Journal of Korea Game Society 2021 Dec; 21(6): 31-40
http://dx.doi.org/10.7583/JKGS.2021.21.6.31

게임 프로그래밍

몬테카를로 방법과 점 패턴 매칭을 활용한 바둑에서의 사활문제 해결을 위한 원형 안형의 분류

이병두
용인대학교 AI학부
blee026@korea.com

Prototypical Eye Shape Classification to Solve Life-and-Death Problem in Go, using Monte-Carlo Method and Point Pattern Matching

Byung-Doo Lee
School of Artificial Intelligence, Yong In University

요 약

바둑은 2,500년 이상의 역사를 지녔고, 바둑에서의 사활문제는 컴퓨터 바둑을 구축 시에 반드시 해결해야 되는 기본 문제영역이 된다. 본 논문에서는 사활문제와 직결되는 3, 4, 5, 6궁에 대한 원형 안형의 개수 확인과 4-튜플 형식으로 표현된 원형 안형을 분류하고자 했다. 실험은 몬테카를로 방법과 점 패턴 매칭에 의해 수행되었다. 실험 결과에 따르면 원형 안형의 개수는 3궁 2개, 4궁 5개, 5궁 12개, 6궁 35개가 된다. 아울러 4-튜플 형식으로 된 원형 안형을 분류하면 3궁 1가지, 4궁 3가지, 5궁 4가지, 6궁 8가지로 분류된다.

ABSTRACT

Go has a history of more than 2,500 years, and the life-and-death problems in Go is a fundamental problem domain that must be solved when implementing a computer Go. We attempted to determine the numbers of prototypical eye shapes with 3, 4, 5, and 6 eyes that are directly related to the life-and-death problems, and to classify the prototypical eye shapes represented in 4-tuple forms. Experiment was conducted by Monte-Carlo method and point pattern matching. According to the experimental results, the numbers of prototypical eye shapes were 2 for 3-eye, 5 for 4-eye, 12 for 5-eye, and 35 for 6-eye shapes. Further, using a 4-tuple form, we classified prototypical eye shapes into 1 for 3-eye, 3 for 4-eye, 4 for 5-eye, and 8 for 6-eye shapes.

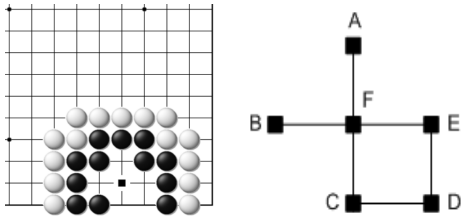
Keywords : Go(바둑), life-and-death problem(사활문제), prototypical eye shape(원형 안형), Monte-Carlo method(몬테카를로 방법), point pattern matching(점 패턴 매칭)

Received: Sep. 17. 2021 Revised: Oct. 27. 2021
Accepted: Oct. 30. 2021
Corresponding Author: Byung-Doo Lee(Yong In University)
E-mail: blee026@korea.com

ISSN: 1598-4540 / eISSN: 2287-8211

1. 서 론

적어도 2,500년 이상의 역사를 지닌 바둑에서의 사활문제(life-and-death problem)[1]는 컴퓨터 바둑[2]을 구현하기 위해 반드시 해결해야 할 기본 문제영역이 된다[1,2]. 바둑에서의 궁도(eye shape)는 상대방의 돌에 의해 둘러싸인 자신의 집모양이 된다. 사활문제와 직접적으로 관련된 궁도는 3궁, 4궁, 5궁, 6궁이 되며, 이러한 궁도 내의 사활문제를 궁도사활 문제라고 한다. 참고로 3궁, 4궁, 5궁, 6궁이 아닌 1궁, 2궁은 무조건 죽음이 되고, 7궁 이상의 궁도는 무조건 삶이 보장된다[3]. 한 예로 [Fig. 1](a)는 6궁의 형태로 흑이 ■의 곳을 두어 살거나 또는 백이 ■의 곳을 치중해서 흑을 잡을 수 있는 궁도사활 문제가 된다.



[Fig. 1] (a) A 6-eyes shape. (b) The graph with 6 vertices and 6 edges for representing the eye shape in (a).

