

A EXPERIMENTAL RESULTS

A.1 EMPIRICAL RESULTS FOR PROPOSITION 5.1

We evaluated the performance of lazy-agents trained in environments with random delays, where $o_{\max} = 10$, and compared them to normal agents trained in those with constant delays, where $o = o_{\max}$, as illustrated in Fig. 5. The empirical results demonstrate that the performance of lazy-agents is comparable to that of normal agents. This supports our argument that RMDPs can be transformed into equivalent CDMDPs through our lazy-agents, thereby enabling conventional methods designed for handling constant delays to be naturally extended to environments with random delays.

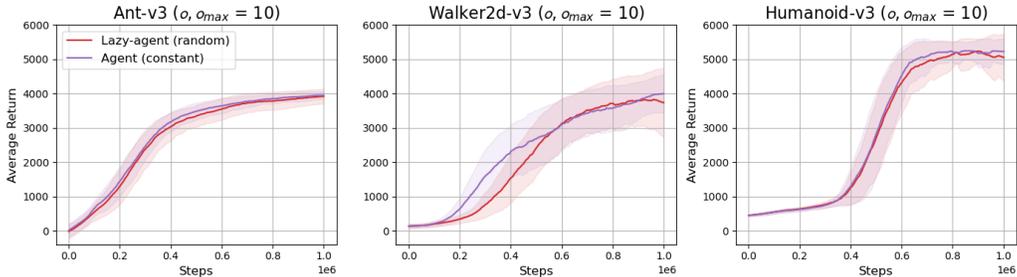


Figure 5: Performance curves of lazy-agents trained in environments with random delays of $o_{\max} = 10$ and normal agents trained in environments with constant delays of $o = o_{\max}$ for continuous control tasks in the MuJoCo benchmark. All tasks were conducted with five different seeds for one million time-steps. The shaded regions represent the standard deviation of average returns.

A.2 PLOTS OF PERFORMANCE COMPARISON

In this section, we present the performance curves of each algorithm on the MuJoCo tasks with random delays of $o_{\max} \in \{5, 10, 20\}$. All tasks were conducted with five different seeds for one million time-steps. The shaded regions represent the standard deviation of average returns.

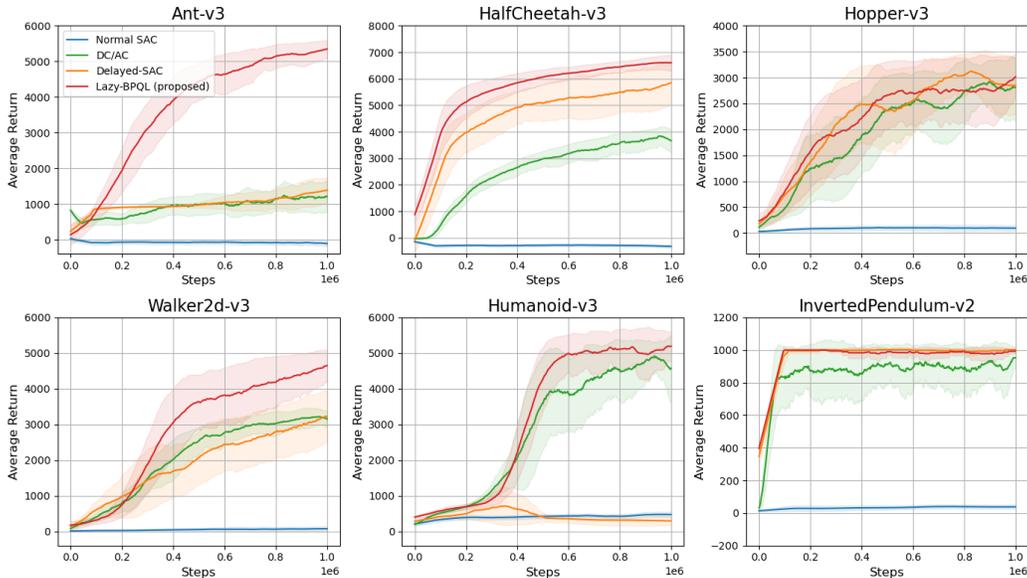


Figure 6: Performance curves of each algorithm on the MuJoCo tasks with $o_{\max} = 5$.

