Journal of Korea Game Society 2020 Apr; 20(2): 91-100
http://dx.doi.org/10.7583/JKGS.2020.20.2.91

게임 프로그래밍

바둑에서의 사활문제 해결을 위한 외곽선 추적

이병두
용인대학교 컴퓨터과학과
blee026@korea.com

Contour Tracing to Solve Life-and-Death Problem in Go

Byung-Doo Lee
Department of Computer Science, YongIn University

요 약

바둑에서의 사활문제는 컴퓨터바둑을 구현하기 위해 극복되어야 하는 기본적인 문제이다. 그것을 해결하기 위한 중요 고려 사항은 흑백 대국자 간에 누가 둘러싸고 있고 또는 둘러싸여 있는지를 파악하는 것이다. 흑백 그룹간의 경계선을 알아내기 위해 세력함수와 외곽선 추적 알고리즘을 적용하였다. 여러 외곽선 추적 알고리즘 중에서 무어의 이웃 추적을 적용하면 경계선을 생성할 수 있음을 알아냈으며, 아울러 게임트리의 탐색공간을 획기적으로 줄일 수 있는 가능성을 제시했다.

ABSTRACT

Life-and-death problem in Go is a fundamental problem to be overcome for implementing a computer Go. To solve it, an important consideration is to find out who surrounds or is surrounded between black and white players. To figure out the boundary between black and white groups, we applied an influence function and a contour tracing algorithm. We found that applying the Moore-neighbor tracing among various contour tracing algorithms can create boundaries, and also suggested the possibility of tremendously reducing the search space of a game tree.

Keywords : Go(바둑), life-and-death problem(사활문제), influence function(세력함수), contour tracing(외곽선 추적), Moore-neighbor tracing(무어의 이웃 추적)

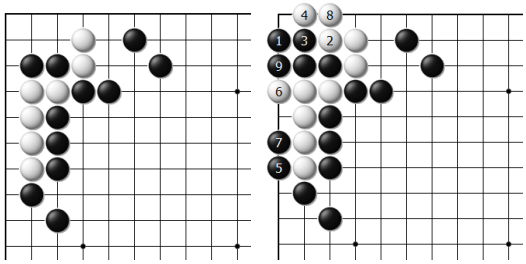
Received: Nov. 11. 2019 Revised: Dec. 06. 2019
Accepted: Dec. 14. 2019
Corresponding Author: Byung-Doo Lee (YongIn University)
E-mail: blee026@korea.com

ISSN: 1598-4540 / eISSN: 2287-8211

1. 서 론

바둑은 적어도 2,500여 년의 역사를 갖고 있으며, 컴퓨터가 그 동안 인간을 제압하지 못했던 유일한 보드게임이었다[1]. 그러나 2015년에 알파고는 최초로 19줄바둑에서 유럽 챔피언 Fan Hui를 5:0으로 제압하였고, 2016년에는 세계 최정상급 프로기사인 이세돌 9단을 4:1로 완승하여 전 세계인들을 놀라게 했다. 이후 고도화된 알파고는 세계 최정상급 프로기사들을 대상으로 60:0이라는 경이적인 승리 기록을 남긴 후 2017년에 바둑계를 은퇴했다[2].

바둑에서의 사활문제(Life-and-Death Problem 또는 Tsumego)는 돌들의 삶과 죽음을 다루는 문제가 된다. 사활문제 해결은 컴퓨터바둑을 구현하기 위해 반드시 극복해야 하는 많은 문제 중의 하나이다[3]. 사활문제를 해결하기 위해 고려되는 가장 중요한 사항은 [Fig. 1](a)에서 보듯이 흑백 대국자 간의 둘러싸임, 즉 누가 둘러싸고 있는지 또는 둘러싸여 있는지를 파악하는 것이 중요하다. 참고로 [Fig. 1](b)에 있는 일련의 수순은 흑백간의 최상의 착수가 되며, 흑1부터 흑9까지로 인해 백의 눈모양(eye-shape)은 오궁도화(五宮桃畵)가 되어 좌하귀에 있는 백돌 전체가 죽게 된다.