사활문제 해결에 있어 대부분의 연구자들은 개방된 문제 공간이 아닌 폐쇄된 문제 공간에 한정해서 나름대로의 방식을 갖고 문제를 해결하고자 했다. 초창기는 사활문제 해결을 위해 탐색 및 휴리스틱 규칙에 의존했다. D.B. Benson은 정적평가를 이용하여 간단한 사활문제를 해결하였으며[3], 대부분의 컴퓨터 바둑은 정합 및 휴리스틱 규칙을 기반으로 베일에 싸인 별도의 사활문제 모듈을 프로그램 내에 탑재하여 사활문제를 처리했다[4]. 이후 연구자들은 사활문제 해결을 위해 인공지능을 활용하기 시작했다. N. Sasaki 등은 바둑에서의 사활문제 해결을 위해 기존의 탐색방법이 아닌 인공신경망을 이용한 통제학습을 구현하여 프로 1단 정도의 해결 능력을 보였다[5]. B.D. Lee 등은 인공신경망을 활용한 패턴군집을 통해 사활문제 해결을 위한 첫 번째 착수를 구하고자 했다[2,6]. 또한 D. Silver 등은 알파고의 최신 버전인 알파고 제로를 통해 사활문제 해결을 위한 기본 개념을 파악할 수 있는 능력을 보였다[7]. T. Cazenave는 알파고와 달리 전술적 탐색을 추가한 심층학습을 이용하여 사활문제와 같이 전술을 요하는 제반 문제를 풀고자 했다[8]. 이와 같이 사활문제 해결을 위해 탐색, 휴리스틱 규칙, 인공신경망 등을 이용하여 문제를 해결해 왔으나, 문제 공간에 대한 근본적인 분석은 거의 없어 최신 기술에 의해 도출된 좋은 결과값에 대해 정확하게 해석학적으로 이해가 쉽지 않다.

이에 본 연구에서는 사활문제에 대한 직접적인 문제 해결보다는 문제 공간에 대한 정확한 이해를 위해 문제 공간 자체를 분석하고자 했다. 즉, 사활과 직결되는 3궁, 4궁, 5궁, 6궁에 대한 원형 안형의 개수 확인과 4-튜플 형식으로 표현된 원형 안형의 분류를 하고자 했다. 이러한 원형 안형에 대해 수학적인 4-튜플 형식의 표현 및 분류, 그리고 궁도에 따른 원형 안형의 개수 파악 등은 향후 체계적인 컴퓨터 바둑 구축시에 많은 도움이 될 것으로 기대한다.

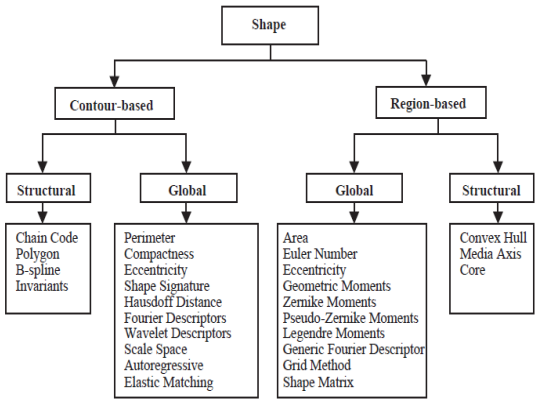
2. 관련 연구

형상(shape)은 어떤 것의 특정한 물리적 형태나 모양을 의미하며, 형상의 표현 및 분류 작업은 두

1) 둘러싸인 돌들이 살아있는지, 죽었는지 또는 빅(비김 또는 공생)이 되었는지를 판별하는 문제. 바둑계에서는 '삶과 죽음'을 생사(生死) 대신에 사활(死活)로 사용하고 있음

2) 인공지능을 활용하여 컴퓨터에서 개발된 바둑 프로그램이 되며, 전 세계적으로 컴퓨터 바둑(computer Go)으로 호칭을 하나 한국에서는 이를 인공지능 바둑(AI Go)으로 부름

단계로 진행이 된다. 첫 번째 단계는 형상이 갖고 있는 고유한 성질을 표현하는 것이고, 두 번째 단계는 적당한 거리 척도로 형상을 분류하는 것이다[9]. 디지털 영상 처리에서는 형상 기술자(shape descriptor)를 이용하여 형상을 표현하며, 형상 기술자는 일반적으로 형상의 외곽선을 중요시하는 외곽선 기반 기술자(contour based descriptor)와 형상의 내부를 중요시하는 영역 기반 기술자(region based descriptor)로 구분된다[10].



[Fig. 2] Classification of shape representation and description[9]

2.1 안형 기술자

안형(eye shape)은 궁도에 대한 눈모양을 지칭한다. 일반적인 바둑판은 19줄 바둑판이 되며, 가로 19줄, 세로 19줄의 격자 형태를 이루고 있다. 정점(vertex)은 바둑판 내 가로줄과 세로줄이 겹치는 점이 되며, 간선(edge)은 두 정점을 대각선이 아닌 수평 또는 수직으로 잇는 선이 된다. 이러한 이유로 바둑판 내 특정 위치에 있는 한 개의 점은 5개 이상의 간선을 보유할 수 없고, 최대 4개까지 간선을 보유할 수 있다. 본 실험에서는 안형을 정점과 간선으로 연결된 완전 그래프(complete graph)로 고려하였으며, 이를 [Fig. 3]과 같이 4개의 요소로 구성된 (v_1, v_2, v_3, v_4)라는 4-튜플의 형식으로 안형 기술자를 대신하였다.

v_1	v_2	v_3	v_4

[Fig. 3] 4 elements in a 4-tuple form

[Fig. 3]의 4-튜플 형식 내에 있는 요소인 v_1은 1개의 간선, v_2는 2개의 간선, v_3는 3개의 간선, v_4는 4개의 간선을 포함하는 정점의 수가 된다. 한 예로 [Fig. 1](b)와 같은 그래프에서 정점 A, B는 1개의 간선, C, D, E는 각각 2개의 간선, F는 4개의 간선을 포함하고 있기 때문에 $(2,3,0,1)$이라는 4-튜플 형식으로 표현할 수 있다.

