Journal of Korea Multimedia Society Vol. 12, No. 9, September 2009(pp. 1333-1341)

# 컴퓨터 게임에서 전술적 경로 찾기를 위한 휴리스틱 학습

유 견 아†


## 요 약

컴퓨터 게임에서 전술적 경로 찾기란 캐릭터의 이동 경로를 결정할 때, 최단 거리나 최소 시간 등의 요소만이 아니라 주변의 전술 정보를 고려하여 경로를 선택해야 하는 경로 찾기를 말한다. 경로 찾기에 전술 정보를 포함하는 한 가지 방법은 게임에 필요한 전술 정보를 각 정보의 중요도에 따라 가중치를 부여하고 가중 합으로 휴리스틱 함수를 표현하는 것이다. 전술 정보의 가중치의 결정은 경로를 찾기 위한 탐색의 성능과 구해지는 경로의 특성을 결정짓기 때문에 매우 중요하다. 본 논문에서는 레벨 설계자가 캐릭터의 특성에 맞는 경로 표본을 제공하면 현재 가중치에 의해 탐색된 경로와 주어진 표본 경로와의 차이를 이용하여 더 나은 가중치로 조정함으로서 휴리스틱 함수를 개선하는 방법을 제안한다. 제안된 방법은 탐색 오차를 발견하여 휴리스틱을 학습하기 위해 수정된 탐색 알고리즘과 퍼셉트론-유사 가중치 갱신 공식을 포함한다. 시뮬레이션 결과를 통해 전술 정보를 이용한 경로 계획과 기존의 경로 찾기의 차이를 보여주며 학습의 성능에 영향을 주는 요인들에 대해 분석하고 실제 게임 환경에 적용한 예를 보여 준다.


# Learning Heuristics for Tactical Path-finding in Computer Games

Kyeonah Yu†


## ABSTRACT

Tactical path-finding in computer games is path-finding where a path is selected by considering not only basic elements such as the shortest distance or the minimum time spend but also tactical information of surroundings when deciding character's moving trajectory. One way to include tactical information in path-finding is to represent a heuristic function as a sum of tactical quality multiplied by a weighting factor which is determined based on the degree of its importance. The choice of weighting factors for tactics is very important because it controls search performance and the characteristic of paths found. In this paper, we propose a method for improving a heuristic function by adjusting weights based on the difference between paths on examples given by a level designer and paths found during the search process based on the current weighting factors. The proposed method includes the search algorithm modified to detect search errors and learn heuristics and the perceptron-like weight updating formular. Through simulations it is demonstrated how different paths found by tactical path-finding are from those by traditional path-finding. We analyze the factors that affect the performance of learning and show the example applied to the real game environments.


**Key words:** Computer Game(컴퓨터 게임), Tactical Path-finding(전술적 경로 찾기), Heuristic Function(휴리스틱 함수), Machine Learning(기계학습)

※ 교신저자(Corresponding Author) : 유견아, 주소 : 서울시 도봉구 쌍문동 419(132-714), 전화 : 02)901-8346, FAX : 02)901-8341, E-mail : kyeonah@duksung.ac.kr
접수일 : 2008년 11월 21일, 수정일 : 2009년 5월 20일
완료일 : 2009년 6월 12일
† 정회원, 덕성여자대학교 컴퓨터공학부 교수
※ 본 연구는 2008년 덕성여자대학교 교내연구비 지원으로 수행되었음

## 1. 서 론

컴퓨터 게임에서 기존의 경로 찾기는 대부분 최단 경로를 빠르게 찾는데 목적을 두어 왔으나 최근 게임에서는 적의 시야로부터의 차단이나 보급소의 위치 등과 같은 주변 환경의 전술적 정보를 고려하여 경로를 찾아야 하는 필요성이 대두되고 있다. 이와 같이 경로를 찾는데 전술적 정보들을 이용하는 경로 찾기를 전술적 경로 찾기(tactical path-finding)라고 하며 복잡한 시나리오의 게임이 증가하고 있는 최근 게임 분야에서 관심을 모으는 주제이다[1]. 전술적 정보를 이용하는 방법 중의 하나는 경로 찾기를 위해 노드를 탐색할 때 이용되는 휴리스틱 함수에 이와 같은 정보들을 포함하는 것이다. 즉 각 전술 정보의 중요성에 따라 가중치를 부여하고 이들의 합으로 휴리스틱 함수로 나타내어, 탐색할 때 노드의 가치를 평가하기 위한 비용함수에 이용하는 것이다. 이때 가중치의 선택은 최종 경로의 품질이나 경로가 선택되는 과정에 영향을 미치는 중요한 문제인데 이를 기계학습 방법을 이용하여 해결하고자 하는 것이 본 논문의 목적이다.

