Towards Online Planning for Dialogue Management with Rich Domain Knowledge

Pierre Lison

Abstract Most approaches to dialogue management have so far concentrated on offline optimisation techniques, where a dialogue policy is precomputed for all possible situations and then plugged into the dialogue system. This development strategy has however some limitations in terms of domain scalability and adaptivity, since these policies are essentially static and cannot readily accommodate runtime changes in the environment or task dynamics. In this paper, we follow an alternative approach based on online planning. To ensure that the planning algorithm remains tractable over longer horizons, the presented method relies on probabilistic models expressed via *probabilistic rules* that capture the internal structure of the domain using high-level representations. We describe in this paper the generic planning algorithm, ongoing implementation efforts and directions for future work.

1 Introduction

Dialogue management is at its core a decision-making operation: given a specific conversational situation the agent finds itself in, the objective of the dialogue manager is to find the optimal action to perform at that stage, depending on some performance criteria. In order to search for this optimal action, the agent often needs to take into account not only the local effect of the action, but also how it might influence future states and actions. An action might therefore be locally sub-optimal but still be selected if it contributes to a higher objective within the interaction. A typical example of these lookahead strategies pertains to clarification requests. These requests might indeed have a slightly negative effect in the short term but are often beneficial on the longer term, since they can help reducing the state uncertainty and therefore lead to more successful dialogues [15].

Pierre Lison
University of Oslo (Norway), Department of Informatics, Language Technology Group
e-mail: plison@ifi.uio.no

This observation has led many researchers to cast dialogue management as a decision-theoretic *planning problem* [4, 23, 22]. Such formalisation relies on the specification of rewards associated to particular (state,action) pairs. In this setting, the role of the dialogue manager is to select the action which yields the highest return (cumulative discounted reward) over the course of the interaction. One major benefit of this formalisation is that it enables the system designer to flexibly encode the trade-offs between the various, sometimes conflicting objectives of the interaction. Dialogue strategies can then be automatically optimised for the domain at hand, leading to conversational behaviours which are often more flexible and natural than handcrafted strategies [8].

So far, most approaches to dialogue management relying on decision-theoretic planning perform this optimisation entirely *offline*, by precomputing a dialogue policy mapping every possible state of the dialogue to the optimal action to perform at that state. While this approach is attractive in terms of runtime computational savings, it also presents a number of limitations. The first challenge is that policy optimisation becomes increasingly hard as the size of the dialogue state grows, since the policy must be calculated for every hypothetical situation which might be encountered by the agent. The problem is especially critical when the dialogue state is partially observable and therefore expressed in a high-dimensional, continuous space. The second challenge is the difficulty of adapting or refining the policy once it has been calculated. Precomputed policies must be recalculated every time the domain-specific models are modified or extended. This is an important drawback for application areas such as human-robot interaction, cognitive assistants or tutoring systems, since these domains often rely on environment or task models whose dynamics can vary at runtime, while the interaction unfolds (for instance in order to adapt to shifting user preferences).

An alternative approach that has recently gained popularity in the POMDP planning literature is to perform planning *online*, at execution time [18]. Compared to offline policies, the major advantage of online planning is that the agent only needs to consider the current state to plan, instead of enumerating all possible ones. It can also more easily adapt to changes in the environment or task models. The available planning time is however more limited, since planning is in this case interleaved with dialogue system execution and must therefore meet real-time constraints[1].

To address these performance constraints, we investigate in this paper the use of prior domain knowledge to filter the space of possible actions and transitions that need to be considered by the planning algorithm. The key idea is to structure the probability and reward models used for planning in terms of high-level, probabilistic *rules*. These rules can notably express which actions are deemed to be relevant for a given dialogue state, and thus filter the space of actions to consider at planning time. The main intuition is that by exploiting the internal structure of the domain

[1] Interestingly, offline and online approaches to planning are not mutually exclusive, but can be combined together to offer "the best of both worlds". The idea is to perform offline planning to precompute a rough policy, and use this policy as a heuristic approximation to guide the search of an online planner [18]. These heuristic approximations can for instance be used to provide lower and upper bounds on the value function, which can be exploited to prune the lookahead tree.

and integrating it in our models, we can develop algorithms which are significantly more efficient than approaches relying on unstructured models.

