. *Modeling of Interactions and Planning*

For the planning and adaptation, the system needs to represent knowledge on the history of interactions with the user. This is done by means of several probabilistic models, which are sketched here in the following and are described in detail in [11]. The dialog manager first builds up a persistent probabilistic transition model at runtime, that is representing the probability of reaching a certain state S' given a predecessor state S and the executed action A. Here, user specific decisions and reactions, as well as the internal restrictions on the state sequences are learned, such that the planning system does not need to be configured with that knowledge beforehand. The representation of that model is a sum of samples $s_j = (S', S, A)_j$ each of them equipped with a weight w_j. Remembering the various components of the state vector S, it is obvious, that these samples have a very high dimensionality and therefore, it is very unlikely that exactly the same states appear very often. To generalize consequences among similar states, a similarity function $\delta(s_a, s_b) : \Re^d \times \Re^d \mapsto [0, 1]$ defines a neighborhood among samples $s_a \in \Re^d$ and $s_b \in \Re^d$. By this means, the set of samples defines a continuous probability function over the space of possible transitions.

$$P(S'_t, S_t, A_t) \propto \sum_j w_j \delta(s_t, s_j) \qquad (2)$$

For realizing a goal directed planning of action sequences, additionally a model of goal states $P(G|S)$ and a model of rewards $P(R|S)$ gained in a certain state are defined similarly as weighted set of samples. The goal state probability represents the system internal interaction targets, while the rewards stand for the user internal targets. Our planning algorithm is able to take both parts into account properly. The rewards are only recorded, if positive or negative reward events took place. By ignoring zero rewards and using the former average in that state instead, the policy remains stable, even if the user is pleased with dialog course and does not reward every action individually.

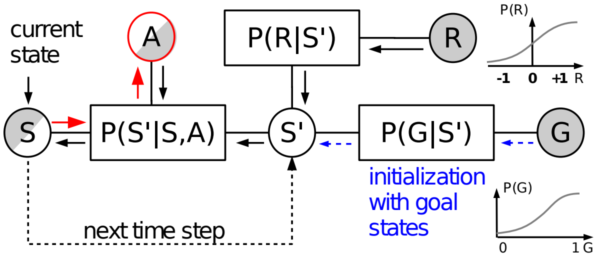


Fig. 4. Dynamic factor graph used for planning of a most promising action A, given the current state S and a goal state distribution (blue/dashed message for initial step). Inference with Max-Product algorithm is executed in a loop following the black message passing arrows looking for n time steps in the future. Once a matching state S could be reached, the actual action A is inferred following the red/bold arrows.

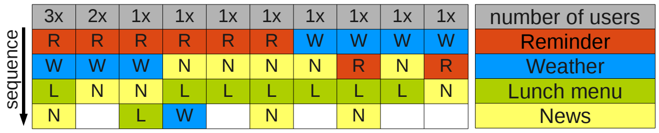
Initially, these three user-specific models for each sub-dialog are empty and have to be filled during the interactions by observing and counting the real transitions, rewards, and occurrence of goal labels. Fig. 4 shows the structure of the dynamic factor graph that is used for inference of the most promising action by means of a message passing algorithm called Max-Product algorithm [12]. The Max-Product algorithm can find the maximum probability in the probabilistic model belonging to the factor graph, that is reachable with a given set of fixed or observed variables. In [11], we explained in detail, how the algorithm is realized for the special kind of model representation we use. In our setting, we search for the maximum probability for each possible action to reach a goal and get maximum rewards on intermediate states, while the number of steps needed to reach a goal is unknown. That yields a distribution $P_{plan}(A)$. To overcome that drawback of the unknown number of steps needed, we can benefit from the structure of the problem, which is asking for the first action in the chain only. The inference of possible predecessor states, therefore, can be done in a loop until we find a state, that matches our current dialog state. Thus, for each intermediate time step in the planning horizon, we can infer the probability for the actions, and by searching for the maximum probability for each action over all time steps we gain the desired $P_{plan}(A)$.

C. *Prediction of user preferences*

In many situations in a repeatedly conducted dialog between the robot and the user, the annoying questions for options (e.g. which website should be shown) can be omitted, when using the former choices in a similar simulative context. The transition model we built from the dialog history exactly contains this information.

Thus, instead of asking the user for the information, the robot can try to infer the desired value, which is changing the state S of the sub-dialog similarly without any inputs from the user. Depending on the outcome of that, the dialog can continue either with a confirmation of that fact or with a question for specification of the information, if the inference did not yield a significant probability. By means of that, also the proactive situation-dependent offer of services can be realized easily by integration of prediction in the main menu sub-dialog.

D. Results From the OfficeMate Experiment



3x	2x	1x	1x	1x	1x	1x	1x	1x	1x	number of users
R	R	R	R	R	R	W	W	W	W	Reminder
W	W	W	N	N	N	N	R	N	R	Weather
L	N	N	L	L	L	L	L	L	N	Lunch menu
N		L	W		N		N			News

Fig. 5. Individual patterns in the order of application usage evolving for the participants of our evaluation study; This shows the ability for adaptation to individual preferences of the users.

