Learning Multi-Goal Dialogue Strategies Using Reinforcement Learning with Reduced State-Action Spaces

Learning Multi-Goal Dialogue Strategies Using Reinforcement Learning With Reduced State-Action Spaces

Heriberto Cuayáhuitl[1]*, Steve Renals*[1]*, Oliver Lemon*[2] *and Hiroshi Shimodaira*[1]

CSTR[1], HCRC[2], School of Informatics, University of Edinburgh
2 Buccleuch Place, EH8 9LW, Edinburgh, Scotland, UK
{h.cuayahuitl,s.renals,olemon,h.shimodaira}@ed.ac.uk

Abstract

Learning dialogue strategies using the reinforcement learning framework is problematic due to its expensive computational cost. In this paper we propose an algorithm that reduces a state-action space to one which includes only valid state-actions. We performed experiments on full and reduced spaces using three systems (with 5, 9 and 20 slots) in the travel domain using a simulated environment. The task was to learn multi-goal dialogue strategies optimizing single and multiple confirmations. Average results using strategies learnt on reduced spaces reveal the following benefits against full spaces: 1) less computer memory (94% reduction), 2) faster learning (93% faster convergence) and better performance (8.4% less time steps and 7.7% higher reward).

Index Terms: reinforcement learning, spoken dialogue systems.

1. Introduction

The main task of a Spoken Dialogue Manager (SDM) in goal-oriented dialogue systems is to control the dialogue flow between user and system with the following aims: successful, efficient and natural conversations. More specifically, an SDM has the following subtasks: a) to gather information from the user, b) to clarify information explicitly or implicitly, c) to resolve ambiguities that arise due to speech recognition (ASR) errors or incomplete specifications, d) to suggest subsequent dialogue goals, e) to offer assistance upon request or when necessary, f) to provide alternatives when the information is not available, g) to provide additional constraints, h) to interact with other components in order to retrieve or provide information, and i) to control the degree of initiative [1].

The design of SDMs is typically hand-crafted by system developers, based on their intuition about the proper dialogue flow. As a consequence, dialogue strategies designed by humans are prone to errors, labour-intensive and non-portable. This makes semi-automatic design an attractive alternative. Levin and Pieraccini [2], pioneered the idea of dialogue design as an optimization problem using Markov Decision Processes (MDPs) and reinforcement learning [3]. However, the search space grows exponentially according to the state variables taken into account, making the task of dialogue optimization difficult, even for simple dialogue systems.

Previous research efforts have investigated how to learn uni-goal dialogue strategies for optimizing confirmation [4-9], initiative [4] and database queries [6]. They have mostly adopted the formalism of MDPs [2,4-7], Partially Observable MDPs [8], and function approximation [9,10], using either real [4] or simulated environments [5-9]. However, little attention has been devoted to the problem of learning on reduced state-action spaces, with the aim of faster learning and reduced computational demands.

In this paper we investigate how to reduce state-action spaces for learning multi-goal dialogue strategies. For such purpose we propose the $sapReduction$ algorithm, which aims to formalize the idea of avoiding unnecessary learning by using prior knowledge that reduces the search space to only valid state-actions.

2. Spoken Dialogue Management Using Markov Decision Processes

Informally, the idea of spoken dialogue management as an optimization problem consists of taking the best action for every situation in a conversation by following an optimal dialogue strategy. The reinforcement learning paradigm is particularly appealing for this scenario, where an agent takes optimal actions for every situation in the environment described by a Markov Decision Process (MDP), or any other formalism (POMDP, SMDP or POSMDP). Formally, an MDP is defined as a 4-tuple $< S, A, T, R >$ characterized as follows: S is a set of states of the environment; A is the set of actions; T is a transition probability function that observes the next state s' given the current state s and action a according to the probability distribution $P(s'|s, a)$; and R is the reward function that specifies the rewards given to the agent for choosing action a when the environment makes a transition from s to s'.

