경영과학
제5권 제2호
1988. 12
pp.71-80

지식 베이스를 이용한 오목게임에 관한 연구 (A Study on Knowledge-Based Five-in-a-Row)

정 봉 주*
박 순 달*

ABSTRACT

The object of this thesis is to develop an effective method for Five-in-a-Row. For this purpose, we develop a method that describes a move more accurately, and a procedure that chooses a move more effectively.

In describing a move, we represent each move as eight vectors generated from four directions for both offensive and defensive strategies. Each vector consists of three components, that is, connectivity, agressiveness, and directions of stones in a line. In choosing a move, we introduce a preference order among all vectors by an expert's knowledge. Then, an efficient algorithm is developed to test a cycle in the preference order.

Experimental results show that the success rate of this new method is about 90% against Weisenbaum's program known as the prototype of the Five-in-a-Row program.

1. 서 론

오목은 19×19선으로 구성된 판위에서 두 명의 참가자가 흑과 백의 돌을 번갈아 놓으면서 게임이 진행된다. 이 게임은 한 경기자가 먼저 자기의 돌들을 수직, 수평 또는 대각선 방향으로 5개가 연결되게 하면 승리로 끝나게 된다.

오목 게임의 특징은, 게임의 진행상태를 완전히 알 수 있는 (perfect information) 게임이며, 유한번만에 끝난다. 체스, 바둑등도 이러한 게임의 종류에 속하나, 포카 게임은 상대방의 패가 드러나지 않으므로 이에 속하지 않는다. 따라서, 오목 게임은 정보의 부족으로 인한 확률적인 요소가 배제되며 승리를 위한 특정한 해가 존재한다[8].

그러나 이러한 해를 찾는다는 것은 체스나 바둑에서와 같이 가능한 모든 수(move)를 나열하여야 하므로 현재의 컴퓨터의 연산속도로서는 사실상 불가능하다. (오목 게임이나 바둑 게임의 경우 (19×19)! 개의 가능한 수가 존재한다) 결국 효율적인 발견적 기법(heuristics)이 필요한 것이다.

오목 게임에 대한 기존의 연구는 Joseph Weisenbaum[14,15]과 Deena Koniver에 의해 이루어졌는데, 이들 두 기법간의 차이점은 거의 없다[11]. 이 기법의 기본 개념은 현재 놓으려고 하는 수의 위치에 대한 가치를 선형평가함수(linear evaluation function)로 산출하여 가장 높은 값을 가지는 수를 다음의 컴퓨터의 수로 선택하는 것이다. 그러나 선형평가함수의 값이 현재 놓여

* 서울대학교 산업공학과

져 있는 상대방이나 자신의 돌의 갯수에 의해서만 거의 결정되므로 여러가지 게임상황들을 적절히 반영하지 못할 뿐 아니라, 항상 같은 형태의 선형평가함수를 사용하므로 게임 수준에 한계를 보이고 있다.

일반적으로 게임 참가자는 그 게임에 대한 자신의 경험과 지식에 의하여 여러가지 가능한 수(move)들을 비교하여 가장 우선 순위가 높은 수를 다음수로 선택한다. 따라서 이 논문에서는 오목 게임에서의 수(move)들을 그때의 게임 상황을 정확히 반영할 수 있게 표현하고, 전문가의 게임지식을 활용할 수 있는 방법을 제시하고자 한다. 이를 위하여

첫째, 수의 가치를 보다 정확하게 표현하는 방법과 이에 따른 지식 베이스(Knowledge -Base)의 구성,

둘째, 지식 베이스의 탐색을 통한 수의 선택 기법,

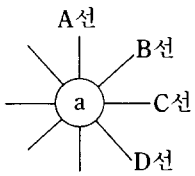
셋째, 지식 베이스의 유용성 검사를 위하여, 유방향 그래프에서 Cycle의 존재여부 검사를 하는 효율적인 해법

등을 연구하고자 한다.

