Dynamic Programming을 적용한 트리구조 미로내의 목표물 탐색 알고리즘

Target Object Search Algorithm under Dynamic Programming in the Tree-Type Maze

이동훈 · 윤한얼 · 이동욱 · 심귀보
중앙대학교 전자전기공학부

Dong-Hoon Lee, Han-Ul Yoon, Dong-Wook Lee, and Kwee-Bo Sim
School of Electrical and Electronic Engineering, Chung-Ang University
E-mail :sky52929@wm.cau.ac.kr

요 약

어떤 미로환경 내에서 로봇이 스스로 목표물을 찾기 위해서는 탐색경로를 결정하는 알고리즘이 요구된다. 본 논문에서는 'Y'형 미로에서 목표물을 탐색하기 위하여 Dynamic Programming을 적용한 미로 탐색 알고리즘을 제안한다. 실험에서는 규격화된 미로 블록을 만들고, 먼저 기존에 연구 되었던 좌수법 알고리즘을 자율이동 로봇에 적용해 "Y"형 미로 블록을 탐색하게 한다. 그리고 본 논문에서 제시한 Dynamic Programming을 이용한 미로탐색 알고리즘을 자율이동로봇에 적용하고 미로를 탐색한 후 이두가지 알고리즘을 적용한 로봇의 주행결과를 각각 비교해 봄으로서 Dynamic Programming을 적용한 자율이동로봇의 미로탐색 방법의 성능을 확인한다.

1. 서론

현재 로봇 시스템은 공장 자동화 및 우주탐사, 군사 등 광범위한 분야에서 응용되고 있다. 특히 퍼지, 신경회로망, 유전 알고리즘, 강화학습 등의 인공지능 이론은 로봇에게 자율성을 부여하여 로봇 혼자서도 주어진 임무를 수행할 수 있는 단계에 이르렀다. 이동 로봇은 21세기에 가장 성장할 것으로 예상되는 분야 중 하나이지만, 아직 실생활과 산업계에서는 큰 비중을 차지하지 못하고 있다. 이러한 현실의 가장 큰 원인은 agent가 수많은 불확실성을 다룰 수 있는 지능을 확보하지 못하고 있기 때문이다 [1][2].

본 논문에서는 상기에 언급한 것과 같은 문제점을 보완 하는 한 가지 방법으로 자율 이동 로봇의 알고리즘에 Dynamic Programming(DP)을 사용하여 'Y'형 미로 블록을 탐색 하는 DP를 적용한 미로탐색알고리즘을 제안한다. 본 논문에서 제안한 DP를 적용한 미로탐색 알고리즘의 실험을 위해 소형 이동 로봇을 제작하였고, 제작된 이동 로봇에 좌수법과 DP를 적용한 미로탐색 알고리즘을 각각 적용하여 목표물 탐색을 하고, 이 실험을 통한 결과를 통하여 DP를 적용한 미로탐색 알고리즘의 성능을 확인하였다.

2. Dynamic Programming의 개념 및 실제 사용 예

2.1 Dynamic Programming

Dynamic Programming(DP)의 특징은 첫째로 문제의 순환적인 성질을 이용한다. DP는 큰 문제를 이루는 작은 문제들을 먼저 해결하고 작은 문제들의 최적해를 이용하여 순환적으로 큰 문제를 해결한다. 둘째 DP에서는 중복된 계산을 하지 않는다. DP는 문제의 순환적인 성질 때문에 이전에 계산되어졌던 작은 문제의 해가 다른 어

딘가에 필요하게 되는데 이를 위해 DP에서는 이미 해결된 작은 문제들의 해를 저장 공간에 저장하게 된다. 그리고 이렇게 저장된 해들이 다시 필요할 때 마다 해를 얻기 위해 다시 문제를 재계산하지 않고 테이블의 참조를 통해서 중복된 계산을 피하게 된다. 셋째로 DP가 어느 문제에나 적용될 수 있는 것은 아니다. 주어진 문제가 최적화의 원칙을 만족해야만 DP를 적용할 수 있다 [3][4][5][6].

