ISSN : 1226-7244 (Print)
ISSN : 2288-243X (Online)
논문번호 20-04-10

j.inst.Korean.electr.electron.eng.Vol.24,No.4,991~997,December 2020
http://dx.doi.org/10.7471/ikeee.2020.24.4.991
71

Q-learning을 이용한 이동 로봇의 실시간 경로 계획

Real-Time Path Planning for Mobile Robots Using Q-Learning

김 호 원*, 이 원 창**★

Ho-Won Kim*, Won-Chang Lee**★

Abstract

Reinforcement learning has been applied mainly in sequential decision-making problems. Especially in recent years, reinforcement learning combined with neural networks has brought successful results in previously unsolved fields. However, reinforcement learning using deep neural networks has the disadvantage that it is too complex for immediate use in the field. In this paper, we implemented path planning algorithm for mobile robots using Q-learning, one of the easy-to-learn reinforcement learning algorithms. We used real-time Q-learning to update the Q-table in real-time since the Q-learning method of generating Q-tables in advance has obvious limitations. By adjusting the exploration strategy, we were able to obtain the learning speed required for real-time Q-learning. Finally, we compared the performance of real-time Q-learning and DQN.

요 약

강화학습은 주로 순차적인 의사 결정 문제에 적용되어 왔다. 특히 최근에는 신경망과 결합한 형태로 기존에는 해결하지 못한 분야에서도 성공적인 결과를 내고 있다. 하지만 신경망을 이용하는 강화학습은 현장에서 즉각적으로 사용하기엔 너무 복잡하다는 단점이 있다. 본 논문에서는 학습이 쉬운 강화학습 알고리즘 중 하나인 Q-learning을 이용하여 이동 로봇의 경로를 생성하는 알고리즘을 구현하였다. Q-table을 미리 만드는 방식의 Q-learning은 명확한 한계를 가지기 때문에 실시간으로 Q-table을 업데이트하는 실시간 Q-learning을 사용하였다. 탐험 전략을 조정하여 실시간 Q-learning에 필요한 학습 속도를 얻을 수 있었다. 마지막으로 실시간 Q-learning과 DQN의 성능을 비교하였다.

Key words : reinforcement learning, Q-learning, DQN, path planning, mobile robot

Ⅰ. 서론

이동 로봇의 자율주행에 있어 경로 생성은 필수적인 기능이다. 경로 생성을 위한 방법에는 크게 두 가지가 있다. 첫째는 사전에 정보를 알고 있는 경우이고, 둘째는 실제 환경에서 로봇이 지식을 점진적으로 증가시키는 경우이다. 전자의 방법으로는 Dijkstra, A*와 같은 알고리즘이 있고, 후자의 방법

* Dept. of Smart Robot Convergence and Application Engineering, Pukyong National University
** Dept. of Electronic Engineering, Pukyong National University
★ Corresponding author
E-mail : wlee@pknu.ac.kr, Tel : +82-51-629-6219
※ Acknowledgment
This work was supported by a Research Grant of Pukyong National University(2019)
Manuscript received Nov. 26, 2020; revised Dec. 15, 2020; accepted Dec. 18, 2020.

으로는 D*, Incremental A*와 같은 알고리즘이 있다[1-3]. 그러나 위 알고리즘들에는 메모리 문제, 성능의 편차, 경직성 등의 단점이 있다.

알파고의 등장 이후 인공 신경망을 이용한 머신러닝 분야가 크게 주목받게 되었다. 알파고는 머신러닝의 한 종류인 강화학습을 주요한 알고리즘으로 사용한다. 강화학습은 순차적인 의사 결정 문제에 적합하다는 점에서 경로 생성에 적합한 알고리즘이라 할 수 있고, 해당 연구도 활발히 진행되고 있다. 본 논문에서는 대표적인 강화학습 알고리즘인 Q-learning을 사용하여 경로 생성을 진행하였다[4-7]. 일반적으로 Q-table을 생성하는 Q-learning은 상태의 가짓수가 많아지는 경우 모든 상태에 대한 학습 테이블을 저장할 수 없다는 한계를 가진다. 이를 극복하기 위해 신경망을 사용하는 방법이 있으며 대표적인 방법으로는 Deep Q-Networks (DQN)가 있다. 알파고는 DQN을 사용하여 상태의 가짓수가 무한대에 가까운 바둑에서 정상급 인간 프로기사를 이겼다.

