한국컴퓨터정보학회 하계학술대회 논문집 제19권 제2호 (2011. 6)

멀티에이전트 전략을 위한 방향벡터 활용과 동적 환경에 적응하는 경로 추천시스템에 관한 연구

윤석현°
°청강문화산업대학 컴퓨터정보과
e-mail : shyoon@ck.ac.kr

A research on utilizing direction vector and course recommendation system adapting dynamic environment for multi agents strategy

Seok-Hyun Yoon°
°Dept. of Computer Information, Chungkang College of Cultural Industries

● 요 약 ●

본 논문은 사용자 및 동적환경의 변화를 파악하고 분석된 정보를 바탕으로 최적화된 경로를 제공하기 위한 시스템을 멀티에이전트를 이용하여 해결하고자 하였다. 멀티에이전트를 통해 설정된 목표를 찾아가는 먹이추적 문제에 적용하였고 현실 세계와 흡사한 무한 공간 환경에서 알고리즘의 성능을 실험하였다. 적용된 환경의 모델은 순환구조(circular)형 격자 공간이라는 새로운 실험 공간으로 방향 벡터 함수 알고리즘을 통해 새롭게 멀티에이전트의 목표를 획득하기 위한 해법이다. 기존의 연구와 비교하여 먹이의 효율적 포획, 에이전트간의 충돌문제 해결에 대한 새로운 해법을 제시할 수 있었다.

키워드: 멀티 에이전트(Multi Agent), 먹이추적 문제(Prey Pursuit Problem), 방향 벡터(Direction Vector)

I. 서론

유비쿼터스의 핵심기술인 개인화정보 서비스는 사용자에게 최적화된 정보를 제공하는 것으로 우선 개인의 특성을 고려한 프로파일 값이 필요하다. 개인의 프로파일 값 생성 방법에는 여러 가지가 있는데 사용자의 활동데이터(이동경로, 움직임, 조작)를 누적하여 데이터를 추출하는 방법과 유사한 성향을 가진 기존의 사용자들의 데이터를 바탕으로 추천하는 방법들이 있다.

통신기술 및 장비의 발달로 양방향 데이터 송수신 서비스가 수월하고 다양한 데이터 수집 센서들이 개발됨에 따라 본 논문에서는 다양한 환경변수를 고려하여 개인에게 가장 최적화된 경로를 제공할 수 있는 시스템을 제안하고자 한다. 설정된 기본 데이터를 바탕으로 동적 환경정보, 운전자의 특성들을 n개의 에이전트라 하고 이 에이전트들은 찾고자 하는 먹이(목표, 데이터 결과)에 대입하여 멀티에이전트의 목표획득을 위한 연구 환경과 해법을 제안해 본다.

기존 먹이추적 문제의 실험 환경은 격자 공간(grid) 상에 4개의 에이전트와 1개의 먹이가 존재하고, 에이전트들이 협동을 통해 먹이를 포획하여 에피소드를 끝내게 된다[3]. 먹이는 에이전트가 도달하고자 하는 목표이며, 각 에이전트는 최소의 비용을 통해 효율적으로 목표 획득을 하는 것이 멀티에이전트 연구의 목표라 할 수 있다. 그러나 기존 실험 환경은 30×30의 격자 구조로 이루어져 현실성이 결여되어 있으며, 에이전트의 먹이 포획만을 가지고 해법을 찾고자 하였다.

본 논문에서는 순환구조(circular)형 격자 공간이라는 실험 환경을 제안하여 복잡하고 넓은 현실세계와 비슷하게 표현하고자 하였고, 에이전트와 에이전트, 먹이와 에이전트 사이의 관계를 방향 벡터로 표현하고, 거리, 위치, 방향성 등을 고려한 학습을 통하여 새로운 휴리스틱을 만들 수 있도록 하였다. 또한 먹이에게도 에이전트와 동일한 능력을 설정하여 에이전트가 보다 진보된 학습을 할 수 있는 방법을 제안하였다.