게임 개발 단계에서 고려되어야 하는 여러 가지 전술 정보가 정해졌을 때, 레벨 설계자가 각 정보의 중요도에 따라 직접 가중치를 부여하는 방법은 설계자의 주관에 의존해야 할 뿐 아니라 대략적인 추정에 의존하기 때문에 정확하지 않다. 레벨 설계자가 바로 가중치 값을 추정하는 대신에 캐릭터의 특성에 맞는 좋은 품질의 경로 표본을 보여 주는 것은 충분히 구체적이므로 설계자에게 더 용이한 일이 될 것이다. 이렇게 주어진 표본들을 가중치를 학습하기 위하여 훈련 예제로 이용할 수 있는데 본 논문에서는 경로를 탐색하면서 주어진 훈련 예제들과의 비교를 통해 탐색 오차를 발견하면 휴리스틱을 갱신하는 판별적 학습 (discriminative learning) 방법을 사용한다. 즉, 현재의 가중치로 이루어진 휴리스틱을 이용해 탐색을 하는 가운데 탐색의 결과가 훈련 예제와 다른 경우, 가중치를 갱신하고 갱신된 가중치를 이용하여 탐색을 계속해 나가면서 반복적으로 가중치를 갱신하는데 정확한 휴리스틱 모델을 학습한다기보다 '좋은' 노드와 '나쁜' 노드를 구분해내어 목표를 찾기에 충분한 정도의 휴리스틱을 학습한다는 관점에서 '판별적 학습 방법'이라고 할 수 있다. 이 접근 방식은 구묶음(chunking)이나 태깅(tagging)등의 자연언어 처리 문제의 해결을 위해 제안된 탐색 최적화에 의한 학습(Learning as Search Optimization)[2] 방식을 인공 지능 계획(AI planning) 분야에 적용하기 위해 수정한 [3]에 기반한다.

전술적 경로 찾기에 대한 연구는 컴퓨터 게임 분야의 일반 경로 찾기에 비해 최근에 주목받기 시작하여 자료의 양이 많지 않으나 그중 대부분은 경로 찾기를 위한 로드맵에 전술적 정보를 저장하는 방법에 대한 것이다[1,4-7]. [4]와 [5]에서는 게임 배경에 지정된 경로점(waypoint)에 전술적 평가 점수를 부여하는 방법을 제안하였다. 즉 위치의 좋고 나쁨을 결정하는 특성들로 위치 평가 함수를 정의하여 캐릭터의 위치를 선정하는데 이용하였다. [4]에서는 위치 평가 함수를 A*를 이용한 경로 탐색에도 이용하였는데 위치 평가 함수를 경로 탐색을 위한 비용 함수로 전환할 때 특성 값들의 가중치 합으로 표시하였으며 설계자가 임의로 가중치를 정하는 문제점을 지적하였으나 이를 해결하는 방법을 제시하지는 않았다. [5]에서는 경로점마다 특성 값을 행렬 형태로 미리 저장해 놓아 경로 탐색 등의 응용에서 빠르게 경로점의 정보를 이용할 수 있는 방법을 제안하였다. 그러나 [5]에서 다룬 특성들은 가시성 여부와 같이 이진 정보로서 경로를 탐색할 때 경로점이 특성을 만족하는가만 확인하도록 되어 있어 정량적인 취급에 필요한 특성들은 다루지 않았다. [6]과 [7]에서는 전술적 정보를 고려하여 영역을 사다리꼴로 나누는 방법에 대하여 논하였다. 전술적 정보의 값들을 공통으로 가질 수 있는 작은 영역으로 나누면 이 영역을 하나의 노드로 하여 A*와 같은 알고리즘을 이용하여 경로를 탐색하는데 최소의 영역으로 나누는 문제에 초점이 있으며 나누어진 영역이 유일한 값을 가지므로 경로 탐색은 단순하게 이루어졌다. [1]에서는 탐색에 이용되는 평가 함수를 전술적 정보의 가중치 곱의 합으로 표현하고 가중치의 선정이 중요하다는 것을 언급하였으나 역시 설계자에 의해 특성의 중요도에 따라 가중치를 임의로 정하였으며 가중치를 기계학습에 의해 적응적으로 변환하려는 시도는 없었다. 본 논문에서는 앞에서 소개한 탐색 최적화에 의한 학습 방식을 게임의 경로 찾기에 적합하도록 수정하여 자동적으로 가중치를 갱신하는 방법을 제안하며 실제 게임에 적용하였을 때 기존의 경로 찾기 결과와 비교해

어떤 차이가 있는지를 시뮬레이션 결과를 통해 확인하며 학습 성능에 영향을 주는 요인들에 대한 분석 결과를 제시한다.

## 2. 전술적 경로 찾기

에이전트가 한 위치에서 다른 위치로 이동하는 루트를 정하는 일을 지칭하는 경로 찾기는 최근 실시간 전략게임이 인기를 얻음에 따라 컴퓨터 게임 개발에서의 비중이 더욱 높아지고 있다. 경로 찾기를 프로그래밍의 관점에서 본다면 그래프 탐색이다[8,9]. 시작 노드로부터 출발하여 이웃한 노드들을 탐색하고 평가하여 최상의 노드를 선택하는 과정을 목표 노드가 나올 때까지 반복한다. 본 장에서는 그래프 탐색을 위한 휴리스틱 함수에 전술 정보를 어떻게 반영하는가와 전술적 경로 찾기를 위한 그래프 탐색 알고리즘에 대해 살펴본다.

### 2.1 전술 정보를 포함하는 휴리스틱 함수의 표현

기존의 경로 찾기에서는 대부분 최단 경로를 빠른 시간 내에 찾는 것을 목표로 하며 이를 위해 노드가 목표에 도달하기 위해 소요되는 비용을 추정한 휴리스틱 함수를 이용한다. 최단 경로를 찾는 문제에서 대표적인 휴리스틱으로는 노드와 목표 지점 사이의 직선 거리를 나타내는 유클리디언 거리를 들 수 있다.

