



Towards Online Planning for Dialogue Management with Rich Domain Knowledge

Pierre Lison

Language Technology Group,
University of Oslo

IWSDS, November 29, 2012



Introduction

- Dialogue management (DM) as a *decision-theoretic planning problem*
- Most approaches perform this planning offline (precomputed policies)

	+	-
Offline planning	Action selection is straightforward and quite efficient (direct lookup)	<ul style="list-style-type: none">• Must consider all possible situations• Policy must be recomputed after every model change



Introduction

Alternative: perform planning *online*, at execution time

	+	-
Offline planning	Action selection is straightforward (and efficient)	<ul style="list-style-type: none">• Must consider all possible situations• Policy must be recomputed after every model change
Online planning	<ul style="list-style-type: none">• Must only plan for current situation• Easier to adapt to runtime changes	<ul style="list-style-type: none">• Must meet real-time requirements

3



Approach

- To address these real-time constraints, we must ensure the planner concentrates on *relevant actions* and states
- **Intuition:** we can exploit *prior domain knowledge* to filter the space of possible actions and transitions
- We report here on our ongoing work with the use of *probabilistic rules* to encode the probability/reward models of our domain

4



Approach

- We represent the dialogue state as a **Bayesian Network**
- Probabilistic rules are applied upon this dialogue state to update / extend it
- Advantages:
 - (exponentially) fewer parameters to estimate
 - Can incorporate prior domain knowledge

[Pierre Lison, «Probabilistic Dialogue Models with Prior Domain Knowledge», SIGDIAL 2012]

5



Probabilistic rules

- The rules take the form of structured if...then...else cases, mappings from *conditions* to (probabilistic) *effects*:

```
if (condition1 holds) then  
  P(effect1) =  $\theta_1$ ,   P(effect2) =  $\theta_2$   
else if (condition2 holds) then  
  P(effect3) =  $\theta_3$ 
```

- For action-selection rules, the effect associates *rewards* to particular *actions*:

```
if (condition1 holds) then  
  R(actions) =  $\theta_1$ 
```

6

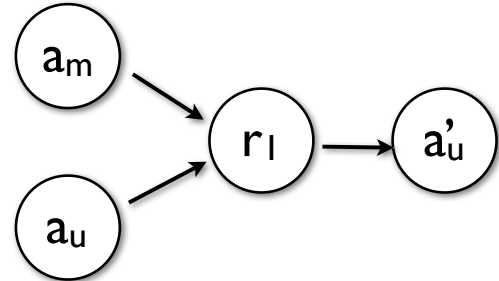
Rule examples

- Example r_1 (probability rule):

if ($a_m = \text{AskRepeat}$) **then**

$$P(a_u' = a_u) = 0.9$$

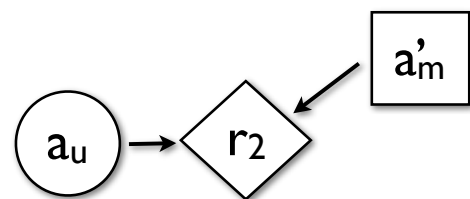
$$P(a_u' \neq a_u) = 0.1$$



- Example r_2 (reward rule):

if ($a_u \neq \text{None}$) **then**

$$R(a_m' = \text{AskRepeat}) = -0.5$$



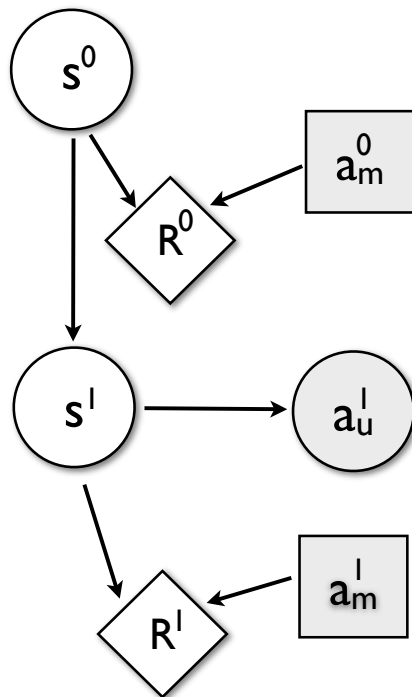
7

Approach

- The transition, observation and reward models for the domain are all encoded in terms of these probability & reward rules
- We devised a simple *forward planning* algorithm that:
 - samples a collection of **trajectories** (sequence of actions) starting from the current state, until a specific horizon is reached
 - and records the **return** obtained for each trajectory
- The algorithm then searches for the action sequence with highest average return, and executes the first action in this sequence

8

Planning algorithm



Repeat until enough trajectories are collected

Loop (for each trajectory):
 $Q = 0$
Loop (until horizon reached):
 Sample system action a_m
 $Q = Q + \gamma^t R$
 Predict next state s
 Sample next user action a_u
 Do belief update
 Record Q for trajectory
 \forall trajectory, calculate average Q
Return trajectory with max. Q

⋮ until horizon is reached

9

Discussion

- What is original in our approach?
 - The rules provide high-level constraints on the relevant actions (and subsequent future states)
 - For instance, if the user intention is $\text{Want}(\text{Mug})$, the rules will provide utilities for the actions $\text{PickUp}(\text{Mug})$ or AskRepeat , but will consider $\text{PickUp}(\text{Box})$ as irrelevant
 - In other words, they enable the algorithm to quickly focus the search towards high-utility regions
 - ... and discard irrelevant actions and predictions



Discussion

- We did some preliminary experiments with an implementation of this algorithm
 - The rules did make a difference in guiding the search for «relevant» trajectories
 - But unfortunately, the algorithm doesn't yet scale to real-time performance (sorry)
 - Need to find better *heuristics* to aggressively *prune* the applied rules, improve action sampling and the performance of the inference algorithm

11



Conclusions

- Online planning can help us build more **adaptive** dialogue systems
 - Can be combined with offline planning, using a precomputed policy to guide the search of an online planner!
 - Using *prior domain knowledge* (encoded with e.g. *probabilistic rules*) should help the planner focus on relevant actions and filter out irrelevant ones
- More work needed to find tractable planning techniques, and collect empirical results

12



Online reinforcement learning

- Links with the reinforcement learning literature:
 - Our approach can be seen as a *model-based* approach to reinforcement learning, where a model is learned and then used to plan the best actions
 - Most other approaches rely on a *model-free* reinforcement learning paradigm, and try to learn the optimal action directly from experience, without trying to estimate explicit models (and plan over them)