2. 수의 표현과 선택

수(手)의 표현 : 수(move)란 특정한 게임 상황에서 참가자가 취할 수 있는 가능한 행동이며, 수의 위치는 수의 선택 결과로서 일대일 대응되는 바둑판상의 위치이다. 따라서 수의 가치는 곧 수의 위치에 대한 가치를 의미한다. 수를 표현한다는 것은 수의 가치를 판단할 수 있는 방법을 제공한다는 것이며 결국 수의 위치에 대한 가치를 나타내는 방법이 된다. 오목 게임에서 수의 위치에 대한 가치에 영향을 미치는 요소는 돌의 연결상태, 연결된 돌의 막힘상태, 연결된 돌의 방향, 연결된 돌의 공격상태등을 들 수 있다. 이 중에서 연결된 돌의 방향은 [그림 1]의 네 가지 선(line)중의 하나가 된다.

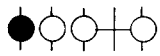
결국 하나의 수는 A, B, C, D 네 선상에서 돌의 연결상태, 막힘상태, 공격상태를 반영하여 표현될 수 있다.



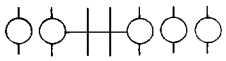
[그림 1] a수에 대한 네가지 선(line)

먼저 돌의 연결상태를 나타내기 위하여 연결선(chain)이라는 개념을 도입한다. 연결선이란 하나의 선(A,B,C,D선 중)상에서 연결된 같은 돌의 집합을 뜻하며, 가장 인접한 두 돌 사이에는 최대 하나의 빈 위치만 있을 수 있다. 이 연결선중에서 연결선상이 두 돌 사이에 빈 위치가 없는 연결선을 강연결선(strong chain, SC)이라고 하며, 연결선상에 적어도 하나의 빈 위치가 있는 연결선을 약연결선(weak chain, WC)이라고 한다. 그리고 연결선상의 돌의 갯수를 연결값(chain value, CV)이라고 한다.

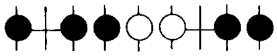
따라서, 연결값에는 강연결값(strong chain value, SCV)과 약연결값(weak chain value, WCV)이 있다. [그림 2]는 C선상에서의 여러가지 연결값들을 보여주고 있다.



(a) 백 : WCV=3
흑 : CV=1



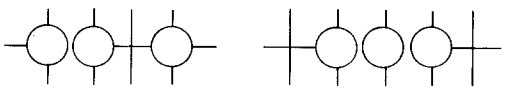
(b) 좌측 : SCV=2
우측 : SCV=3



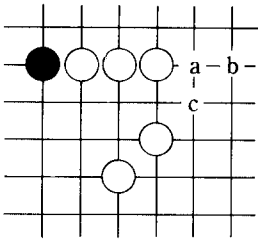
(c) 백 : SCV=2 흑(좌) : WCV=3
흑(우) : SCV=2

[그림 2] 여러 가지 연결값

여기서 약연결선이 필요한 이유는 먼저, 강연결선과 같은 가치의 약연결선이 존재하기 때문이다. [그림 3]의 (a)에서 만일 약연결선의 개념이 없다면 강연결값은 2, 1 두가지가 존재하지만 사실상 (b)의 강연결선과 같은 가치를 지닌다. 따라서, 이러한 상황은 약연결선이 있음으로서 표현가능하다. 즉, (a)의 약연결값은 3이다. 다음으로 연결선의 가치를 정확하게 반영할 수 있기 때문이다. [그림 3]의 (c)에서 a, b, c수에 대한 백의 강연결값은 c선상의 SCV(a)=4, 모든 선상의 SCV(b)=1, B선상의 SCV(c)=3이므로 b수가 가장 나쁜 수로 판단된다. 그러나 b수는 사실상 승리 수이다. 즉 약연결선으로 생각하면 c선상의 WCV(b)=4, B선상의 WCV(b)=3이므로 정확한 가치를 반영할 수 있다.



(a) (b)



(c)

[그림 3] 약연결선의 필요성

다음, 특정한 수의 위치 (i,j)에 대한 표현벡터를 다음과 같이 나타낸다.