전술적 경로 찾기는 기존 경로 찾기에서 고려 대상이던 거리나 시간 이외에 전술적 가치를 고려하여 루트를 정해야 하는 경로 찾기이다. 예를 들어 그림 1에서 경로 1은 기존의 최단 거리 경로 찾기에 의해 정해진 루트를 나타내며 경로 2, 3은 적군으로부터 피하기 위해 장애물을 이용하여 우회한 루트를 나타

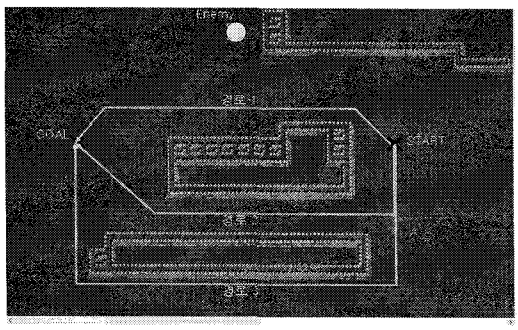


그림 1. 전술적 정보를 고려하였을 때 가능한 경로들

내는데 이중에서 어떤 루트가 택해질 것인가는 어떤 전술적 가치에 주안점을 두느냐에 따라 달라질 것이다. 이와 같은 정보를 경로 찾기에 반영하는 한 가지 방법은 게임에서 고려해야 하는 모든 전술적 정보를 휴리스틱에 포함하는 것이다. 참고문헌 [1]의 $H=D+\Sigma_i w_i T_i$와 같이 전술적 정보의 비중에 따라 가중치를 곱하고 이들을 모두 합하여 나타낸다. 이 공식에서 D는 목표까지의 직선 거리를 나타내고 $T_i$는 i번째 전술적 정보, $w_i$는 $T_i$에 해당하는 가중치를 나타내어 전술적 정보와 비전술적 정보를 분리하여 표기하였다. 이와 같이 분리함으로서 각 전술적 정보에 대한 가중치의 경중을 기존의 거리 정보에 상대적으로 판단하여 중한 정보는 1 이상의 값을, 경한 정보는 1 이하의 값을 할당하는 방법으로 가중치를 어림하였다. 본 논문에서는 가중치의 자동 갱신을 위한 프로그램을 위해 전술적 정보와 비전술적 정보를 분리하는 대신에 거리 정보도 전술적 정보와 같이 특성의 하나로 통일되게 표시하여 다음과 같은 공식으로 나타내기로 한다.

$$H = \Sigma_j w_j f_j \quad (1)$$

각 노드의 특성 값들을 어떻게 효과적으로 저장하고 호출해 휴리스틱 정보로 이용할 것인가에 관한 연구는 게임의 배경을 표현하는 자료 구조와 정보를 저장하는 기법과 관련된 부분으로 본 연구의 범위에 포함하지 않고 [1]의 방법과 같이 전술 정보를 위치-기반(location-based)정보와 연결-기반(connection-based) 정보로 구분하여 다룬다. 지형의 종류나 경사 등과 같은 위치-기반 정보는 노드마다 미리 저장하여 사용하고 적군이나 보급기지로부터의 거리 등과 같은 연결-기반 정보는 휴리스틱 값이 필요할 때마다 계산하여 사용하도록 한다.

### 2.2 탐색 문제로서의 경로 찾기

일단 휴리스틱 함수가 결정되고 나면 전술적 경로 찾기는 일반적인 탐색 문제가 된다. 컴퓨터 게임에서의 경로 찾기를 위해 가장 보편적으로 사용되는 탐색 알고리즘은 A*이다[8,10]. A*알고리즘은 과소 추정된 휴리스틱(underestimate heuristic)을 이용하면 최적의 경로를 찾는 것이 보장된다는 장점으로 여러 가지 보완 방법과 함께 다수의 게임에서 경로 찾기를 위해 사용되어 왔다[11]. 그러나 위에서 제안한 형태

의 휴리스틱을 사용하면 과소 추정을 보장하는 것이 항상 가능하지는 않으므로 A* 와 같은 최적해(optimal solution) 알고리즘을 사용하는 대신, 탐색을 통한 기계학습의 구현에 적합한 빔 탐색(beam search) 알고리즘을 사용하고자 한다. 빔 탐색 알고리즘은 탐색 그래프의 레벨마다 모든 후손들을 생성하고 휴리스틱 값에 따라 정렬한 후, 미리 정해진 일정 수(빔폭, beam width)의 자손들만 OPEN 리스트에 남기고 나머지는 제거하는 탐색 방식이다. 그림 2는 빔 탐색 알고리즘의 의사 코드(pseudo code)이다. 빔폭을 무한대로 하는 경우는 제거되는 자손들이 없어 너비 우선 탐색과 같은 탐색이 되고 빔폭을 작게 하면 많은 노드들이 제거된다. 따라서 빔폭의 선택에 따라 목표 노드가 제거될 수 있기 때문에 최적성이나 완전성(completeness)은 보장되지 않으나 OPEN 리스트에 남아있는 노드수를 빔폭으로 제한하기 때문에 필요한 공간 자원을 한정시킬 수 있다는 장점이 있다. 빔 탐색 알고리즘이 판별적 학습에 적합한 이유는 미리 정해진 빔폭이 '좋은' 노드와 '나쁜' 노드를 구분하는 경계의 역할을 하며 탐색 레벨마다 주어진 훈련 예제의 해당하는 레벨과의 비교를 통해 현재의 가중치에 의한 탐색 결과가 좋은지 나쁜지 판별이 가능하기 때문이다. 그러므로 본 논문에서 전술적 경로 찾기는 식 (1)의 휴리스틱을 비용함수로 하는 빔탐색 문제로 정의하고 이를 학습에 이용하도록 한다. 그림 2의 유사 코드는 일반적 빔탐색 알고리즘을 나타내며 경로 찾기에 적용하기 위해서는 상태

```
BeamSearch ((s0,g), b)
// b is a beam width
   B ← {(s0)}
   while (B is not empty)
         node ← RemoveFront(B)
         if GoalTest(node) then return node
         B ← BeamExpand (B, b)
   return failure;
-----------------------------------------
BeamExpand (B, b)
   candidates ← {}
   for each n ∈ B
         candidates ← candidates ∪ Successors(n)
   for each n ∈ candidates
         computes H(n) = w · f(n)
   return b best nodes ordered by heuristic value
```