II. 관련 연구

1. 관련 연구

다양하고 복잡한 환경에서 사용자의 요구와 문제 해결을 위해 에이전트가 개발 되어졌으며, 그 결과 단일 에이전트로 해결하지 못하는 복잡한 문제에 대한 해결 방안을 위해 멀티 에이전트간의 협동을 통한 멀티 에이전트 시스템(multi-agent system)이 제안되었다[1,2,3,5]. 효율성 있는 멀티 에이전트의 대표적인 실험모델인 먹이추적문제는 현실세계를 표방하는 격자 공간(grid)내에 4개의

독립적 에이전트가 하나의 먹이를 포획하는 실험으로서 복잡한 현실 세계를 표현하기 위해 M. Benda에 의해 제안되었다[2].

Ⅲ. 본론

1. 방향벡터를 이용한 먹이 포획전략과 탈출전략

기존 포획 전략들의 문제는 에이전트들이 한곳에 모여 있을 때의 상황이다. 무한 공간 상태에서 먹이는 에이전트의 반대 방향으로 이동시 분명 포획이 불가능하다. 본 논문에서 제안한 새로운 실험환경은 무한 공간 상태와 유사한 환경이기에 기존 연구들의 전략으로는 포획하기가 힘들다. 실험 환경 크기가 커질수록 효율성이 떨어지는 것을 알 수 있었다. 이런 점에서 본 논문에서는 먹이와 에이전트간의 거리관계를 방향벡터로 표현하고 이를 새로운 전략법으로 적용하여 구현하였다.

각 에이전트들은 먹이를 기점으로 서로의 상태 정보 값을 확인하고 먹이를 중심으로 이동한다. 먹이는 분명 에이전트를 피하기 위해 에이전트와의 공간관계를 계산하고 에이전트들이 없는 상태 공간 쪽으로 이동을 한다. 초기 설정에서 에이전트와 먹이의 이동 속도가 같고 먹이가 에이전트보다 한 단계 먼저 이동할 때 포획이 상당히 어렵다는 것을 알 수 있다. 에이전트들은 먹이 포획을 위해 효율적으로 좇아 가야하는 것을 판단하는 함수를 만들고, 먹이는 근접상태의 공간을 파악할 수 있는 능력을 주어 도망할 수 있는 정책이 필요하다. 이 함수들을 만들기 위해 방향벡터를 도입하였다. 인접한 에이전트와 먹이와의 거리와 다른 에이전트들의 상관관계를 반영하는 벡터를 만든다. 에이전트는 기존의 연구들과 동일하게 $A_i \geq 4$로 설정하였다. 최소 에이전트 값을 4로 해야만 완전하게 먹이를 포획할 수 있기 때문이다.

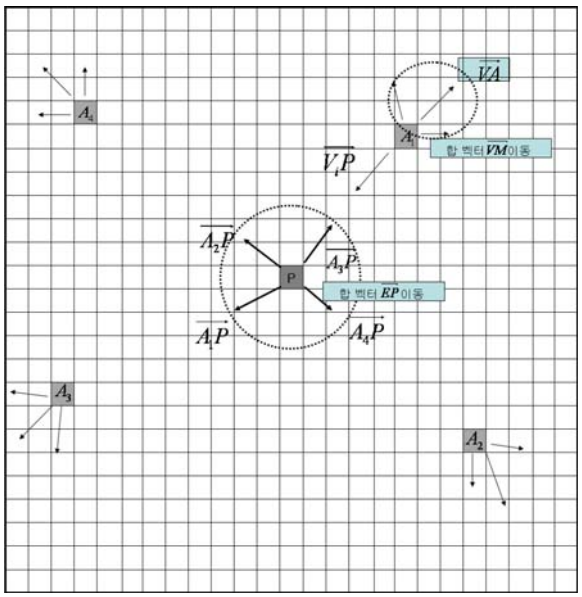

그림 1. 방향벡터를 이용한 전략
Fig. 1. Direction vector strategy

수식(1)은 에이전트(A_i)가 먹이(P)쪽으로 이동하는 방향벡터로 먹이와 에이전트와의 거리를 반영한 것이다. 각 에이전트와 먹이간의 방향 벡터 $\overrightarrow{VP}$의 크기는 먹이와 각 에이전트간의 거리에 반비례한다. 이것은 에이전트가 먹이와 가까울수록 먹이 방향으로 가려고 하는 에이전트의 벡터 값이 커져 에이전트의 이동 함수를 만들게 된다.