(CV(i,j), D(i,j), L(i,j))

여기서,

(i,j) : 바둑판상의 좌표. i,j=1,2,…,19

CV(i,j): (i,j)에 돌을 놓을 경우의 연결값

$1 \leqq SCV(i,j) \leqq 5$ (식 2.1)

$1 \leqq WCV(i,j) \leqq 4$ (식 2.2)

CV(i,j)=max[SCV(i,j), WCV(i,j)]

D(i,j) : (i,j)에 돌을 넣을 경우, 연결선의 막힘상태

=1, 연결선의 양쪽이 막히지 않은 상태

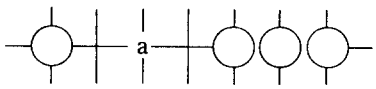
=0, 연결선의 한쪽만 막히지 않은 상태

L(i,j) : 선(line)의 종류

=1, B또는 D선

0, A또는 C선

(식 2.1)은 SCV≧6일 경우 수로서의 가치가 없기 때문에 성립한다. (식 2.2)는 WCV≧5일 경우 이는 승리를 의미하는 것이 아니며, 또한 수의 가치를 반영할 수도 없기 때문에 성립한다. [그림 4]에서 a수에 대한 WCV=5라고 할 수 있으나 이는 승리를 의미하는 것이 아니다. 따라서 a수를 중심으로 좌측의 WCV=2, 우측의 WCV=4로 나누어 결국 CV=4로 하는 것이 타당하다.



[그림 4]

또한, 표현벡터는 공격전략 표현벡터와 수비전략 표현벡터로 나눌 수 있는데 공격전략 표현벡터란 빈 위치 (i,j)에 컴퓨터의 돌을 둘 경우, 컴퓨터의 돌에 대해서 도출되는 표현벡터를 말하며, 수비전략 표현벡터란 빈 위치 (i,j)에 참가자(player)의 돌을 둘 경우, 참가자의 돌에 대해서 도출되는 표현벡터를 말한다.

공격(또는 수비)전략 표현벡터는 하나의 빈 위치 (i,j)에 대해 각 선(line)상에서 1개씩 최대 4개가 도출될 수 있다(양쪽이 막힌 상태의 연결선은 표현벡터를 도출하지 않음)

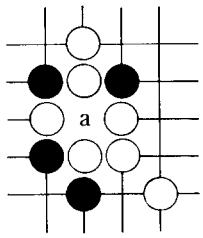
이상에서, 하나의 수(i,j)는 공격과 수비전략

표현벡터들로 최대 4개씩, 총 8개의 표현벡터들로서 표현한다. [그림 5]의 a 수는 다음 5개의 표현벡터로서 표현된다.

(백 : 컴퓨터)

공격전략 표현벡터 : (4, 0, 0), (3, 1, 0)
(3, 0, 1)

수비전략 표현벡터 : (3, 1, 1), (2, 0, 1)



[그림 5]

지식 베이스의 구성 : 지식 베이스(knowlege-basa)의 역할은 수를 표현할 수 있고, 표현된 수에 따라 다음의 가장 좋은 수를 선택할 수 있는 제반 지식을 제공하여야 한다. 따라서, 수를 표현하는 표현벡터의 내용에 관한 지식이 우선 필요하다. 다음으로, 수의 우선관계를 판가름하기 위해서는 표현벡터들간의 우선관계를 나타내는 지식이 필요하다. 이 우선관계지식은 자명하게 결정할 수 있는 우선관계가 대부분이나 그렇지 않은 경우는 전문가의 지식을 참고로 한다.

●표현벡터를 나타내는 지식(KB1)

연결값(CV)	막힘상태(D)	선의종류(L)	index

하나의 수를 표현하기 위해 최대 8개의 표현벡터들을 항상 주기억장치(main memory)에 저장하려면 많은 용량이 필요하므로 대응되는 index만을 찾아서 저장함으로서 사용되는 기억용량을 절약할 수 있고, 또한 우선관계 지식을 표현하고 저장하는 데도 아주 효율적이다. 최대 Record의 수는 5×2×2=20개가 된다.