그림 2. 빔 탐색 알고리즘

공간(state space)을 구성할 상태를 정의하고 휴리스틱을 구성할 특성과 각 특성의 가중치를 결정하면 된다. 유사 코드에서 $s_0$는 초기상태, $g$는 목표상태를 나타낸다. 상태는 게임 배경을 표현한 방식에 따라 결정되는데 경로점 그래프나 가시성 그래프를 사용하는 경우, 그래프의 노드들이 상태가 되며 격자기반(grid-based)으로 표현되는 경우에는 상태공간은 전체 격자로 이루어지게 된다.

## 3. 휴리스틱 학습 알고리즘

전술적 정보를 이용한 경로 찾기의 성공의 핵심은 특성들의 가중치 합으로 나타내어진 휴리스틱에 있다고 할 수 있다. 가중치를 어떻게 할당하는가에 따라 찾는 경로의 질이 달라질 뿐 아니라 탐색의 성능도 영향을 받기 때문이다. 그러므로 본 논문에서는 2장에서 정의한 휴리스틱 함수와 탐색 문제에 훈련 데이터를 이용하여 휴리스틱 함수의 가중치를 향상시키는 학습 방법을 제안하며 본 장에서는 이와 같은 학습을 위한 기능이 추가된 탐색 알고리즘과 가중치를 수정하는 방법에 대해 기술한다.

### 3.1 휴리스틱 학습을 위한 빔 탐색

본 논문에서 제안하는 학습 방법은 학습 과정을 기존의 탐색과 분리하지 않고 탐색 과정에서 학습이 일어나도록 한 탐색 최적화에 의한 학습(LaSO)에 기반한다. LaSO는 언어 구조 분석 분야에서 처음으로 제안되었는데 [2] 주어진 문제 $x_i$와 그에 대한 이상적인 답(ideal solution) $y_i$로 이루어진 훈련 예제 집합 $(x_i, y_i)$를 주고 각 $x_i$에 대해 해답 $y_i$을 찾을 수 있도록 상태 공간 탐색을 유도하는 학습 방식이다. 여기서 각 $x_i$는 시작점과 목표점으로 이루어진 경로 찾기 문제 (s0, g)로 $y_i$는 $x_i$에 대한 해답 경로(solution path), $(s_0, s_1, \cdots, s_T=g)$로 나타내면 LaSO를 경로 찾기 문제에 응용할 수 있다. 경로 찾기 분야가 기존의 문제와 근본적으로 다른 점은 경로 찾기에서 가능한 좋은 경로가 단 하나가 아니라는 점이다. 다수의 좋은 경로들을 모두 훈련 예제 집합에 포함하여 처리한다면 이 문제를 해결할 수는 있지만 중복 자료로 인해 비효율성을 가중시키게 된다. 그러므로 본 논문에서는 동일 레벨에서 큐에 남아 있는 노드들은 모두 동일한 품질을 가지는 것으로 간주하여 너비 우선 빔 탐색으

로 이를 처리하고자 한다.

기존 상태 공간 탐색의 틀 안에서 학습이 일어나는 것이 가능한 부분은 큐 역할을 하는 빔의 구성 요소들을 변화시키는 것이다. 즉, 기준이 되는 비용 함수에 의해 '좋은' 노드들이 '나쁜' 노드들에 비해 빔에서 우선순위를 갖도록 위치된다면 탐색의 기본 절차상, 최적 해를 구하는 문제에서는 최적 해를 더 빨리 구할 수 있고 근사 해를 구하는 문제에서는 최적에 더 가까운 해를 구할 수 있게 되는 것이다. 본 논문에서의 비용 함수는 특성의 가중치 합으로 정의되었고 따라서 이를 결정하는 요인은 바로 가중치이므로 학습의 목표는 가중치의 조정(weight adjustment)이다. 학습 알고리즘은 가중치 갱신을 위한 기준으로 훈련 예제를 이용하는 지도 학습으로 구현되며 레벨 설계자가 훈련 예제를 제공하면 이를 이용하여 탐색을 진행하다가 탐색 결과가 주어진 훈련 예제와 불일치하면 이를 보완하는 방향으로 가중치를 수정해 나간다. 이를 정리하면 그림 3의 의사 코드로 나타낼 수 있다.