본 논문에서는 Q-learning의 한계를 극복하는 방법으로 실시간 Q-learning을 소개한다. 실시간 Q-learning은 단 하나의 Q-table만을 사용함으로써 기존의 방식이 가지는 상태의 가짓수만큼 Q-table이 필요하다는 한계를 극복할 수 있다. 실시간 Q-learning을 사용하기 위해선 가능한 빠른 학습 속도가 필요하다. 이를 위해 Q-learning의 탐험 전략에 따른 학습 속도를 비교해보고, 가장 빠른 학습 속도를 지니는 탐험 전략을 이용하여 실시간 경로 계획을 구현하였다.

II. 배경 이론

1. 강화학습

강화학습은 지도학습, 비지도 학습과 구분되는 머신러닝의 한 방법이다. 강화학습은 환경(environment)과 상호작용 하는 에이전트(agent)를 학습시키는 것을 목표로 한다. 에이전트는 상태(state)를 인지하고 행동(action)한다. 에이전트가 행동함으로써 다른 상태로 천이하며, 환경은 에이전트에게 양(+), 음(-) 또는 0의 보상(reward)을 반환한다. 환경은 확률적이며, 정책에 따라 특정 상태에서 행동을 선택한다. 이러한 강화학습 알고리즘은 MDP(Markov decision process)를 이용하여 수식화할 수 있다. MDP의 경우 다음과 같은 식을 만족한다.

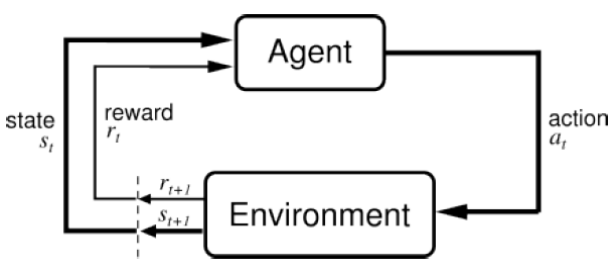

Fig. 1. Agent-Environment interaction.
그림 1. 강화학습 모델

$$\begin{aligned}&\Pr\{R_{t+1}=r, S_{t+1}=s'|S_0, A_0, R_1, \ldots, \\ &\qquad S_{t-1}, A_{t-1}, R_t, S_t, A_t\} \\ &=\Pr\{R_{t+1}=r, S_{t+1}=s'|S_t, A_t RIGHT\end{aligned} \tag{1}$$

2. Q-learning

에이전트의 목표는 episode의 시작부터 종료까지 받는 보상을 최대화하는 것이다. 강화학습에서는 상태 가치함수(state-value function)와 행동 가치함수(action-value Function)를 사용하여 보상을 최대화하는 문제를 해결한다. 상태 가치함수는 아래와 같은 기댓값으로 표현된다.

$$V_\pi(s)=E[R_{t+1}+\gamma R_{t+2}+\gamma^2 R_{t+3}+\ldots|S_t=s] \tag{2}$$

즉, 시간 t에서 정책 π를 따를 때 기대되는 상태 s의 가치는 미래에 받을 보상의 총합으로 표현된다. 식 (2)에서 γ는 [0, 1] 값을 가지는 할인율이다. 할인율은 미래에 받는 보상과 즉각적인 보상의 가치가 다름에서 나온 요소이다.