2.2 점 패턴매칭

점 패턴매칭(point pattern matching)은 컴퓨터 비전이나 패턴 매칭에서 중요하게 다루는 연구 분야가 된다[10]. 점 패턴 매칭 문제는 주어진 두 개의 점 데이터 세트에 대해 대응되는 각 점들 간의 일치성을 확인하는 작업이 된다[11]. 2차원 평면상에서 두 개의 점 데이터 세트인 $P=\{p_1, p_2, ..., p_m\}$, $Q=\{q_1, q_2, ..., q_n\}$가 주어진 경우에, 표본집단 P와 비교집단 Q의 일치성을 파악하기 위해 P 내의 각 점들은 식 (1)과 같이 선형변환에 의한 변환 작업이 이루어진다.

$$P^T=\begin{pmatrix}p_x^T\\ p_y^T\end{pmatrix}=\begin{pmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{pmatrix}\begin{pmatrix}p_x\\ p_y\end{pmatrix}+\begin{pmatrix}t_x\\ t_y\end{pmatrix} \quad (1)$$

여기서 (p_x, p_y)는 $p\in P$인 점들의 선형변환 전의 좌푯값, (p_x^T, p_y^T)는 선형변환 후의 좌푯값, θ는 회전각, (t_x, t_y)는 이동벡터가 된다[11,12].

이후 거리 척도에 의해 P^T와 Q 내의 대응되는 각 점들 간의 거리가 계산되어진다. 두 개의 점 데이터 세트 내의 각 점들간의 거리를 계산하는 대표적인 방법은 식 (2)와 같은 하우스도르프 거리(Hausdorff distance) 계산이 된다[13,14].

$$H(P^T, Q) = \max(h(P^T, Q), h(Q, P^T)) \quad (2)$$

여기서 $h(P^T, Q) = \max_{p^T \in P^T}\left(\min_{q \in Q} \| p^T - q \|\right)$가 되며, $h(Q, P^T) = \max_{q \in Q}\left(\min_{p^T \in P^T} \| q - p^T \|\right)$가 된다. 만약 $H(P^T, Q)$의 값이 작으면 부분 일치를 의미하고, 0이 되는 경우에는 완전 일치가 된다.

3. 실험 방법 및 결과

3.1 실험 목적 및 방법

본 실험의 목적은 (1) 점 패턴 매칭을 사용하여 3궁, 4궁 5궁 6궁에 대한 원형 안형의 개수 확인 (2) 4-튜플 형식으로 표현된 원형 안형의 분류가 된다. 실험을 위해 사용된 주요 방법은 몬테카를로 방법과 점 패턴 매칭이 된다.

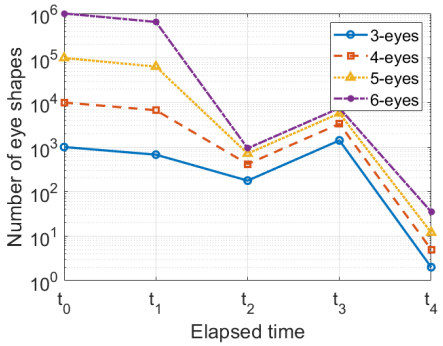
3.2 원형 안형의 개수 확인

안형은 정점과 간선으로 구성되는 완전 그래프 형태가 되어야 한다. 유효한(또는 정상적인) 안형을 생성하기 위한 작업은 (1) 후보 안형 세트를 생성 (2) 이 중에서 완전 그래프를 형성시키지 못하는 무효한(또는 비정상적인) 후보 안형을 제거시켜 유효한 안형 세트를 생성 (3) 이후 유효한 안형 세트를 갖고 추가 안형 세트를 생성시켜 전체 유효 안형 세트를 생성 (4) 마지막으로 전체 유효 안형 세트 내의 안형들 간의 일치성 여부를 확인하여 안형들을 대표하는 원형 안형 세트를 생성이 된다.