Under this formalism, a sequence of states s, actions a and rewards r within a dialogue (or episode) receives a total expected reward expressed as $\mathcal{R} = \sum_{t=0}^{T} \gamma r_{t+k+1}$, where the discount rate $0 \leq \gamma < 1$ makes future rewards less valuable than immediate rewards. Thus, the solution for an MDP is to learn a dialogue strategy (or policy) that maximizes $\mathcal{R}$, which optimizes the interaction with its environment by choosing optimal actions. The expected value of the reward can be computed recursively by value functions $V^{\pi}(s)$ or action-value functions $Q^{\pi}(s)$ as described in [2], where the optimal policy is expressed as $\pi^{*}(s) = \arg\max_a Q^{*}(s, a)$, and can be learnt by either dynamic programming methods or reinforcement learning methods.

A main limitation in dialogue optimization is the expensive computational cost due to the fact that state-action spaces grow exponentially. As an example, consider an MDP where the states S are formed by combinations of slots Q and state variables V as shown in figure 1.a, and the system actions A are formed by combinations of slots Q and single actions A^s (see fig. 1.b). Thus, the size of the state space would be $|S| = |V|^{|Q|}$ and the size of the state-action space would be $|S \times A| = |V|^{|Q|} * (|Q| * |A^s|)$. For a small-scale dialogue system with 7 slots, 5 state variables and 6 single actions, the search space is 3.3 million state-actions, and the growth is exponential assuming no constraints at all.

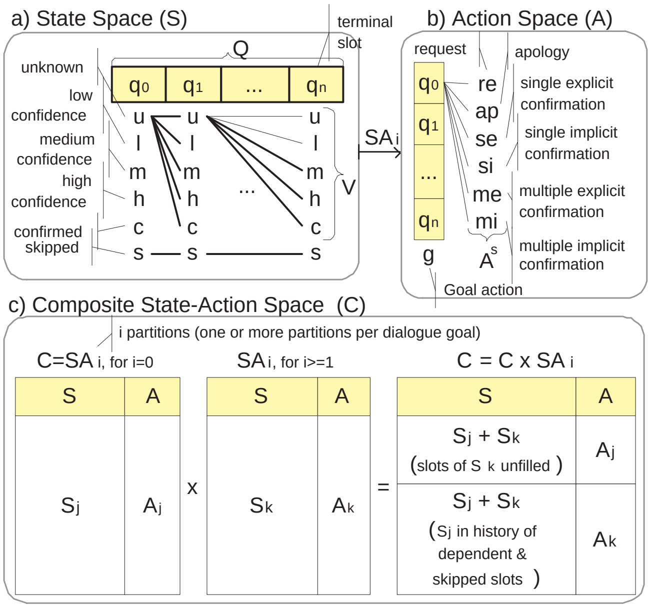
a) State Space (S)
terminal slot
Q
unknown
low confidence
medium confidence
high confidence
confirmed
skipped
q0
q1
...
qn
u
l
m
h
c
s
...
V
SAi
b) Action Space (A)
request
apology
q0
q1
...
qn
re
ap
se
si
me
mi
single explicit confirmation
single implicit confirmation
multiple explicit confirmation
multiple implicit confirmation
g
A^s
Goal action
c) Composite State-Action Space (C)
i partitions (one or more partitions per dialogue goal)
C=SA i, for i=0
SA i, for i>=1
C = C x SA i
S
A
Sj
Aj
x
Sk
Ak
=
Sj + Sk (slots of S k unfilled)
Sj + Sk (Sj in history of dependent & skipped slots)

Figure 1: *Scheme used to generate search spaces for multi-goal dialogue systems, where sapReduction constraints each space.*

3. The sapReduction Algorithm

Figure 1 illustrates a proposed scheme in order to generate state-action spaces (or search spaces) for learning multi-goal dialogue strategies, where the $sapReduction$ algorithm applies reduction to each space. Under this scheme, this algorithm generates reduced search spaces with constraints at three levels of granularity: states, actions and partitions. Whilst the first and second levels are used for uni-goal spoken dialogue systems, the third level is used for multi-goal dialogue systems. This algorithm makes the following assumptions: 1) states S are combinations of slots Q and state variables V (see fig. 1.a), 2) actions A are combinations of slots Q and single actions A^s (see fig. 1.b), and 3) the dialogue design is specified in partitions roughly equivalent to dialogue goals, where the merge of partitions form a composite search space (see fig. 1.c). In this way, states S are reduced using the $sReduction$ step, actions A for each state are reduced using the $aReduction$ step, and the search spaces for each partition SA_i are merged using the $pReduction$ step. This is more formally described in the algorithm shown in figure 2 (see next subsections for details).