The structure of this paper is as follows. We start by briefly reviewing the mathematical foundations of our work, and then describe our representation formalism, based on the concept of a probabilistic rule. We then detail our planning algorithm, which relies on these rules to find the optimal action sequence to perform at a given state. We then describe our current implementation efforts to develop an efficient planner based on this algorithm. Finally, we compare our approach to related work in the field, and conclude the paper.

2 Background

2.1 Dialogue state

Virtually all dialogue management frameworks rely on the notion of a *dialogue state*, which can take various forms, depending on the chosen level of expressivity and account of uncertainty. In this work, we encode the dialogue state as a *Bayesian Network* [21], where each node represents a distinct state variable deemed to be relevant for decision-making, such as the hypothesised user intention or contextual features. These variables might be conditionally dependent on each other, which is easily encoded in a Bayesian Network via directed edges.

Formally, let $X_1 ... X_n$ denote a set of random variables. Each variable X_i is associated with a range of mutually exclusive values. A Bayesian Network defines the joint probability distribution $P(X_1 ... X_n)$ via conditional dependencies between variables using a directed graph where each node corresponds to a variable X_i. Each edge $X_i \rightarrow X_j$ denotes a conditional dependence between the two nodes, in which case X_i is said to be a *parent* of X_j. A conditional probability distribution $P(X_i|Parents(X_i))$ is associated with each node X_i, where $Parents(X_i)$ denotes the parents of X_i.

A Bayesian Network can be straightforwardly extended for capturing utility-related information. In practice, this is realised by adding *utility* and *decision* nodes to the network. A utility node encodes the utility associated with a particular set of dependent variables. Typically, at least one of these dependent variables is a decision node, which describes a set of possible actions that the agent can perform. Figure 1 illustrates a Bayesian Network extended with such nodes.

2.2 Decision-theoretic planning

In dialogue management, we are interested in the action which has the highest expected discounted cumulative reward for a given horizon. In order to find this op-

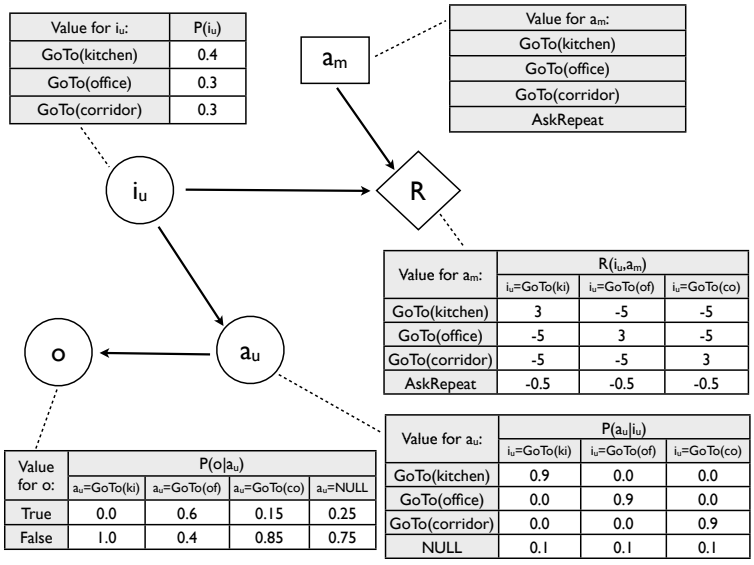

Fig. 1 Example of Bayesian Network extended with utility nodes (shown as diamonds) and decision nodes (shown as rectangles). The network represents a domain for a robot that is able to perform 4 actions (going to the kitchen, the office, the corridor, or asking the user to repeat). The node i_u represents the user intention and the node a_u the last dialogue act from the user. The latter is connected to an evidence node o which represents the actually observed N-best list of user acts (e.g. coming from speech recognition). Depending on the user intention, the execution of a specific action will yield different values for the reward R. In our case, we can easily calculate that $R(a_m = \mathsf{GoTo(office)}|o = \mathsf{True}) = 0.96$, $R(a_m = \mathsf{GoTo(corridor)}|o = \mathsf{True}) = -3.3$, $R(a_m = \mathsf{GoTo(kitchen)}|o = \mathsf{True}) = -4.65$, and finally $R(a_m = \mathsf{AskRepeat}|o = \mathsf{True}) = -0.5$.

timal action, the dialogue manager must be able to perform some form of forward planning to estimate the expected future rewards of every action.