2.2 DP의 실제 사용 예

일반적으로 Dynamic Programming(DP)은 최적의 value function, 최적의 policy function, recurrence relation과 boundary condition으로 이루어져 있다. 그림 1은 간단한 cost path problem을 설명하는데 도움이 되는 실례이다.

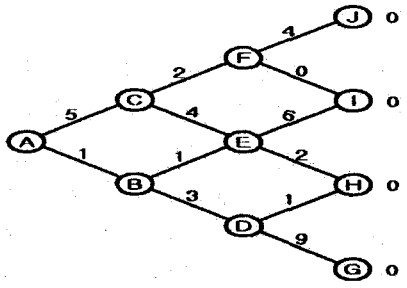

그림 1. 간단한 비용 경로 문제.

우리는 agent가 action을 선택하여 minimum cost를 축적함으로서 goal state $g \in G \in S$ 로 주행 하도록 agent를 학습 시켜야 한다. 여기서 G는 goal state의 집합, S는 모든 state의 집합, g는 goal이다. 그림 1에서 최적의 value function $V(s)$는 = '*state s*에서 *goal g*로가는 최소의 *value*'이다. 순환관계에 의하여 state s의 minimum cost는 가장 잘 선택 된 action a의 cost와 같다 [3]. 그리고 다음 state의 minimum cost는 action a에 의하여 지시되어 진다.

$$V(s) = \min_{a \in actions} cost(s,a) + V(next(s,a)),\ \forall s \in S - G \qquad (1)$$

위 수식에서 $cost(s,a)$는 action a에 의해 지시되는 state의 cost를 나타내고, $next(s,a)$는 action a 에 의하여 지시하는 state s의 다음 state의 cost를 나타낸다 [4]. goal state value를 정의하면 다음 식과 같다.

$$V(s) = 0, \forall s \in G. \qquad (2)$$

방정식 (2)는 boundary condition을 나타낸다. 최적의 policy function $\pi(s) = a\ such\ that\ V(s) = \min a \in actions, cost(s,a) + V(next(s,a))$ [5].

우리가 찾고자 하는 목표지점인 goal을 그림 1에서 Ⓗ, start position을 Ⓐ라고 가정하자. 첫 번째 state에서 agent가 Ⓐ에서 Ⓑ로 action을 취한다면, 그것은 방정식 (1)과 (2)에 의하여 다음과 같이 산출되어 진다.

$$\begin{aligned} V(s_0) &= \min_{actions} cost(s_0, down) + V(next(s_1, a_{next})) \\ &= \min_{a \in action} 1 + V(next(s_1, a_{next})) \end{aligned} \qquad (3)$$

여기서, s_0, s_1, s_2, s_3 는 S의 원소이다. $V(s_0)$를 정의하기 위해서 $V(next(s_1, a\ next))$를 결정하는 것이 필요하다. 두 번째로 agent가 Ⓑ에서 Ⓔ로 이동한다면 식 (4)와 같이 산출 된다.

$$\begin{aligned} V(s_1) &= \min_{a \in actions} cost(s_1, up) + V(next(s_2, a_{next})) \\ &= \min_{a \in actions} 1 + V(next(s_2, a_{next})) \end{aligned} \qquad (4)$$

세 번째로, agent가 Ⓔ에서 Ⓗ로 이동한다면,

$$\begin{aligned} V(s_2) &= \min_{a \in actions} cost(s_2, down) + V(next(s_3, a_{next})) \\ &= \min_{a \in actions} 1 + V(next(s_3, a_{next})) \end{aligned} \qquad (5)$$

마지막 state가 0값이면 agent는 종결조건을 만족하기 때문에 s_3를 goal state로 인식한다.

$$V(s_3) = 0 \qquad (6)$$

따라서 방정식 (3), (4), (5)에 따라 $s_0 \rightarrow s_3$ 까지의 minimum cost는 식 (7)과 같다.

$$V(s_0 \rightarrow s_3) = 1 + 1 + 2 + 0 = 4. \qquad (7)$$

우리가 Ⓗ를 goal state로 가정하였기 때문에 agent의 학습은 완료 된다.

