Proceedings of the 2003 IEEE/RSJ
Intl. Conference on Intelligent Robots and Systems
Las Vegas, Nevada · October 2003

A Reinforcement Learning Approach Involving a Shortest Path Finding Algorithm

Woo Young Kwon[1], Sanghoon Lee[2], Il Hong Suh[1,2]

[1]The Graduate School of Information and Communications

[2]Department of Electrical Engineering and Computer Science

Hanyang University, Seoul, KOREA

Tel : +82-2-2290-0392, E-mail : ihsuh@hanyang.ac.kr

Abstract

Reinforcement learning (RL) has been widely used as a learning mechanism of an artificial life system. However, RL usually suffers from slow convergence to the optimum state-action sequence or a sequence of stimulus-response (SR) behaviors, and may not correctly work in non-Markov processes. In this paper, first, to cope with slow-convergence problem, if some state-action pairs are considered as disturbance for optimum sequence, then they are to be eliminated in long-term memory (LTM), where such disturbances are found by a shortest path-finding algorithm. This process is shown to let the system get an enhanced learning speed. Second, to partly solve non-Markov problem, if a stimulus is frequently met in a searching-process, then the stimulus will be classified as a sequential percept for a non-Markov hidden state. And thus, a correct behavior for a non-Markov hidden state can be learned as in a Markov environment. To show the validity of our proposed learning technologies, several simulation results will be illustrated.

1. Introduction

Artificial life system(ALS) can be defined as an artificial(mechanical or graphical) organism which is similar to the real living organism. ALS is often required to have an ability to learn what behaviors would be well-suited for a given environmental situation for adaptation on dynamically changing real and/or virtual environment. Animal learning can be classified as classical conditioning and operant conditioning: Classical conditioning is to learn associations between unconditional stimulus and conditional stimulus, which gets living organisms to find out what stimulus is more beneficial among stimuli the living organism might experience [1]. On the other hand, operant conditioning is to learn proper stimulus-response(SR) behaviors by using reinforcement(or rewards) which may be obtained whenever a correct behavior is shown to a given stimulus [2]. A complex behavior can be made by a sequence of SR behaviors.

Reinforcement learning technique by Minsky [3] can be applied to the learning of artificial life system, since it is very similar to operant conditioning in the sense that both methods utilize only reward signal to learn appropriate associations between stimulus and response, and thus a *priori* environmental information is no longer required. However, such reinforcement learning technique may show some difficulties when they are to be applied to the reality: First, rewards may not be given immediately after a behavior is completed. Rewards are usually given after a goal is achieved by performing a sequence of stimuli and behaviors. This delayed reward causes the learning to be very time-consuming, or even to be not practical.

Second, RL technique may not correctly work in non-Markov environments. Non-Markov environment is an environment in which a current state is correlated with past history of states, and behaviors. Such a non-Markov environment can be met when environmental changes are not simple, but complex, or when perception capability of an ALS is too low to differentiate necessary environmental states.

In this paper, we will propose a mission (sequence) learning method which shows a relatively fast speed of convergence, and can be applied to a non-Markov environment(as well as Markov environment). For this, in section 2, delayed reward problems are to be reviewed, and some features on non-Markov systems are to be revisited. In section3, a novel type of reinforcement learning algorithm is proposed to partially cope with delayed-reward problem, where some unnecessary state-action pairs are found by means of a well-known single shortest path finding algorithm, Dijkstra's algorithm, and then their reliability values are reduced. As in Monte Carlo approach these reasoning process are performed after a reward signal is received. On the other hand, an algorithm is also proposed to find out a time-sequence which consists of current states, previous states and behaviors, and can be considered as a new state to recognize non-Markov hidden states. In section 4, to show the validity of our proposed learning technologies, several simulation results will be illustrated.

2. Background

2.1. S-R learning

Lorenz defined animal learning as adaptive changes of behavior [4]. Animal Learning has two types of associations. One is classical conditioning, the other is operant conditioning [5]. In particular, operant conditioning is to learn associations between stimulus and response. And, complex behaviors can be analyzed as a sequence of stimuli and response.

