한국게임학회 논문지
제7권 제4호 2007년 12월

예측 정보를 이용한 Q-학습의 성능 개선 기법

이충현[0], 조경은*, 엄기현**
㈜ Wisdomain[0], 동국대학교 영상미디어대학 게임멀티미디어공학과*,**
{tgleech[0], cke*, khum**}@dongguk.edu

A Strategy for improving Performance of Q-learning with Prediction Information

Choonghyeon Lee[0], Kyungeun Cho*, Kyhyun Um**
Wisdomain Co., Ltd.[0], Dept. of Game & Multimedia Engineering, Dongguk University*,**

요 약

게임 환경에서의 학습은 다양한 분야에서 유용하게 활용될 수 있다. 그러나, 학습이 게임에서 만족스러운 결과를 산출하기까지는 많은 학습 시간이 요구된다. 이러한 점을 개선하기 위하여 학습시간을 단축시킬 수 있는 방법론들이 필요하다. 본 논문에서는 예측 정보를 이용한 Q-학습의 성능개선 방안을 제안한다. Q-학습 알고리즘에서는 Q-테이블의 각 상태별 선택된 액션을 참조한다. 참조한 값은 예측 모듈의 P-테이블에 저장되고, 이 테이블에서 출연 빈도가 가장 높은 값을 찾아 2차 보상 값을 갱신할 때 활용한다. 본 연구에서 제시한 방법은 상태내의 전이가 가능한 액션의 수가 많을수록 성능이 높아짐을 확인하였다. 또한 실험결과로 실험 중반 이후부터 제안한 방식이 기존 방식보다 평균 9%의 성능 향상을 보였다.키워드: 실감형 게임, 모바일 게임, SIFT, 물체 인식.

Abstract

Nowadays, learning of agents gets more and more useful in game environments. But it takes a long learning time to produce satisfactory results in game. So, we need a good method to shorten the learning time. In this paper, we present a strategy for improving the learning performance of Q-learning with prediction information. It refers to the chosen action at each status in the Q-learning algorithm. It stores the referred value at the P-table of prediction module, and then it searches some values with high frequency at the table. The values are used to renew second compensation value from the Q-table. Our experiments show that our approach gets the efficiency improvement of average 9% after the middle point of learning experiments, and that the more actions in a status space, the higher performance.

Keyword : Reinforcement learning, Q-learning, Prediction information

※교신저자(Corresponding Author):조경은,
주소:서울시 중구 필동 3가 26 동국대학교(100-715), 전화:02)2260-3834, E-mail:cke@dongguk.edu

1. 서론

1.1 연구의 배경 및 필요성

게임 내 에이전트에게 학습 능력을 부여하는 것이 오늘날 게임의 트랜드가 되고 있다. 에이전트의 AI가 지능적인 적응성을 갖추고 있다면 과거의 게임처럼 특정 기술 몇 가지만으로도 충분히 공략할 수 있었던 시대와는 다르게 끊임없이 새로운 기술이나 전략을 찾아서 공략해야 할 것이다. 그 결과 플레이어는 게임에 좀 더 많은 시간을 투자하게 되며, 그 과정에서 또 다른 흥미가 유발된다[1].

학습 능력은 개발자와 플레이어 모두에게 많은 혜택을 제공한다. 사람이 직접 풀기 어려운 문제를 학습 알고리즘의 사용으로 해결하거나 게임의 출시 이전에 예측할 수 없었던 플레이어의 플레이 스타일, 취향, 성격 등을 게임 내 에이전트가 적응하도록 만드는데 쓰일 수 있다 [2,3]. 전자의 경우 사람의 감독이 거의 필요하지 않으며, 후자의 경우는 플레이어의 플레이 스타일 예측을 위한 레벨 디자이너의 디자인 과정이 생략되어 개발 기간의 단축을 꾀할 수 있다. 또한 다양한 플레이 스타일로 게임을 플레이 할 때, 각각의 플레이 스타일에 따른 대응 전략이 달라져 플레이어가 상호 적응적인 플레이를 할 수 있게 도와줌으로써 게임에 대한 집중도와 흥미 유발을 높일 수 있다[4,5].

학습 능력의 효율을 극대화하는 이유는 다음과 같다. 먼저 개발자 측면에서는 에이전트의 행동 스타일을 다양한 방향으로 훈련하는 기간을 단축하고 게임 내 에이전트 레벨 밸런싱 기간의 단축으로 개발기간을 단축하는 효과를 가져올 수 있다. 다음으로 플레이어 측면에서는 학습 능력의 효과는 '블랙 앤 화이트'와 같은 플레이어의 스타일에 맞춰가는 게임의 경우에서 쉽게 볼 수 있다. 즉, 플레이어의 게임스타일에 적응하는 속도를 빠르게 하여 플레이어에게 만족도를 높일 수가 있다.

1.2 연구의 목적

본 논문에서는 현재 그 활용 범위가 가장 넓은 강화 학습의 학습 능력을 극대화하는 방안을 제안한다. 강화 학습에서 널리 사용되는 Q-학습 알고리즘을 기반으로 예측 정보를 이용하여 최적의 결과를 빠르게 산출할 수 있도록 한다. 기존 Q-학습의 강화 값(이하 Q 값)이 들어있는 Q-테이블을 참조하여 예측한 값을 또 하나의 Q 값 즉, 2차 Q 값을 생성한다. 1회 학습 시 두 번의 Q 값이 갱신된다. 단, 2차 Q 값의 생성 과정은 기존 방식의 Q 값 생성시보다 프로세스 부담이 적어야 한다. 여기서 학습 능력의 향상은 예측 정보의 정확도에 따라 판가름된다. 예측 정보의 정확도를 높이기 위해서는 그 만큼의 계산이 필요하므로 자칫 기존 방식보다 효율이 떨어질 수 있는 문제점을 안고 있다. 적당한 정확도와 효율을 높이기 위한 보완이 필요하다. 이에 대한 해결책으로 예측 정보를 만들기 위한 통계 자료는 Q-테이블을 공유하며 소량의 정적인 형태로 정의한다. 이를 위한 최적의 공유 범위에 대한 연구도 수반되어야 한다. 그로 인한 2차 Q 값의 생성 비용 절감으로 본 논문이 제안한 연구의 목적인 학습 효율의 극대화를 꾀할 수 있다.

본 연구의 구성은 다음과 같다. 2 장 관련 연구에서는 강화 학습에서의 효율을 극대화하기 위한 다양한 방법에 대해 기술한다. 3 장에서는 제안하는 예측 정보를 이용한 Q-학습의 성능 개선 방안이 수행되는 방법과 구현 설계 및 적용 사례를 설명한다. 4 장 실험 및 분석에서는 기존 방식과 제안하는 방식에 대한 성능 예측으로 모의실험을 통해 성능을 예측하고 분석한다. 마지막 5 장에서는 결론과 향후 계획을 제시한다.