3.2.1 후보 안형 세트 생성

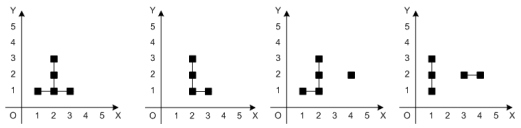
n궁에 대한 후보 안형을 생성시키기 위해 몬테카를로 방법을 이용하였다. 작업 순서는 (1) 무작위 n개의 정점을 갖는 후보 안형 세트를 생성 (2) 생성된 후보 안형 세트들 중에서 중복된 정점을 포함하는 후보 안형을 제거 (3) 정점 간에 연결이 안된 후보 안형, 즉 단절 그래프(disconnected graph)로 형성된 후보 안형을 제거가 되며, 이를 통해 최종 유효 안형 세트를 추출해 내었다.

세부 작업을 살펴보면 몬테카를로 방법을 이용하여 n궁에 대한 충분한 후보 안형 세트를 생성하기 위해, $0 \sim (n-1)$의 정수값을 갖는 후보 안형을 무작위로 [Fig. 4]의 t_0에서 보듯이 3궁 10^3개, 4궁 10^4개, 5궁 10^5개, 6궁 10^6개를 생성시켰다.



[Fig. 4] Numbers of eye shapes with respect to varying elapsed time

한 예로 [Fig. 5]는 몬테카를로 방법을 이용하여 생성된 5궁의 후보 안형들이 되며, 이 중에서 [Fig. 5](a)는 유효한 후보 안형이 되나, [Fig. 5](b), [Fig. 5](c), [Fig. 5](d)는 무효한 후보 안형들이 된다. 즉 [Fig. 5](b)는 중복된 정점을 포함하고 있으며, [Fig. 5](c)는 간선이 없는 정점이 있으며, [Fig. 5](d)는 연결 그래프(connected graph)가 아닌 단절 그래프이기 때문에 무효한 후보 안형이 된다.



(a) (b) (c) (d)

[Fig. 5] Valid and invalid eye shapes with 5 eyes. (a) Valid eye shape. (b) Invalid eye shape with duplicated vertices. (c) Invalid eye shape without edges. (d) Invalid eye shape with a disconnected graph

3.2.2 유효 안형 세트 생성

몬테카를로 방법에 의해 생성된 후보 안형 세트들 중에서 중복된 정점을 갖는 후보 안형들을 제거하였다. [Table 1]의 t_1에서의 값들은 중복된 정점을 포함하는 무효한 후보 안형들을 제외한 나머지 후보 안형들의 개수가 된다.

[Table 1] Number of eye shapes with respect to varying time

Time / Eyes	t_0	t_1	t_2	t_3	t_4
3	10^3 (100%)	674 (67.4%)	176 (17.60%)	1,408 (140.80%)	2 (0.200%)
4	10^4 (100%)	6,705 (67.1%)	415 (4.15%)	3,320 (33.20%)	5 (0.050%)
5	10^5 (100%)	64,939 (64.9%)	713 (0.71%)	5,704 (5.70%)	12 (0.012%)
6	10^6 (100%)	644,429 (64.4%)	937 (0.09%)	7,496 (0.75%)	35 (0.004%)

[Table 1]을 살펴보면 생성된 초기 후보 안형(t_0)들에 대해 대략 35% 정도가 중복인 정점을 갖는 무효한 후보 안형이기 때문에, 이를 제거하게 되면 대략 65% 정도(3궁 67.4%, 4궁 67.1%, 5궁 64.9%, 6궁 64.4%)가 후보 안형 세트로 남게 된다.

유효한 후보 안형 내에 있는 각 정점들은 간선으로 서로 연결되어 연결 그래프를 형성하여야 하나, [Fig. 5](c)와 [Fig. 5](d)에서 보듯이 1개 이상의 정점이 떨어져 있는 단절 그래프로 된 안형들은 후보 안형에서 제거시켰다. 제거 작업은 (1) [Fig. 5](c)와 같이 간선이 없는 안형 제거 (2) [Fig. 5](d)와 같이 모든 정점이 간선을 포함하나 단절 그래프로 형성된 안형에 대한 제거로 구분하여 순차적으로 진행하였다.

간선이 없는 안형에 대한 제거 대상은 n궁을 구성하는 모든 n개의 정점을 기준으로 동서남북으로 이웃하는 정점이 한 개도 없는 경우가 되며, 이를 후보 안형 세트에서 제거시켰다. 모든 정점이 간선을 포함하나 단절 그래프로 형성된 안형에 대한 제거 대상은 (1)의 작업 이후에 생성된 n궁에 대한 후보 안형 세트가 되며, 이를 대상으로 [Fig. 6]과 같이 깊이우선탐색(DFS)을 실시하여 방문한 노드(또는 정점)의 개수가 n이 아닌 후보 안형을 후보 안형 세트에서 제거시켰다.