648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701

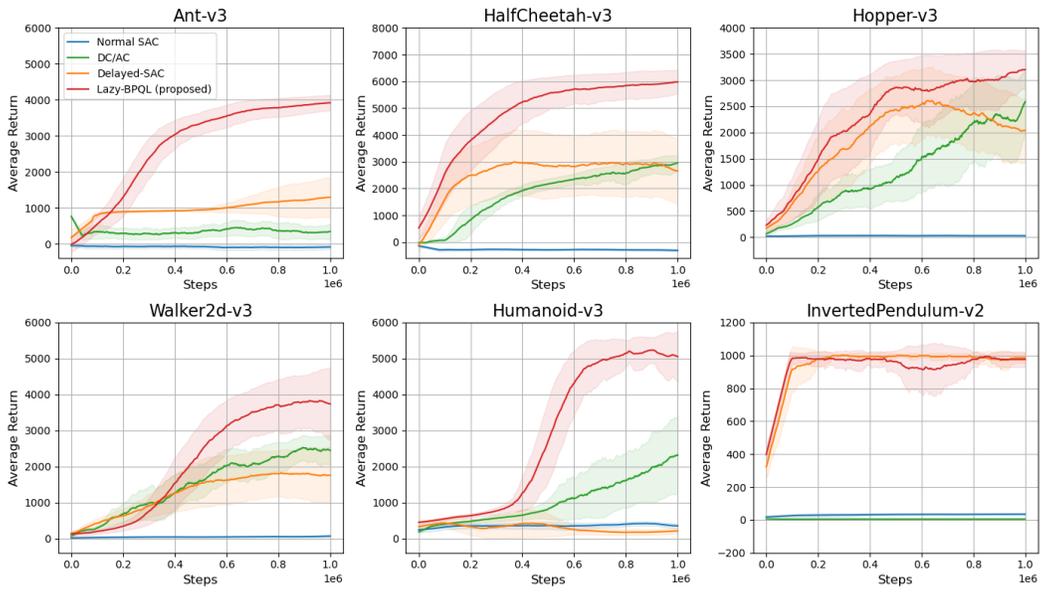


Figure 7: Performance curves of each algorithm on the MuJoCo tasks with $o_{\max} = 10$.

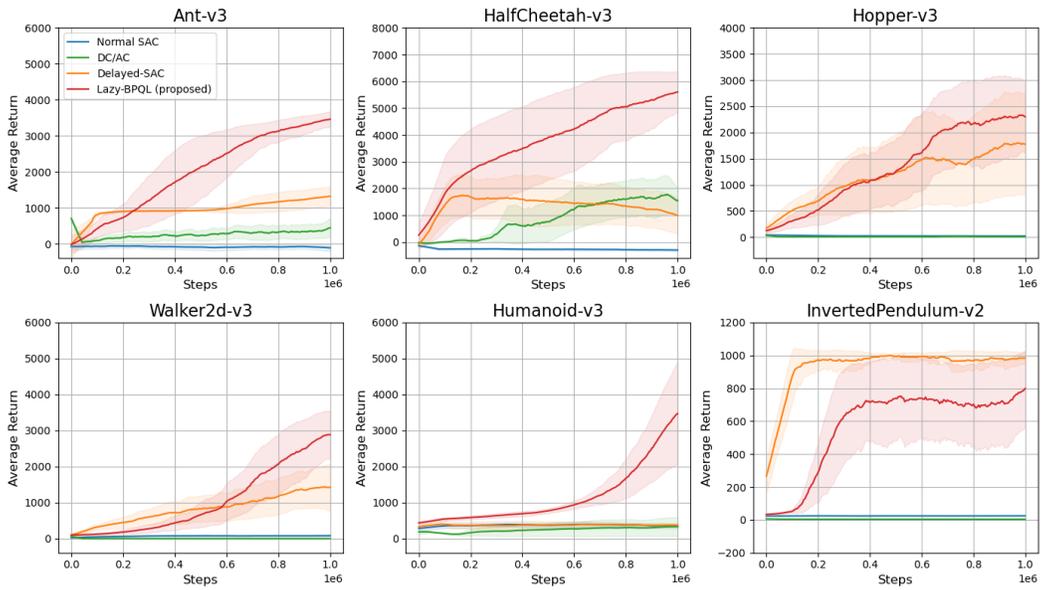


Figure 8: Performance curves of each algorithm on the MuJoCo tasks with $o_{\max} = 20$.