To this end, let us assume that we have a dialogue state (also called belief state) b, with $b(s) = P(s)$ being a joint probability distribution over the possible state values. This joint probability distribution can for instance be described as a Bayesian Network of state variables (cf. previous section). In addition, we will also assume that we can structure our models in the generic form of a *Partially Observable Markov Decision Process* (POMDP), with a reward model $R(s,a)$ describing the rewards associated with particular actions, a transition model $P(s'|s,a)$ describing the state following the execution of action a in state s, and an observation model $P(o|s,a)$ encoding the expected observations (in our case, an N-best list of user dialogue acts) for a given state s after action a. The expected cumulative reward of a state-action sequence $\langle b_0, a_0, b_1, a_1, ..., b_n, a_n \rangle$ with a discount factor γ is then defined as:

$$Q[(b_0, a_0, b_1, a_1, ...b_n, a_n]) = \sum_{t=0}^{n} \gamma^t R(b_t, a_t) = \sum_{t=0}^{n} \gamma^t \sum_{s \in S} b_t(s) R(s, a_t) \qquad (1)$$

where the dialogue state b_{t+1} is defined as an update from b_t:

$$b_{t+1}(s') = P(s'|o_{t+1}, a_t, b^t) = \alpha\, P(o_{t+1}|s', a_t) \sum_{s \in S} P(s'|s, a_t)\, b^t(s) \tag{2}$$

with α being a normalisation constant. Of course, the observations o^t are not known in advance, so the planning strategy must take into account the range of possible observations which might follow from the execution of a given action.

Using the fixed point of Bellman's equation [2], we know that the expected return for the optimal policy can be written with the following recursive form:

$$Q(b,a) = R(b,a) + \sum_{o \in O} P(o|b,a) max_{a'} Q(b', a') \tag{3}$$

where b' is the updated dialogue state following the execution of action a and the observation of o, as in Eq. 2. Furthermore, for notational convenience, we used $R(b,a) = \sum_{s \in S} R(s,a)b(s)$ and $P(o|b,a) = \sum_{s \in S} P(o|s,a)b(s)$.

Extracting an optimal policy for such POMDP is known to be a hard problem, with intractable exact solutions. Fortunately, many good approximations exist, often based on sampling a limited number of trajectories [13, 19]. The online planning algorithm we present in the next section makes use of such sampling techniques.

3 Approach

The general architecture of our approach revolves around a shared dialogue state, which is read and written asynchronously by a collection of modules (for dialogue understanding, interpretation, decision-making, generation, etc.). Each module can update the current state with new information. We have already described in our previous work the general dialogue system workflow [10], and will not repeat it here. Instead, we will concentrate on the dialogue manager module, and in particular on its internal, domain-specific models.

As we will shortly describe, our planning algorithm relies on rich domain knowledge to speed up the action selection process. Our starting point is the observation that the probability and reward models used in dialogue management usually contain quite a lot of *internal structure* that can be readily exploited to yield more efficient algorithms. For instance, we can see in Figure 1 that the probability $P(a_u = \mathsf{NULL}|i_u)$ does not actually depend on the specific value of i_u. Similarly, the reward $Q(a_m = \mathsf{AskRepeat}, i_u)$ does not depend on i_u either, since it is equal to -0.5 for all possible values of i_u. Generally speaking, we can often group the enumeration of possible values for the dependent variables into a set of distinct, mutually exclusive *partitions* that yield similar outcomes.

3.1 Probabilistic Rules

Probabilistic rules are a generic description formalism to capture such structure. They take the form of *if...then...else* cases mapping a list of *conditions* on input variables to specific *effects* on output variables. At runtime, these rules are then directly applied on the dialogue state, thereby extending the Bayesian Network with new nodes and conditional dependencies. This Bayesian Network can then be directly used for inference, e.g. to compute the marginal distribution of a particular variable or the utility of a given action. The probabilistic rules thus function as high-level *templates* for the incremental construction of a classical probabilistic model.