행동 가치함수는 아래와 같은 기댓값으로 표현된다.

$$\begin{aligned}Q_\pi(s,a)=E[&R_{t+1}+\gamma R_{t+2}+\gamma^2 R_{t+3}+\ldots \\ &|S_t=s, A_t=a]\end{aligned} \tag{3}$$

Q-함수 또는 Q-값이라고도 하며, 상태 가치함수에서 행동이 추가된 모습이다. Q-값은 Bellman 방정식에 따라 아래와 같이 정의할 수 있다.

$$Q(s,a)=r+\gamma \max_{a'} Q(s',a') \tag{4}$$

a'는 다음 상태(s')에서 Q-값을 최대로 만드는 행동이다. Q-learning의 학습 과정은 Q-값을 식 (4)의 Bellman 방정식으로 근사하는 것이다[4]. Q-값은 다음 식을 반복적으로 계산하여 식 (4)에 근

사할 수 있다.

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a)) \quad (5)$$

α는 [0, 1] 값을 가지는 학습률이다. 이전의 값을 얼마만큼 반영할지를 나타낸다. 전통적인 방식의 Q-learning은 충분한 학습을 통해 Q-table을 만들어 에이전트의 정책으로 사용한다. 모든 상태에 대한 Q-table을 만든 뒤 상태를 인지하면 해당하는 Q-table을 불러오는 방식으로 Q-learning을 문제 해결에 적용할 수 있다. 하지만 상태의 가짓수가 너무 많은 경우 모든 Q-table을 저장하는 것은 한계가 있다.

3. DQN

Q-table이 가지는 한계를 신경망을 사용하는 Q-network 알고리즘을 사용하여 극복하려는 노력이 있었다. 하지만 학습 데이터 간의 높은 상관관계 문제와 경사 하강법을 통한 신경망 업데이트 시 목표가 같이 움직이는 문제가 있었다. 딥마인드의 DQN은 경험 재생(experience replay)과 분리된 신경망(separate networks)을 통해 이를 극복하였다 [8-9].

강화학습에서 학습 데이터는 에이전트의 행동에 따른 결과이기에 시간의 흐름이 반영된 순차적인 데이터이다. 이러한 데이터들은 근접한 데이터끼리 높은 상관관계를 띄게 되고 불안정한 학습 결과를 낳게 된다. 딥마인드는 경험 재생을 통해 이문제를 해결했다. 에이전트의 경험 $e_t = (s_t, a_t, r_t, s_{t+1})$를 시간 스텝 단위로 $D_t = \{e_1, \ldots, e_t\}$에 저장한 뒤 균일한 무작위 표본화를 통해 미니 배치를 구성하여 학습을 진행했다.

$$(s,a,r,s') \sim U(D) \quad (6)$$

이런 방법은 입력 데이터 간의 상관관계를 상당히 줄일 수 있었다. 또한, 서로 같은 구조이지만 독립적인 네트워크를 만들어 목표가 움직이는 문제를 해결했다. 학습을 진행할 주 네트워크인 $Q(s,a;\theta)$와 학습의 목표가 되는 네트워크인 $\hat{Q}(s,a;\theta^-)$을 만들어 Q-learning 목표 y_i에 이용한다. 목표 네트워크는 일정 횟수 동안은 업데이트되지 않아 신경망의 업데이트 시 목표가 움직이는 현상을 방지할 수 있다.

$$y_i = r + \gamma \max_{a'} \hat{Q}(s',a';\theta_i^-) \quad (7)$$

$$L_i(\theta_i) = E_{(s,a,r,s') \sim U(D)}[(r + \gamma \max_{a'} \hat{Q}(s',a';\theta_i^-) - Q(s,a;\theta_i))^2] \quad (8)$$

θ_i는 i번째 반복에서의 주 네트워크이고, θ^{-i}는 i번째 반복에서의 목표 네트워크이다.

이러한 방법을 통해 DQN은 훌륭한 성능을 내게 되었지만, 학습 성능을 크게 좌우하는 하이퍼 파라미터는 숙련된 사람의 직관에 좌우되는 경우가 많고, 학습에 높은 컴퓨팅 파워와 많은 시간을 요구한다는 단점이 있다.