학습을 위한 탐색 알고리즘의 입력인 (x, y)는 훈련 예제로서 x는 기존 탐색 알고리즘의 입력이었던 $(s_0,g)$이며 y는 그에 해당하는 올바른 경로 해 $(s_0,s_1, \ldots ,s_T=g)$이다. 이와 같이 구성된 훈련 예제 이외에 가중치 w가 훈련을 위한 알고리즘의 입력이 되는데 기존의 탐색에서 가중치 값이 고정되어 있던 것과는

```
LearningbySearch ((x,y), w, b)
// x = (s0, g)
// y is a solution trajectory (s0, s1, ···, sT=g)
   B ← {(x, (s0))}
   for i = 1,···, T
         B ← BeamExpand (B, w, b);
       n* ← si
       if n* ∉ B // search error
          w ← Update (w, B, n*)
          B ← {n*}
   return w;
```

```
BeamExpand (B, w, b)
   candidates ← {}
   for each n ∈ B
         candidates ← candidates ∪ Successors(n)
   for each n ∈ candidates
         computes H(n) = w · f(n)
   return b nodes with low heuristic value
```

그림 3. 학습을 위해 수정된 빔 탐색

대조적으로 프로그램 실행에 따라 갱신되는 매개변수로 사용된다. b는 기존의 탐색 알고리즘과 마찬가지로 빔폭을 나타낸다.

알고리즘을 살펴보면, 주어진 경로 찾기 문제 x에 대해서 경로의 해(solution)가 미리 주어지므로 노드를 확장해야 하는 레벨이 T로 정해져 있고 또한 GoalTest가 불필요하다. 학습을 위한 가장 중요한 변경은 탐색 오차(search error)가 발생하는 부분이다. 탐색 오차는 임의의 레벨에서 빔을 생성하였을 때, 훈련 예제 경로의 노드인 n*가 그 빔 안에 존재하지 않으면 발생하는 오차로 정의되는데 탐색 오차가 발생하면 다음 두 가지를 실행한다.

1. n*가 빔 안에 존재하지 않는다는 것은 현재의 가중치로 확장한 결과가 원하는 노드를 포함하지 않을 정도로 좋지 않다는 것을 의미하므로 가중치를 갱신한다. (가중치를 갱신하는 방법에 대해서는 다음 절에 다룬다.)
2. 현재의 빔으로 계속 탐색을 하는 대신에 빔을 비우고, 훈련 예제로 주어진 n*를 삽입하여 탐색을 계속한다.

한 가지 주목할 사항은 이 알고리즘은 실패하지 않는다는 것이다. 탐색 알고리즘이 실패한다는 것은 목표에 이르기 전에 빔이 비는 것을 의미하는데 다음과 같이 간단히 증명할 수 있다. 그림 3의 알고리즘의 경우, i) 빔이 n*를 포함하는 경우와 ii) 빔이 n*를 포함하지 않는 경우로 나누어지며 ii)의 경우에도 {n*}로 다시 정의하여 탐색을 진행하므로 빔이 비어 있게 되는 경우가 발생하지 않는다. 그러므로 이와 같은 방식으로 목표 노드를 포함한 빔까지의 확장을 실패 없이 수행하게 된다. 빔 확장의 경우 (BeamExpand)에는 기존의 탐색 알고리즘과 동일하나 학습을 하는 동안에는 빔을 우선순위 큐로 만들지 않아도 된다.

### 3.2 가중치 수정

탐색 오차가 발생하면 빔이 n*를 포함할 수 있도록 휴리스틱의 가중치를 수정한다. 그림 3에서 **Update** *(w, B, n*)*에 해당하는 부분이다. 본 논문에서는 고전적인 가중치 갱신 방법인 퍼셉트론 갱신(perceptron updates) 방법을 인공지능의 계획 문제의 학습에 맞도록 수정한 다음의 식을 이용한다[3].

$$w = w + \alpha \cdot \left( \frac{\Sigma_{n \in B} F(n)}{|B|} - F(n^*) \right), 0 < \alpha \leq 1$$

여기서 α는 학습 속도 변수이고 F(n)은 노드 n의 특성 벡터, B는 현재 빔을 나타내며 낮은 값이 더 선호되는 휴리스틱을 사용하는 경우이다.

가중치의 갱신은 탐색 오차가 발생했을 때만 이루어지기 때문에 위 식이 의미하는 것은 항상 일정하다. n*는 훈련 예제로 주어진 바람직한 노드(desired node)이며 현재 빔 B는 n*를 포함하지 않으므로 위 식에 의해 가중치는 ① 바람직한 노드의 휴리스틱 값은 낮추고 ② 빔에 있는 노드들의 휴리스틱 값은 높이는 방향으로 갱신됨을 의미한다. 가중치 갱신은 각 훈련 예제의 T 단계 동안 지속되며 하나의 훈련 예제에 대해 가중치 갱신이 종료되면 결과 가중치에 의해 학습을 계속할 지의 여부를 결정한다. 이를 반영하여 그림 3의 탐색에 의한 학습 알고리즘을 완성한 주프로그램을 그림 4의 유사 코드로 나타내었다. 그림 4의 종료 조건에서와 같이 더 이상 사용할 훈련 예제가 없을 때 학습을 중지할 뿐 아니라 가중치의 변화가 어떤 임계치(threshold) 이하가 되면 중단하도록 하는데 이는 원칙적으로 가중치의 변화가 더 이상 발생하지 않은 경우에 학습을 중단하는 것을 의미한 것이다.

학습 속도 변수인 α의 선정에 있어서는 α를 너무 크게 하면 하나의 훈련 예제에 의한 학습에서 가중치의 변화가 크게 일어나 학습 중단의 첫 번째 조건을 만족하지 못하는 경우가 발생하고 α를 너무 작게 하면 가중치의 변화가 작게 일어나 주어진 훈련 예제 안에서 가중치 학습이 완성되지 않는 경우가 발생한다. 휴리스틱 학습을 위한 적절한 α는 실험적으로 정하도록 한다.