Probability models

For probabilistic models of the form $P(X_1,...X_n|Y_1,...Y_m)$, a rule is formally expressed as an ordered list $\langle c_1,...c_n\rangle$, where each case c_i is associated with a condition ϕ_i and a distribution over stochastic effects $\{(\psi_i^1,p_i^1),...,(\psi_i^k,p_i^k)\}$, where ψ_i^j is a stochastic effect and probability $p_i^j = P(\psi_i^j|\phi_i)$, where $p_i^{1...m}$ satisfy the usual probability axioms. The rule reads as such:

if (ϕ_1) **then**
  $\{[P(\psi_1^1) = p_1^1],\ ...\ [P(\psi_1^k) = p_1^k]\}$
else if (ϕ_2) **then**
  $\{[P(\psi_2^1) = p_2^1],\ ...\ [P(\psi_2^l) = p_2^l]\}$
...
else if (ϕ_n) **then**
  $\{[P(\psi_n^1) = p_n^1],\ ...\ [P(\psi_n^m) = p_n^m]\}$

A final **else** case is implicitly added to the bottom of the list, and holds if no other condition applies. If not overridden, the default effect associated to this last case is void – i.e. it causes no changes to the distribution over the output variables.

The rule conditions ϕ_i are expressed as logical formulae grounded in the dependent variables. They can be arbitrarily complex formulae connected by conjunctive, disjunctive and negation operators. Formally speaking, a condition is therefore a function mapping state variable assignments to a boolean value. The conditions on the input variables can be seen as providing a compact partition of the state space to mitigate the dimensionality curse. Without this partitioning in alternative conditions, a rule ranging over m variables each of size n would need to enumerate n^m possible assignments. The partitioning with conditions reduces this number to p mutually exclusive partitions, where p is usually small.

The rule effects ψ_i^j are similarly defined: given a condition holding on a set of input variables, the associated effects define specific *value assignments* for the output variables. The effects can be limited to a single variable or range over several output

variables. Each effect is assigned a probability, and several alternative stochastic effects can be defined for the same case. The effect probabilities are parameters which can be hand-coded or estimated from data .

As an illustrative example, consider the probability model $P(a_u|i_u)$ from Figure 1. This model can be encoded with the rule r_1:

$$\textbf{Rule } r_1 : \textbf{ if } (i_u = \mathsf{GoTo(X)}) \textbf{ then}$$
$$\{[P(a_u = \mathsf{GoTo(X)}) = 0.9],\ ...\ [P(a_u = \mathsf{NULL}) = 0.1]\}$$

The rule simply expresses that the value of a_u should be identical to the one in i_u with probability 0.9, and equal to NULL with probability 0.1. The exact value for the argument X will be filled at runtime given the instantiation in i_u.

Reward models

Rules can also be applied to describe reward models, with minor notational changes. Assume a reward model $R(X_1,...X_n,A_1,...A_m)$, where $X_1...X_n$ are random variables and $A_1,...A_m$ decision variables. The rule is defined as an ordered list $\langle c_1,...c_n\rangle$, where each case c_i has a condition ϕ_i ranging over the random variables $X_1,...X_n$ and a set of reward values $\{(\psi_i^1,r_i^1),...,(\psi_i^k,r_i^k)\}$, where ψ_i^j is an assignment of values for the decision nodes and $r_i^k = R(\psi_i^k,\phi_i)$. The rule reads similarly:

$$\textbf{if } (\phi_1) \textbf{ then}$$
$$\{R(\psi_1^1) = r_1^1,\ ...\ R(\psi_1^k) = r_1^k\}$$
$$...$$
$$\textbf{else if } (\phi_n) \textbf{ then}$$
$$\{R(\psi_n^1) = r_n^1,\ ...\ R(\psi_n^m) = r_n^m\}$$

By convention, the reward of an action value which is not explicitly expressed in the effect is assumed to be 0. The conditions ϕ_i are defined in the exact same way as for probability models – that is, as functions mapping assignments of values for $X_1...X_n$ to a boolean value. The effects express assignments for the decision variables $A_1,...A_m$.

To illustrate the use of such rules to capture the model structure, consider the reward model $R(i_u,a_m)$ from Figure 1, which can be encoded with the rule r_2:

$$\textbf{Rule } r_2 : \textbf{ if } (i_u = \mathsf{GoTo(X)}) \textbf{ then}$$
$$\{[R(a_m = \mathsf{GoTo(X)}) = 3],[R(a_m = \mathsf{AskRepeat}) = -0.5],$$
$$[R(a_m = \mathsf{GoTo(Y)} \wedge Y \neq X) = -5]\}$$

The general rule structure is provided by the system designer, while their parameters (probabilities or utilities) can be estimated from data, as shown in [11].

