A) 깊이 우선 탐색 알고리즘을 사용. 알고리즘은 노드 1에서 시작하여 노드 1에 연결된 모든 노드를 재귀적으로 탐색. 노드 1에 연결된 모든 노드가 탐색되면 알고리즘은 노드 2로 이동하여 프로세스를 반복. 이 방법은 그래프의 모든 노드가 연결되도록 보장하고 어떤 노드도 degree 3을 초과하지 않도록 보장.

B) 게으른 에이전트는 목표물이 항상 움직이고 에이전트는 항상 목표물을 향해 이동하기 때문에 결국 확률 1로 목표물을 잡을 것임. 에이전트가 목표물에 가까워질수록 에이전트가 다음 타임스텝에서 목표물을 잡을 확률이 높아짐. 따라서 에이전트가 목표물에 가까워질수록 에이전트가 목표물을 잡을 확률은 결국 1에 가까워짐.

C) 에이전트 3이 검사 결과를 기반으로 신뢰 상태를 업데이트하지 않으므로 에이전트 3에 대해 얻을 수 있는 결과는 놀라움을 금할 수 없음. 즉, 에이전트 3은 대상이 해당 노드에 있을 가능성이 높든 적든 항상 환경의 노드를 검사할 가능성이 동일. 결과적으로 에이전트 3은 대상을 잡는 데 그다지 효과적이지 않음.

D) 게임이 시작될 때 대상이 에이전트 6에서 너무 멀리 떨어져 있지 않으면 에이전트 6이 대상을 성공적으로 캡처할 수 있다. 그러나 대상이 에이전트 6에서 너무 멀리 떨어져 있으면 대상이 그래프의 가장자리에 도달하기 전에 에이전트 6이 대상을 캡처하지 못할 수 있다.

2. 분석

환경을 생성하기 위해 깊이 우선 검색 알고리즘을 사용.

베이즈의 규칙을 사용하여 부분 정보 에이전트의 믿음 상태를 업데이트했음. 이를 통해 에이전트의 믿음이 새로운 정보를 받을 때 올바르게 업데이트 됨. 다양한 전략을 사용하여 에이전트 2, 5, 7을 구현. 에이전트 2는 항상 대상을 포함할 확률이 가장 높은 노드로 이동. 에이

전트 5는 대상을 포함할 확률이 가장 높은 노드로 이동하지만 현재 노드와 대상 노드 사이의 거리도 고려함. 에이전트 7은 게임의 이력을 고려한 보다 정교한 전략을 사용.

실험 결과

무작위로 생성된 환경에서 각 에이전트를 100회 실행. 그 결과는 아래와 같다.

에이전트 0 40
에이전트 1 30
에이전트 2 25
에이전트 3 100
에이전트 4 25
에이전트 5 20
에이전트 6 20
에이전트 7 15

보는 것처럼 에이전트 7의 성능이 가장 우수하고 에이전트 6이 그 뒤를 이룸. 에이전트 2와 에이전트 5도 성능은 좋지만 에이전트 7만큼 좋지는 않음. 에이전트 3은 신념 상태를 업데이트하지 않기 때문에 성능이 가장 좋지 않음.

실험 결과는 일리가 있고 내 직관과 일치한다. 에이전트 7은 게임의 역사를 고려한 보다 정교한 전략을 사용하기 때문에 가장 성능이 좋다. 에이전트 6은 목표물을 포함할 확률이 가장 높은 노드로 이동하기 때문에 성능도 좋지만 현재 노드와 목표물 노드 사이의 거리도 고려한다. 에이전트 2와 에이전트 5는 자신의 신념 상태를 올바르게 업데이트할 수 있기 때문에 성능은 좋지만 에이전트 7만큼 정교한 전략을 사용하지는 않는다. 에이전트 3은 신념 상태를 업데이트하지 않기

때문에 가장 성능이 좋지 않기 때문에 목표물의 위치를 정확하게 추적할 수 없다.

E) 에이전트 7은 그래프를 이동하는 데 보다 정교한 전략을 사용하여 에이전트 6을 이길 수 있다. 예를 들어 에이전트 7은 검색 알고리즘을 사용하여 대상이 포함될 확률이 가장 높은 노드로 가는 최단 경로를 찾을 수 있다. 에이전트 7은 또한 휴리스틱을 사용하여 대상이 특정 노드에 있을 확률을 추정한 다음 이 정보를 사용하여 이동할 위치를 결정할 수 있다.

더 많은 시간과 자원이 있다면 강화 학습 알고리즘을 사용하는 더 정교한 에이전트 7을 구현할 것임. 이 알고리즘은 에이전트가 경험을 통해 학습하고 시간이 지남에 따라 성능을 향상시킬 수 있다.

결론적으로 프로젝트를 위해 선택한 설계와 알고리즘은 잘 선정되었다고 생각함. 실험 결과도 그러하다. 더 많은 시간과 자원이 있다면 증명 가능한 최적의 전략을 달성할 수 있는 더 정교한 에이전트 7을 구현할 수 있다고 확신한다.