3. Dynamic Programming를 적용한 미로탐색 알고리즘의 실험

3.1 실험 환경 및 시나리오

본 논문의 실험에서 agent로 시작점에서부터 goal 탐색을 시도 하였다. Dynamic Programming(DP)을 적용한 미로탐색 알고리즘으로 agent가 미로를 탐색 할 때 종결지점에서 목표물인 goal 지점과 그 이외의 종결지점을 구분 할 수 있도록 goal 지점에는 흰색을 칠하고 그 이외의 종결지점에는 모두 검정색을 칠하였다. 미로 벽은 흰색 하드 보드지를 사용 하였다.

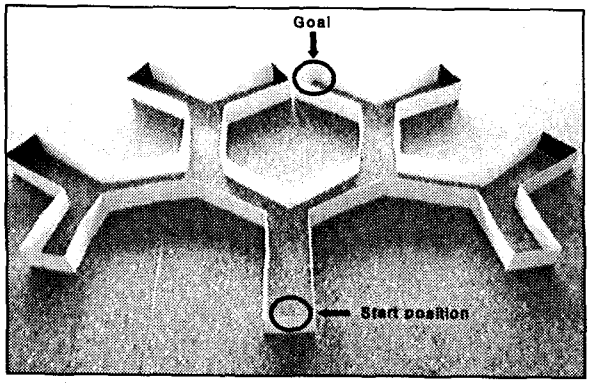

그림 2. 미로환경 외관.

Dynamic Programming(DP)을 적용한 미로탐색 알고리즘으로 agent가 미로를 탐색 할 때 종결지점에서 목표물인 goal 지점과 그 이외의 종결지점을 구분 할 수 있도록 goal 지점에는 흰색을 칠하고 그 이외의 종결지점에는 모두 검정색을 칠하였다. 미로 벽은 흰색 하드 보드지를 사용 하였다.

3.2 실험결과

첫 번째로, 좌 수법을 agent에 적용하여 목표물 탐색을 시도하였다. 좌 수법은 왼쪽 벽을 무조건 따라가기 때문에 상당한 시간을 소요한 후에 결국에는 goal 지점에 도달할 수 있었다. 두 번째로, Dynamic Programming(DP)을 agent에 적용하여 미로탐색 실험을 하였다.

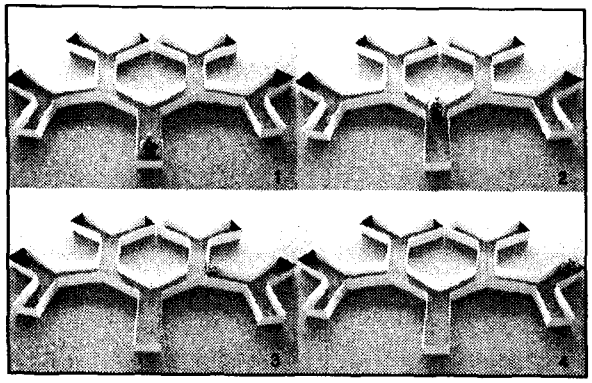

그림 3. DP를 적용한 goal 탐색.

이동 로봇이 시작점에서부터 탐색을 하면서 각 state마다 랜덤하게 action을 취하고 이 action에 의해 지시되어진 다음 state의 값에 따라 현재 위치의 state와 이전의 state사이의 action에 대해 점수를 부여하고 결국에는 골 지점을 찾아 갈 수 있었다. 세 번째로, 처음 좌수법으로 미로탐색을 마친 로봇으로 미로탐색을 다시 실험하였다. 예상했던 대로 처음 실험과 동일한 방향으로 동일한 시간을 소비하고 결국에는 목표지점을 찾아갔다. 마지막으로, DP를 적용하여 목표물을 탐색을 완료한 agent를 시작점에서 다시 구동 하여 실험을 하였다.