Simple stimulus-response associations cannot be used to make inferences. Tolman postulated that animals could learn something like S–R–S' associations [6]. These tell the animal that if it is in situation S and performs response R it will end up in situation S'. Such a way of describing associations is much more powerful than the others we have so far considered. For example, if the animal is in possession of the two associations S0–R0–S1 and S1–R1–S2, it can, at least potentially, infer that by performing the responses R0 and R1 at S0 it will reach S2. Thus, it can perform sequences of responses in order to obtain a goal even if that particular sequence has never been performed before [7].

Reinforcement learning is a way of learning what to do – how to map situation to action so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most form of learning, but instead must discover which actions yield the most reward by trying them. In the most interesting and challenging case, actions may affect not only the immediate reward but also the next situation and subsequent reward [8][9].

Reinforcement learning has some similarity to operant conditioning, because reinforcement learning learns relations between state and action. And thus, learners do not need to know correct actions as training state [10]. TD, Q and Monte Carlo learning method to implement are typical method for reinforcement learning[11].

However, general reinforcement learning usually suffers from delayed rewards which cause slow speed of learning especially when state space is large [12].

2.2. Sequence learning

The Markov property in (1) defines that transitions and outcomes depend only on the current state and the action.

$$\Pr\left(S_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, s_0, a_0\right) = \Pr\left(s_{t+1} \mid s_t, a_t\right) \qquad (1)$$

In (1), s_t is the model state at time t, and a_t is the action chosen at time t, and subscripts $t-1$ and $t+1$ indicate steps in the past and future.

If an environment satisfies Markov property in (1), an agent has no need to look backward to know past states and actions. And behavior sequence can be viewed as a stimulus-response chain. In the Fig. 1, if an agent possesses three relations, s1-a1, s2-a2 and s3-a3, the agent can reach s4 from s1.

In a Markov environment, a behavior sequence is considered as a sequence of stimuli and response. Thus, an agent may learn such a sequence by employing a reinforcement learning technique. For Markov environment, a variety of different reinforcement learning algorithms has been devised(such as TD(λ) and Q-learning) algorithms.

Features that are not immediately observable, but are correlated with past state, observations or action are called non-Markov hidden state. This implies that

$$\Pr\left(S_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, s_0, a_0\right) \neq \Pr\left(s_{t+1} \mid s_t, a_t\right) \qquad (2)$$

Uncovering non-Markov hidden state is a huge selective perception problem. This results in a problem of feature selection, where features should be selected not from the current percept, but from the current and past percepts. There are some techniques to deal with non-Markov hidden state [13][14][15][16].

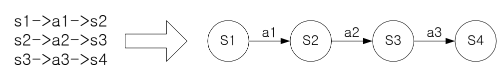


Fig. 1. An example representing a behavior sequence.

3. A fast RL technique involving a shortest-path finding algorithm

An artificial life system is required to learn associations between stimulus and response by a small number of trials. Exploration for learning usually needs much cost, and may get the system to be in danger. On the other hand, internal reasoning may usually cause computational or algorithmic complexity, but its cost is relatively low when compared with cost of exploration process [13].

Reinforcement learning (RL), as a widely used learning technique for artificial life system, allows the system to get learned only if a reward signal is received by the system. Actually, learning is progressed by examining what S-R behaviors should be credited among past S-R behaviors in getting a reward. Thus, RL shows a relatively slow speed of convergence, and may not practically work due to such slow learning speed when size of state-space becomes large.

In this paper, a fast RL algorithm is proposed, where internal reasoning process is incorporated to reduce probability of selecting wrong behaviors. Specifically, first, RL is applied until a successful episode is completed. Then, all S-R behaviors of the episode, which have been recorded in a short-term memory (STM), are transferred to long-term memory (LTM), in which a shortest path-finding algorithm is applied to find out the shortest path from initial S-R behavior to the goal stimulus. S-R behaviors on the shortest path are the highly credited by increasing their reliability values, and S-R behaviors which are not on the shortest path are punished b reducing their reliability values. Since an S-R behavior will be selected based on behaviors and their reliability value of LTM, it is expected to reduce the probability of choosing unnecessary behaviors, and thus to increase the learning speed.