$$\overrightarrow{VP} = \overrightarrow{A_iP}$$

$$\rightarrow \overrightarrow{|A_iP|} = \frac{1}{(Px - A_ix)^2 + (Py - A_iy)^2} \qquad (1)$$

기존의 연구에서 가장 많이 제기되었던 문제는 에이전트간의 충돌 문제였다. 이것을 해결하고자 본 논문에서는 에이전트와 다른 에이전트간의 상관관계(거리와 먹이를 중심으로 발생하는 각도)를 고려한 벡터 함수를 만들었다. 다른 에이전트들과 거리가 가까워질수록 반대방향으로 값이 커지는 벡터 $\overrightarrow{VA}$를 생성하게 된다. 이것은 에이전트간의 일정한 거리를 유지하며 먹이에게로 접근할 수 있는 함수 값이 되며 이 값은 에이전트사이의 모든 방향 벡터의 합으로 만들어 진다.

/* Initialization phase */
Set an initial position for prey and 4 agents position.
/* Main algorithm */
Loop /* This loop is an iteration of the algorithm */
1. /* Initialization of weighted values α and β */
Initialize weighted values α and β for agents
α : agent to prey , β : agent to other agent
2. /* In this step prey calculates direction vector to move and moving */
$\overrightarrow{PE}$
Prey applies the prey escape vector using (Eq.1) to choose position to escape from each agents.
3. /* In this step agents calculates direction vector to prevent conflict with each agents */
$\overrightarrow{RA_i}$
Agents(Ai) applies the vector using (Eq.2) to prevent conflict with each agents.
4. /* In this step agents real moving for prey capture*/
Applies weighted value α and β to (Eq.3), and
$\overrightarrow{VA_i}$
Agents(Ai) applies there all moving vector using (Eq.4) to move the position to capture prey
Until (End_condition = True)

그림 2. 먹이포획문제 알고리즘
Fig. 2. Prey Pursuit Problem Algorithm

$$\overrightarrow{VA} = \sum_{i \in J(A)} \overrightarrow{R_i}$$

$$\rightarrow |\overrightarrow{R_i}| = \left(\frac{1}{(R_i x - Ax)^2 + (R_i y - Ay)^2} \right) \qquad (2)$$

$J(A)$는 에이전트(A)의 이웃 에이전트의 집합, R은 에이전트(A)와 이웃 에이전트, $\overrightarrow{R}$은 에이전트(A)와 에이전트(R)의 방향 벡터이다. 이것은 에이전트들 간의 거리가 가까울수록 반대 방향으로 커진다. 그리고 에이전트와 다른 에이전트와의 벡터 값은 두 에이전트 거리의 제곱에 반비례하는 값을 갖게 하면 되는데, 이것은 에이전트와 에이전트 사이를 멀어지게 하는 작용을 한다. 수식(2)를 통해 에이전트는 다른 에이전트들과 다른 방향으로 가려고 하는 함수가 만들어져 에이전트간의 충돌문제를 해결하며 다른 에이전트들과는 먹이를 잡는데 있어 공동작전을 효과적으로 펼 수 있게 된다..