Rule instantiation

At runtime, the rules are *instantiated* on the current dialogue state by creating new nodes and dependencies, thereby converting ("grounding") the rules into a standard probabilistic model which can then be straightforwardly used for inference based on standard algorithms such as variable elimination or importance sampling. The outlined procedure is an instance of *ground inference* [6], since the rule structure is grounded in a standard Bayesian Network.

Practically, this instantiation is realised by creating one node for each rule:

- For probability models, the rule node is a chance node that is conditionally dependent on the input variables, and expresses the effect which is likely to hold given their values. This rule node also has outward dependencies to the set of output variables it determines – for instance, a_u for rule r_1.
- For utility models, the rule node is a utility node dependent on the input variables, and expresses the reward associated with specific action values given the inputs.

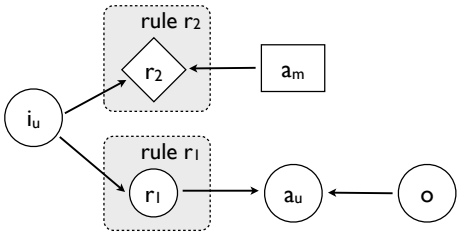

Fig. 2 Example of Bayesian Network generated by the instantiation of rules r_1 and r_2, producing a distribution similar to Figure 1. The nodes r_1 and r_2 are rule nodes expressing which of the rule effects hold given the value of the i_u variable. The node r_1 represents the probabilistic effect on a_u given i_u, while r_2 encodes the utility of a_m given i_u. Finally, the distribution $P(a_u|r_1)$ is a simple deterministic distribution following the value assignment ascribed in the effect.

3.2 Planning algorithm

The pseudo-code of the planning algorithm we are currently developing is given in Algorithm 1. It works by sampling a set of *trajectories* starting from the current dialogue state until a given horizon limit is reached. One benefit of this sampling-based strategy is the ability to work in anytime mode, which means that at any point in time, the procedure is able to deliver a solution. The quality of the solution will of course depend on the number of trajectories that are sampled – more trajectories leading to a more accurate plan, but at a higher computational cost. The anytime

nature of the algorithm is important since the planner operates online and must thus satisfy real-time constraints.

Algorithm 1 : PLAN (**b**)

Require: **b**: Current dialogue state (expressed as a Bayesian Network)
Require: $nbTrajectories$: Number of trajectories to sample
Require: γ: Discount factor
Require: $horizon$: Planning horizon

$sequences \leftarrow \emptyset$
for $i = 0 \rightarrow nbTrajectories$ **do**
  $\mathbf{b}' \leftarrow$ copy of **b**
  $a_m \leftarrow$ sample system action
  $trajectory \leftarrow [a_m]$
  $Q \leftarrow \sum_{s \in S} R(s, a_m)\mathbf{b}'(s)$
  for $t = 0 \rightarrow horizon$ **do**
    $a_u \leftarrow$ sample next user action given $\mathbf{b}'$
    $\mathbf{b}' \leftarrow$ BELIEFUPDATE($\mathbf{b}' \cup a_u$)
    $a_m \leftarrow$ sample system action
    $trajectory \leftarrow trajectory \cup [a_m]$
    $Q \leftarrow Q + \gamma^t \sum_{s \in S} R(s, a_m)\mathbf{b}'(s)$
  end for
  if $trajectory \in sequences$ **then**
    $sequences[trajectory] \leftarrow sequences[trajectory] \cup [Q]$
  else
    $sequences[trajectory] \leftarrow [Q]$
  end if
end for
$sequence^* \leftarrow \arg\max_i \frac{\sum_{Q \in sequences[i]} Q}{|sequences[i]|}$
return $sequence^*$

The trajectories are sampled by repeatedly selecting system actions and subsequent user actions until the horizon limit is reached. The system actions can be sampled uniformly or following heuristic distributions if some are available to guide the search towards high-utility regions [7]. For the user actions, the sampling relies on a (also rule-structured) user action model $P(a_u|s, a_m)$ that predicts the next user action given the current state and last system action, modulo some standard noise inserted into the distribution to simulate speech recognition errors.

In order to navigate through the trajectories, the algorithm needs to update its dialogue state after the selection of a user action. The belief update algorithm is described in detail in [10]. For each trajectory, the algorithm records the rewards accumulated after each action. Once enough trajectories have been sampled, the algorithm computes the average expected return for each possible sequence, and selects the sequence with the highest score. This sequence will correspond to the optimal plan, and the only remaining step for the dialogue system is to execute the first action of this plan.