[Fig. 1] (a) A life-and-death problem. (b) The sequence of optimal moves

일반적으로 사활문제에 있어 흑백 간의 경계가 명확하게 구분되어 있으면 다소 쉽게 문제를 풀 수 있으나, [Fig. 1](a)와 같이 흑백 간의 경계가 불명확한 경우에는 프로기사 역시 문제에 대한 답을 즉각적으로 내놓기가 용이하지 않다. 대부분의 실전에 등장하는 사활문제는 경계가 불명확하여 프로기사나 일반인들이 문제 해결을 위해 많은 시간을 할애하고 있다. 본 논문에서는 흑백 간에 불명확하게 구분되어 있는 사활문제 해결을 위해 (1) 세력함수를 이용하여 흑백 간의 가상의 영역을 생성하고, (2) 생성된 가상의 영역으로부터 흑백 간의 경계선을 찾고, (3) 흑백간의 둘러싸임 여부를 파악하고자 했다.

2. 관련 연구

사활문제 해결을 위해 Sasaki 등은 통제학습용 신경망(supervised neural network)을 이용하여 사활문제 해결을 위한 첫수를 구하고자 했으며, 실험을 통해 통제학습용 신경망의 성능이 프로 1단 정도의 실력이 되나 일련의 최상의 수순을 구하는 것은 한계가 있음을 보였다[4]. Lee 역시 인공신경망과 눈모양 분석을 활용하여 사활문제 해결을 위한 첫수를 구하고자 했으며, 실험결과에 따르면 통제학습용 신경망은 65.0%, 비통제학습용 신경망(unsupervised neural network)은 62.8%, 눈모양 분석은 36.7%의 정확도를 각각 보였다[3]. 최근에 사활문제 풀이기인 XuanXuanGo는 바깥영역이 확실하게 구분되어 있는 경우, 즉 흑백 간의 경계가 다소 명확한 경우에는 사활문제의 70% 정도를 풀 수 있다[5]. 결국 현재까지의 사활문제 풀이기는 (1) 흑백 간의 경계가 명확한 경우에서의 최상의 수순 또는 (2) 일련의 수순이 아닌 최상의 첫수를 구하는 데에 그치고 있다. 본 연구에서는 향후 완벽한 사활문제 풀이기를 구현하기 위한 초석으로 흑백 간의 경계가 명확하지 사활문제에 대한 경계선을 구하고자 세력함수와 외곽선 추적기법을 적용하였다. 아울러 분할된 경계선을 바탕으로 제한된 영역 내에서의 돌들을 이용하여, 향후 트리탐색의 성능을 향상시킬 후보 첫수 군을 제시하고자 했다.

3. 배경 이론

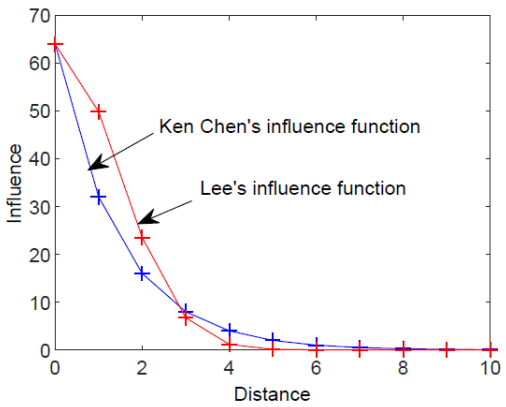
3.1 세력함수

세력(influence)은 바둑에서 형세판단을 하기 위해 요긴하게 사용되는 주먹구구식(rule of thumb) 방법으로, 현재의 돌들의 상태가 “두텁다(thick)” 또는 “엷다(thin)”라고 표현되는 바둑에서의 상위 개념이 된다.

세력함수는 흑백 간의 가상의 영역을 구분하는 데 사용되며, 현재 대부분의 컴퓨터바둑은 이를 사용하여 흑백 간의 가상의 영역을 구분하고 있다[6]. Zobrist는 박사논문에서 세력함수를 활용하여 최초의 컴퓨터바둑을 구현해 냈으며[7], Chen 역시 [Fig. 2]에서 보듯이 간단한 세력함수인 식 (1)을 이용하여 바둑돌 간의 세력을 구하여 형세판단을 하고자 했다[3].

$$F(d) = 2^{6-d} \qquad (1)$$

여기서 d는 돌 간의 거리가 된다.