3.1. The sReduction Step

The full state space S consists of all combinations of slots Q and state variables V (see fig. 1.a), including valid and invalid combinations. Thus, the task of this step is to avoid invalid combinations. The space reduction is driven by the assumption that a terminal slot requires the non-terminal slots to be confirmed, where typically a transaction is performed. In addition, this step adds a state with skipped slots in case of an unwanted dialogue goal (see figure 2, lines 10-16). Notice that all slots require confirmation, which is not the case for “yes/no” slots, in this case we assume that such slots can be considered confirmed if they were collected with the highest ASR confidence level. Also, notice that fig. 1.a shows one state-variable per slot but it may have a vector of state variables.

```
01.Algorithm sapReduction($P, O, D, Q, V, A^s$) return $SA$
02.input: partitions ($P$), optional partitions ($O$), dependent partitions ($D$),
03.       slots per partition ($Q$), state variables ($V$), single actions ($A^s$)
04.    initialize variables $S$,$SA$

05.    for each partition $p_i \in P$ do
06.        $S_i \leftarrow$ sReduction($p_i, O, Q_i, V$)
07.        $SA_i \leftarrow$ aReduction($S_i, Q_i, V, A^s$)
08.    end for
09.    $SA \leftarrow$ pReduction($SA, P, D, Q$)

10.function sReduction($p_i, O, Q, V$) return $S$
11.    initialize variable $S$
12.    for each slot $q_j \in Q$ do
13.        $S \leftarrow$ combinations of substates $S$, slot $q_j$ and variables $V$;
14.            where $q_n$ require the non-terminal slots to be confirmed
15.    end for
16.    $S \leftarrow S \cup$ state with skipped slots ($q_j.s$) if partition $p_i \in O$

17.function aReduction($S, Q, V, A^s$) return $SA$
18.    initialize variables $A, SA$
19.    for each state $s_i \in S$ do
20.        for each slot $q_j \in s_i$ do
21.            for each $a_k \in A^s$ do
22.                $A \leftarrow A \cup (q_j, a_k)$, under the conditions of figure 3
23.                $A \leftarrow A \cup (q_j, \{g\})$, if $q_j = q_n$ and slots confirmed
24.            end for
25.        end for
26.        $SA \leftarrow SA \cup (s_i \mapsto A)$, reinitialize $A$
27.    end for

28.function pReduction($SA, P, D, Q$) return $C$
29.    initialize variables $C \leftarrow SA_0, M$
30.    for each partition $p_i \in P \, \forall i > 0$ do
31.        $history \leftarrow$ sequences of confirmed & skipped slots up to $p_{i-1}$
32.        for each state $s_j \in C$ do
33.            for each state $s_k \in SA_i$ do
34.                $M \leftarrow M \cup (s_j + s_k \mapsto A_{s_j})$, if slots of $s_k$ unfilled
35.                $M \leftarrow M \cup (s_j + s_k \mapsto A_{s_k})$, if $s_j \in history$
36.            end for
37.        end for
38.        $C \leftarrow M$, reinitialize $M$
39.    end for
```

Figure 2: *The sapReduction Algorithm.*

3.2. The aReduction Step

The full action space A consists of all combinations of slots Q and single actions A^s (see fig. 1.b), which includes valid and invalid combinations. Thus, the task of this step is to avoid invalid actions per state. Figure 3 illustrates the conditions used to generate valid combinations, where the presence of circles represent valid combinations and the absence invalid combinations. In addition, we clustered sets of circles that must satisfy a condition. For instance, to validate action $(q4, re)$ in figure 3(a) the condition $ntsc$ (non-terminal slots confirmed) must be satisfied, see figure 2 (lines 17-27). Also, further ad-hoc reductions are possible.