에이전트의 이동방향을 결정짓는 함수는 수식(3)으로 생성된다. 에이전트는 $\overrightarrow{VP}$와 $\overrightarrow{VA}$를 합친 쪽으로 이동하면 먹이쪽으로 움직이게 된다. 이렇게 만들어진 에이전트의 이동 함수를 $\overrightarrow{VM}$이라 한다.

$$\overrightarrow{VM} = \overrightarrow{VP} + \overrightarrow{VA} \qquad (3)$$

여기서 $\overrightarrow{VM}$을 구할 때 단순히 합하지 않고 $\alpha(\overrightarrow{VP}) + \beta(\overrightarrow{VA})$ 형태로 변형시킨다. α, β 값은 기존의 학습 값으로 먹이를 추적하는 방향을 결정하는 중요한 환경변수가 된다. (먹이 쪽으로 이동하는 것이 방향의 다양성보다 중요하므로 일반적으로 $\alpha > \beta$ 으로 설정한다. α값은 기존의 ACS를 통해 획득할 수 있다[6]. ACS를 통한 α획득 알고리즘은 다음과 같다.

$$\alpha = \max(Q(s_t, a)), a \in A(s_t)$$

$$\overrightarrow{VM} = \alpha * \overrightarrow{VP} + \beta * \overrightarrow{VA} \qquad (4)$$

수식(4)를 통해 생성된 $\overrightarrow{VM}$ 벡터 함수로 에이전트의 이동 방향을, 벡터의 크기는 현재 좌표의 평가 값이 된다. 따라서 생성된 $\overrightarrow{VM}$을 가진 에이전트는 생성된 벡터 방향과 가장 가까운 셀(좌표)로 이동하는 것을 시도할 수 있다. 이동 후보들과 현재 위치 각각에 대해서 $\overrightarrow{VM}$을 구해서 제일 큰 $\overrightarrow{VM}$의 위치로 이동하게 된다. 여기서 고려해야 할 사항은 먹이나 에이전트는 반드시 움직이지 않아도 된다는 것이다. 제자리에 머물러 있는 것도 하나의 전략으로 $\overrightarrow{VM}$으로 생성 할 수 있다. 즉 생성된 $\overrightarrow{VM}$ 을 통해 에이전트가 이동할 방향이 현재 위치보다 나쁘다면 움직일 필요가 없는 전략인 것이다.

수식(5)는 먹이의 탈출을 위한 벡터 함수이다. 단, 먹이의 탈출을 결정하는 $\overrightarrow{EP}$는 에이전트의 학습능력을 결정하는 중요한 요소이기도 하다. 본 논문에서는 에이전트들이 벡터함수를 이용한 먹이추적을 하는데 있어 반복되는 실험을 통해 얻게 되는 강화학습방법을 활용하게 되는데 $\overrightarrow{EP}$는 학습능력을 향상시키는 요소이기도 하다. $\overrightarrow{EP}$는 α를 결정하는데 중요한 작용을 하는 것이다. $\overrightarrow{EP}$를 증가시키거나 감소시킴에 따라 에이전트의 학습능력이 변화된다.

$$\overrightarrow{EP} = \overrightarrow{A_1P} + \overrightarrow{A_2P} + \overrightarrow{A_3P} + \overrightarrow{A_4P} = \sum_i \overrightarrow{A_iP} \qquad (5)$$

$$\rightarrow |\overrightarrow{A_iP}| = \frac{1}{(Px - A_ix)^2 + (Py - A_iy)^2}$$

3. 실험결과

제안한 알고리즘의 성능 평가는 먹이 포획의 성공률(success rate), 각 에이전트의 상태 전이 수(number of transitions)를 기준으로 하였다. 새롭게 제시한 순환구조(circular)형 격자 공간을 통해 멀티 에이전트의 목표에 최대한 근접하기 위한 실험으로 먹이의 완전포획(100%포획)과 에이전트간의 충돌 방지를 동시에 할 수 있는 전략을 적용하여 기존의 연구들에서 제안된 알고리즘과 비교하였다. 본 실험 환경은 현실성을 고려하여 불완전 포획이라는 요소를 없애고 초기 먹이와 에이전트의 위치는 무작위로 주어지며 먹이의 완전포획을 기준으로 에이전트의 학습 값 α를 생성하였다. 표 1의 실험결과와 같이 먹이의 포획과 에이전트간의 충돌 방지, 학습을 통해 획득한 α의 증가에 따른 방향벡터를 고려한 알고리즘의 효율성을 입증 할 수 있었다. 상태공간이 커질수록 에이전트의 이동횟수가 증가하지만 이에 따른 α를 통한 학습이 효율적임을 알 수 있었다.