Algorithm 1 Depth First Search Algorithm(DFS)

```
1: procedure DFS(G, u)
2:   u.visited = true
3:   for each v ∈ G.Adj[u]
4:     if v.visited == false
5:       DFS(G, v)
```

[Fig. 6] Procedure of DFS[15]

[Table 1]에서 보듯이 단절 그래프로 된 무효한 후보 안형을 제거한 이후 남아 있는 유효 안형(t_2)의 개수는 3궁 176개(17.60%), 4궁 415개(4.15%), 5궁 713개(0.71%), 6궁 937개(0.09%)가 되었다.

3.2.3 추가 유효 안형 세트 생성

유효 안형 세트 내의 모든 안형에 대해 선형변환을 실시하여 7개(회전 3개, 대칭 4개)의 추가 안형을 생성시켜, 이를 갖고 새로운 추가 유효 안형 세트를 생성시켰다. 추가 유효 안형 세트 생성은 (1) 각 유효 안형에 대한 도심(centroid of shape) 계산 (2) 원점으로의 유효 안형 이동 (3) 선형변환인 회전 및 대칭을 이용한 추가 유효 안형 생성으로 진행되었다.

유효 안형의 도심 위치를 나타내는 유효 안형의 평균값 (μ_x, μ_y)는 각 정점의 좌푯값들이 (x_1, y_1), (x_2, y_2), …, (x_n, y_n)으로 주어진 경우에 식 (3)과 같이 계산된다.

$$\mu_x = \frac{\sum_{i=1}^{i=n} x_i}{n}, \qquad \mu_y = \frac{\sum_{i=1}^{i=n} y_i}{n} \tag{3}$$

여기서 n은 정점의 개수가 된다.

이후 유효 안형에 대한 원점으로의 이동은 각 정점의 좌푯값과 식 (3)의 평균값을 이용하여 식 (4)에 의거 수행이 된다.

$$\text{trans}_{\text{origin}} = \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_n & y_n \end{pmatrix} - \begin{pmatrix} \mu_x & \mu_y \\ \mu_x & \mu_y \\ \vdots & \vdots \\ \mu_x & \mu_y \end{pmatrix} = \begin{pmatrix} x_1 - \mu_x & y_1 - \mu_y \\ x_2 - \mu_x & y_2 - \mu_y \\ \vdots & \vdots \\ x_n - \mu_x & y_n - \mu_y \end{pmatrix} \quad (4)$$

원점으로 이동된 유효 안형을 갖고 회전 및 대칭을 시켜 추가 유효 안형들을 생성시켰다. 즉, 유효 안형에 대한 회전은 회전행렬(rotation matrix)인 $\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$를 이용하여 식 (5)와 같이 유효 안형을 90°, 180°, 270° 회전시켜 추가로 3개의 유효 안형을 생성시켰다.

$$\text{rot} = \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_n & y_n \end{pmatrix} \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} = \begin{pmatrix} x_1\cos\theta - y_1\sin\theta & x_1\sin\theta + y_1\cos\theta \\ x_2\cos\theta - y_2\sin\theta & x_2\sin\theta + y_2\cos\theta \\ \vdots & \vdots \\ x_n\cos\theta - y_n\sin\theta & x_n\sin\theta + y_n\cos\theta \end{pmatrix} \quad (5)$$

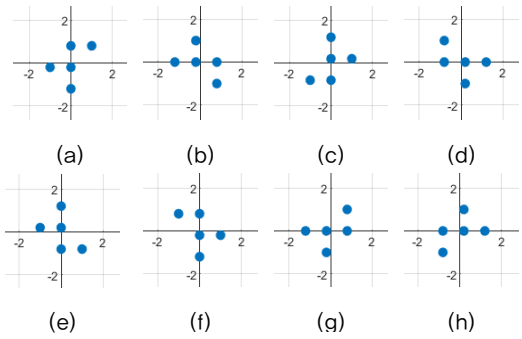
여기서 θ는 x-축으로부터 반시계 방향으로의 회전각이 된다.

아울러 유효 안형에 대한 대칭은 대칭행렬(reflection matrix)인 $\begin{pmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{pmatrix}$를 이용하여 식 (6)과 같이 유효 안형을 x-축, y-축, 직선 $y=x$, 직선 $y=-x$ 대칭을 시켜 추가로 4개의 유효 안형을 생성시켰다.

$$\text{rot} = \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_n & y_n \end{pmatrix} \begin{pmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{pmatrix} = \begin{pmatrix} x_1\cos 2\theta + y_1\sin 2\theta & x_1\sin 2\theta - y_1\cos 2\theta \\ x_2\cos 2\theta + y_2\sin 2\theta & x_2\sin 2\theta - y_2\cos 2\theta \\ \vdots & \vdots \\ x_n\cos 2\theta + y_n\sin 2\theta & x_n\sin 2\theta - y_n\cos 2\theta \end{pmatrix} \quad (6)$$

여기서 θ는 원점을 관통하는 대칭축인 직선과 x-축과의 사잇각이 된다.