3.1. Exploration of state space of reinforcement learning

Internal memory consists of short-term memory (STM) and long-term memory (LTM). In the STM, stimulus-response (SR) pairs are recorded along the time. Fig. 2 shows a process in which data in STM are moved to LTM. Once all data in STM is moved to LTM, then STM is cleared to store data of new exploration. All SR pairs in STM are assumed to be appetitive behaviors to get rewards. With references to the time at which a reward is received, SR behaviors near the reference time should be more credited than SR behaviors far from the reference time. This can be reflected as a reliability value, V_{ij}, given by

$$V(s_i, a, s_{i+1}) \leftarrow V(s_i, a, s_{i+1}) + \frac{\eta(1 - V(s_i, a, s_{i+1}))}{i^{\lambda}} \quad (3)$$

where s_i is the index of the i^{th} stimulus, a is the index of response behavior, η is a learning rate, λ is decay rate, and i^{λ} is weightings of distance from reference time. It is remarked that (3) is similar to equation for Monte-Carlo method. Fig. 3 shows pseudo-code of STM and LTM operations for exploration of reinforcement learning.

It is also remarked that *V(s,a,sp)* plays a similar role of Q*(s,a)* value in Q-learning theory, and V is also used as a probability sampler to choose an action or behavior for exploration of a new episode during STM operation. That is, to choose an action for a state, Boltzman exploration method is here used. Since V is a function of 3 arguments, Boltzman exploration equation is given by

$$\Pr(s,a) = \frac{e^{V(s,a,sp)/T}}{\sum_{a'}\sum_{s'} e^{V(s',a',sp)/T}} \,. \quad (4)$$

Note again that V is the function of 3 arguments, and Q is the function of 2 arguments. Specifically, $V(s_i,a,s_{i+1})$ has the third argument s_{i+1} to let us know what stimulus is the outcome of behavior, a, for the stimulus, s_i. This makes our reasoning processing easy, and will be efficiently utilized to deal with non-Markov hidden state to be explained in later section.

STM

time	state	action
1	s_0	a_0
2	s_1	a_1
3	s_2	a_2
4	s_3	a_3

received reward

value update

LTM

state	action	consequence of action	reliability $V(s,a,sp) \le 1$
s_0	a_0	s_1	$V(s_0, a_0, s_1)$
s_1	a_1	s_2	$V(s_1, a_1, s_2)$
s_2	a_2	s_3	$V(s_2, a_2, s_3)$
s_3	a_3	goal index	$V(s_3, a_3, s_G)$

Fig. 2. STM and LTM

```
repeat(for each episode)
  initialize STM
  repeat (for each step of episode)
    choose action and state a,s with respect to V(s,a,sp)     (Boltzman exploration)
    store a, s to STM
    take action a
  LOOP i = 1 to STMsize
    IF i=STMsize
       V(s_i,a_i,Goal Index)=1
    ELSE
       V(s_i,a_i,s_{i+1}) <- V(s_i,a_i,s_{i+1}) + η (1-V(s_i,a_{i+1},s_{i+1}))/i^λ
V(s,a,sp) : LTM Value
sp : consequence of action
η : learning rate 0<η<1
λ : decay rate    0<η<1
```

Fig. 3. Pseudo code of suggested learning algorithm

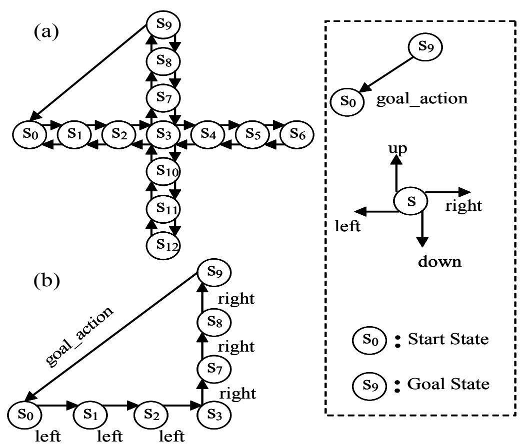


Fig. 4. An example of FSM to find the shortest path

```
Create Graph G(V,E)
repeat for all l ∈ LTM
  get s,a,sp from LTM entry l
  G(V,E) <- l(s start Vertex, a as Edge sp as end Vertex, weight=1)
end

find all pair shortest path P from G using Djkstra's algorithm

repeat for all l ∈ LTM
  get s,a,sp from LTM entry l
  make V,E from l
  if V,E ∈ G then
     V(s,a,sp) <- V(s,a,sp) + α
     clamp(V(s,a,sp),0,1)
  else
     V(s,a,sp) <- γ × V(s,a,sp)
end
```

Fig. 5. pseudo-code of the reasoning process.