A.3 STATE-SPACE EXPLOSION ISSUE

In this section, we present the performance curves of lazy-augmented-SAC and lazy-BPQL on the MuJoCo tasks with random delays of $\sigma_{\max} \in \{5, 10, 20\}$. As shown in Fig. 9, the proposed lazy-BPQL outperforms lazy-augmented-SAC in terms of asymptotic performance and sample efficiency. Notably, lazy-augmented-SAC completely fails to learn for the tasks even with the random delay of $\sigma_{\max} = 5$, highlighting the importance of avoiding the state-space explosion issue.

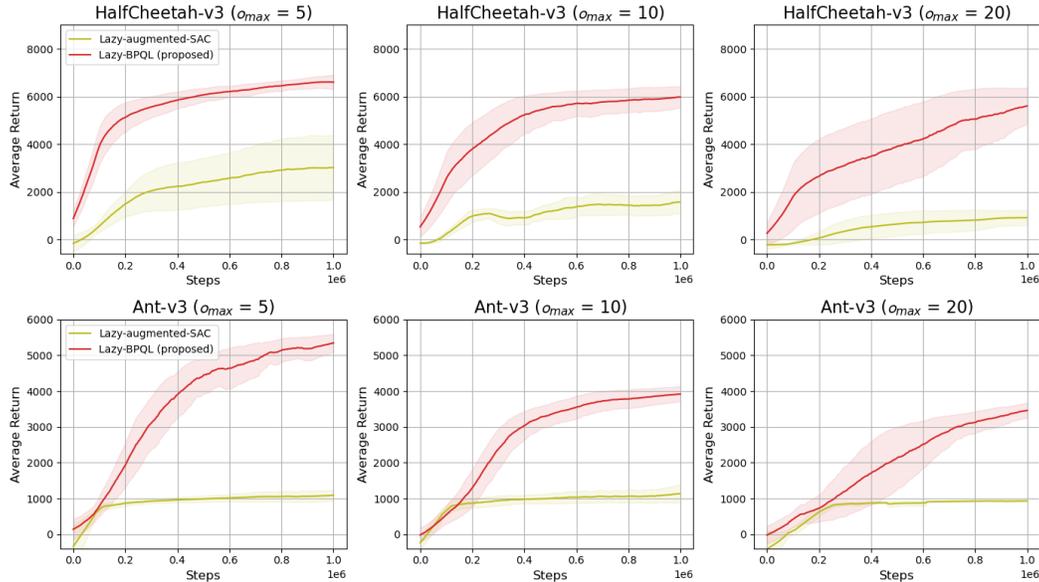


Figure 9: Performance curves of lazy-augmented-SAC and lazy-BPQL for continuous control tasks from the MuJoCo benchmark with random delays of $\sigma_{\max} = \{5, 10, 20\}$. From the results, lazy-BPQL dominates lazy-augmented-SAC, underscoring the importance of avoiding the state-space explosion issue.

B EXPERIMENTAL DETAILS

B.1 ENVIRONMENTAL DETAILS

Table 2: Environmental details of the MuJoCo benchmark.