The originality of our approach lies in the use of probabilistic rules for updating the dialogue state and determining the possible actions available at that state. More specifically, the reward rules will determine a set of actions which can be performed if their conditions hold, and which reward is expected from their execution. The major benefit is that instead of having to search through the whole space of possible actions, the planning algorithm can be limited to consider only a subset of *relevant* actions. The reward rules can therefore be seen as providing a high-level filter on the action space [9], and we expect them to significantly speed up the search for the optimal action.

We are currently in the process of implementing the details of this online planner and integrating it in the dialogue system architecture described in [10]. Unsurprisingly, the main bottleneck we currently encounter remains the runtime performance, which doesn't yet scale to real-time requirements for more than trivial domains. We hope to solve these tractability issues soon, and be able to report empirical results on the performance of this online planner for a human-robot interaction domain similar to the one described in [11].

4 Related work

Online planning has a long history in dialogue systems [1, 20], but it has usually been confined to classical planning, relying on a clear-cut set of goal states instead of relative utilities and with no account of observation uncertainty. Decision-theoretic approaches on the other hand have mostly focused on offline optimisations of MDPs [12, 16] or POMDPs [23] via reinforcement learning.

The field of POMDP planning has recently witnessed a surge of interest for online methods [18, 19]. As mentioned in the introduction, online planning offers clear advantage in terms of scalability to large domains, and adaptivity to dynamic changes in the environment or task models. Moreover, it can be combined with offline methods, the precomputed policy being in this case employed as a heuristic approximation to guide the online search for the optimal action. Another related development is the use of online planning for model-based Bayesian reinforcement learning [17]. These approaches rely on the inclusion of model uncertainty as part of the state space. Due to the increased size and continuous nature of the resulting state space, direct policy optimisation is not feasible, and online planning based on sampling methods is the only viable alternative.

Online reinforcement learning is another alternative for adapting the system behaviour to its context. Most work so far has concentrated on model-free learning algorithms such as Gaussian Process SARSA or Kalman Temporal Differences [5, 3]. Model-free reinforcement learning seeks to directly derive an optimal policy through interaction with the environment. The approach we take in this work is closer to (Bayesian) model-based approaches to reinforcement learning [14, 17], where the learner first seeks to estimate a model of the environment and then uses this model to plan an optimal behaviour. The model estimation is often done by

incorporating model uncertainty into the state space. Model-based approaches are able to directly incorporate prior knowledge and constraints into their models, which make them attractive for dialogue management tasks.

The use of high-level representations such as probabilistic rules in planning has been explored in previous work [24, 7]. The common intuition behind most of these approaches is that capturing the inner structure of the domain via high-level representations can yield models which are:

- easier to learn and generalise better to unseen data, since they depend on a greatly reduced number of parameters (as shown in [11])
- more efficient to use, since the model structure can be exploited by the inference algorithms (as we have tried to show in this paper).

5 Conclusions

We have presented in this paper a general approach to online planning for dialogue management, based on the use of high-level probabilistic rules. These rules enable the system designer to provide important domain knowledge which can help filtering the space of possible actions to consider at a given time point.

Our underlying hypothesis is that online or hybrid planning can be beneficial for dialogue management, especially for open-ended domains such as human-robot interaction or tutoring systems. The environment's dynamics of these domains is rarely static, and is likely to change over time due to shifting user preferences or contextual factors. Online planning strategies are able to naturally cope with such changes without having to recompile policies.

The major bottleneck for online planning remains however the runtime performance. We still need to find ways to make online planning tractable for real domains. In this respect, we would like to investigate the use of precomputed policies as heuristic approximation to guide the lookahead search.

Another interesting venue for future work is the combination of online planning with reinforcement learning. As described in [17], online planning can be ideally combined with Bayesian approaches to reinforcement learning, where uncertainty in the model parameters is directly captured in terms of additional variables in the state space. Given the high-level representations provided by the probabilistic rules, the rule parameters could potentially be estimated from limited amounts of (raw) interaction data, and be continuously refined as more interaction experience becomes available, either from real users or from simulation.