2. 관련연구

Q-학습의 학습 효과를 높이기 위한 관련 연구들을 기술한다. Q-학습의 학습 효과를 높이기 위한 연구는 상태공간의 축소문제와 학습시간의 단축 문제 등이 있다.

본 논문에서는 강화 학습에서의 학습 효율을 극대화하기 위한 학습 시간의 단축을 위한 기법을 연구한다. 학습의 시간 단축을 목표로 한 기존 연구는 다음과 같다.

ONRELS(Online Reinforcement Learning to Search the Shortest Path in Maze Environments)는 미로 환경에서 최단 경로를 빠르게 탐색할 수 있는 실시간 강화 학습 시스템이다. ONRELS는 현재 상태에서 상태 전이를 하기 전에 선택 가능한 모든 (상태-행동) 쌍에 대한 평가값을 갱신하고 나서 상태 전이를 한다. ONRELS는 미로 환경의 상태 공간을 압축하고 나서 압축된 환경과 시행-착오를 통해 상호 작용하면서 학습을 수행한다[6].분포 기억도를 이용한 Q-학습에서는 학습시간을 단축하기 위해서 한 번의 학습으로 하나 이상의 Q-값이 갱신되는 분포 기억도 방식을 제안하였다

[7]. 분포 기여도를 이용해서 인접한 Q-값에 기여정도에 따라서 Q-값을 갱신한다. 영향력분포도를 이용한 Q-학습[8]에서는 Q-학습의 이산 상태 중간값을 영향력분포도를 이용해서 채우는 방식을 사용하였다.

위 논문들은 본 논문의 제안과 마찬가지로 강화 학습의 효율을 높이는데 목적을 두고 있다. 특정한 환경 또는 상태에 제약을 걸고 그에 맞게 알고리즘 전체에 변형을 가한다. 하지만, 본 논문의 제안은 이와 다르게 응용 범위가 넓다. 환경과 환경에서 얻어지는 정보를 바탕으로 행동을 결정하고 그에 따라 얻어지는 보상이 있는 경우라면 거의 모든 경우에서 응용이 가능하다.

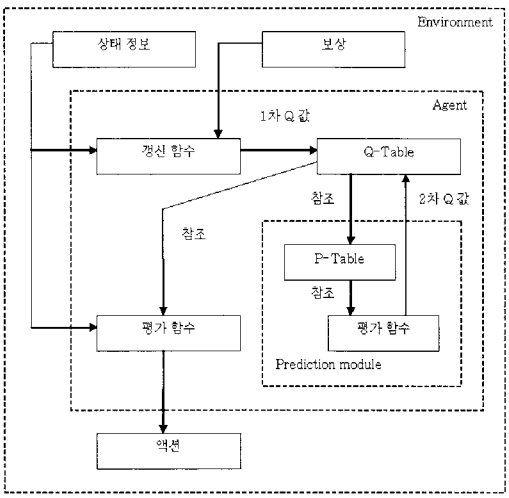
본 논문의 제안은 위에서 제시한 논문들과는 다르게 선택기와 학습기 등으로 강화 학습의 구조 자체를 바꾸지 않는다. 예측 정보 산출 모듈을 기존의 학습 알고리즘에 삽입하는 것으로 적용이 가능하다. 이러한 이유로 본 논문의 제안은 위 논문들과는 다른 차별성을 가지고 있다.

3. 예측 정보를 이용한 Q-학습 기법

학습은 에이전트가 속해 있는 환경에서 자신의 위치 또는 그 외의 상황에서 가장 좋은 결과를 얻기 위해 필요하다. 예를 들면 실시간 FPS 게임의 경우 엄폐물의 위치, 개수, 종류, 방향, 적의 위치, 주변 건물 배치 등 외부에서 입력 받을 수 있는 모든 정보를 상태 정보로 규정한다. 상태 정보를 입력받은 강화 학습 알고리즘은 현재 상태에서 가장 적합한 행동을 산출하게 된다. 이것을 행동의 정책이라 한다. 산출한 정책이 가장 좋은 결과를 얻을 수 있도록 이미 행동한 액션에 의한 결과의 좋고 나쁨을 판단한다. 이것은 액션에 대한 보상값을 결정하는 기준이 된다. 행동한 액션으로 인한 결과가 좋으면 양의 값을 보상하고, 나쁘면 음의 값을 보상하게 된다. 이렇게 같은 상황을 되풀이하고 현재 상황에서 다양한 액션을 결정한다. 그 결과에 대한 보상값을 정책의 결정에 반영하게 되면서 가장 좋은 액션을 선택할 수 있도록 적응하게 된다. 이와 같은 방식으로 강화 학습은 이루어진다.

3.1 예측 정보를 이용한 Q-학습의 기본 구조

본 논문에서는 예측 정보를 산출하기 위해 Q-학습 알고리즘에서 Q-테이블의 자료를 참조한다. 참조는 Q-테이블에 양의 보상값이 갱신될 때 이루어진다. 이 때 양의 보상값을 받은 액션은 예측 정보를 산출하는 예측 모듈의 새로운 테이블에 저장된다. 테이블의 이름은 P-테이블로 규정한다. 이렇게 저장된 액션 중 출연 빈도의 통계를 계산하고 이 중 가장 확률이 높은 액션을 다시 Q-테이블에 양의 보상값을 받을 액션으로 갱신한다. P-테이블은 정적으로 구성된다. 큐(queue)와 같은 구조를 갖는다. 일정 크기 이상 저장되면 가장 먼저 받은 값은 삭제되어 하나씩 앞으로 이동한다. 테이블은 계속 새로운 값으로 갱신된다. Q-테이블의 값이 학습을 거듭할수록 정확도가 높아지므로 테이블을 참조해서 얻은 통계 자료 역시 정확도가 높아진다. 예측 모듈은 Q-학습 알고리즘 내 처리 과정의 한 부분으로 삽입된다. 예측 모듈의 처리 시간이 기존 방식의 2회 학습 시간보다 길어지면 본 논문의 타당성이 낮아질 수 있다. 그러므로 예측 모듈의 처리 시간을 줄이기 위해 P-테이블의 크기를 적절하게 정해야 한다. 테이블의 크기를 너무 줄일 경우 통계로 얻은 값의 신뢰성이 떨어지게 되므로 처리시간을 줄이는 것과 테이블의 크기를 어느 정도로 결정하느냐에 따라 예측 모듈의 성능이 좌우된다.