Task	State dimension	Action dimension	Time-step (s)
Ant-v3	27	8	0.05
HalfCheetah-v3	17	6	0.05
Walker2d-v3	17	6	0.008
Hopper-v3	11	3	0.008
Humanoid-v3	376	17	0.015
InvertedPendulum-v2	4	1	0.04

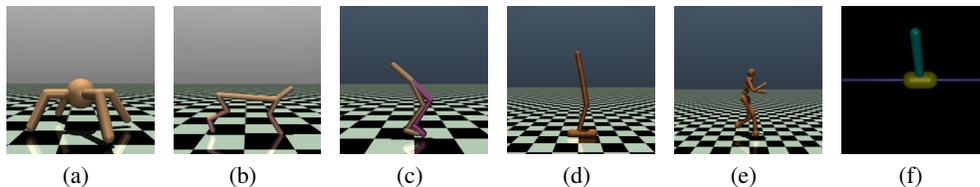


Figure 10: Experimental environments in the MuJoCo benchmark: (a) Ant-v3 (b) HalfCheetah-v3, (c) Walker2d-v3, (d) Hopper-v3, (e) Humanoid-v3, and (f) InvertedPendulum-v2

B.2 IMPLEMENTATION DETAILS

The implementation details of the proposed lazy-BPQL align with those presented in Kim et al. (2023), with the specific hyperparameters listed in Table 3. Since the baseline algorithms included in our experiments employ the SAC algorithm as their foundational learning algorithm, the hyperparameters are consistent across all approaches, except for the DC/AC algorithm.

Table 3: Hyperparameters for lazy-BPQL and the baselines.

Hyperparameters	Values
Actor network	256, 256
Critic network	256, 256
Learning rate (actor)	3e-4
Learning rate (critic)	3e-4
Temperature (α)	0.2
Discount factor (γ)	0.99
Replay buffer size	1e6
Mini-Batch size	256
Target entropy	$-\dim \mathcal{A} $
Target smoothing coefficient (ξ)	0.995
Optimizer	Adam (Kingma, 2014)
Total time-steps	1e6

B.3 PSEUDO CODE OF LAZY-BPQL

The proposed lazy-agent can be seamlessly integrated into the BPQL framework with minimal modifications by *using* the initial state for decision-making at its maximum delayed times. Subsequently, all states become naturally available for use at their respective maximum delayed times.

In the implementation, a temporary buffer \mathcal{B} has been employed, as utilized by Kim et al. (2023), to store *observed* states, corresponding rewards, and action histories, which enables the agent to access timely and relevant information for constructing augmented states.

Algorithm 1 Lazy Belief Projection-based Q -Learning (Lazy-BPQL)

```

1: Input: actor  $\bar{\pi}_\phi(a|\hat{x})$ , beta critic  $Q_{\theta,\beta}(s,a)$ , target beta critic  $Q_{\tilde{\theta},\beta}(s,a)$ , replay buffer  $\mathcal{D}$ , temporary buffer  $\mathcal{B}$ , maximum delay  $o_{\max}$ , beta critic learning rate  $\lambda_Q$ , actor learning rate  $\lambda_\pi$ , soft update rate  $\xi$ , episodic length  $H$ , and total number of episodes  $E$ .
2: for episode  $e = 1$  to  $E$  do
3:   for time-step  $t = 1$  to  $H$  do
4:     if  $t < o_{\max}$  then
5:       select random or ‘no-ops’ action  $a_t$ 
6:       execute  $a_t$  on environment
7:       put  $a_t$ , observed states, rewards to  $\mathcal{B}$ 
8:     else if  $t = o_{\max}$  then ▷ wait for  $o_{\max}$  time-steps
9:       select random or ‘no-ops’ action  $a_t$ 
10:      execute  $a_t$  on environment
11:      put  $a_t$ , observed states, rewards to  $\mathcal{B}$ 
12:     else
13:       get  $s_{t-o_{\max}}, a_{t-o_{\max}}, \dots, a_{t-1}$  from  $\mathcal{B}$ 
14:       ▷ get most recent usable state and action histories
15:        $\hat{x}_t \leftarrow (s_{t-o_{\max}}, a_{t-o_{\max}}, \dots, a_{t-1})$  ▷ construct augmented state
16:        $a_t \leftarrow \bar{\pi}_\phi(\hat{x}_t)$ 
17:       execute  $a_t$  on environment
18:       put  $a_t$ , observed states, rewards to  $\mathcal{B}$ 
19:     if  $t > 2o_{\max}$  then
20:       get  $s_{t-2o_{\max}}, s_{t-2o_{\max}+1}, s_{t-o_{\max}}, r_{t-o_{\max}}, a_{t-2o_{\max}}, \dots, a_{t-o_{\max}}$  from  $\mathcal{B}$ 
21:        $\hat{x}_{t-o_{\max}} \leftarrow (s_{t-2o_{\max}}, a_{t-2o_{\max}}, \dots, a_{t-o_{\max}})$ 
22:        $\hat{x}_{t-o_{\max}+1} \leftarrow (s_{t-2o_{\max}+1}, a_{t-2o_{\max}+1}, \dots, a_{t-o_{\max}+1})$ 
23:       store  $(\hat{x}_{t-o_{\max}}, s_{t-o_{\max}}, a_{t-o_{\max}}, r_{t-o_{\max}}, \hat{x}_{t-o_{\max}+1}, s_{t-o_{\max}+1})$  in  $\mathcal{D}$ 
24:       pop  $s_{t-2o_{\max}}, a_{t-2o_{\max}}$  from  $\mathcal{B}$ 
25:     end if
26:   end if
27: end for
28: for each gradient step do
29:    $\theta \leftarrow \theta - \lambda_Q \nabla \mathcal{J}_{Q_\beta}(\theta)$  ▷ update beta critic
30:    $\phi \leftarrow \phi - \lambda_\pi \nabla \mathcal{J}_\pi(\phi)$  ▷ update actor
31:    $\tilde{\theta} \leftarrow \xi\theta + (1 - \xi)\tilde{\theta}$  ▷ update target beta critic
32: end for
33: end for
34: Output: actor  $\bar{\pi}_\phi$ 