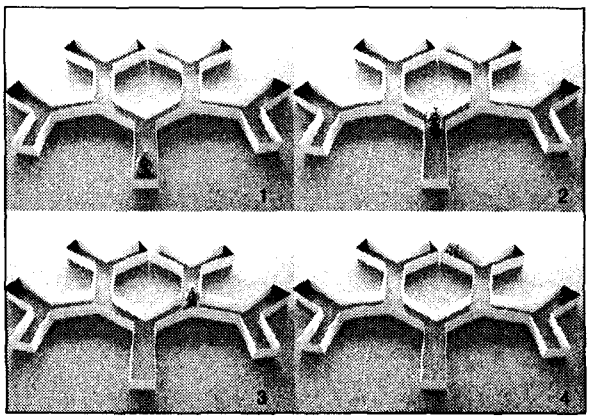

그림 4. DP를 적용한 goal 탐색완료.

DP를 적용한 agent는 이전에 실험을 통하여 현재 미로내의 각 state와 action의 값을 모두 저장하고 있기 때문에 'Y'형 미로내의 state와 state의 구역별 최적의 부분 주행로를 결정하고 이 부분 주행로들의 결합으로 agent가 가장 최적화된 주행을 할 수 있었다. 그림 4는 agent가 DP를 사용하여 목표물을 탐색 한 후 두 번째로 목표물을 탐색 하였을 때의 모습을 보여준다.

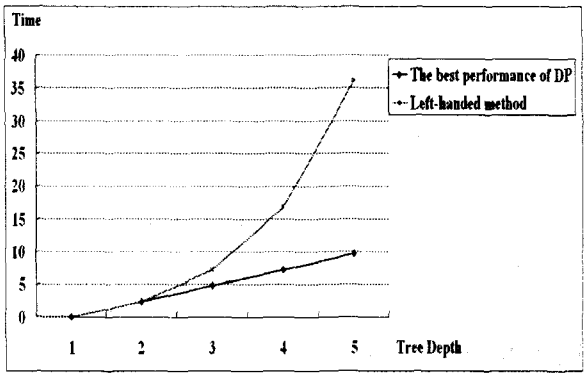

그림 5. DP와 좌수법의 성능비교.

이 실험에서 확인한 바와 같이 각각의 알고리즘이 goal 지점을 찾은 후 두 번째 탐색을 할 때에는 좌수법을 사용한 agent는 처음과 동일한 방법으로 동일한 시간을 소비하며 골 지점을 찾는 반면에 DP를 적용한 미로탐색 알고리즘으로 미로를 탐색한 agent는 한번 지나간 자리의 값을 모두 기억하고 있기 때문에 시작점에서 출발하자

마자 곧바로 goal 지점을 찾아 가는 것을 확인할 수 있었다. 이번 실험 에서 좌수법과 DP을 적용한 agent의 첫 번째 실험에서는 DP를 적용한 agent가 좌수법을 적용한 agent보다 빠른 시간에 목표지점에 도달할 수 있었지만 처음 agent가 미로를 탐색 할 때, 모든 경우에 DP를 적용한 agent가 좋은 성능을 보이는 것은 아니다. DP를 적용한 agent가 갈림길에서 선택한 방향은 랜덤하게 선택한 방향이기 때문에 첫 주행 시에는 좌수법보다 더 나쁜 성능을 보일 때도 있었다. 하지만 본 논문에서 제안한 'Y'형 미로내의 목표물 탐색 알고리즘의 가장 핵심적인 부분은 처음 미로탐색을 마친 agent의 다음 주행의 성능을 평가하는 것으로 이 부분에 대해서는 충분히 만족할 만한 결과를 얻었다.