●공격전략 표현벡터들간의 우선관계지식(KB2)

index 1	index 2

여기서, index1, index2는 KB1의 index들이며 index1이 index2에 우선한다. 공격전략 표현벡터들간에 우선표현벡터를 결정하고자 할 때 사용된다. pair compairison으로 우선관계지식을 구성한 것은 하나의 file를 이루는 record의 길이는 일정하여야 하고, 또한 pair comparison으로 모든 표현벡터들간의 우선관계를 결정할 수 있기 때문이다. 최대 Record의 수는 ${}_{20}C_2=190$개가 된다.

●수비전략 표현벡터들간의 우선관계지식(KB3)

index 1	index 2

여기서, index1, index2는 KB1의 index들이며 index1이 index2에 우선한다. 수비전략 표현벡터들간에 우선표현벡터를 결정하고자 할 때 사용된다. 최대 Record의 수는 ${}_{20}C_2=190$개가 된다.

●공격과 수비전략 표현벡터들간의 우선관계지식(KB4)

index1	index2	우선 index

여기서, index1은 공격전략 표현벡터, index2는 수비전략 표현벡터이며 우선 index는 이 들중 우선되는 표현벡터를 의미한다. 공격과 수비전략 표현벡터들간에 우선 표현벡터를 결정하고자 할 때 사용된다. 최대 Record의 수는 20×20=400개가 된다.

지식 베이스 KB2, KB3, KB4의 경우 표현벡터들간에 모순된 우선관계가 있는지를 검사하여야 한다. 먼저, 표현벡터들간의 우선관계는 유방향그래프로 표현할 수 있는데, 표현벡터를 vertex로, 우선관계를 arc로 표현한다. 즉 표현벡터 1과 2가 있을 경우 1 → 2는 표현벡터 1이 표현벡터 2에 우선함을 의미한다. 이러한 그래프를 선호 그래프(preference Graph)라고 한다. [6]. 따라서 우선관계의 모순이란 곧 선호 그래프상에 Cycle이 존재하는 것을 말한다. 또한 선호 그래프가 연결된 그래프(connected graph)가 아닐 경우, 연결되지 않은 vertex들간에는 우선관계를 결정할 수 없다. 이상에서 지식 베이스의

유용성은 두가지 사항을 검사함으로서 결정된다.

먼저, 선호 그래프상에서 Cycle의 존재여부를 검사하여 Cycle이 존재하지 않을 경우, 지식 베이스는 유용하다. 해법은 3절에서 다룬다. 다음으로 선호 그래프의 연결성(connectivity)을 검사하여 선호 그래프가 연결된 그래프(connected graph)일 경우 지식 베이스는 유용하다. 무방향 그래프의 연결성(connectivity)검사를 위한 기존의 해법은 참고 문헌 ([2], [6], [7])에 수록되어 있으므로 선호 그래프의 유방향 arc를 무방향 arc로 변환시킨 다음 적용하면 될 것이다. 만일, 지식 베이스 KB2, KB3, KB4 중 어느 하나라도 유용하지 않을 경우는 미리 구성된 표현벡터들 간의 우선관계를 이용한다.

후보수의 결정 : 후보수(candidate move)는 현재 돌이 놓여지지 않은 모든 위치가 될 수 있다. 그러나, 게임 상황에 유리한 위치만을 고려함으로서 불필요한 후보수를 탐색하지 않도록 다음 규칙에 의하여 후보수를 결정한다.

규칙 1. 가장 최근의 컴퓨터와 참가자(Player)의 두 수를 중심으로 강연결선(SC)의 양쪽 빈 위치를 후보수의 위치로 한다.

이는 가장 최근의 수가 곧 현재의 가장 중요한 게임 상황을 반영한다는 일반적인 사실에 근거한다.

규칙 2. 연결값 즉, CV≧3인 표현벡터를 가지는 위치를 후보수로 한다.

단, CV=3인 경우는 양쪽이 열린 상태이어야 한다.

이는 게임 승패에 결정적인 수는 항상 고려하고자 하는 것이다.