[그림 3-1] 제안 방식의 기본 구조

제안 방식의 기본 구조는 [그림 3-1]과 같다. 환경에서 얻은 상태 정보를 에이전트 내의 학습 알고리즘에서 입력받는다. 이 정보는 갱신 함수와 평가 함수에서 전달된다. 갱신 함수에서는 입력 받은 상태정보를 가지고 Q-테이블을 갱신할 준비를 한다. 평가 함수는 입력 받은 상태 정보에 최상의

액션을 결정하기 위해 Q-테이블의 값을 참조한다. Q-테이블은 각각의 상태와 액션의 쌍에 대한 자세한 값을 가지고 있다. 자세히 말하면 각 상태에서 행동한 액션에 의해 나타난 결과에서 얻은 보상값을 누적하고 있다. 액션마다 다른 보상값을 누적해서 가지고 있다. 이 값을 참조하여 평가 함수는 현재 상태의 행동할 액션을 결정한다. 예측 모듈은 Q-테이블에서 양의 보상값을 갱신할 때 갱신한 액션을 P-테이블에 저장한다. 이 테이블에 저장된 액션 중 출연 빈도가 가장 높은 액션을 결정해 Q-테이블에 갱신할 2차 Q 값(양의 보상값)을 받는 액션으로 지정한다. 이 과정을 반복하여 학습을 수행하게 된다.

3.2 예측 정보의 산출

3.2.1 예측 정보 산출 모듈의 기본 개념

예측 정보 산출 모듈은 Q-테이블 내의 정보를 참조하고 각 상태의 액션의 출연 빈도를 예측하기 위해 P-테이블에 저장한다. 이것은 각 상태의 양(positive)의 Q 값을 받은 액션을 저장한다. 2차 Q 값을 생성하기 위해 P-테이블에 저장된 값 중 출연 빈도가 가장 높은 액션을 검색한다. 검색된 액션이 1개일 경우에만 2차 양의 Q 값을 받는 액션으로 인정한다. 2개 이상의 액션이 검색된 경우 2차 Q 값의 신뢰성이 떨어질 수 있으므로 2개 이상의 경우에는 2차 Q 값을 무시한다.

```
// 보상값 저장할 배열 정적 선언
// 상태에서 전이 가능한 액션 만큼의 배열 선언
const int maxStates = 10;    // 전체 상태
const int maxActions = 20;   // 각 상태별 입력 받는 최대 액션
// actionType = 상태에서 전이 가능한 액션(ex 상, 하, 좌, 우 이동)
const int actionType = 4;
int pTable[maxStates][maxActions];
// nowPosition = pTable에 입력받은 값의 현재 위치
int nowPosition = 1;

int Prediction(int state, int action, int reward){
   int temp[actionType];
   int topScore;
   int sameScore;          // 가장 높은 값의 액션이 두 개 이상인지 체크
   int i;
   for(i=0;i<actionType;i++){
      temp[i] = 0;
   }
   if(reward > 0){
      // 맨 앞의 값 삭제 후 한 자리씩 앞으로 이동 후 action 삽입
      for(i=maxActions-nowPosition; i<maxActions; i++){
         if(i<maxActions-1){
            pTable[state][i] = pTable[state][i+1];
         }
         if(nowPosition > 1)
            temp[pTable[state][i]]++;
      }
      pTable[state][i-1] = action;
      topScore = 0;
      sameScore = -1;
      for(i=1; i<actionType; i++){
         if(temp[i] > temp[topScore]) topScore = i;
         else if(temp[i] == temp[topScore])
                        sameScore = topScore;
      }
      // 입력 받은 값만큼만 통계를 내기 위해 현재 위치 설정
      if(nowPosition < maxActions) nowPosition++;
      // 가장 높은 액션이 두 개 이상일 경우 음의 값 리턴
      if(sameScore == topScore) return -1;
      else if(nowPosition > (maxActions/100)*30)
                        return topScore;
      else return -1;
   }else{
      return -1;
   }
}
```

[그림 3-2] 예측 정보 산출 알고리즘

[그림 3-2]는 예측 정보 산출 알고리즘이다. 예측 정보를 산출하는 알고리즘의 진행 순서는 다음과 같다.

1) P-테이블의 사이즈를 정의할 상태와 액션의 최대값을 정의한다.
2) 상태에서 전이 가능한 액션에 대한 경우의 수를 정의한다.
3) P-테이블을 정의한다.
4) P-테이블에 입력 받은 값의 현재 위치를 나타내는 변수 nowPosition을 정의한다.
5) 실제 예측 정보를 산출하는 Prediction 메소드를 정의한다.
6) 이 메소드는 상태(state), 액션(action), 보상(reward)값의 총 3개의 인자 값을 받는다.
7) reward가 양의 값일 때 다음의 계산을 수행한다.
8) P-테이블의 값이 계속 새로운 값으로 갱신되도록 P-테이블의 입력받은 값이 꽉 차면 가장 먼저 받은 값을 삭제하고 한자리씩 자리를 이동한다. (queue 구조로 구성한다.)
9) 임시 배열 변수 temp에는 입력 받은 액션의 출연 빈도를 체크하기 위해 P-테이블의 액션값을 누적한다.
10) 자리 이동을 마친 P-테이블의 끝 값에 action 인자 값을 입력한다.
11) temp 값을 비교해 가장 큰 값을 찾는다. 또한 가장 큰 값이 두 개 이상 인지도 체크한다.

12) 액션이 두 개 이상 검색된 경우이거나 P-테이블 전체 크기의 30% 이하일 경우에는 예측 정보로 사용될 값의 신뢰성을 높이기 위해 액션의 리턴 값을 반환 하지 않는다.

3.2.2 예측 모듈의 2차 Q 값 생성 방법

이 절에서는 예측 모듈의 2차 Q 값 생성 방법을 설명한다. 예를 들면, [그림 3-3]에서 A, B, C, D는 상태에서 전이 가능한 액션의 경우의 수이다. 각 상태별로 양의 보상값을 받은 액션들을 P-테이블에 저장할 때 A, B 등으로 표현한다.