```

C VISUAL REPRESENTATION OF LAZY-AGENT

In this section, we provide a visual representation of the proposed lazy-agent employed in RDMDPs, where the maximum delay is set to $o_{\max} = 3$.

* s_a^b : a = generated time, b = delay

Times	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 8	t = 9	t = 10
Generated states										
Observed states										
Usable states										
Augmented states										
Actions										

(a) Time $t = 0$

* s_a^b : a = generated time, b = delay

Times	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 8	t = 9	t = 10
Generated states	s_1^2									
Observed states										
Usable states										
Augmented states	'no-ops'									
Actions	a_1									

(b) Time $t = 1$

* s_a^b : a = generated time, b = delay

Times	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 8	t = 9	t = 10
Generated states	s_1^2	s_2^1								
Observed states										
Usable states										
Augmented states		'no-ops'								
Actions	a_1	a_2								

(c) Time $t = 2$

Figure 11: At times 1 and 2, the states s_1^2 and s_2^1 are generated but remain unobserved by the lazy-agent due to delays. In this scenario, the lazy-agent does nothing ('no-ops') until the initial state s_1^2 becomes usable.

918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971

* s_a^b : a = generated time, b = delay

Times	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 8	t = 9	t = 10
Generated states	s_1^2	s_2^1	s_3^2							
Observed states			s_1^2 s_2^1							
Usable states										
Augmented states		'no-ops'								
Actions	a_1	a_2	a_3							

(a) Time $t = 3$

* s_a^b : a = generated time, b = delay

Times	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 8	t = 9	t = 10
Generated states	s_1^2	s_2^1	s_3^2	s_4^3						
Observed states			s_1^2 s_2^1							
Usable states				s_1^2						
Augmented states		'no-ops'		\hat{x}_4						
Actions	a_1	a_2	a_3	a_4						

(b) Time $t = 4$

* s_a^b : a = generated time, b = delay

Times	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 8	t = 9	t = 10
Generated states	s_1^2	s_2^1	s_3^2	s_4^3	s_5^1					
Observed states			s_1^2 s_2^1		s_3^2					
Usable states				s_1^2	s_2^1					
Augmented states		'no-ops'		\hat{x}_4	\hat{x}_5					
Actions	a_1	a_2	a_3	a_4	a_5					

(c) Time $t = 5$

Figure 12: At time 3, states s_1^2 and s_2^1 are observed simultaneously. As the lazy-agent uses these observed states at their maximum delayed times, s_1^2 is used at time 4 and s_2^1 is used at time 5. These states are reformulated as augmented states before being fed into the policy, thereafter determining the appropriate actions. States s_3^2 , s_4^3 , and s_5^1 are generated at corresponding times, with s_3^2 being observed at time 5.