한 예로 [Fig. 7](a)와 같은 유효 안형이 주어진 경우에 이를 회전 및 대칭을 하면 [Fig. 7](b)에서 [Fig. 7](h)까지의 안형이 추가로 생성된다. 참고로 추가 유효 안형을 포함한 최종 전체 유효 안형의 개수는 [Table 1]의 시간 t_3에서 보듯이 시간 t_2에서의 개수의 8배인 3궁 1,408개(140.80%), 4궁 3,320개(33.30%), 5궁 5,704개(5.70%), 6궁 7,496개(0.75%)가 되었다.



[Fig. 7] Eye shapes with 5 eyes. (a) Prototypical eye shape. (b) Rotation by 90° counterclockwise. (c) Rotation by 180° counterclockwise. (d) Rotation by 270° counterclockwise. (e) Reflection by the x-axis. (f) Reflection by y-axis. (g) Reflcetion by the line $y=x$. (h) Reflection by the line $y=-x$.

3.2.4 원형 안형 세트 생성

원형 안형 세트의 생성은 유효 안형 세트 중에서 각 안형을 대표하는 원형을 추출하는 작업이 된다. 원형 안형 세트의 생성 작업은 (1) 최초 원형 안형 세트 생성 (2) 전체 유효 안형과 원형 안형 세트 내의 원형 안형들 간의 점 패턴 매칭이 된다. 최초 원형 안형 세트 생성을 위해 전체 유효 안형 중에서 첫 번째 유효 안형을 초기 원형 안형 세트 내에 첫 번째 원형 안형으로 등록시켰다. 전체 유효 안형과 원형 안형 세트 내의 원형 안형들 간의 매칭은 전체 유효 안형을 순차적으로 하나씩 받아들여, 입력된 유효 안형을 기존의 원형 안형

세트 내의 원형 안형들과 일치하는 가를 확인하는 작업이 된다. 순차적으로 입력된 유효 안형의 좌푯값들이 $E = \{(E_{x_1}, E_{y_1}), (E_{x_2}, E_{y_2}), ..., (E_{x_n}, E_{y_n})\}$인 경우에 이들을 x-, y-좌푯값 별로 오름차순 정렬을 하여 새로운 유효 안형의 좌푯값인 $E' = \{(E'_{x_1}, E'_{y_1}), (E'_{x_2}, E'_{y_2}), ..., (E'_{x_n}, E'_{y_n})\}$을 생성시킨다. 점 패턴 매칭 작업은 식 (7)과 같이 E'와 원형 안형 세트 내의 원형 안형의 좌푯값들인 $P = \{(P_{x_1}, P_{y_1}), (P_{x_2}, P_{y_2}), ..., (P_{x_n}, P_{y_n})\}$와와 거리 척도를 활용한 점 패턴 매칭이 된다.

$$\sum_{i=1}^{n} \| E_i - P_i \| = \sum_{i=1}^{n} \sqrt{(E'_{x_i} - P_{x_i})^2 + (E'_{y_i} - P_{y_i})^2} \le \epsilon \quad (7)$$

여기서 허용오차 ϵ은 0.001로 설정하였으며, n은 n궁에 대한 정점의 개수가 된다.

식 (7)에 의거하여 입력된 유효 안형과 기존의 원형 안형 세트 내의 원형 안형이 (1) 일치가 되면 새로운 점 패턴 매칭을 위해 또 다른 유효 안형이 입력이 되고 (2) 일치가 안되는 경우에는 입력된 원형 안형이 원형 안형 세트 내의 새로운 원형 안형으로 첨가가 된다. 이와 같은 작업을 입력되는 유효 안형이 없을 때까지 반복 실행하였다. 위와 같은 반복적인 작업을 수행한 후의 결과는 [Fig. 4]의 t_4, [Table 1]의 t_4, [Table 2]에서 보듯이 3궁 2개(0.200%), 4궁 5개(0.050%), 5궁 12개(0.012%), 6궁 35개(0.004%)가 된다. 참고로 [Table 2]의 오른쪽에 있는 안형들은 점 패턴 매칭 작업에 의해 생성된 최종 원형 안형들의 모습으로, 각 안형의 도심을 원점으로 이동한 모양이다.