4. 탐색과 이용

에이전트는 누적된 보상 값을 최대로 만들기 위하여 탐색과 이용 반복한다. 탐색은 미지의 보상을 찾기 위해 행동하는 것이고, 이용은 알려진 보상을 받기 위해 행동하는 것이다. 탐색하지 않는 에이전트는 보상 값을 최대로 만들지 못할 수 있고, 탐색만 하는 에이전트는 경험을 이용하지 못해 비효율적이다. 이러한 문제를 exploration-exploitation tradeoff dilemma라 한다. 에이전트는 적절한 탐험 전략을 통해 학습 속도를 높일 수 있다. 본 논문에선 다음과 같은 네 가지 경우를 생각한다.

- Random : 에이전트가 무작위로만 움직이는 순수한 탐색
- Greedy : 에이전트가 알려진 더 큰 보상을 향해서만 움직이는 순수한 이용
- ϵ-greedy : 에이전트는 greedy 하게 움직이지만, ϵ만큼의 확률로 random으로 움직임
- Decaying ϵ-greedy : 기본적으로 ϵ-greedy와 같지만, 시간이 지날수록 ϵ이 줄어듦

본 논문에선 탐험 전략을 비교하기 위해 전략과 파라미터를 바꿔가며 결과를 비교한다.

III. 시뮬레이션 및 결과

모든 상태에 대한 Q-table을 구하고 저장하는 것은 현실적으로 불가능함으로 Q-learning을 문제 해결에 사용하기는 쉽지 않다. 이를 극복하기 위해 DQN과 같은 알고리즘을 쓸 수 있으나 간단한 문제에 적용하기엔 상대적으로 큰 비용을 요구한다. 따라서 본 논문에선 Q-learning의 한계를 극복하기 위해 미리 Q-table을 만드는 것이 아닌 상태를 줬을

때 실시간으로 Q-table을 만드는 실시간 Q-learning을 구현하였다. 실시간으로 Q-table을 만들 수 있는 학습 속도를 얻기 위해 탐험 전략에 따른 학습 속도를 비교해보았으며, 시뮬레이션을 통해 얻은 최적 전략을 사용하여 DQN과 성능을 비교해보았다.

1. 시뮬레이션 방법

본 논문은 비영리 인공지능 연구회사인 OpenAI에서 제공하는 강화학습 개발도구 Gym을 사용하여 시뮬레이션 환경을 구현하였다. 이 시뮬레이션은 이동 로봇이 무작위로 주어진 시작 지점에서 출발하여 장애물을 회피하며 무작위로 주어진 목표 지점으로 이동하는 상황이라 가정한다. 에이전트는 동, 서, 남, 북 네 방향으로만 움직일 수 있으며 움직일 때마다 step이 1 증가한다. 장애물에 충돌할 시 음(-)의 보상을 받고 시작 지점으로 돌아간다. 에이전트가 목표 지점에 도착할 시 양(+)의 보상을 받고 episode가 끝난다. 에이전트가 Q-table을 완성할 만큼 충분한 횟수의 episode를 수행하면 새로운 상태에서 다시 학습한다. 이를 case라고 한다. 에이전트는 정해진 횟수의 case를 학습하며 모든 case에서 나온 값의 평균을 결괏값으로 사용한다.

Table 1. ϵ and rewards according to strategy.
표 1. 전략에 따른 ϵ과 보상

	ϵ	이동시 보상	장애물 보상	목적지 보상
1	0	0	-1	+1
2	0	-1	-10	+10
3	0.3->0	0	-1	+1
4	0.3->0	-1	-10	+10
5	0.3	0	-1	+1
6	0.3	-1	-10	+10

Q-learning에서 학습 속도에 영향을 미치는 요소는 탐색의 정도를 나타내는 ϵ과 환경에서 받는 보상이 있다. ϵ은 0.3으로 시작해서 episode마다 ϵ에 0.99를 곱해주는 것이 흔히 사용된다. 보상은 성공시 양(+), 장애물에 부딪힐 시 음(-)을 주는 것은 공통이고, 그 외의 경우에 0의 보상을 주거나 음(-)의 보상을 주는 방법으로 나뉜다. 테스트한 전략은 총 여섯 가지로 표 1과 같다.

1, 5, 6번의 경우 3,500 episode, 40,000 case 동안 유의미한 학습 결과를 내지 못하여 학습을 중단하였다. 2, 3, 4번의 경우 학습이 되는 것이 확인돼 3,500 episode, 400,000 case를 학습하였다.