상태1	ABAACBABCAC ... ACB	A 40% , B 30% , C 30%
상태2	DACADBBDCCA ... DAC	A 15% , B 15% , C 35% , D 35%
상태3	ABBABABBAAB ... ABB	A 40% , B 60%

A - 액션1
B - 액션2
C - 액션3
D - 액션4

[그림 3-3] P-테이블 갱신 과정 1

상태1	ABAACBABCAC ... ACB
상태2	DACADBBDCCA ... DAC
상태3	ABBABABBAAB ... ABB

↓

상태1	BAACBABCACA ... CBA
상태2	ACADBBDCCAD ... ACD
상태3	BBABABBAABA ... BBA

↓ 새로운 액션이 삽입되면 앞에서부터

상태1	AACBABCACAB ... BAC
상태2	CADBBDCCADA ... CDA
상태3	BABABBAABAB ... BAB

A - 액션1
B - 액션2
C - 액션3
D - 액션4

[그림 3-4] P-테이블 갱신 과정 2

상태1에서는 출연 빈도가 가장 높은 A가 2차 Q 값을 받는 액션으로 결정된다. 상태2에서는 출연 빈도가 가장 높은 액션은 2개 이상이므로 2차 Q 값을 받는 액션은 결정되지 못한다. 상태3에서는 출연 빈도가 가장 높은 B가 2차 Q 값을 받는 액션으로 결정된다.

P-테이블에서 저장하는 공간은 정적으로 구성한다. 일정 개수 이상 누적 되면 가장 먼저 누적된 액션부터 삭제한다. 위의 과정들을 통해 얻어진 2차 Q 값을 Q-테이블에 갱신하게 된다.

3.3 P-테이블을 이용한 학습

위에서 언급한 예측 정보 산출 알고리즘을 학습 알고리즘에 삽입한다. 기존의 Q-학습에서 학습을 수행하는 부분에서 1차 보상값을 계산할 때 보상값이 양의 값으로 갱신되는 경우 예측 모듈의 P-테이블에 양의 보상값을 받는 액션값을 저장한다. 예측 정보의 산출은 양의 보상값을 받는 시점에서 이루어지므로 매번 계산할 필요는 없다.

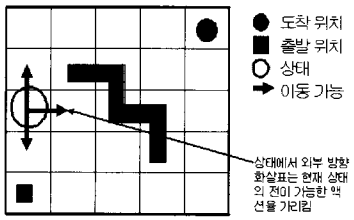
```
While learning
  excute action a
  collect reward r
  collect reward r2    // 2차 보상값 선언
  determine new state s'
  best = - unlimit
  # 새로운 상태로부터 전이 가능한 모든 액션 검사
  for each a' in s'
    // Prediction()은 [그림 3-3]에서 자세히 설명한다.
    r2 = Prediction(s' ,a' ,r)
    // 예측 모듈에서 2차 보상값을 얻어온다.
    Q(s,a) += * (r2 + * best ? Q(s,a))
    // 2차 보상은 1차 보상과 보상 대상(액션)이 다를 수 있다.
    # 가장 좋은 액션만 기억
    if Q(s' ,a' ) ) best then
      best = Q(s' ,a' )
    end if
    # 이전 상태/액션 쌍의 값을 갱신
    Q(s,a) += * (r + * best ? Q(s,a))
  end for
  # 다음 상태 처리
  s = s'
End while
```

[그림 3-5] P-테이블을 이용한 학습 알고리즘

[그림 3-5]는 본 논문에서 제안한 방식의 전체적인 알고리즘이다. 각 부분의 변수 설명은 다음과 같다. 변수 s는 현재 상태, 변수 a는 액션, 변수 r은 1차 보상값, 변수 r2는 2차 보상값, Q(s,a)는 상태 s에서 액션 a를 취한 후의 보상값을 산출하는 함수, 상수 는 학습율, 상수 는 감소율을 나타낸다. 알고리즘에서 굵게 기울여서 쓴 부분이 본 논문에서 제안한 예측 모듈이 삽입된 부분이다. 위 알고리즘에서 반복문으로 각 상태에 대한 보상을 처리하는 과정에 또 하나의 보상값 생성과정을 삽입한다. 변수 r2에 Prediction()에서 예측 값을 받아 온다. 이 값을 Q(s,a)에 변수 r과 같은 방식으로 갱신한다. 예측 모듈의 상세 설명은 [그림 3-2]에서 언급한 바와 같다.

3.4 적용 사례

본 논문의 실험에 앞서 실험 방법을 설명하기 위해 5*5 최단 경로 찾기 학습을 예로 든다. 이러한 학습 방법은 환경으로부터 상태의 정보를 받아 적절한 액션을 결정할 수 있는 정책이 필요한 모든 경우에 응용할 수 있다. 예를 들면, 단순한 길 찾기부터 복합적인 목표를 가지고 있는 전략 게임에 이르기까지 활용 범위가 넓다. 경로 찾기를 예로 든 이유는 설명이 용이하면서, 본 제안 방식을 그대로 적용할 수 있기 때문이다. 5*5 최단 경로 찾기 학습의 환경은 다음과 같다 [그림 3-6].



[그림 3-6] 5*5 최단 경로 찾기

학습은 에이전트가 출발 지점에서 시작하여 도착 지점에 도달 할 때까지 이동과 탐색을 반복한다. 첫 번째로 도착 지점에 도달한 총 이동 거리는 다음 학습 시 도달한 총 이동 거리에 대한 보상(강화) 기준이 된다. 이동 거리는 학습 시 상태 공간을 얼마나 거쳐 갔는지를 나타낸다. 보상값은 도착 위치에 있는 마지막 상태의 액션 결과 값부터 순차 계산한다. 보상값은 음(negative)의 값, 양의 값, 0의 값의 총 3가지로 부여한다. 상세한 방법은 다음 절에서 설명한다.

3.4.1 맵의 상태 정보

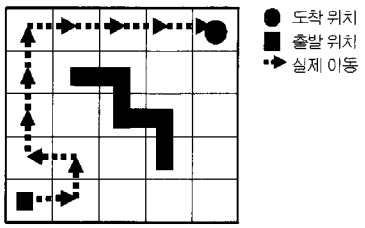
본 실험에서 사용한 맵의 상태 공간은 각각의 위치 정보를 가지고 있다[그림 3-7]. 인덱스된 상태 정보는 학습 시 시작 지점의 상태에서 다른 상태로 이동하기 위해 필요하다.

s0	s1	s2	s3	s4
s5	s6	s7	s8	s9
s10	s11	s12	s13	s14
s15	s16	s17	s18	s19
s20	s21	s22	s23	s24

[그림 3-7] 맵의 상태 정보 인덱스

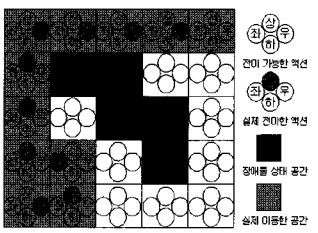
3.4.2 학습에 따른 이동 경로 변화

[그림 3-8]은 최초 학습 시 도달한 이동 경로를 나타낸다. 그림을 보면 이동 거리는 11칸이다. 최초 학습이므로 이 거리는 현재의 최단 거리가 되며, 다음 학습 결과의 판단 기준이 된다. 단, 이 경로에 대한 보상은 하지 않는다. 현재까지의 최단 거리보다 짧은 이동 거리를 기록하는 경우에는 양의 보상을 한다. 그 값이 현재의 최단 거리가 된다.