3.3 원형 안형의 분류

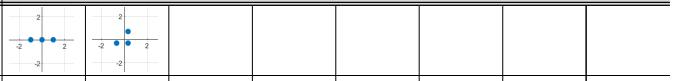
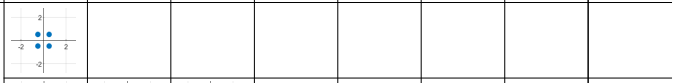
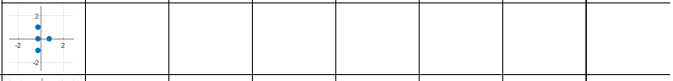
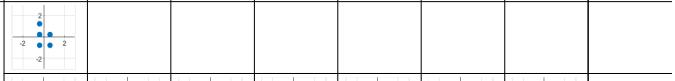
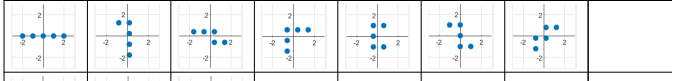
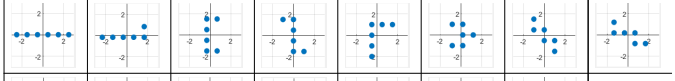
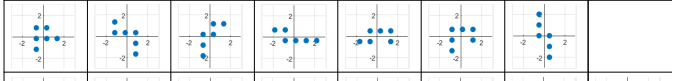
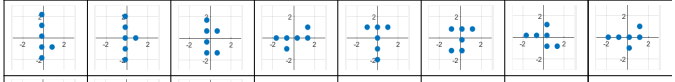
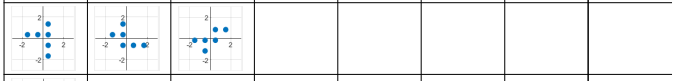
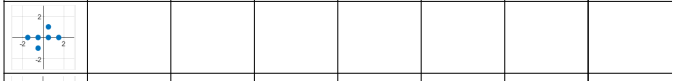
점 패턴 매칭 이후 생성된 원형 안형 세트를 구성하는 3궁 2개, 4궁 5개, 5궁 12개, 6궁 35개의 원형 안형들에 대해 4개의 요소로 구성된 4-튜플 형식인 (v_1, v_2, v_3, v_4)로 표현을 한 후, 이를 분류하게 되면 [Table 2]에서 보듯이 2개의 원형 안형으로 구성되어 있는 3궁인 경우 (2,1,0,0)인 1가지, 5개의 원형 안형으로 구성되어 있는 4궁인 경우 (0,4,0,0), (2,2,0,0). (3,0,1,0)인 3가지, 12개의 원형 안형으로 구성되어 있는 5궁인 경우 (1,3,1,0), (2,3,0,0), (3,1,1,0), (4,0,0,1)인 4가지, 35개의 원형 안형으로 구성되어 있는 6궁인 경우 (0,4,2,0), (1,4,1,0), (2,2,2,0), (2,3,0,1), (2,4,0,0), (3,2,1,0), (4,0,2,0), (4,1,0,1)인 8가지로 최종 분류가 된다.

3.4 실험 결과

본 실험에서는 사활문제와 직결되는 궁도인 3궁, 4궁, 5궁, 6궁의 원형 안형 개수 확인 및 4-튜플 형식으로 표현된 원형 안형에 대한 분류를 했다. 실험은 원형 안형 개수 확인을 위해 4단계인 (1) 후보 안형 세트 생성 (2) 유효 안형 세트 생성 (3) 추가 유효 안형 세트 생성 (4) 원형 안형 세트 생성으로 진행되었으며, 생성된 원형 안형 세트를 갖고 최종적으로 4-튜플 형식으로 된 원형 안형의 분류를 하였다.

후보 안형 생성은 몬테카를로 방법을 이용하여 충분한 후보 안형 세트인 3궁 10^3개, 4궁 10^4개, 5궁 10^5개, 6궁 10^6개를 생성시켰다. 이 중에서 무효한 안형을 제거시킴으로서 유효한 안형 개수는 3궁 176개, 4궁 415개, 5궁 713개, 6궁 937개가 되었다. 추가 유효 안형 세트 생성은 유효 안형을 갖고 선형변환인 회전 및 대칭을 시켰으며, 이를 통해 추가 유효 안형을 포함한 최종 유효 안형 개수는 3궁 1,408개, 4궁 3,320개, 5궁 5,704개, 6궁 7,496개가 되었다. 이후 원형 안형 세트 생성을 위해 점 패턴 매칭을 실시하였으며, 이로서 전체 유효 안형에 대한 원형 안형의 개수는 3궁 2개, 4궁 5개, 5궁 12개, 6궁 35개가 되었다.

[Table 2] 4-tuples and set of prototypical eye shapes with n eyes

Eyes	4-tuples		Number of prototypical eye shapes	Set of prototypical eye shapes
	Number	Expression		
3	1	(2,1,0,0)	2	
4	3	(0,4,0,0)	5	
		(2,2,0,0)		
		(3,0,1,0)		
5	4	(1,3,1,0)	12	
		(2,3,0,0)		
		(3,1,1,0)		
		(4,0,0,1)		
6	8	(0,4,2,0)	35	
		(1,4,1,0)		
		(2,2,2,0)		
		(2,3,0,1)		
		(2,4,0,0)		
				
		(3,2,1,0)		
				
		(4,0,2,0)		
		(4,1,0,1)		

마지막으로 원형 안형에 대한 분류를 위해 4-튜플 형식인 (v_1, v_2, v_3, v_4)를 활용하였으며, 이를 통해 전체 원형 안형 세트에 대한 분류 결과는 3궁 1가지, 4궁 3가지, 5궁 4가지, 6궁 8가지가 되었다.