2. 시뮬레이션 환경 구현

환경은 일반적으로 사용되는 그리드 월드이다. 그리드 월드는 다음과 같은 조건을 만족한다.

- 가로 = m, 세로 = n
- 장애물 개수 = (m x n) / 4
- 시작 지점과 목표 지점의 최소 거리 = (m + n) / 2
- 학습을 할 수 없는 공간은 없어야 한다.
- 시작 지점과 목표 지점은 연결돼야 한다.

그림 2는 허용되지 않는 상태의 예시이다. 에이전트는 검은색, 장애물은 빨간색, 목표 지점은 파란색이다. 왼쪽 위 상태부터 시계방향으로 첫 번째 그림은 장애물로 둘러싸여 학습할 수 없는 공간이 있는 경우이며, 두 번째 그림은 장애물과 목표 지점으로 둘러싸여 학습할 수 없는 공간이 있는 경우를 나타낸다. 세 번째 그림은 시작 지점과 목표 지점이 너무 가까운 경우이고, 마지막, 네 번째 그림은 시작 지점과 목표 지점이 이어지지 않는 경우이다. 첫 번째, 두 번째 및 세 번째 그림은 학습 시간을 단축해 왜곡을 일으킬 수 있다고 생각해 허용하지 않았으며, 네 번째 그림은 학습이 불가능하므로 허용하지 않았다.

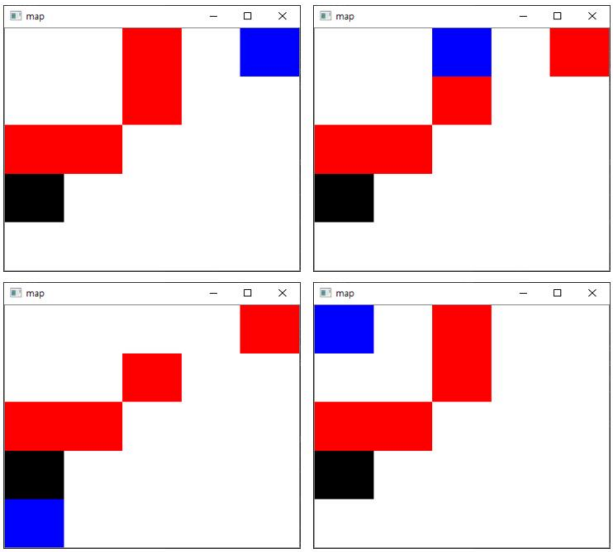

Fig. 2. Unacceptable states.
그림 2. 허용되지 않는 상태

시뮬레이션은 10 × 10 그리드 환경에서 진행하였다. 장애물의 개수는 25개이다. 장애물이 없는 경우 최소거리는 10이고 최대거리는 18이다. 장애물을 고려했을 때 시작 지점과 목표 지점의 거리의 기댓값을 15~16이라 할 수 있다.

3. 시뮬레이션 결과

전략에 따른 성능을 그래프로 나타내었다. 그림 3은 한 episode를 끝내는데 몇 번의 step이 있었는지 나타내는 그래프이다. 그림 4는 한 case를 끝내는데 총 몇 번의 step이 있었는지 나타내는 그래프이다. 전략 3, 5는 total step이 너무 많아 그래프 밖으로 벗어났다. 그림 3에서 전략 1, 3, 5가 2, 4, 6보다 step이 적어 보이지만 그림 4를 확인하면 그렇지 않다는 것을 알 수 있다.