[그림 3-8] 최초 학습의 이동 경로

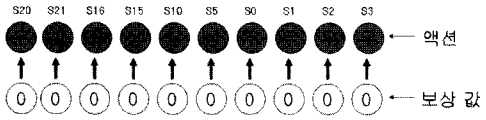
[그림 3-9]는 [그림 3-8]의 이동 경로에 대한 상태 공간의 변화 과정을 자세하게 보여주는 그림이다. 출발 지점의 상태 공간에서 오른쪽으로 이동하는 액션을 시작으로 도착지점까지 여러 상태 공간을 지나간다. 지나간 상태 공간은 회색 바탕으로 표현한다. 상태 공간 내에 있는 네 개의 작은 원은 이동 가능한 방향을 나타낸다. 여기서 이동 가능한 방향은 상태 공간의 액션에 해당한다. 학습이 누적될수록 각 상태 공간에 있는 액션은 다양한 보상값을 입력받게 된다. 이렇게 누적된 값을 가지고 있는 액션은 상태 공간 내에서 액션을 결정하는 선택의 기준이 된다. 각 상태 공간에서 누적된 보상값이 가장 큰 액션이 선택될 확률이 높아진다.



[그림 3-9] [그림 3-8]의 상태 공간 표현

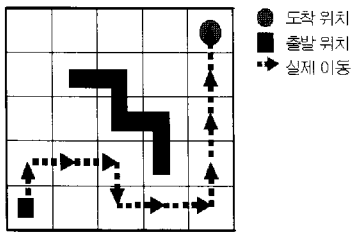
다음에 나오는 그림들은 학습이 진행됨에 따라 각 학습에서 얻어진 결과에 대한 보상이 주어지는 과정을 설명한다. [그림 3-10]은 [그림 3-9]의 상태 공간에서 전이한 액션에 대한 보상값을 누적하는 과정을 보여준다. 학습을 거듭할수

록 각각의 누적된 값은 다음 학습의 액션 선택에 영향을 준다. 각 상태 공간에서 누적된 보상값이 높은 액션이 선택될 확률이 높아진다.



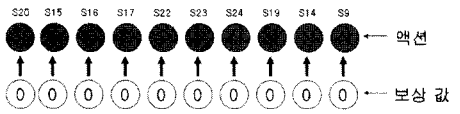
[그림 3-10] [그림 3-9]의 보상값 계산

[그림 3-10]은 두 번째 학습 시 도달한 이동 경로를 나타낸다. 그림을 보면 이동 거리는 11칸이다. 이 거리는 현재까지의 최단 거리와 같다. 그러므로 이 경로에 대한 보상값은 0이 된다.



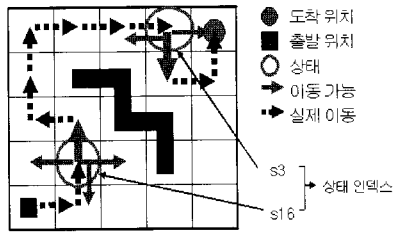
[그림 3-10] 두 번째 학습의 이동 경로

[그림 3-11]은 [그림 3-10]의 각 상태 공간의 전이한 액션에 대한 보상값을 누적하는 과정을 보여준다. 현재까지의 최단 경로와 이동 거리가 같으므로 0 값을 보상한다.



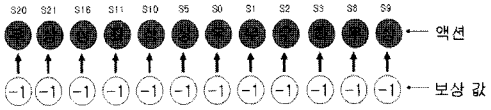
[그림 3-11] [그림 3-10]의 보상값 계산

[그림 3-12]는 n-2 번째 학습 시 도달한 이동 경로를 나타낸다. 이 그림을 보면 이동 거리는 13칸이다. 이 거리는 최단 거리보다 큰 값이다. 그러므로 이 경로에 대한 보상값은 음의 값이 된다.

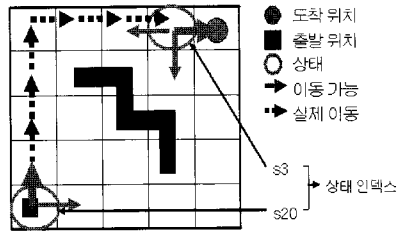


[그림 3-12] n-2 번째 학습의 이동 경로

[그림 3-13]은 [그림 3-12]의 각 상태 공간의 전이한 액션에 대한 보상값을 누적한다. 현재까지의 최단 경로의 이동 거리보다 값이 크므로 음의 보상값인 -1을 보상한다.



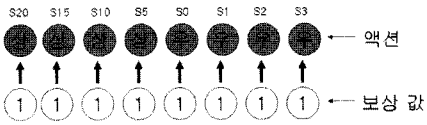
[그림 3-13] [그림 3-12]의 보상값 계산



[그림 3-14] n-1 번째 학습의 이동 경로

[그림 3-14]는 n-1 번째 학습 시 도달한 이동 경로를 나타낸다. 그림을 보면 이동 거리는 9칸이다. 이 거리는 현재까지의 최단 거리보다 작은 값이다. 그러므로 이 경로에 대한 보상값은 양의 값이 된다.

[그림 3-15]는 [그림 3-14]의 각 상태 공간의 전이한 액션에 대한 보상값을 누적한다. 현재까지의 최단 경로의 이동 거리보다 값이 작으므로 양의 보상값인 1을 보상한다.

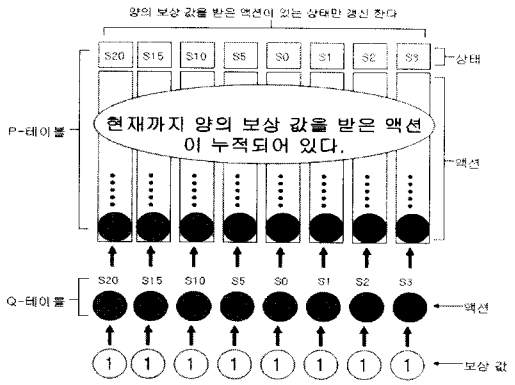


[그림 3-15] n-1 번째 학습의 보상값 계산

각각의 행동한 액션은 n-2 번째 학습의 경우 기준이 되는 최단 거리보다 큰 값의 이동 거리로 Q 값은 음의 값을 받게 된다. n-1 번째 학습의 경우 최단 거리 보다 작은 값이므로 양의 Q 값을 받게 된다. 이렇게 보상값이 누적된 액션 중 가장 큰 값이 각 상태 공간에서 선택될 확률이 높아지게 된다.