4. 결론 및 제언

바둑에서의 궁도는 상대방의 돌에 의해 둘러싸인 자신의 집모양이 된다. 사활문제와 직결되는 궁도는 3궁, 4궁, 5궁, 6궁이 되며, 본 연구에서는 이들에 대한 원형 안형의 개수 확인 및 분류를 하고자 했다. 실험 결과에 따르면 3궁, 4궁, 5궁, 6궁의 원형 안형의 개수는 3궁 2개, 4궁 5개, 5궁 12개, 6궁 35개가 되며, 이를 4-튜플 형식인 (v_1, v_2, v_3, v_4)로 분류하게 되면 3궁 1가지, 4궁 3가지, 5궁 4가지, 6궁 8가지가 됨을 알 수 있었다. 향후 본 실험에서 밝혀진 3궁, 4궁, 5궁, 6궁에 대한 원형 안형의 해석학적 성질을 활용하면 컴퓨터 바둑 구축에서의 사활문제 해결에 많은 도움을 줄 것으로 기대된다. 특히 단순한 4-튜플 형식이 아닌 궁도의 사활 여부를 결정짓는 요소를 포함한 5-튜플 형식으로 된 궁도의 표현을 하게 되면, 즉각적으로 이를 컴퓨터 바둑 구현에 적용될 수 있다. 결국 본 실험의 유용성은 크게 두 가지인 (1) 문제 공간에 대한 해석학적 분석 가능성을 예시했으며 (2) 문제 해결에 있어 확률론적 방식이 아닌 해석학적 문제 해결 접근 가능성을 제시한 것이 된다.

REFERENCES

[1] Wikipedia, “Life and death”, https://en.wikipedia.org/wiki/Life_and_death, September, 2021.
[2] B. D. Lee and Y. W. Keum, “Candidate First Move for Solving Life-and-Death Problems in the Game of Go” Journal of Korea Game Society, Vol. 9, No. 1, pp.105-114, 2009.
[3] D. B. Benson, “Life in the game of Go” Information Intelligence 10, Vol. 9, No. 1, 91-124, 1976.
[4] K. Chen and Z. Chen, “Static analysis of life and death in the game of Go” Journal of Information Sciences 121, 113-134, 1999.
[5] N. Sasaki, Y. Sawada, J. Yoshimura, “A Neural Network Program of Tsume-Go” Computer and Games 1998, pp.167-182, Springer-Verlag, Germany, 1999.
[6] B. D. Lee, “Multi-strategic Learning, Reasoning and Searching in the Game of Go”, PhD thesis, University of Auckland, New Zealand, 2005.
[7] D. Silver, et al., “Mastering the game of Go with deep neural networks and tree search” Journal of Nature, Vol. 529, Issue 7587, pp.484-489, 2016.
[8] T. Cazenave and B. Helmsetter, “Combining tactical search and deep learning in the game of Go”, In: CIG’05, pp.171-175, 2005.
[9] B. H. Shekar and B. Pilar, “Shape Representation and Classification through Pattern Spectrum and Local Binary Pattern” 2014 Fifth International Conference on Signal and Image Processing, 2014.
[10] M. A. Akta, “Shape Descriptors”, PhD thesis, University of Exter, England, 2012.
[11] M. Tico and C. Rusu, “Point Pattern Matching using a Genetic Algorithm and Voronoi Tessellation” 9th European Signal Processing Conference (EUSIPCO 1998), pp. 1-4, 1998.
[12] W. Xiaoyun and Z. Xianquan, “Point Pattern Matching Algorithm for Planar Point Sets under Euclidean Transform” Journal of Applied Mathematics, Vol. 2012, 2012.
[13] Wikipedia, “Hausdorff distance”, https://en.wikipedia.org/wiki/Hausdorff_distance, September, 2021.
[14] D. Aiger and K. Kedem, “Approximate input sensitive algorithms for point pattern matching”, Pattern Recognition 43, pp.153-159, 2010.
[15] Programiz, “Depth First Search (DFS)”, https://www.programiz.com/dsa/graph-dfs, September, 2021.

이 병 두 (Byung-Doo Lee)

약 력 : 1982 한양대학교 원자력공학 학사
1991 서강대학교 정보처리학 석사
2005 Auckland University 컴퓨터공학 박사
현재 용인대학교 AI학부 조교수

관심분야 : 컴퓨터공학, 인공지능, 컴퓨터바둑