3.3. The pReduction Step

There is indeed exponential grow in search spaces if we assume no boundary across dialogue goals, but this yields incoherent dialogues by allowing sequences of slots from different dialogue goals. In this step we propose to generate multiple search spaces according to the partitions specified in the dialogue design, roughly equivalent to dialogue goals. This step assumes the fol-

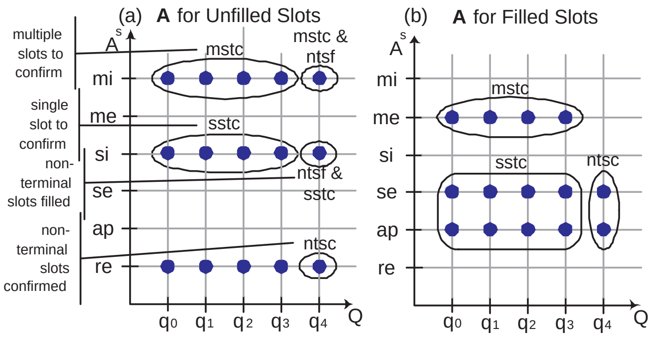

Figure 3: *Conditioned combinations of slots Q and single actions A^s representing valid system actions A, which occur in the presence of a circle that satisfy a condition given by its cluster.*

lowing assumptions: 1) partitions may be optional by only allowing skipped slots (with state variable s), 2) partitions may be dependent or independent (e.g., a partition with dependency can occur if and only if the dependent partition has its slots confirmed), and 3) a dialogue goal might be specified with more than one partition. Thus, the task of this step is to merge multiple search spaces into a composite reduced one. Figure 1.c illustrates this step by using an incremental merge C of search spaces per partition SA_i, which takes into account the history of dependent and skipped slots; see figure 2 for a more formal description (lines 28-39).

4. Experiments and Results

4.1. Experimental Setup

The aim of our experiments was to investigate the performance of $full$ and $reduced$ search spaces (using the proposed algorithm), specifically in the optimization of single and multiple confirmations for goal-oriented mixed-initiative dialogue systems. We performed three experiments in the travel domain (denoted as "Exp1", "Exp2" and "Exp3"), with a similar structure to the DARPA Communicator systems [11]; using 1, 2 and 5 dialogue goals consisting of 5, 9 and 20 slots respectively. Because of a large number of dialogues is required to learn the optimal dialogue strategies, we utilized a simulated environment in order to control the amount of randomness in ASR confidence levels and user responses.

Figure 4 shows the partitions P, dependent partitions D, optional partitions O, and slots Q used in our experiments. Exp1 used the first dialogue goal, Exp2 used the first two goals, and Exp3 used the five goals. The MDP was configured as follows: The state-action spaces were generated as illustrated in figure 1, whilst reduced spaces used the $sapReduction$ algorithm, full spaces used only step 3; T used deterministic transitions by observing the next state based on the current state-action, user response and confidence level (see section 4.2); and the reward function R consisted of $+100$ if all slots were confirmed or skipped, -20 if there was nothing to confirm/apologize, and -1 otherwise.

Finally, we used the following learning setup: algorithm = Q-Learning; step size $\alpha = 100/(100+t)$ with t elapsed time-steps; discount factor $\gamma = 0.9$; selection strategy = ϵ-greedy (with 20% exploration); initial Q-values = 0; and convergence = explored space $\geq 99.9\%$ and difference in average MaxQ values (Q-values with maximum value) of last against previous 10^4 episodes ≤ 0.

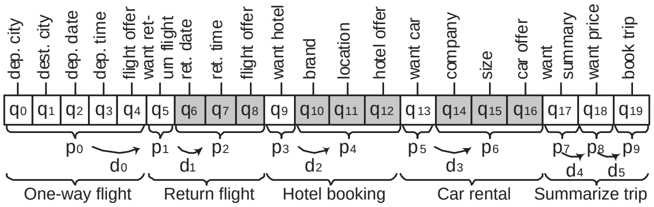

Figure 4: *Structure of dialogue goals, partitions P, dependent partitions D (e.g., p_1 require the slots of p_0 confirmed), and slots Q. The optional partitions O are represented with shaded slots.*

Table 1: *Sizes of full and reduced composite state-action spaces.*

Experiment	Full (S)	Reduced (S)	Full (SA)	Reduced (SA)	% of SA Reduction
Exp1	3126	630	93722	5636	93.98
Exp2	3255	663	95979	5760	93.99
Exp3	4125	957	110097	6672	93.93