## 4. 시뮬레이션 결과

본 논문에서 휴리스틱 학습에 의해 추구하고자 하는 바를 단계적으로 보여주는 시뮬레이션 결과가 그림 5이다. 그림 5 (a)는 최단 거리 경로이며 (b)는 기존의 거리 요소에 위협 요소의 존재를 추가했을 때의 훈련 예제로서 먼거리로 우회하더라도 위협 요소를 피하는 경로를 찾아 가도록 한 것이다. 그림 5 (c)와 (d)는 탐색에 의한 학습을 진행한 후, 우회 경로를 찾은 결과인데 (d)의 경우는 학습이 과도하게 되었을 때, 탐색 과정의 효율성이 떨어질 뿐 아니라 경로의 품질도 저하되는 경우를 보여준다. 이 예로부터 학습을 언제 중단하는가가 좋은 가중치를 결정하는데 매우 중요한 요소라는 것을 알 수 있다. 본 논문에서는 가중치의 변화가 더 이상 일어나지 않는 시점을 학습 중단 시점으로 정의하였으나 주어지는 훈련 예제 수에 따라 학습의 횟수는 제한되었다. 우선 α와 빔폭에 의한 영향을 분석하기 위하여 그림 5의 예제와 같은 형태의 휴리스틱에 10개의 훈련 예제를 제공하고 학습을 실행하였다.

α는 학습 속도를 조정하는 요소이며 보통 실험적으로 정해진다. α 값의 결정이 학습 성능의 요인들에 어떤 식으로 영향을 주는지 살펴보았다. 그림 5와 같은 시뮬레이션을 통해 적절한 학습이 이루어지는 α의 범위에서 0.01, 0.05, 0.1을 선택하였을 때 다음 세 가지 요인을 비교하였다. 첫째는 학습을 달성하기 위해 소요되는 훈련 예제수이며 둘째는 경로 탐색을 위한 방문 노드수, 마지막으로 탐색에 의해 구해진 경로의 길이인데 그 결과는 그림 6과 같다. α값이 0.01인 경우의 값들을 기준 1로 하고 α값이 0.05, 0.1인 경우의 비율을 나타내었다. α값이 증가함에 따라 훈련에 소요되는 예제 수는 줄어들며 경로를 탐색하는 동안 방문하는 노드수와 찾아진 경로의 길이도

```
HeuristicLearn (Tdata, b)
// Tdata is a training data set, {(xi,yi)}
wold ← 1/n          // n is a number of features
repeat
        wnew = LearningbySearch((xi,yi), wold, b)
until (|wnew-wold| < threshold) or Tdata is empty
```

그림 4. 탐색에 의한 학습의 주프로그램

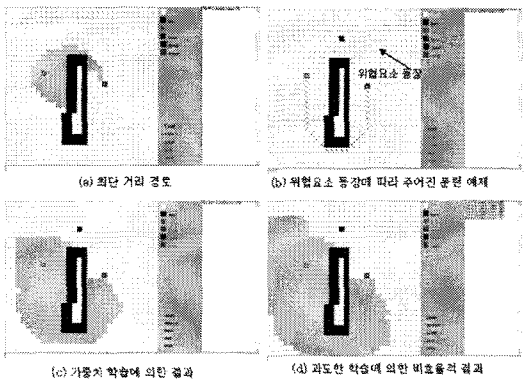


그림 5. 기존 경로 찾기의 결과와 가중치 학습 결과의 비교

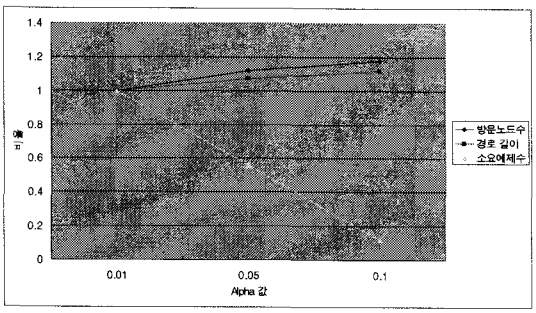


그림 6. α 값에 따른 학습 성능 비교

증가함을 볼 수 있다. 이는 α값에 따른 결과의 패턴을 보여줄 뿐이며 문제에 따라 α의 범위가 변할 수 있고 비율값도 달라질 수 있다. α값이 커지면 훈련 속도가 빨라지는 것은 자명한 일이다. 그러나 주어진 훈련 예제에 대해 탐색을 하는 동안에는 학습을 중단하지 않으므로 큰 α값에 의하여 가중치의 변화가 과도하게 일어날 수 있는 것이 단점이 된다. 그 결과, 학습된 가중치에 의해 탐색하는 성능이 급격히 저하되는 경우가 있다. 그림 6에서 방문 노드수와 경로 길이는 학습된 가중치에 의해 예제를 돌린 결과의 평균이며 최악의 경우에는 방문 노드수는 2배에 가까운 차이를 보이기도 한다. α값이 커짐에 따라 학습에 소요되는 예제의 수가 급격히 줄어드는 비율에 비하면 방문 노드수나 경로의 길이는 급격하게 늘어나지 않는다. 그러나 α값이 0.05 이상이 되었을 때, 지적한 바와 같이 과도한 학습에 의해 성능이 급격히 떨어지는 문제의 발생이 생기기 시작하므로 α값을 0.01로 하여 이후의 실험을 진행하였다.