3.4.3 예측 정보 산출 값 적용

위의 Q-테이블에 보상값이 갱신되는 동안 양의 보상값이 갱신되는 시점에서 예측 정보의 산출이 이루어진다. 이 때 양의 보상값을 받는 액션은 P-테이블에 저장된다. 또한 P-테이블에 저장된 값들의 통계를 계산하고 이중에서 가장 출연 빈도가 높은 액션을 선택한다. 선택한 액션은 2차 Q 값을 받는 액션으로 Q-테이블에 갱신된다.



[그림 3-16] P-테이블 갱신 과정

[그림 3-16]은 앞서 언급한 n-1의 학습 시 보상 과정을 예로 들어 P-테이블의 갱신 과정을 나타낸다. Q-테이블에 양의 보상값을 받는 액션을 P-테이블에 저장한다.

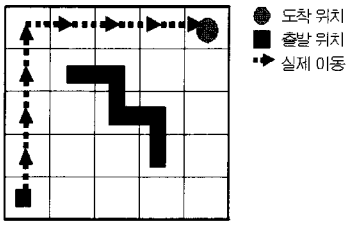
실험에서 사용하게 되는 Q-table은 다음과 같이 정의된다.

```
int nowTileData[LEARNINGTYPE][MAPW*MAPH][ACTION];
```

여기서 LEARNINGTYPE은 2가지로 기존 Q-학습과 본 논문에서 제시하는 예측정보를 활용한 Q-학습이다. 그리고 본 실험에서는 4개의 ACTION을 사용한다. MAPW, MAPH는 각각 25개이다. Q-학습에서 학습을 수행하는 부분에서 1차 보상값을 계산할 때 보상값이 양의 값으로 갱신되는 경우 예측 모듈의 P-테이블에 양의 보상값을 받는 액션값을 저장한다. 그리하여, 가장 높은 액션이 선택되면, Q-테이블에서 그 액션에 해당하는 보상값을 추가하는 학습방법이다.

3.4.4 결과

기존 방식으로 학습을 했을 때보다 빠르게 최적의 결과를 얻기 위해 Q-테이블에 갱신되는 Q 값 중에서 양의 값을 갖는 액션을 예측 모듈 내의 P-테이블에 저장하고 이 중에서 출연 빈도가 가장 높은 액션을 찾아 2차 Q 값으로 Q-테이블에 다시 갱신하여 학습을 수행한다.



[그림 3-17] 학습을 통해 얻은 최단 경로

[그림 3-17]은 위 학습 과정을 통해 최종적으로 얻어진 최단 경로이다.

4. 실험 및 분석

4.1 실험의 필요성 및 목적

본 실험의 목적은 기존 방식과 제안하는 방식의 학습 성능 비교를 통해 제안하는 방식의 성능 향상을 입증하는 것이다. 입증하는 방법으로는 같은 방식의 학습을 수행할 때 동일 시간에 더 빨리 최적의 결과에 도달하는 지를 비교 평가한다. 테스트 베드로는 최단 경로 찾기 실험을 설계한다. 여기서 실험 범위로는 10개의 서로 다른 맵에서 에이전트가 시작점에서 출발하여 도착점에 도달 하는 횟수가 일정한 시간 동안 가장 많은 방식을 찾고 이 과정에서 최단 거리와 평균 이동 거리를 비교 평가한다. 양 방식의 학습 결과는 비교하기 쉽게 한 화면에 동시에 보여 줄 수 있도록 설계한다.

4.2 시스템 설계

이 시스템은 양 방식을 동시에 구현하기 위해 2개의 에이전트로 구성 된다. 각 에이전트는 동일한 환경에서 동일한 조건으로 주어진 맵의 시작점을 출발하여 도착지점까지 이동하는 경로를 최단 거리로 가장 빨리 도달하는 방식을 찾기 위해 약간의 제약을 둔다. 학습에서의 최적의 결과는 시간이 흐름에 따라 모두 비슷한 수준에 도달한다. 때문에 일정 시간 내에 가장 최단 경로에 도착하는 것으로 제약을 둔다. 최종적으로 이 기록을 가지고 성능을 평가한다. 테스트 베드의 세부적인 설명으로 에이전트는 맵에서 현재 위치 정보와 주변의 장애물을 판별한 정보를 분석한다. 에이전트가 결정할 수 있는 액션은 상, 하, 좌, 우의 네 가지의 이동으로 한정된다. 맵에는 시작 지점과 도착 지점이 있고 에이전트는 시작 지점에 위치한다. 에이전트는 시작 지점에서 탐색과 이동을 반복해 도착 지점까지 도달한다. 이 과정을 반복하는 과정에서 최단 거리의 루트를 찾게 된다. 다음은 테스트 베드에 대한 설명이다.

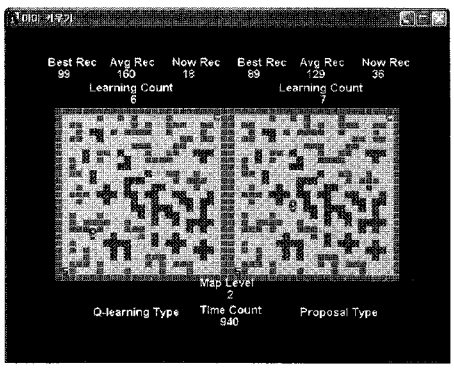
1) 가로, 세로 25칸씩 총 625칸으로 최대 크기를 정한다. 맵은 각기 다른 내용(장애물의 위치 변경)의 총 10개로 구성한다.
2) 맵은 1개의 시작점과 1개의 목표(도착지점), 1개의 에이전트, 특정 위치의 장애물로 구성한다.
3) Q-학습 방식과 제안 방식의 비교가 용이하도록 한 화면에 동시 실행이 가능하게 화면을 구성한다.
4) 처음 목표 지점에 도달한 시간을 기준으로 다음 행동 결과에 보상값을 결정한다.
5) 에이전트는 이동 행동 4개, 정지 행동 1개로 총 5개의 행동이 가능(실제 행동은 4개 상, 하, 좌, 우)하게 설계한다.

본 논문의 시스템 환경은 다음과 같다. 테스트 베드를 제작한 컴퓨터의 스펙은 CPU AMD Sempron 2800+, RAM 1GB, Video Geforce 5600이다. 운영체제로는 Windows XP Professional 서비스팩2를 사용하였고, 제작 툴로는 Visual Studio 6를 사용하였으며, DirectX9.0 기반으로 테스트 베드를 제작하였다.