우선 표현벡터의 결정해법 : 후보수들 중 가장 좋은 수를 선택하기 위해서는 각 후보수들로부터 도출된 표현벡터들 간에 가장 좋은 표현벡터를 결정하여 그 표현벡터를 포함하는 수를 선택하면 된다. 따라서, 여기서는 주어진 표현벡터들 중 우선 표현벡터를 결정하는 해법을 제시하고자 한다. 먼저, 지식 베이스들은 모순된 우선관계가 없어 유용하다고 가정하자. 지식 베이스 KB2, KB3, KB4에 대해 각각 선호 그래프 G2, G3, G4를 구성하여 놓는다. 이때, 선호 그래프는 모순된 우선관계가 없으므로 무순환 유방향 그래프(acyclic directed graph)가 된다. 이상의 선호 그래프들은 게임을 시작하기전에 지식 베이스 들을 탐색하여 구성하여 놓고, 매 번 수를 선택할 때마다 도출된 표현벡터들 중 우선 표현벡터를 결정하여야 할 때 필요한 선호 그래프 하나를 이용한다. 우선 표현벡터의 결정해법은 다음과 같다.

단계 1. zero in - degree vertex의 탐색

이용하고자 하는 선호 그래프를 G라고 하자. 유방향 그래프에서 Cycle의 존재여부를 검사하는 해법인 EXIST - CYCLE (제 4 장 참고)을 G에 대하여 수행하면서 주어진 표현벡터들 중 zero in - degree vertex가 생기면 단계 2.로 간다. (zero in - degree vertex : 들어오는 arc가 없는 vertex)

단계 2. 우선 표현벡터의 결정

경우 1. zero in - degree vertex가 하나일 때

그 vertex가 우선 표현벡터가 된다.

경우 2. 두 개 이상의 zero in - degree vertex가 존재할 때

G상의 다른 vertex까지 가장 긴 경로를 가진 zero in - degree vertex가 우선 표현벡터가 된다. (두 개 이상의 우선 표현벡터가 선택되면 이들 중 임의로 하나를 선택한다.)

zero in - degree vertex를 선택하는 이유는 들어오는 arc가 없다는 것이 곧 가장 우선 표현

벡터이기 때문이며 경우2의 선택방법은 보다 많은 우선 횟수를 가진 표현벡터를 선택하고자 함이다.

수의 선택해법 : 컴퓨터가 매 번 수를 선택하는 절차는 다음과 같다.

단계 1. 후보수의 결정

후보수의 결정 규칙에 따라 후보수들을 결정하여 후보수 목록 L=[1,2,…,n]을 작성한다. (편의상 각 후보수를 1,2,…,n이라고 부르기로 한다.) 이 때, 후보수 i는 공격전략 표현벡터 목록 Li(i)와 수비전략 표현벡터 목록 L2(i)로 표현된다.

단계 2. 각 후보수별 우선 표현벡터 결정

목록 L의 모든 후보수 i에 대하여 전략별 우선 표현벡터 목록인 O, D를 작성한다.

즉, $O=\sum_{i\in L}$ {L1(i)중 우선 표현벡터} : KB2탐색

$D=\sum_{i\in L}$ {L2(i)중 우선 표현벡터} : KB3탐색

가 되는 공격전략 우선 표현벡터 목록 O와 수비전략 우선 표현벡터 목록 D를 작성한다.

단계 3. 전략별 최우선 표현벡터 결정

공격전략 우선 표현벡터 목록 O에서 최우선 표현벡터 p를 (KB2 탐색), 수비전략 우선 표현벡터 목록 D에서 최우선 표현벡터 q를 (KB3 탐색) 결정한다.

단계 4. 최종 우선 표현벡터 결정

표현벡터 p와 q중 최종 우선 표현벡터 k를 결정한다. (KB4 탐색)

단계 5. 수 선택

최종 우선 표현벡터 k를 포함하는 후보수를 선택한다.

경우.1 선택된 후보수가 하나인 경우

이 후보수가 컴퓨터의 선택 수가 된다.

경우.2 선택된 후보수가 두 개 이상인 경우

선택된 후보수들의 표현벡터 목록에서 표현벡터 k를 제거한 다음, 이 후보수들만으로 후보수 목록 L을 작성한다. 단계.2로 간다.

만일 선택된 후보수들의 표현벡터들이 모두 같다면, 이 후보수들 중 임의의 수를 선택한다.