그림 5는 'Y'형 블록이 증가함에 따라 좌수법을 적용한 agent와 DP를 적용한 agent의 성능을 비교 분석한 것이다. 좌수법을 적용한 agent는 'Y'형 미로가 증가 할수록 탐색 시간은 2에 지수승으로 증가 하였다. 첫 주행에서는 DP를 적용한 agent도 좌수법과 비슷한 시간을 소요하면서 'Y'형 미로를 탐색 하였다. DP의 첫 주행에서 action을 선택하는 방법이 랜덤 선택 방법이기 때문에 경우에 따라 좌수법보다 효율이 좋을 수도 있고 나쁠 수도 있다. 하지만 그림 5에 나타난 것처럼 두 번째 주행을 했을 때에는 좌수법을 이용한 방법과 DP를 이용한 방법은 현격히 차이가 났다. 좌수법을 사용한 agent는 처음 주행한 것과 동일한 결과가 나오지만 DP를 사용한 agent는 시작점에서 목표물까지의 가장 빠른 길로 주행을 하였다. 트리의 개수가 2^n-1 개 증가할 때의 예상 시뮬레이션을 해 본 결과 각각의 알고리즘을 적용한 agent의 처음 탐색 시간은 트리 개수가 2^n-1 개 증가 할 때마다 2에 지수승으로 증가 하였다. 하지만 두 번째 탐색에서는 좌수법을 적용한 agent는 이전과 동일한 시간을 소비한 반면 DP를 적용한 agent는 한 depth만 증가 할 뿐이었다.

4. 결론 및 향후과제

본 논문에서는 선험적 지식이 없는 'Y'형 미로 공간에서의 goal 탐색을 위해 Dynamic Programming(DP)을 적용한 미로탐색 알고리즘을 제안하였다. 또한 실험을 위해 실제 제작된agent에 DP와 좌수법 알고리즘을 적용하여 실험을 하였고 이 실험의 결과를 통해 본 논문에서 제안한 DP를 적용한 미로탐색 알고리즘의 성능을 확인할 수 있었다. 본 논문에서는 DP를 적용한 알고리즘을 보다 쉽게 설명하기 위하여 이진트리로 미로를 구성하여 실험을 하였지만 본 논문에서 제안한 방법을 이용하면 어떤 구조의 미로든지 목표지점을 찾아 가는 것이 가능 하다.

감사의 글 : 본 연구는 과학기술부의 뇌신경정보학연구사업의 '뇌 정보처리에 기반한 감각정보 융합 및 인간행위 모델 개발'의 연구비 지원으로 수행되었습니다. 연구비 지원에 감사드립니다.

5. 참고문헌

[1] 이원창, 옥진삼, 강근택, "웹을 이용한 이동로봇의 원격제어," *한국동력기계공학회지*, vol. 4, no. 4, pp. 78-83, 2000.

[2] 정완균, "이동로봇의 위치인식기술," *한국통신학회지*, vol. 21, no. 10, pp. 67-75, 2004.

[3] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*, Prentice Hall, 1997.

[4] C. W. Xu, "Fuzzy model identification and self-learning for dynamic systems," *IEEE Trans. Syst. Man. Cybern.*, Vol. 17, No. 4, pp. 683-689, 1987.

[5] H. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. on Sys. Man and Cybern.*, Vol. 15, pp. 116-132, 1985.

[6] J. H. Holland, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley: MA, 1989.

[7] H.-U. Yoon, S.-H. Whang, D.-W. Kim, and K.-B. Sim, "Strategy of cooperative behaviors for distributed autonomous robotics systems," *Proc. of the 10th International Symposium on Artificial Life and Robotics*, pp. 151-154, 2005.

[8] H.-U. Yoon and K.-B. Sim, "Hexagon-based Qlearning for object search with multiple robots," *Proc. of the 1st International Conference on Natural Computation*, pp. 713-722, 2005.

[9] 이동훈, 김대욱, 심귀보, "DARS 로봇에서의 영상전송 시스템 개발," *한국퍼지 및 지능 시스템 학회*, vol. 15, no. 1, pp. 229-232, 2005.