4.3 실험

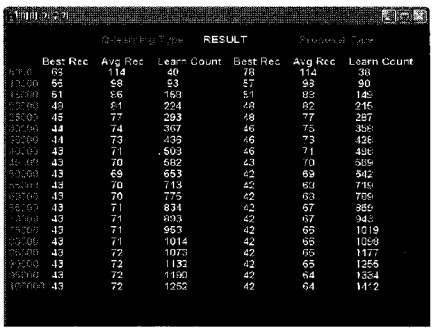
Q-학습 방식과 제안한 방식으로 학습되는 모의실험을 통해 성능을 비교한다. 실험을 위해 경로 찾기 맵의 크기는 25*25를 최대 값으로 설정한다. 이번 실험에서는 학습에 대한 에이전트의 수행결과를 측정하여 최적화 성능을 평가한다. 이와 같은 방식으로 실험하고 비용을 계산하여 그 결과를 출력한다.

본 논문의 제안 방식과 기존 방식의 테스트 화면은 [그림 4-1]이다. 각각의 방식을 비교하기 위해 한 화면에 동시에 보여주고 있다. 화면의 각 부분을 설명하면, Learning Count는 시작 지점에서 도착 지점으로 도착 했을 때마다 카운트되는 수치이다. Best Rec은 시작 지점에서 도착 지점까지의 최소 이동거리이다. Avg Rec은 평균 이동거리이다. Now Rec은 현재의 이동거리를 나타낸다. 화면 중앙 밑에 보이는 Time Count는 게임 내에서 적용되는 시간이다. 이 시간 동안 Learning Count를 가장 많이 올린 방식이 두 방식 중에서 최적화에 가깝다는 것을 나타낸다.



[그림 4-1] 테스트 베드 실험 화면

[그림 4-2]는 테스트 베드의 실험 결과 화면이다. 왼쪽은 기존 방식의 수치를 나타내고 오른쪽은 제안한 방식의 수치를 나타낸다. 화면 맨 왼쪽의 빨간색 숫자는 실험 시간을 나타낸다. 5,000sec 단위로 표시된다.



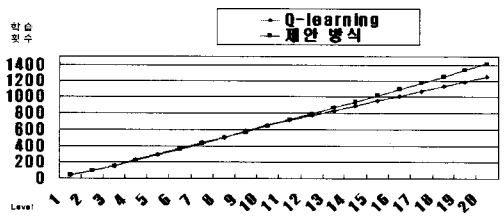
[그림 4-2] 테스트 베드 실험 결과 화면

이 실험에서 비교할 항목은 크게 두 가지이다. 첫째, 두 방식의 학습 횟수 비교, 둘째, 두 방식의 평균 이동 거리 비교이다. 이렇게 두 가지 항목을 일정 시간동안 비교하여 두 방식의 성능을 평가한다.

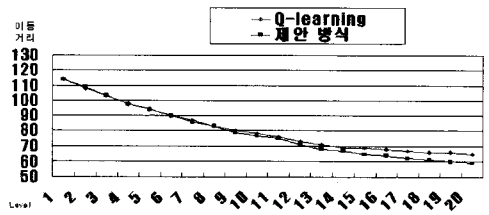
실험은 게임 내 시간을 기준으로 두 학습 방식을 총 100,000sec동안 학습을 수행한다. 5,000sec 단위로 각 방식의 수치를 체크한다. 큰 단위인 5,000sec를 다시 10개로 나누어 500sec마다 맵을 다르게 하여 학습을 수행한다. 수치는 큰 단위인 5,000sec마다 기록한다. 위에서 언급한 세 가지 항목을 총 20단계에 걸쳐 비교한다.

4.4 실험 결과

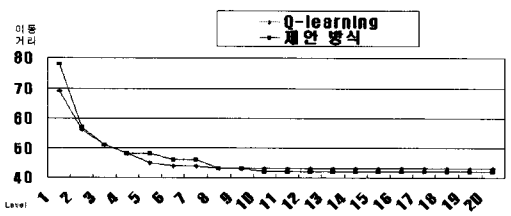
본 실험은 강화 학습을 이용한 경로 찾기이다. 강화 학습의 특성상 학습의 최종적인 단계는 임의로 정해야 한다. 또한 최종적 단계의 성능은 거의 모든 방식이 동일하게 된다. 그런 이유로 성능 평가의 기준은 임의의 학습 시간까지 얼마나 많이 학습하느냐에 달려 있다.



[그림 4-3] 학습 횟수 비교



[그림 4-4] 평균 이동 거리 비교



[그림 4-5] 최소 이동 거리 비교

[그림 4-3]은 본 논문의 학습 횟수 비교 그래프이다. 왼쪽의 눈금은 Learning Count의 수치를 나타내며, 오른쪽의 눈금은 각 단계별 시간을 나타낸다. 총 20단계로 나누어 5,000sec 마다 기록한 Learning Count 수치를 나타낸다. Learning Count의 수치가 높은 쪽이 효율이 높다.

그래프의 결과를 보면 학습 초기부터 중반까지는 두 방식의 차이가 거의 없었다. 두 방식의 차이는 중반이후부터 두드러진다. 중반이후부터 제안한 방식이 기존 방식보다 평균 9%의 성능 향상을 가져왔다. [그림 4-4]의 경우에서도 비슷한 결과를 보였다. 이는 제안의 특성상 생성된 2차 Q 값을 받는 액션이 신뢰성을 얻기 위한 시간을 필요로 하기 때문이다. 다시 말해 통계를 내기 위해서는 P-테이블의 값이 일정 수준이상 저장되어 있어야 한다는 것을 의미한다. 본 제안 방식은 통계 자료로 사용할 범위만큼의 P-테이블의 값이 모두 수집된 이후부터 완전한 2차 Q 값을 생성할 수 있다. 학습 초기에는 기존 방식과 동일하거나 약간 저조한 성능을 보여준다. 또한 본 방식은 상태 공간 내의 전이 가능한 액션이 많을수록 좋은 성능을 보여준다. 이는 1차 Q 값의 생성과 다른 방식인 통계 모델을 적용했기 때문이다. 전이 가능한 여러 액션 중 P-테이블 내에서 가장 많은 출연 빈도를 나타낸 액션이 2차 Q 값을 받게 된다. 그 결과 1차 Q 값과 2차 Q 값을 받는 액션은 서로 다를 수 있게 된다. [그림 4-5]의 최소 이동 거리는 학습의 특성상 평균 이동 거리와는 다르다. 랜덤하게 어느 한쪽이 더 빠른 경로를 찾을 수 있다. 하지만, 시간이 흐름에 따라 일정 시간이 지난 후에는 비슷한 결과를 가져오게 되므로 두 방식의 성능 평가에는 영향을 주지 못하였다.