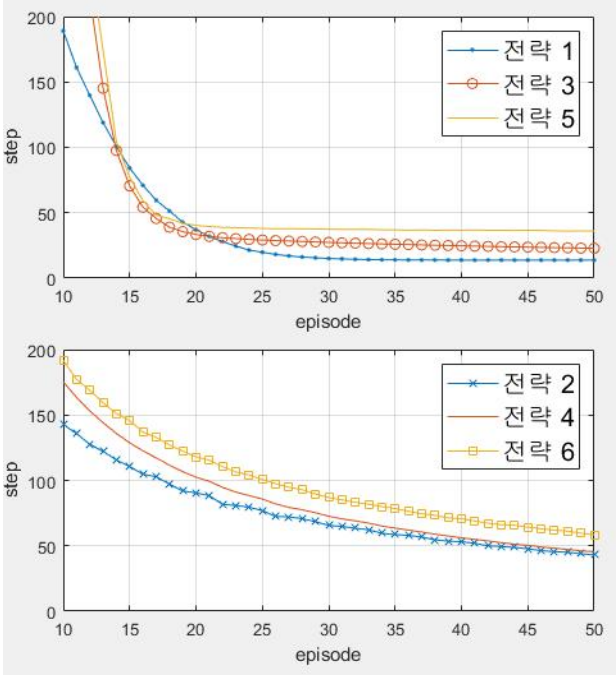

Fig. 3. Episode-step graph.
그림 3. Episode-step 그래프

그림 4를 확인했을 때 가장 빠른 학습 속도를 보인 전략은 전략 2이다. 전략 2는 ϵ이 0이고 step마다 음의 보상을 받는다. 식 (5)의 Q-learning 업데이트 식을 보면 초기에 모든 Q-값은 0으로 시작해서 이동 시 음의 보상을 받으면 그 Q(s, a)의 값이 Q-값 전체에서 가장 낮은 값이 됨을 알 수 있다. 이러한 점을 고려했을 때 음의 보상이 방대한 탐색을 끌어낸 것으로 보인다. ϵ과는 다르게 중복 없이 탐색하게 되기에 더 빠른 학습 속도를 보인 것으로 생각된다. 전략 2는 episode 당 step이 15 이하로 내려가는데 총 10,000번 이하의 step이 필요하였다. 전략 2를 최적 전략이라 생각하고 실시간 Q-learning을 시뮬레이션해보았다.

실시간 Q-learning에서 에이전트는 일정 시간 학습한 후 움직이기를 반복한다. 10,000 step을 시행하는데 0.5초면 충분하기에 에이전트의 step마다 0.5초의 학습 시간을 주었다. 에이전트는 1~2번의 추가 step 이내로 목표 지점에 도착하였고, 이동 도중에 상태가 바뀌는 경우도 실시간적으로 경로 계획을 수정하는 모습을 보여주었다.

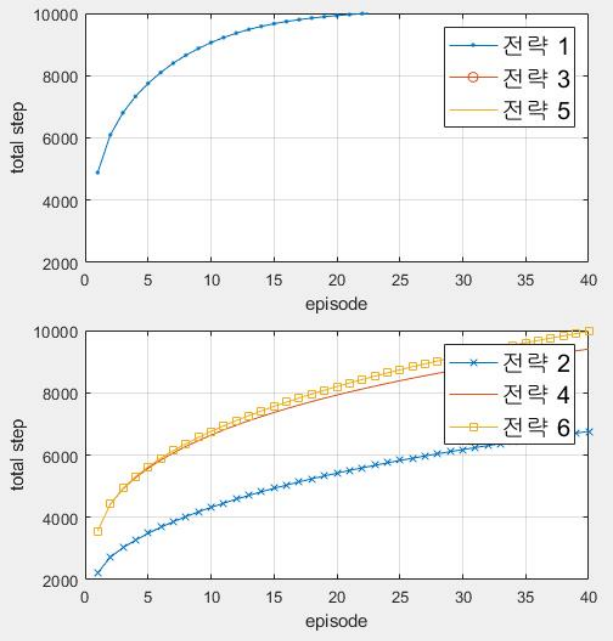

Fig. 4. Episode-total step graph.
그림 4. Episode-total step 그래프

4. 실시간 Q-learning과 DQN의 성능 비교

실시간 Q-learning과 DQN의 성능을 비교하기 위해 간단한 환경인 Cliff 환경과 복잡한 환경인 10 × 10 그리드 환경일 때 각각 비교해보았다. 1 episode마다 경로 찾기를 진행해 성능을 비교하였다. 복잡한 환경에서는 총 100개의 case를 학습했다. 경로 찾기의 성공과 실패는 step 수로 구분하였다. 50 step이 넘으면 경로 찾기에 실패한 것으로 간주하였다. 복잡한 환경에서 학습 시 1 episode의 최대 step 수는 5,000번으로 제한하였고, 10번 연속으로 경로 찾기에 성공하면 해당 case는 경로 찾기에 성공한 것으로 간주하였고 100 episode 동안 경로 찾

기에 실패하면 해당 case의 경로 찾기는 실패한 것으로 간주하였다.