972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025

* s_d^b : a = generated time, b = delay

Times	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 8	t = 9	t = 10
Generated states	s_1^2	s_2^1	s_3^2	s_4^3	s_5^1	s_6^0				
Observed states			s_1^2 s_2^1		s_3^2	s_5^1 s_6^0				
Usable states				s_1^2	s_2^1	s_3^2				
Augmented states		'no-ops'		\hat{x}_4	\hat{x}_5	\hat{x}_6				
Actions	a_1	a_2	a_3	a_4	a_5	a_6				

(a) Time $t = 6$ * s_d^b : a = generated time, b = delay

Times	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 8	t = 9	t = 10
Generated states	s_1^2	s_2^1	s_3^2	s_4^3	s_5^1	s_6^0	s_7^3			
Observed states			s_1^2 s_2^1		s_3^2	s_5^1 s_6^0	s_4^3			
Usable states				s_1^2	s_2^1	s_3^2	s_4^3			
Augmented states		'no-ops'		\hat{x}_4	\hat{x}_5	\hat{x}_6	\hat{x}_7			
Actions	a_1	a_2	a_3	a_4	a_5	a_6	a_7			

(b) Time $t = 7$ * s_d^b : a = generated time, b = delay

Times	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 8	t = 9	t = 10
Generated states	s_1^2	s_2^1	s_3^2	s_4^3	s_5^1	s_6^0	s_7^3	...		
Observed states			s_1^2 s_2^1		s_3^2	s_5^1 s_6^0	s_4^3			
Usable states				s_1^2	s_2^1	s_3^2	s_4^3	s_5^1		
Augmented states		'no-ops'		\hat{x}_4	\hat{x}_5	\hat{x}_6	\hat{x}_7	\hat{x}_8		
Actions	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8		

(c) Time $t = 8$

Figure 13: States s_6^0 and s_7^3 are generated at respective times. At time 6, states s_5^1 and s_6^0 are observed simultaneously but are not immediately usable because the previously generated states, s_3^2 and s_4^3 , have not yet been used in decision-making processes. Instead, s_3^2 is used at this time. At time 7, state s_4^3 is observed and is available for use immediately. At time 8, state s_5^1 becomes usable, as all previously generated states have now been both observed and used.

1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079

* s_a^b : a = generated time, b = delay

Times	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 8	t = 9	t = 10
Generated states	s_1^2	s_2^1	s_3^2	s_4^3	s_5^1	s_6^0	s_7^3	
Observed states			s_1^2 s_2^1		s_3^2	s_5^1 s_6^0	s_4^3			
Usable states				s_1^2	s_2^1	s_3^2	s_4^3	s_5^1	s_6^0	
Augmented states		'no-ops'		\hat{x}_4	\hat{x}_5	\hat{x}_6	\hat{x}_7	\hat{x}_8	\hat{x}_9	
Actions	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	

(a) Time $t = 9$

* s_a^b : a = generated time, b = delay

Times	t = 1	t = 2	t = 3	t = 4	t = 5	t = 6	t = 7	t = 8	t = 9	t = 10
Generated states	s_1^2	s_2^1	s_3^2	s_4^3	s_5^1	s_6^0	s_7^3
Observed states			s_1^2 s_2^1		s_3^2	s_5^1 s_6^0	s_4^3			s_7^3
Usable states				s_1^2	s_2^1	s_3^2	s_4^3	s_5^1	s_6^0	s_7^3
Augmented states		'no-ops'		\hat{x}_4	\hat{x}_5	\hat{x}_6	\hat{x}_7	\hat{x}_8	\hat{x}_9	\hat{x}_{10}
Actions	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}

(b) Time $t = 10$

Figure 14: At times 9 and 10, states s_6^0 and s_7^3 are used in sequence. Despite the state observations occurring simultaneously or being out of order, all the delayed states are consistently used in sequence at their maximum delayed times, i.e., $\tau(s_n^{o_n}) = n + o_{\max}, \forall n > 0$.