본 논문의 장점으로는 크게 두 가지를 들 수 있다. 첫째, 기존 알고리즘의 큰 변환 없이 예측 정보 모듈 추가로 2차 Q 값 생성이 가능하므로 전통적인 Q-학습 외에 반복적 학습의 보상(강화 값) 개념이 있는 거의 모든 기법에 적용이 용이하다. 둘째, 기존 방식의 1회 학습 과정에서 Q 값을 2회 부여가 가능하다. 상대적으로 적은 처리 시간에 2회 학습 이상의 효과를 얻을 수 있다. 결과적으로 학습 시간의 단축이 가능하다.

5. 결론 및 향후 연구계획

본 논문에서 제안하는 예측정보를 적용한 최적화 방안은 기존 Q-학습 방식 보다 학습 시간의 단축을 가져왔다. 이것은 동일 시간에 대한 학습 성능 향상을 나타내며 최적화된 학습 결과를 보여줌으로써 성능을 입증하였다. 따라서 보다 빠른 시간에 최적의 정책이 수행됨에 따라 학습의 효율성 증가와 비용 절감을 기대할 수 있다. 또한 반복적 학습의 보상(강화 값) 개념이 있는 거의 모든 기법에 범용적인 적용이 용이하다.

본 제안 방식은 2차 Q 값의 신뢰성을 얻을 시간이 필요하다. 신뢰성을 얻은 이후인 실험 중반 이후부터 성능 향상이 가능했다. 상태 내의 전이 가능한 액션의 수가 많을수록 성능이 높아진다. 실험 결과 실험 중반 이후부터 제안한 방식은 기존 방식보다 평균 9%의 성능 향상을 보였다. 1차 Q 값 생성 방식처럼 보상값이 가장 높은 액션을 결정하는 것과는 달리 최근 일정 범위 내에서 가장 많은 출연 빈도를 가진 액션을 결정하기 때문이다. 하지만 위 실험에서는 테스트 베드의 특성상 많은 액션이 가능한 상태를 전제로 하는 실험이 불가능하였다. 또한 탐색능력의 기준을 시작지에서 목적지까지 어느 정도의 시간이 흐른 다음에 도달하는지의 평균값인 FPS(First Passage Time)를 구하여, 일반 Q-학습과 본 연구에서 제안한 방법을 비교분석할 필요가 있다. 향후 연구 계획은 향상된 성능을 보다 높이기 위해 예측 모듈의 정확성을 높일 수 있도록 알고리즘을 개선해야 한다. 그러기 위해 예측 모듈 내 자료 구조의 효율적 설계로 저장 공간의 절약 및 계산 비용 절감을 이끌어내야 한다.

참고 문헌

[1] Manslow, John, "Adaptation of Learning", AI Game Programming Wisdom 1, Charles River Media, pp. 749 ~ 760, 2002

[2] Laramee, Francois Dominic, "Using N-Gram Statistical Models to Predict Player Behavior", AI Game Programming Wisdom 1, Charles River Media, pp. 797 ~ 804 , 2002

[3] Carla Marceau, "Characterizing the behavior of a program using multiple-length N-grams",Communications of the ACM, pp.101 ~ 110, 2001

[4] Champandard, Alex J. , AI Game 인공지능 게임 프로그래밍 실전 가이드, 에이콘, pp. 617 ~ 640, 2005

[5] Manslow, John, "강화 학습을 이용한 AI 제어 문제 해결", AI Game Programming Wisdom 2, 정보문화사, pp. 729 ~ 741, 2005

[6] 김병천, 김삼근, 윤병주, "미로 환경에서 최단 경로 탐색을 위한 실시간 강화 학습", 한국정보처리학회 논문지 B, vol. 9-B NO. 02, pp. 0155 ~ 0162, 2002

[7]정석일, 이연정, "분포 기여도를 이용한 퍼지 Q-learning", 퍼지 및 지능시스템학회 논문지 제11권 5호, pp. 388-394, 2001

[8] 성연식, 조경은, "영향력 분포도를 이용한 Q-학습", 멀티미디어학회 논문지 제9권 5호, pp. 649-657, 2006

이 충 현

2004.2 남서울대학교 멀티미디어학과 졸업 (이학사)
2006.8 동국대학교, 영상대학원 멀티미디어학과 졸업 (예술공학사)
2006.9 ~ 현재 (주)Wisdomain

관심분야 : 게임 인공지능, 게임 프로그래밍

조 경 은

1993.2 동국대학교,전자계산학과(공학사)
1995.2 동국대학교, 컴퓨터공학과 대학원(공학석사)
2001.8 동국대학교, 컴퓨터공학과 대학원(공학박사)
2003.9 ~ 2005.8 동국대학교 정보산업대학 컴퓨터멀티미디어공학과 전임강사
2005.9 ~ 현재 동국대학교 영상미디어대학 게임멀티미디어공학과 조교수
관심분야 : 컴퓨터 게임 알고리즘, 게임 인공지능, 멀티미디어 정보처리

엄 기 현

1975년 서울대학교 공과대학 응용수학과 공학사
1997년 한국과학기술원 전산학과 이학석사
1994년 서울대학교 대학원 컴퓨터공학과 공학박사
978년3월~2005년8월 동국대학교 컴퓨터멀티미디어공학과 정교수
2005년9월 ~ 현재 동국대학교 영상미디어대학 게임멀티미디어공학과 교수
1995년3월~1999년2월 동국대학교 정보관리처장 역임
1998년9월~2000년8월 데이터베이스 연구회 운영위원장
1998년8월~2000년7월 한국정보과학회 데이타베이스연구회 운영위원장,
1998년12월~2001년12월 한국 멀티미디어학회 부회장
1999년4월~현재 Int. Conf. on Database Systems for Advanced Applications Steering Committee 위원
2001년 3월~2003년 2월 동국대학교 정보산업대학 학장 역임
2001년~2002년 한국정보과학회논문지편집부위원장 (데이타베이스담당)
2004년1월~2005년2월 한국 게임학회 부회장
2005년3월~현재 한국 게임학회 자문위원
2004년1월~2005년12월 한국 멀티미디어학회 자문위원
2006년1월~2006년12월 한국멀티미디어학회 수석부회장
2007년1월~현재 한국멀티미디어학회 회장
관심분야 : 게임시스템, 멀티미디어데이터베이스, 멀티미디어정보시스템