Evaluation of the second questionnaire in our user study showed that all participants who had more than four interactions (not all users were available all the days) did notice that the robot learned their preferences and also changed its behavior over time. By analysis of the system logs of service activation, we found several patterns in the order of service usage by the users (see Fig.5). The robot also managed to suggest these services proactively in these cases. For three of the users the number of interaction sessions was not sufficient, or they canceled the sessions without consuming any service at all. Thus, for these participants no pattern could be found in the log data. Besides the prediction of services to be selected, also the rewarding mechanism was used successfully to prevent from getting the tutorial at the beginning of a session. This was possible because the tutorial was a completely independent branch in the possible dialog flow, which can be liked or not. In other cases giving explicit reward was not intuitive, when only modifications of the expressions should be distinguished. For example, the earlier mentioned main menu actions with and without additional help have not been noticed as alternatives and thus did not get rewarded accordingly.

Users also mentioned that they were a bit confused by the option of rewarding the robot with only a positive or negative button. They wanted to reward explicitly special aspects of the complex behavior, like volume of speech, but it was not transparent to them, which options the system had to select from. Thus, for a real comfortable adaptation, reinforcement based learning has to be combined with direct teaching of required behavior or the possibility for changing properties of the behavior manually. This means combining skills of adaptivity and adaptability.

Furthermore, the difficulties in using the *like* or *dislike* buttons, are an indication for using more implicit rewards in the future. These rewards can be deduced from observations on the user's reactions as well as system-internal events and therefore, they are not necessarily conscious to the user.

Also important is a better observability of the exploration of alternative actions. In some situations, although it had already been rewarded negatively for a behavior, the robot acted again in an unwanted way due to the exploration strategy which involves some random selection of possible actions. In these situations, users get frustrated as they expected that the robot persistently keeps a behavior they are pleased with even if that special behavior results from a random sequence. Therefore, in future work, the exploration strategy has to be reworked completely. It would be possible to apply a more greedy action selection strategy as proposed in the TAMER system [7] to reduce unexpected repetitions of actions that result from these exploration steps. Unfortunately, this would conflict with the ability to find an action sequence that reaches system internal goals.

VI. Conclusion

We could show in an experimental setup, that a dialog manager using an online learning transition model for planning of action sequences, based on a high dimensional state space, succeeded in adapting to different users in a longer period of interaction. With this ability, we already use the system for realizing the prototype of our health assistance robot in the SERROGA project. In the future, the capabilities regarding generalization in the planning process are in the focus of development in order to enable a faster adaptation and less exploration effort combined with a more significant state space (situational context).

References

[1] SERROGA project website: http://www.serroga.de

[2] Gross, H.-M., Schröter, C., Müller, S., Volkhardt, M., Einhorn, E., Bley, A., Martin, C., Langner, T., Merten, M., Progress in Developing a Socially Assistive Mobile Home Robot Companion for the Elderly with Mild Cognitive Impairment, in Proc. of IEEE/RJS Int. Conf. on Intelligent Robots and Systems, IROS 2011, pp. 2430-2437, 2011.

[3] Einhorn, E., Langner, T., Stricker, R., Martin, C., Gross, H.-M. MIRA - Middleware for Robotic Applications. in Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems IROS 2012, pp. 2591-2598, 2012

[4] Thomson, B. et al., Bayesian update of dialogue state for robust dialogue systems, in Proc. of Acoustics, Speech and Signal Processing, ICASSP 2008, pp 4937-4940, 2008.

[5] Toussaint, M. et al., Probabilistic inference for solving (PO)MDPs, Informatics Research Report 934, School of Informatics, University of Edinburgh, 2006.

[6] Pineau, J. and Thrun, S., High-level robot behavior control using POMDPs, AAAI-02 Workshop on Cognitive Robotics, Vol. 107, 2002.

[7] Knox, W. B. and Stone, P., Interactively shaping agents via human reinforcement: The TAMER framework, in Proc. of the fifth international conference on Knowledge capture. ACM, pp. 9-16, 2009.

[8] Meng, H.M. et al., The Use of Belief Networks for Mixed-Initiative Dialog Modeling, in Transactions on Speech and Audio Processing, vol. 11, num. 6, pp. 757-773, 2003.

[9] Martinez, F. F. et al., Evaluation of a spoken dialogue system for controlling a Hifi audio system, in Proc. of Spoken Language Technology Workshop, pp. 137-140, 2008.

[10] Parr, R. and Russell, S., Reinforcement learning with hierarchies of machines. in Advances in neural information processing systems, pp. 1043-1049, 1998.

[11] Müller, S., Sprenger, S., Gross, H.-M., OfficeMate: A Study of an Online Learning Dialog System for Mobile Assistive Robots, in ADAPTIVE 2014, The Sixth International Conference on Adaptive and Self-Adaptive Systems and Applications, pp. 104-110, 2014.

[12] Loeliger, H.A., An introduction to factor graphs,in Proc. Signal Processing Magazine, IEEE, vol. 21, num. 1, pp. 28-41, 2004.