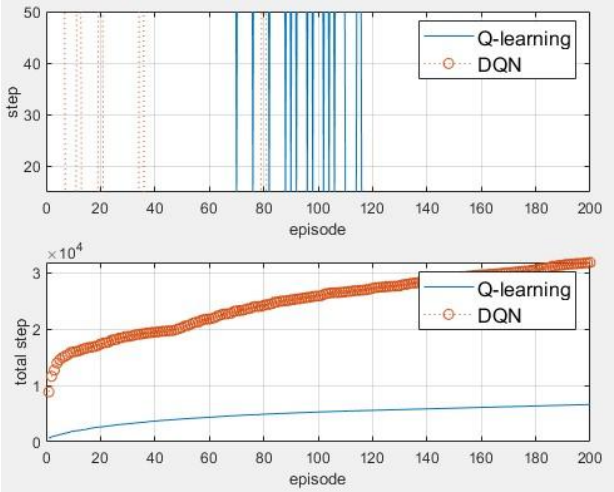

Fig. 5. Episode-step graph of Cliff envirenment.
그림 5. Cliff 환경의 episode-step 그래프

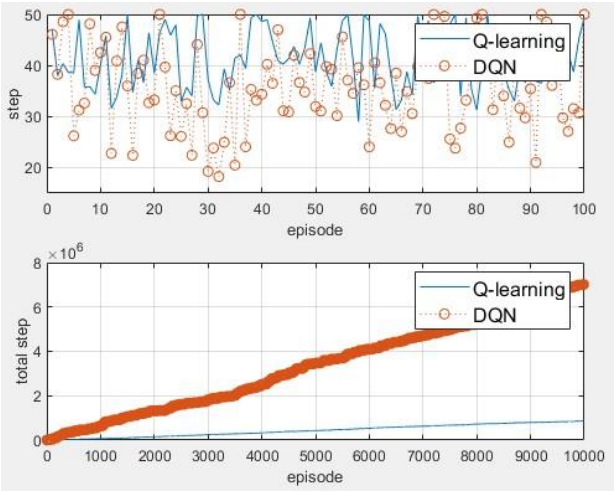

Fig. 6. Episode-step graph of Random environment.
그림 6. Random 환경 episode-step 그래프

두 경우 모두 학습 속도는 Q-learning이 우수한 것으로 나왔다. 간단한 환경에서는 3배 이상 total step이 적게 나왔고 복잡한 환경에서는 7배 이상 total step이 적게 나왔다. Q-learning과 비교해 DQN이 더 많은 연산을 사용한다는 점에서 실제 학습 시간은 더 많이 차이 나게 된다. 하지만 정확도는 DQN이 더 우수한 것으로 보인다. 그림 6을 보면 성공 비율은 DQN이 74%, Q-learning이 61%로 21.3% 높았고, step은 DQN이 35.9 Q-learning이 41.5로 13.5% 더 적게 나왔다.