3.2. Finding of shortest-path from LTM information

Note that LTM may include several paths from a state to the goal state in which a reward was obtained, when all LTM elements are represented as a finite state machine (FSM). Thus, we can find the shortest-path among all such possible paths. And it is expected that if we can reinforce SR behaviors on the shortest paths, and discourage SR behaviors of high *V*-value, which are not on the shortest path, then we can reduce time to explore unreliable behaviors, or reduce searching space owing to the exploration by *V*-value sampler in STM. To be more specific, consider an FSM representing an LTM data as shown Fig. 4, where the FSM is a directed graph. Then, our objective is to find a single source shortest path from such a directed graph. Fig. 4-b shows the shortest-path for FSM graph in Fig. 4-a. Here, Dijkstia's algorithm [17] is employed to find the shortest path. The path from reasoning process is local optimum path in states of one episode. Once shortest path is found, the values of S-R relations out of the shortest path will be decreased. This process increases the possibility of finding optimal path and convergence.

Fig. 5 shows the pseudo-code of the reasoning process including the shortest path finding.

3.3 Convergence and optimality of the proposed algorithm

In typical Monte Carlo method, the convergence and optimality was not been formally proved[11]. Because our learning method is similar to Monte Carlo method, one might consider that it may show problems of convergence and optimality similar to Monte Carlo method. But fortunately, by using our reasoning process we can heuristically prove optimality and convergence as follows;

Note that the reasoning process can always find optimal path within state space that the agent can reach[17]. Then, we can consider two cases. First, when local state-space within LTM includes the global optimal path, the optimal path can be found. Secondly, when local state-space within LTM does not include the global optimal path, a sub-optimal will be found. But, note that in our approach, the size of local state-space will be increased for next trial, since the reliability value of SR behaviors on the sub-optimal path is set to be high, and the probability of S-R behaviors on other paths is not kept to be zero. Therefore, if the number of episode increases to be infinity, the state space within LTM will include the optimal path. Then, global optimal path can be guaranted to be found.

4. Sequential percept generation

To classify what state is a non-Markov hidden state, current state is not defined only by current stimulus, but defined by combination of previous SR's. STM is used to store such previous SR behaviors.

To see how a non-Markov hidden state is found, consider an example in Fig. 6. Fig. 6-a shows a map in which starting state will be 1 and goal state will be G. And Fig. 6-b shows an FSM for an episode. For S_2 in FSM, there are two behaviors, up or down. However, what behavior will be better cannot be determined by only S_2, since as shown in Fig. 6-c, wrong selection of up and down behaviors for state S_2 would cause a never-ending path to the goal. Thus, it is required to know that there are two different S_2 state by referencing previous state and action. In the example in Fig. 6, S_1 – up → S_2 and S_3 – down → S_2 showed be classified. Here, *stimulus – action (or behavior) → current stimulus (result of previous action)* is termed as "sequential percept", and will be used to classify non-Markov hidden states. Two contiguous sequential percept can be handled by a chained process as shown in Fig. 7.

On the other hand, such a sequential percept can be effectively searched out by examining what SR behaviors are repetitive without goal achievement. Since generation of sequential percept corresponds to the increase of number of states, speed of learning will be slow down. But, owing to our reasoning process in Sec. 3, a rather fast speed of learning can be expected. Process to Generate sequential percept is summarized in Fig. 8. Finally, overall algorithm including creating of sequential percept is summarized in Fig. 9.

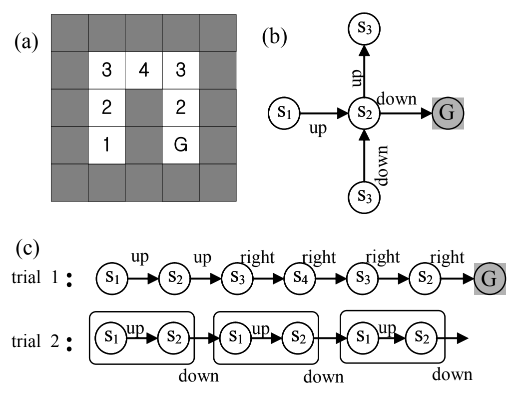
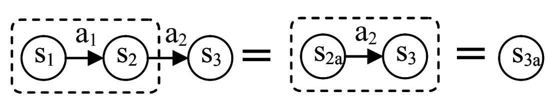