3. 유방향 그래프상에서의 Cycle의 존재여부 검사 해법

G=(V,A)를 유방향 그래프라고 하면 G상의 어떤 vertex에 대해 나가는 arc를 out-degree arc, 들어오는 arc를 in-degree arc라고 한다. 또한 나가는 arc가 없는 vertex를 zero out-degreee vertex, 들어오는 arc가 없는 vertex를 zero in-degree vertex라고 한다. 먼저 Cycle의 존재에 관한 다음 정리를 살펴보자.

(정리 1) G상에 zero in-degree vertex 또는 zero out-degree vertex가 존재하지 않으면, G상에 적어도 하나의 Cycle이 존재한다.

증명 : i) G상에 zero out-degree vertex가 없을 경우

이 경우, G상의 모든 vertex들은 zero in-degree vertex이거나 in-degree arc나 out-degree arc를 모두 가진 vertex이다. G상의 모든 vertex들에 1,2,…,n이라고 이름을 붙이면, 임의의 vertex p(1)에서 출발하여 다른 vertex를 방문하는 최대 길이의 경로 p=(p(1), p(2),…,p(k))를 구성할 수 있다. p(k)에서는 이미 방문한 vertex이외의 다른 vertex로 갈 수 없게 된다. 그런데 p(k)는 zero out-degree vertex가 아니므로, 경로 P상의 어떤 vertex p(k)는 제외)로 연결되는 out-degree arc가 존재한다. 이는 곧 G상에 Cycle이 존재하는 것을 의미한다.

ii) G상에 zero in - degree vertex가 없을 경우

이 경우, G상의 모든 vertex들은 zero out - degree vertex이거나 in - degree arc와 out - degree arc를 모두 가진 vertex이다. G상의 임의의 vertex p(1)에서 역방향으로 다른 vertex를 방문하는 최대 길이의 경로를 구성하면 p(1)←p(2)←……←p(k)가 된다. 이때, 경로 P=(p(k), p(k−1),…,p(1))이 된다. 이 때 p(k)에 연결된 in - degree arc는 경로 P상에 있는 vertex이외의 다른 vertex에 연결된 arc가 될 수 없다. 그런데 p(k)는 zero in - degree vertex가 아니므로, 경로 P상의 어떤 vertex(p(k)는 제외)에 연결된 in - degree arc가 존재한다. 즉, G상에 Cycle이 존재하게 된다.

iii) G상에 zero in - degree, zero out - degree vertex가 모두 없을 경우

이는 i), ii)의 경우에 포함된다. Q.E.D.

정리 1은 G가 Cycle이 있는 유방향 그래프가 되기위한 충분조건을 제시하고 있다. 이로부터 다음 정리가 성립한다.

(정리 2) G상에 zero in - degree vertex와 zero out - degree vertex가 없는 subgraph G'가 존재하는 것은 G상에 Cycle이 존재하기 위한 필요충분조건이다.

증명 : 충분조건 : 정리 1의 iii)의 경우에 해당하므로 성립.

필요조건 : G상에 Cycle이 존재하므로 Cycle을 구성하고 있는 vertex들과 그 vertex들에만 연결되어진 arc들로 이루어진 subgraph G'=(V', A')를 구성할 수 있다. 그러면 G'상의 임의의 vertex p(1)에 대해 p(1)>p(2)>…>p(k)>p(1)이 되는 Cycle을 형성할 수 있다. (여기서, [p(1), p(2),…,p(k)]⊂V') 따라서, G'상의 모든 vertex들은 zero in - degree vertex나 zero out - degree vertex가 될 수 없다. Q.E.D.

정리 2로부터 유방향 그래프 G에서 Cycle의 존재여부에 대한 검사는 곧 G상에서 zero in - degree vertex와 zero out - degree vertex가 없는 subgraph G'가 존재하는가를 알아보는 것과 같은 문제임을 알 수 있다.

이상의 정리에 따라 유방향 그래프 G상에서 Cycle의 존재여부에 대한 검사해법을 서술하면 다음과 같다.