Ⅳ. 결론

본 논문에서는 실시간 경로 생성을 위해 실시간 Q-learning을 사용하였다. 실시간 Q-learning에 필요한 학습 속도를 얻기 위해 Q-learning의 전략에 따른 학습 속도도 비교하여 보았다. 시뮬레이션 결과 가장 나은 전략은 ϵ이 0이고, 이동 시 음의 보상을 받는 것이었다. 일반적으로 강화학습에서 ϵ이 매우 중요하게 여겨지는 것과 다른 결과였다. ϵ이 탐색을 위해 사용되는 것을 생각했을 때, 이동 시 음의 보상을 주는 것이 그리드 월드에서 탐색을 유도한 것으로 보인다. 다른 환경에서도 보편적으로 쓰일 수 있을지는 추가연구가 필요하다고 판단된다. 이 전략을 사용하여 실시간 경로 생성에 실시간 Q-learning을 사용하였을 때, 목표 지점에 도착하는 데 무리가 없었다. 또한, 이동 도중 상태가 바뀌는 경우도 같은 결과를 보였다. 시뮬레이션 환경의 상태가 10 × 10의 그리드 월드에 25개의 장애물, 시작 지점, 목표 지점이 무작위로 생기는 것을 고려했을 때, 최대 1.9×10^{24}개의 상태가 나와 Q-learning으로 해결할 수 없을 것으로 생각되지만, 실시간 Q-learning을 사용하는 경우 문제를 해결할 수 있었다. 또한, 30 × 30의 그리드 월드에서도 동작하는 것을 확인했다. 마지막으로 실시간 Q-learning과 DQN의 성능을 비교했을 때 DQN의 정확도가 더 우수한 것으로 나왔지만 학습 속도는 실시간 Q-learning이 더 우수함을 보였다. 또한 위에서 언급한 것처럼 실시간 Q-learning의 경우 10 × 10 환경에서 30 × 30 환경으로 즉시 변환하여 사용할 수 있었으나 DQN은 새로운 환경을 학습시키기 전까진 사용할 수 없다.

종합적으로 고려했을 때 기존의 Q-learning으로 해결할 수 없는 것으로 알려져서 과도한 성능의 알고리즘을 사용하고 있는 영역이 있다면 높은 유연성과 낮은 비용을 사용하는 실시간 Q-learning은 이러한 영역에 사용할 수 있는 대안 알고리즘으로 보인다.

References

[1] H. T. Cormen, C. E. Leiserson, R. L. Rivest, and Clifford Stein, *Introduction to Algorithms,.*

Second Edition, MIT Press and McGrawHill, 2001.
[2] S. Koenig and M. Likhachev, "D* lite," *National Conference on Artificial Intelligence,* vol.18, pp. 476-483, 2002.
[3] S. Koenig and M. Likhachev, "Incremental A*," *Advances in Neural Information Processing Systems,* vol.14, pp.1539-1546, 2002.
[4] R. Sutton and A. Barto, *Reinforcement learning,* MIT Press, 1996.
[5] Y. Li, C. Li and Z. Zhang, "Q-Learning Based Method of Adaptive Path Planning for Mobile Robot," *IEEE International Conference on Information Acquisition*, pp.983-987, 2006.
DOI: 10.1109/ICIA.2006.305871
[6] D. Tamilselve, S. M. Shalinie and G. Nirmala, "Q Learning for Mobile Robot Navigation in Indoor Environment," *IEEE International Conference on Recent Trends in Information Technology*, pp.324-329, 2011.
DOI: 10.1109/ICRTIT.2011.5972477
[7] J. Muhammad and I. O. Bucak, "An Improved Q-Learning Algorithm for an Autonomous Mobile Robot Navigation Problem," *2013 TAEECE*, pp. 239-243, 2013.
DOI: 10.1109/TAEECE.2013.6557278
[8] Mnih, Volodymyr, et al, "Playing Atari with Deep Reinforcement Learning," *NIPS Deep Learning Workshop 2013,* pp.1-9, 2013.
[9] Mnih, Volodymyr, et al, "Human-level control through deep reinforcement learning," Nature, Vol.518, No.7540, pp.529-533, 2015.

BIOGRAPHY

Ho-Won Kim (Member)

2020 : BS degree in Electronic Engineering, Pukyong National University.
2020~present : MS student in Smart Robot Convergence and Application Engineering, Pukyong National University.

Won-Chang Lee (Member)

1983 : BS degree in Instrumentation and Control Engineering, Seoul National University.
1985 : MS degree in Electrical and Electronic Engineering, KAIST.
1992 : PhD degree in Electronic and Electrical Engineering, POSTECH.
1993~present: Professor, Pukyong National University