Fig. 6. An example of non-Markov hidden state



$s_{2a} := (s_1 - a_1 \rightarrow s_2)$ $s_{3a} := (s_{2a} - a_2 \rightarrow s_3)$

Fig. 7. Chaining sequential percept

```
LOOP i=0 to sizeSTM
{
   get s1,a1 from index i;
   get s2,a2 from index i+1;
   count=0;
    LOOP j=0 to sizeSTM
    {
      get s1',a1' from index j;
      get s2',a2' from index j+1;
        if ( s1 = s1' and a1 = a1' and s2 = s2')
           count=count+1
    }
   if ( count > threshold )
     createNewSequencialPercept(s1,a1,s2)
```

Fig.8. Pseudo-code to generate Sequential Percept

```
repeat(for each episode)
   initialize STM
   repeat (for each step of episode)
  {
       generateSeqentialPerception()
      choose action and state a,s with respect to V(s,a,sp)
      store a, s to STM
      take action a
      LOOP i = 1 to STMsize
     {
        IF i=STMsize
           V(s_i,a_i,Goal Index)=1
        ELSE
           V(s_i,a_i,s_i+1)=V(s_i,a_i,s_i+1)+η (1-V(s_i,a_i+1,s_i+1))/i^λ
     }
     reasoningProcess()     in fig 4
 }
```

Fig.9. Overall learning algorithm

5. Experimental result

To show the validities of our proposed learning algorithms, two experiments are to be performed, where maze of the H type and M type are employed for Markov environment and non-Markov environment, respectively. In each maze, an ALS(mobile robot) is to be learned to find optimal path from starting location S to the goal G. Here, the robot can move from current location to next location by using one of four actions such as "move to north, east, south and west".

5.1. Learning speed in delayed reward

In this experiment, H type maze is employed as sketched in Fig. 10, where all information are available to the robot, and thus location of robot can be uniquely identified.

That is, a Markov process is here assumed. Fig 11 shows experimental result for the Q-learning and for our proposed RL with a shortest path reasoning mechanism. It is observed from Fig.11 that optimal path of 15 steps could be formed by 28 episodic trials for Q-learning case.

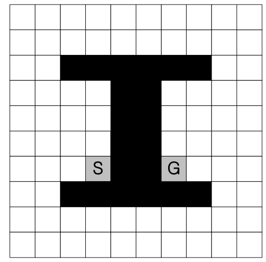


Fig. 10. H type maze

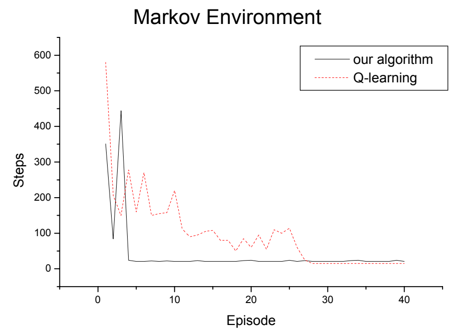


Fig. 11. Experiment results for H type maze

But for the case of our proposed learning algorithm, a sub-optimal path of 20 steps could be found by only 4 episodic trials. Thus our proposed algorithm could enhance speed of learning, but could not get global optimal path due to restriction of searching space

5.2. Learning in non-Markov environment

Consider a maze of M type shown in Fig. 11, where an agent can receive only local state features. And, number on each cell represents the local state feature. For example, cells of number 2 prevent agent from moving to the east and west. Thus, using only number 2 as the feature will make the environment be non-Markov. To cope with non-Markov problem, our agent generates sequential percept in addition to external state feature as explained in sec. 4. Here, we limit the length of sequential percept to be three.

For the exemplar problem in Fig. 12, Q-learning was first applied, but no convergence was obtained. Specifically, in the first trial, a goal was reached. And, an associative learning was done in such a way that state 2 near goal was associated with "down-move" behavior. Therefore, at the next episodic trial, the agent moves up at state S to go to S2, but moves down at state 2 to be back to S, repeatedly. This example simply shows that Q-learning cannot handle such non-Markov hidden state, the agent cannot reach a goal.