단계 1.G상에 zero in - degree vertex 또는 zero out - degree vertex i가 존재하면 단계 2로 간다.

그렇지 않으면 G에는 Cycle이 존재한다. →끝

단계 2.G상에는 vertex i와 vertex i에 연결된 모든 arc들을 제거하여 새로운 유방향 그래프 G를 구성한다.

단계 3.G상에 vertex가 존재하지 않으면 G는 무순환 유방향 그래프(acyclic directed graph)가 된다.→끝

그렇지 않으면 단계 1로 간다.

해법의 개요는 G상에서 zero in - degree vertex 또는 zero out - degree vertex가 존재하지 않을 때까지 그 vertex들을 제거해 나가서 정리 2의 subgraph G'가 존재하는지를 알아보는 것이다.

분석을 위하여 SPARKS언어로 해법을 표현하면 다음과 같다.

```
procedure EXIST−CYCLE(A, n) ;
//input
  A(i, j) : adjacency matrix of directed graph
        G,
  n       : number of vertices//
1 for i ← 1 to n do
```

```
2  vertex(i) ← 0 ; in - degree(i) ← 0 ; out - de-
   gree(i) ← 0 ;
3  for j ← 1 to n do
4  if A(i, j) = i then out - degree(i) ← out - de-
   gree(i) + 1 ;
5  if A(j, i) = 1 then in - degree(i) ← in - deg-
   ree(i) + i ;
6  end
7  end
   // in degree(i) : number of in - degree arcs
   of vertex i,
   out degree(i) : number of out - degree arcs
   of vertex i//
8  for i ← 1 to n do
9  if vertex(i)=0 then [if out - degree(i)=0
                        then call CUT(i, 0) ;
10                      if in - degree(i)=0
                        then call CUT(i, 1)]
11 end
12 for 1 ← 1 to n do
13 if vertex(i) = 0 then [cycle ← true
                          //cyclic directed
                          graph//
14                        go to label 17]
15 end
16 cycle - false // acyclic directed graph //
17 end EXIST-CYCLE

   procedure CUT(i, v) ;
     vertex ← 1 ;
     for j ← 1 to n do
       case
       ; v = 0 : if A(j, i) = 1
                 then [out-degree(j] ← out -
                       degree(j) - 1 ;
                       if out - degree(j) = 0
                       then call CUT(j, 0)]
       ; v = 1 : if A(i, j) = 1
                 then [in - degree(j) ← in -
                       degree(j) - 1 ;
                       if in - degree(j) = 0
                       then call CUT(j, 1)]
     end
   end CUT
```

procedure EXIST-CYCLE의 수행시간은 다음 정리와 같다.

(정리3) procedure EXIST-CYCLE의 수행시간은 $O(n^2)$이다.

증명 : line 1 - line 7에서 2개의 for loop를 수행하는데 소요되는 시간은 $O(n^2)$이다.
line 8의 for loop은 G상의 모든 vertex들을 한 번씩 방문하는 것을 의미한다. 만일 방문한 vertex가 zero in - degree 또는 zero out - degree vertex이면 procedure CUT을 호출하여 제거한다.
procedure CUT은 한번 호출 될 때마다 O(n)의 시간이 소요된다. 한번 제거된 vertex에 대해서는 CUT이 호출되지 않으므로 결국 line 8 - line 11에서의 총 수행시간은 CUT의 수행 시간 O(n)과 모든 vertex를 방문하는데 O(n)시간이 소요되어 총 $O(n^2)$이 된다. line 11 - line 15는 하나의 for loop이므로 수행시간은 O(n)이다. 따라서, procedure EXIST - CYCLE의 총 수행시간은 $O(n^2)$이다. Q.E.D.

그래프상에서 Cycle이 존재하는지를 검사하는 해법은 Roberts와 Flores의 모든 Cycle을 찾는 해법 [10], Warshall의 transitive closure해법 [1] 등을 이용할 수 있고, 무방향 그래프인 경우에는 depth - first search 해법 [7]이 있다. 다음 [표 1]은 기존의 해법들과의 수행시간과 기억용량을 비교한 것이다.