기존의 빔 탐색 알고리즘에서도 빔 사이즈의 결정은 탐색의 성능을 좌우하는 매우 중요한 요인이 된다. 빔 사이즈를 작게 하면 방문 노드수를 줄이고 답을

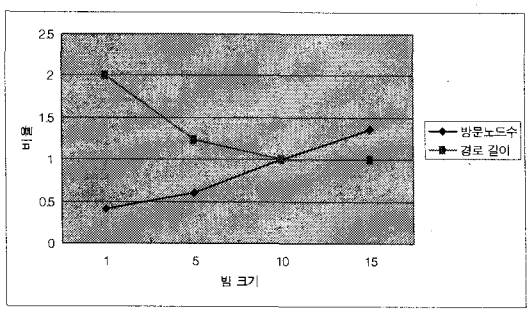


그림 7. 빔 크기에 따른 학습 성능 비교

찾을 수 있지만 알고리즘의 완전성(completeness)을 희생해야 하는 단점이 있으며 반면에 빔 사이즈를 너무 크게 하면 반대로 완전성을 보장하는 대신 탐색 시간이 오래 걸리거나 공간 자원이 너무 많이 소요되어 아예 실행이 불가능하게 되는 경우가 발생하는 단점이 있다. 본 논문에서는 학습을 위한 탐색 시에 빔 크기를 15에서 5씩 줄여가며 탐색의 성능을 비교하여 보았다(그림 7). 빔 사이즈가 1인 경우는 가장 극단적인 경우이다.

빔 크기가 10인 경우를 기준 값으로 하여 방문 노드수와 경로 길이를 상대적인 비율로 표시하였다. 빔 크기가 작아질수록 방문 노드수는 감소하여 빔 크기가 1인 경우에는 절반 이하로 크게 감소한 반면, 경로의 길이는 빔 크기가 커질수록 감소함을 알 수 있다. 즉, 빔 크기가 커질수록 최적에 가까운 경로를 찾아낸다는 것이다. 그러나 이는 빔 크기가 일정 수에 다다르면 더 이상 나아지지 않음을 볼 수 있다. 소요 예제수를 비교하지 않은 이유는 실제 소요 예제수가 크게 차이가 나지 않을 뿐 아니라 빔 크기가 1이나 5와 같이 상태 공간에 비해 크게 작아지면 학습이 실행된 이후에도 사용자의 의도와 다른 잘못된 경로를 찾는 경우가 발생하기 때문에 학습을 달성하는 의미가 없기 때문이다.

이와 같이 학습에 영향을 주는 요인들에 대한 분석을 바탕으로 RTS(real-time strategy) 게임의 제작에 사용된 셀-기반 편집기를 사용하여 생성한 환경에서 시뮬레이션 하였다. 셀-기반 편집기를 이용하여 제작된 게임들도 다양한 상태 공간 표현이 가능하나 [12] 본 논문에서는 셀 하나를 격자 하나로 대응한 격자-기반(grid-based) 표현을 채택하였다. 이와 같이 하면 기존의 편집기를 통해 생성된 자료 구조를 최소로 수정하여 전술 정보를 저장하는 것이 가능하다. 휴리스틱은 목표 지점까지의 거리(DTG, Distance to goal), 위협으로의 가시성(Visibility to threat, VTT), 위협으로부터의 거리(Distance to threat, DTT), 지형의 종류(Type of terrain, TOT)와 차폐 정도(degree of cover, DOC) 등, 5가지 전술 정보의 가중치 합으로 표현하였으며 작은 휴리스틱 값이 선호되도록 하기 위하여 위협으로부터의 거리는 역을 취하였으며 6가지 지형의 종류를 이동하기 쉬운 것을 1로, 차폐 정도가 가장 좋은 것을 1로 하여 각각 1~6의 정수 값을 배정하였다.

표 1. 학습 후의 가중치

| | DTG | VTT | DTT | TOT | DOC |
|---|---|---|---|---|---|
| 캐릭터 A | 1.53 | 0.81 | 2.77 | 0.45. | 1.03 |
| 캐릭터 B | 2.01 | 0.42 | 1.47 | 1.06 | 0.45 |

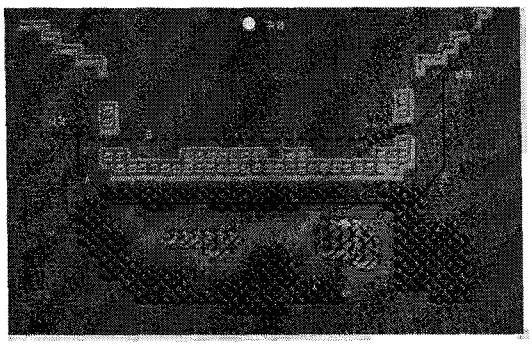

그림 8. 실제 게임에의 적용 예

캐릭터 A를 연락병, 캐릭터 B를 포병대라고 가정하고 각각에 바람직한 경로 예제 10개씩을 주어 가중치를 학습하게 하였으며 각 전술 정보에 대한 가중치가 어떻게 변화되는지 확인하였다. 이 시뮬레이션에서 학습속도 α는 0.01로 빔 크기는 10으로 세팅하였다. 초기치를 0.2로 일정하게 한 후 학습한 가중치는 표 1과 같으며 이에 따라 탐색된 경로의 예는 그림 8과 같다. 이 가중치들은 언제든지 필요에 따라 재조정될 수도 있다. 예를 들어 그림 8에서 캐릭터 B는 위협으로부터의 충분한 거리가 있으면 가시성이 있어도 경로로 채택하였으나 위협으로부터의 가시성 있는 경로를 택한 것이 적절하지 않다고 판단될 경우, 이를 보완할 만한 훈련 예제를 제공하고 학습함으로서 다른 결과를 가져 올 수도 있다.