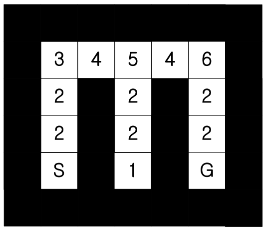


Fig. 12.M-type maze

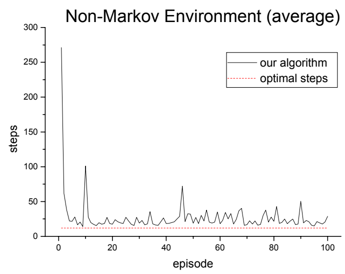


Fig. 13. Learning at non-Markov Environment(average)

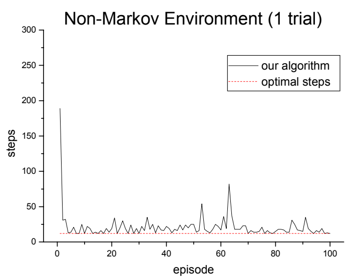


Fig. 14. Learning performance at non-Markov Environment(1 trial)

Simulation result for the application of our proposed algorithm is illustrated in Fig.13, and Fig.14. Fig. 13. shows averaged result of 10 experiments each of which consists of 100 episodic trials. And, Fig.14. shows one of them. After 3rd episode, the result owing to the reasoning processing sec. 3. shows remarkably low steps. But, it does not converge to a global optimum. And in some episodes, the agent needs to get many steps to reach goal. One reason is that sequential percept makes large number of state. Among such sequential percept states, Some states may be on optimal path. But others may be associated with undesirable sequence of actions. For example (6-down)-(2-down)-2 is a unique sequence. But, sequence of (2-down)-(2-left)-2, which is appeared frequently, disturb to reach goal.

6. Conclusions

In this paper, a sequence learning problem has been discussed. Specifically, first, to cope with slow-convergence problem of reinforcement learning algorithm with delayed-reward, some state-action pairs considered as disturbances for an optimum sequence were discouraged in long-term memory (LTM), where such disturbances were found by a shortest path-finding algorithm. This process was shown to let the system get an enhanced learning speed. Second, a sequential percept generation mechanism was proposed to identify what states would be non-Markov hidden states. Validities of our two proposed algorithms were shown by simulations of sequence learning of a goal-finding path in the mazes of H and M type.

References

[1] I.P. Pavolov, *Conditioned Reflexes*, Oxford University Press, 1927.

[2] B.F. Skinner, *Behavior of Organisms* , Appleton-Century-Crofts ,1938.

[3] M.L. Minsky, “Steps towards artificial intelligence”, In Proceedings of the Institute of Radio Engineers, 49, pp8-30, 1961

[4] K. Lorenz, *Foundations of Ethology*, Spronger-Verlag, 1981

[5] B. R. Hergenhahn and M. H. Olson ,*An introduction to theories of learning.* ,6th ed. Prentice Hall, 2001.

[6] E.C. Tolman, *“*A stimulus-expectancy need-cathexis psychology*”*. *Science* 101, 160-166, 1942.

[7] C.Balkenius, “Biological Learning And Artificial Intelligence*”*, University of Rochester 1996.

[8] R. Sutton and A. Barto, *Reinforcement Learning*, MIT Press, 1997.

[9] C.Watkins, “Learning from Delayed Rewards”, PhD thesis, University of Cambridge, England, 1989.

[10] D.S. Touretzky and L.M. Saksida. ,“Operant conditioning in skinnerbots”. *Adaptive Behavior*, 5(3/4) pp 219-247, 1997

[11] R. Sutton , A. Barto, *Reinforcement Learning*, MIT Press, 1997.

[12] L. Kaelbling, M. Littman, A. Moore, “Reinforcement Learning: A Survey” *J. ArtificialIntelligence Research*, vol. 4, pp. 237–285 , 1996

[13] A.K. McCallum , “Reinforcement Learning with selective Perception and Hidden State”, PhD thesis, University of Rochester, 1996.

[14] R. Sun , C. Sessions, “Self Segmentation of Sequences”, *IEEE Trans System Man and Cybernetics, vol. 30. no. 3*, pp.403–418, 2000

[15] Michael Lederman Littman, “Algorithm for Sequential Decision Making”, PhD thesis, Brown University, 1996

[16] S.D. Whitehead and L. J. Lin, “Reinforcement learning in non-Markov environments”*, Artificial Intelligence* , 1993

[17] R.E. Neapolitan, *Foundation of algorithms : using C++ pseudocode*, Jones and Bartlett Publishers, 1998