4.2. The Random Simulated Environment

We used a slot-based confidence level model with the following distribution: $0 \leq l < 0.6$, $0.6 \leq m < 0.8$, and $0.8 \leq h \leq 1$; where averages were computed for multiple filled slots. The random simulated user model consisted of the following features: a) percentage of coherent In-Vocabulary (IV) responses, the complement consisted of random IV responses plus out-of-vocabulary responses; b) two sets of responses for each kind of slot (terminal and non-terminal), the second set aimed to refill the last two slots before the user accepted a terminal slot; c) coherent responses of single and multiple slots used the following distribution: $0 \leq single < 0.7$ and $0.7 \leq multiple \leq 1$; d) correct confirmations used the following distribution: $0 \leq l \leq 0.5$, $0.5 \leq m \leq 0.7$, and $0.7 \leq h \leq 0.9$; and e) finish the conversation if the ratio of number of user responses and number of slots in the system ≥ 4. Finally, two simulated users were utilized, one for learning and one for test, with 90% and 80% of coherent responses respectively. In this way, the policies were learnt with well-behaved users and tested with more difficult ones.

4.3. Results

Table 1 shows that the proposed algorithm reduced the state-action spaces by 94%, meaning that we can avoid unnecessary learning very significantly. However, we must ask: *"Can policies learnt on reduced spaces achieve as good performance as those learnt on full spaces?"* Our results are favourable, and we analyzed them in both learning and test phases with four kinds of plots as shown in figure 5: Average Steps Per Episode (ASPE), Average Reward Per Episode (ARPE), Average MaxQ values per Episode (AMPE), and Explored Space Per Episode (ESPE). On the one hand, we used ASPE and ARPE to observe the strategies performance; on the other, we used AMPE and ESPE to observe their convergence.

Results from the *learning* phase report no degradation in performance, neither in steps per episode nor in average reward. Also, average results from the learning phase report that policies learnt on reduced spaces converged 93% faster than policies learnt on full spaces, which is not surprising because each policy used its own space and the sizes are different. Nevertheless, this result tells us

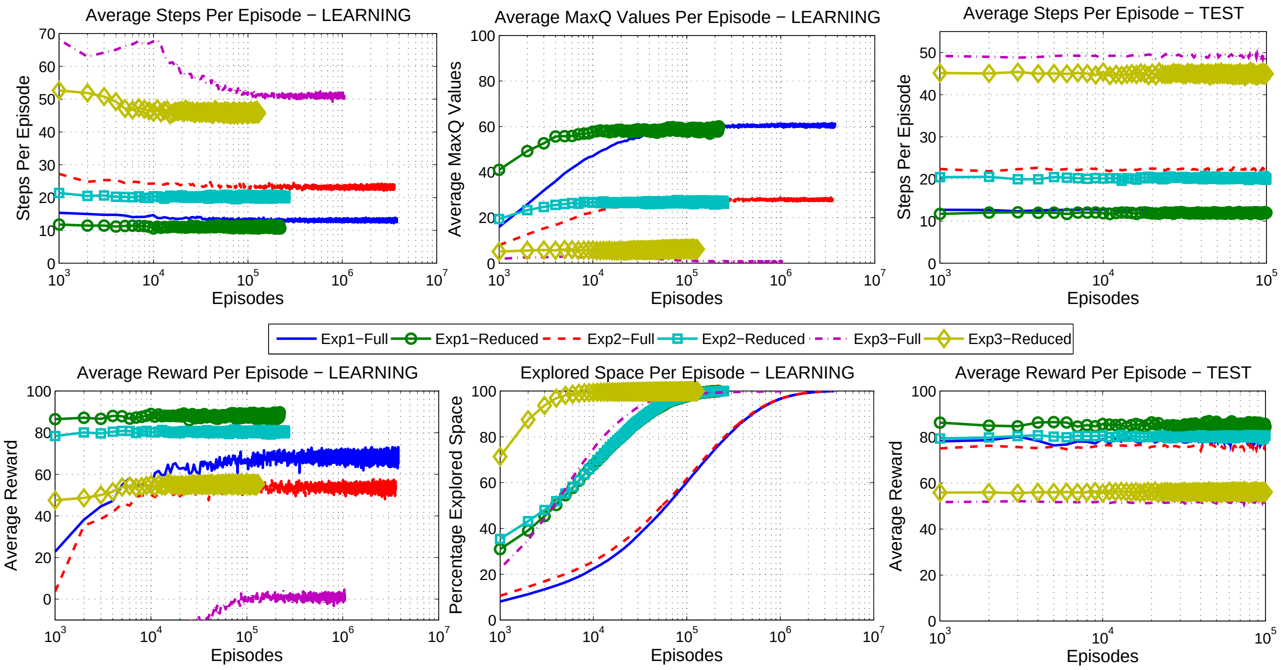

Figure 5: *Performance results (X-axis in log scale) in the learning and test phases, data points are averages of groups of 1000 episodes.*

the importance of learning dialogue strategies on reduced spaces by requiring a much lower amount of dialogues for learning.