## 5. 결 론

본 논문에서는 탐색과 학습을 융합한 효과적인 휴리스틱 학습 방법을 이용하여 전술적 경로 찾기에 대한 연구를 소개하였다. 이 방법으로 여러 가지 전술 정보를 고려한 경로를 캐릭터 특성에 맞게 찾을 수 있음을 시뮬레이션을 통하여 확인하였다. 전술적 경로 찾기를 위해 처음으로 기계학습을 이용했다는 것이 본 연구의 의의인 한편 처음 시도인 만큼 여러 연구 과제가 남아있다.

우선 게임의 상태 공간 표현은 그리드 기반으로 했는데 현재 게임 개발에 사용되는 상태 공간 표현 방식 중에는 시간 및 공간 복잡도 면에서의 효율성을 강조한 표현 방식이 많다[13]. 가시성 그래프나 경로점 그래프 등을 예로 들 수 있는데 이와 같은 효율적인 상태 공간 표현 방식을 이용하기 위해서는 특성값 저장 및 계산 방식에 연구가 별도로 필요하다. 예를 들어 그리드 방식의 경우에는 자식 노드들이 바로 인접해 있기 때문에 지형의 종류나 차폐정도와 같은 특성값을 도달 지점 노드에 국한해서 생각할 수 있다. 그러나 가시성 그래프나 경로점 그래프와 같이 노드들이 멀리 떨어져 링크로 연결되는 경우에는 도달 지점 노드의 특성값만을 고려하면 문제가 될 수 있기 때문에 링크가 걸쳐 있는 노드들의 특성값을 모두 고려하는 방법을 고안해야 한다.

또 한가지 중요한 문제점은 퍼셉트론 갱신 방법에 의한 가중치 조정이다. 이 방법에 의하면 가중치는 탐색 오차가 발생하였을 때만 바람직한 노드의 휴리스틱 값은 낮추고 그렇지 않은 노드들의 휴리스틱 값은 높이는 방향으로만 갱신되므로 빔 크기에 따라 잘못된 탐색 오차가 발생하면 가중치를 반대로 복구할 수 있는 방법이 없을 뿐 아니라 탐색 오차가 발생하지 않으면 아예 학습이 되지 않는 문제점이 있다. 이 문제점의 해결을 위해 부정(negative) 훈련 예제를 이용한다거나 탐색 오차가 발생할 때 뿐 아니라 근사 오차가 발생할 때에도 가중치를 갱신하도록 수정된 퍼셉트론 갱신 방법을 사용하는 등의 연구가 필요하다. 이 밖에도 특성이 많아지면 모든 가중치를 계산하는 대신 관계있는 특성만 선택해서 학습하도록 전처리 단계를 추가하여 탐색과 학습의 성능을 향상시키는 연구도 필요하다.

## 참 고 문 헌

[1] I. Millington, "Tactical and Strategic AI," Artificial Intelligence for Games, pp. 473-562, Morgan Kaufmann, 2006.

[2] H. Daume III and D. Marcu, "Learning as Search Optimization: Approximate Large Margin Methods for Structured Prediction," ICML-05, pp. 169-176, 2005.

[3] Y.Xu, A.Fern, and S.Yoon, "Discriminative

Learning of Beam-Search for Planning," In Proceedings of International Joint Conference on Artificial Intelligence, pp. 2041-2046, 2007.

[4] R. Straatman, W. van der Sterren., and A. Beij, "Killzone's AI: dynamic procedural combat tactics," In Proceedings of Game Development Conference, 2005.

[5] L. Liden, "Strategic and Tactical Reasoning with Waypoints," In: S. Rabin (eds.): AI Game Programming Wisdom, Charles River Media, pp. 211-220, 2002.

[6] A.Kamphuis, M. Rook, and M.H. Overmas, "Tactical Path Finding in Urban Environment," In Proceedings First International Workshop on Crowd Simulation, pp. 51-60, 2005.

[7] M. Rook and A. Kamphuis, "Path Finding using Tactical Information" In Poster Proc. Eurographics/ACM SIGGRAPH Symposium on Computer Animation, pp. 18-19, 2005.

[8] B. Stout, "Smart Moves: Intelligent Path finding," Game Developer Magazine, pp. 28-35, Apr., 1996.

[9] J. Matthews, "Basic A* Pathfinding Made Simple," In: S. Rabin (eds.): AI game Programming Wisdom, Charles River Media, pp. 105-113, 2002.

[10] S. Woodcock, "Game AI The State of the Industry," Game Developer Magazine Aug., pp. 34-43, 2000.

[11] S. Rabin, "A* Speed Optimizations and A* Aesthetic Optimizations," In: M. Deloura, (eds.): Game Programming Gems. Charles Rive Media, pp. 264-287, 2000.

[12] K Yu, "Finding a Natural-Looking Path by Using Generalized Visibility Graphs," PRICAI-2006, pp. 170-179, 2006.

[13] P. Tozour, "Search Space Representations," In: S. Rabin (eds.):AI Game Programming Wisdom 2, Charles Rive Media, pp. 85-102, 2004.

유 견 아

1982년~1986년 서울대학교 공과대학 제어계측공학과 학사

1986년~1988년 서울대학교 공과대학 제어계측공학과 석사

1988년~1989년 한국통신공사 사업지원단

1989년~1995년 미국 USC Computer Science 박사

1996년~덕성여자대학교 컴퓨터공학부 교수

관심분야 : 인공지능, 로봇 알고리즘, 연산 기하학