표 1. α값에 따른 먹이 포획 결과

Table 1. Result of Capture probability by α value

(α)	(β)	30×30		50×50		100×100	
		Capture probability	State transition	Capture probability	State transition	Capture probability	State transition
0.1	0.1	3%	107	1%	230	0.2%	872
0.2	0.1	42%	98	19%	251	8%	864
0.3	0.1	55%	91	26%	223	11%	820
0.4	0.1	78%	79	32%	210	23%	795
0.5	0.1	84%	67	54%	182	31%	757
0.6	0.1	92%	69	62%	171	42%	693
0.7	0.1	100%	63	75%	144	50%	610
0.8	0.1	100%	64	81%	130	62%	625
0.9	0.1	100%	61	92%	128	69%	612
1	0.1	100%	62	100%	111	75%	602

Ⅳ. 결 론

본 논문에서 제안한 순환구조(circular)형 격자 공간은 에이전트와 먹이의 순환구조를 통한 무한 공간 개념의 현실세계와 유사하게 표현하고자 한 것이다. 새로운 실험 환경에 맞는 방향벡터를 이용한 휴리스틱을 통해 에이전트간의 충돌 방지와 효율적인 먹이 포획을 입증할 수 있었으며, 현실에서 응용할 수 있는 멀티에이전트의 목표 획득에 대한 또 하나의 전략을 구현할 수 있었다.

기존 네비게이션 서비스의 획일화된 정보 활용이 아닌 동적 환경변수를 적용한 시스템을 방향 벡터를 이용한 먹이 추적 문제를 통해 접근하여 새로운 해법을 제시할 수 있었다.

본 논문에서 제안한 방향벡터를 이용한 포획 전략은 무작위 배열상태에서도 효율성을 검증할 수 있었지만 특정상황(초기 배열이 한쪽으로 몰리는 현상)에서는 먹이의 포획율이 낮아 이것을 해결할 수 있는 추가적인 연구가 필요하다. 또한 독립된 에이전트간의 먹이포획 전략인 비대화형 에이전트 연구가 필요하다.

참고문헌

[1] Peter Stone, Manuela Veloso, Multiagent coordination with learning classifier systems, In Proceeding of the AAAI 99 Workshop on Negotiation, pp. 44-49, 1999.

[2] M. Benda, V. Jagannathan and R. Dodhiawala, ‘On optimalcooperation of knowledge source-an empirical invarstigation,’ Technical Report BCS-G2010-28, Boeing Advanced Technology Center, Boeing Computing Services, Seattle, Washington, July, 1986

[3] Peter Stone and Manuela Veloso, “Multiagent System : A Survey from a Machine Learning,” Technical Report CMU-CS-97-193, The University of Carnegie Mellon, December-1997

[4] Hyun Kim, SeokHyun Yoon, TaeChoong Chung. A research on non-interactive multi agents by ACS & Direction vector algorithm, KSCI vol. 15, No. 12
pp. 11-18, December-2010

[5] T. Fuku, A. Namatame, and T. Kaizouji. Collective eciency in two-sided matching. In P. Mathieu,B. Beauls, and O. Brandouy, editors, Articial Economics: Agent-Based Methods in Finance, Game Theory and Their Applications, pages 115{126.Springer-Verlag, Berlin, Germany, 2006

[6] M. Dorigo, V. Maniezzo, and A.Colorni. The Ant System: Optimization by A Colony of Cooperating Agents. IEEE Transactions on Systems, Man, and Cybernetics–Part B, vol. 26, No. 2, pp. 29–41, 1996.