Finally, results from the *test* phase reveal that policies learnt on reduced spaces can obtain higher performance. Average results report 8.4% less time steps and 7.7% higher reward. This means that policies learnt on full spaces do not always learn optimal actions, which may be attributed to the fact that the more invalid state-actions, the more chance to learn non-optimal actions.

5. Conclusions and Future Work

The contribution of this paper is the $sapReduction$ algorithm, which generates composite reduced state-action spaces to learn mixed-initiative multi-goal dialogue strategies using the reinforcement learning paradigm. We argue that the proposed algorithm derived from prior knowledge of valid state-actions is: generic to optimize confirmation, can be extended to optimize other state variables and does not require significant development effort. Average results using strategies learnt on reduced spaces reveal the following benefits against full spaces: 1) less computer memory (94% reduction), 2) faster learning (93% faster convergence) and better performance (8.4% less time steps and 7.7% higher reward). To our knowledge, our experiments are the first aiming optimization of large-scale dialogue systems (with up to 20 slots), where the utility of reduced spaces becomes crucial for more complex and larger systems. The last we plan to investigate using hierarchical learning with dialogue simulators learnt from data [7,12].

6. Acknowledgements

This research was mainly sponsored by PROMEP, part of the Mexican Ministry of Education (http://promep.sep.gob.mx) and partially by the Autonomous University of Tlaxcala (www.uatx.mx).

7. References

[1] Zue, V., and Glass, J. Conversational Interfaces: Advances and Challenges. In *Proc. of the IEEE*, 88(8), pp. 1166-1180, 2000.

[2] Levin, E., Pieraccini, R., and Eckert, W. A Stochastic Model of Computer-Human Interaction for Learning Dialog Strategies. In *Proc. of Eurospeech*, pp. 1883-1886, 1997.

[3] Sutton R., and Barto, A. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[4] Singh S., Litman, D., Kearns, M., and Walker, M. Optimizing Dialogue Management with Reinforcement Learning: Experiments with the NJFun System. In *Journal of AI Research*, 16, pp.105-133, 2002.

[5] Scheffler, K., and Young, S. Automatic Learning of Dialogue Strategy Using Dialogue Simulation and Reinforcement Learning. In *Proc. of HLT*, 2002.

[6] Pietquin, O., and Renals, S. ASR System Modeling for Automatic Evaluation and Optimization of Dialogue Systems. In *Proc. of ICASSP*, pp. 46-49, 2002.

[7] Schatzmann, J., Stuttle, M. N., Weilhammer, K., and Young, S. Effects of the User Model on Simulation-Based Learning of Dialogue Strategies. In *Proc. of ASRU*, pp. 220-225, 2005.

[8] Williams, J., and Young, S. Scaling Up POMDPs for Dialogue Management: The "Summary POMDP" Method. In *ASRU*, 2005.

[9] Henderson, J., Lemon, O., and Georgila, K. Hybrid Reinforcement/Supervised Learning for Dialogue Policies from COMMUNICATOR data. In *Proc. of Workshop on K&R-PDS (IJCAI)*, 2005.

[10] Denecke, M., Dohsaka, K., and Nakano, M. Fast Reinforcement Learning of Dialogue Policies Using Stable Function Approximation. In *Proc. of IJCNLP*, pp. 1-11, 2002.

[11] Walker, M., Passonneau, R., and Boland, J. Quantitative and Qualitative Evaluation of Darpa Communicator Spoken Dialogue Systems. In *Proc. of ACL*, pp. 515-522, 2001.

[12] Cuayáhuitl, H., Renals, S., Lemon, O., and Shimodaira, H. Human-Computer Dialogue Simulation Using Hidden Markov Models. In *Proc. of ASRU*, pp. 290-295, 2005.