References

1. Allen, J., Byron, D., Dzikovska, M., Ferguson, G., Galescu, L., Stent, A.: An architecture for a generic dialogue shell. Natural Language Engineering **6**, 213–228 (2000)

2. Bellman, R.: Dynamic programming. Princeton University Press, Princeton, NY (1957)
3. Daubigney, L., Geist, M., Pietquin, O.: Off-policy learning in large-scale pomdp-based dialogue systems. In: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4989 –4992 (2012)
4. Frampton, M., Lemon, O.: Recent research advances in reinforcement learning in spoken dialogue systems. Knowledge Engineering Review **24**(4), 375–408 (2009)
5. Gasic, M., Jurcicek, F., Thomson, B., Yu, K., Young, S.: On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In: 2011 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), pp. 312–317 (2011)
6. Getoor, L., Taskar, B.: Introduction to Statistical Relational Learning. The MIT Press (2007)
7. Lang, T., Toussaint, M.: Planning with noisy probabilistic relational rules. Journal of Artificial Intelligence Research **39**, 1–49 (2010)
8. Lemon, O., Pietquin, O.: Machine Learning for Spoken Dialogue Systems. In: Proceedings of the 10th European Conference on Speech Communication and Technologies (Interspeech'07), pp. 2685–2688 (2007)
9. Lison, P.: Towards relational POMDPs for adaptive dialogue management. In: Proceeding of the Student Research Workshop of the 48th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics (2010)
10. Lison, P.: Declarative design of spoken dialogue systems with probabilistic rules. In: Proceedings of the 16th Workshop on the Semantics and Pragmatics of Dialogue (2012)
11. Lison, P.: Probabilistic dialogue models with prior domain knowledge. In: Proceedings of the SIGDIAL 2012 Conference, pp. 179–188. Seoul, South Korea (2012)
12. Pietquin, O.: Optimising spoken dialogue strategies within the reinforcement learning paradigm. In: M.E. Cornelius Weber, N.M. Mayer (eds.) Reinforcement Learning, Theory and Applications, pp. 239–256. I-Tech Education and Publishing, Vienna, Austria (2008)
13. Pineau, J., Gordon, G., Thrun, S.: Point-based value iteration: An anytime algorithm for POMDPs. In: International Joint Conference on Artificial Intelligence (IJCAI), pp. 1025 – 1032 (2003)
14. Poupart, P., Vlassis, N.A.: Model-based bayesian reinforcement learning in partially observable domains. In: International Symposium on Artificial Intelligence and Mathematics (ISAIM) (2008)
15. Purver, M.: The theory and use of clarification requests in dialogue. Ph.D. thesis (2004)
16. Rieser, V., Lemon, O.: Learning human multimodal dialogue strategies. Natural Language Engineering **16**, 3–23 (2010)
17. Ross, S., Pineau, J., Chaib-draa, B., Kreitmann, P.: A Bayesian Approach for Learning and Planning in Partially Observable Markov Decision Processes. Journal of Machine Learning Research **12**, 1729–1770 (2011)
18. Ross, S., Pineau, J., Paquet, S., Chaib-Draa, B.: Online planning algorithms for POMDPs. Journal of Artificial Intelligence Research **32**, 663–704 (2008)
19. Silver, D., Veness, J.: Monte-carlo planning in large POMDPs. In: J. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R. Zemel, A. Culotta (eds.) Advances in Neural Information Processing Systems 23, pp. 2164–2172 (2010)
20. Steedman, M., Petrick, R.P.A.: Planning dialog actions. In: Proceedings of the 8th SIGDIAL Meeting on Discourse and Dialogue, pp. 265–272. Antwerp, Belgium (2007)
21. Thomson, V., Young, S.: Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. Computer Speech & Language **24**, 562–588 (2010)
22. Williams, J.: A case study of applying decision theory in the real world: POMDPs and spoken dialog systems. In: L. Sucar, E. Morales, J. Hoey (eds.) Decision Theory Models for Applications in Artificial Intelligence: Concepts and Solutions, pp. 315–342. IGI Global (2012)
23. Young, S., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., Yu, K.: The hidden information state model: A practical framework for POMDP-based spoken dialogue management. Computer Speech & Language **24**, 150–174 (2010)
24. Zettlemoyer, L.S., Pasula, H.M., Kaelblin, L.P.: Learning planning rules in noisy stochastic worlds. In: IN AAAI, pp. 911–918. AAAI Press (2005)