[표 1] 해법의 비교분석

해 법	수행시간	소요기억용량
RF	NP-complete	$O(n^2)$
Warshall	$O(n^3)$	$O(n^2)$
EXIST-CYCLE	$O(n^2)$	$O(n^2)$

RF : Roberts와 Flores의 모든 Cycle을 찾는 해법

Warshall : Warshall의 transitive closure 해법

n : 그래프상의 vertex의 수

무방향 그래프인 경우 이용할 수 있는 depth-first search해법의 수행시간과 기억용량은 각각 $O(n^2)$이다.

4 결론

본 연구에서는 오목 게임을 위한 효율적인 발견적 기법을 개발하였다. 이 발견적 기법의 특징은 정성적으로 수의 가치를 평가하고, 수를 선택하는 인간의 보편적인 게임 방식을 체계화 시킨 것이다.

이를 위하여 본 논문에서 연구된 사항은 다음과 같다.

첫째, 오목 게임에서의 수의 표현방법을 개발하였다.

수를 표현하기 위하여 돌의 연결상태, 막힘상태, 연결방향등을 요소로하는 표현벡터라는 개념을 도입하였는데, 이에 따라 하나의 수는 최대 8개의 표현벡터로서 표현된다.

둘째, 표현벡터들 간의 우선관계를 이용하는 수의 선택해법을 개발하였다. 기본개념은 현재의 게임상황에 가장 유리한 우선적인 표현벡터들을 가지는 수를 선택하는 것이다. 이때, 표현벡터들 간의 우선관계가 곧 오목게임을 위한 지식이 된다.

셋째, 우선관계의 모순성을 검사하기 위하여, 유방향 그래프 상에서의 Cycle의 존재여부를 검사하는 $O(n^2)$해법을 개발하였다. 이 해법은 adjacency matrix를 입력자료로 하고 있다.

실험결과는 본 연구에서 제시한 오목게임의 발견적 기법이 기존의 오목 프로그램인 Weisenbaum의 것에 대해 90%의 높은 승률을 보여주고 있다.

참 고 문 헌

1. 정봉주, "지식베이스를 이용한 오목게임에 관한 연구", 석사학위논문, 서울대학교 공과대학, 1988.
2. Basse, S., *Computer Algorithms : Introduction to Design and* Analysis, Addison Wesely, 1978.
3. Bondy, J. A. and U. S. R. Murty, *Graph Theory with Applications*, American Elsevier Publishing Co., Inc., 1976.
4. Bramer, M. A., *Computer Game-Playing : Theory and Practice*, Ellis Horwood, 1983.
5. Charniak, E., and D. McDermott, *Introduction to Artificial Intelligence*, Addison Wesely, 1985.
6. Deo, N., *Graph Theory with Applications to Engineering and Computer Science*, Englewood Cliffs, N. J., Prentice-Hall, 1974.
7. Horowitz, E., and S. Sahni, *Fundamentals of Data Structures*, Computer Science Press, 1984.
8. Nilsson, N. J., *Porblem-Solving Methods in Artificial Intelligence*, McGraw-Hill, 1971.
9. Rich, E., *Artificial Intelligence*, McGraw-Hill, 1983.
10. Roberts, P. D., and B. Flores, "Systematic Generation of Hamilton Circuits," *Comm. ACM*, Vol. 9, No. 9, pp 690-694, 1966.
11. Slagle, J. R., *Artificial Intelligence : The Heuristic Programming Approach*, McGraw-Hill, 1971.
12. Tenenbaum, A. M., and M. J. Augenstein, *Data Structures using Pascal*, Prentice-Hall, 1981.
13. Tiernan, J. C., "An Efficiecnt Search Algorithm to Find the Elementry Circuits of a Graph," *Comm. ACM*, vol. 13, No. 12, pp 722-726, Dec., 1970.
14. Weisenbaum, J., "How to make a Computer Appear Intelligent : Five-in-a-Row offers no guarantee," *Datamation*, pp 24-26, February, 1962.
15. _____, and R. C. Sheperdson, "Gamesmanship," *Datamation*, pp 10, April, 1962.