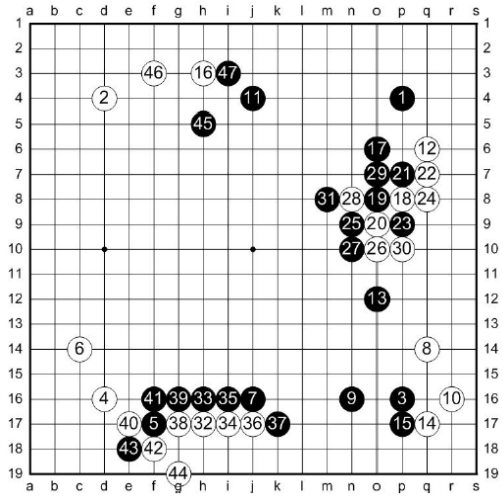


[Fig. 2] Influence functions[3]

한편 Lee는 근접한 돌 간의 연결 강도를 고려하여 새로운 세력함수인 식 (2)를 제안하여 [Fig. 3]과 같이 포석이 진행된 형국에서의 형세판단을 하고자 했다[3].

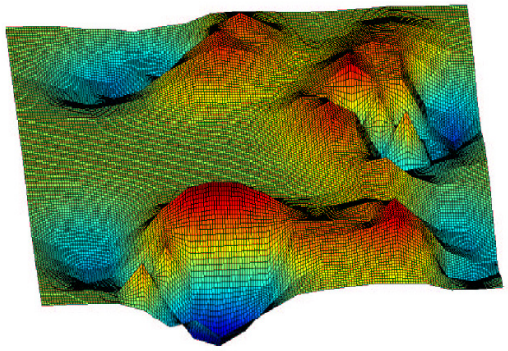
$$I(d) = C \times \exp\left(\frac{-d^2}{4}\right) \qquad (2)$$

여기서 d는 돌 간의 거리가 되며, C는 상수로 흑에 대해서는 +64, 백에 대해서는 -64를 적용하였다.



[Fig. 3] An opening game[3]

형세판단을 위해 Lee는 식 (2)의 세력함수와 유클리디언 거리변환(DT: Distance Transform)을 이용하여 [Fig. 4]와 같은 3차원 세력지도를 그려 흑백간의 형세판단을 하였다[3]. 참고로 [Fig. 4] 내의 봉우리는 흑의 세력권이 되며, 골짜기는 백의 세력권이 된다.



[Fig. 4] The 3-dimensional influence map[3]

3.2 외곽선 추적

외곽선 추적(contour tracing)은 경계선 추적(boundary tracing)이라고도 하며, 이진 디지털 영상의 경계선을 추적하여 영역을 분할하는데 사용된[8,9]. 대표적인 외곽선 추적 알고리즘으로는 정사각형 추적 알고리즘(STA: Square Tracing Algorithm), 무어의 이웃 추적(MNT: Moore-Neighbor Tracing), 방사상 수색 알고리즘(RSA: Radial Sweep Algorithm), Theo Pavlidis′ Algorithm(TPA) 등이 있다[8,9,10,11].

3.2.1 정사각형 추적 알고리즘

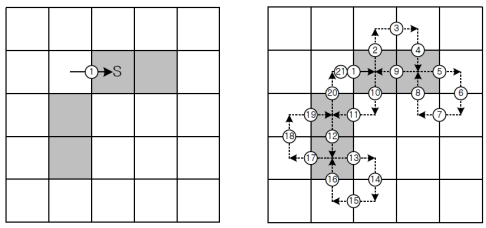
정사각형 추적 알고리즘(STA)은 가장 간단한 알고리즘으로 Simple Boundary Follower(SBF)로도 부른다. 정사각형 추적 알고리즘은 [Fig. 5]에서 보듯이 초기에 픽셀 추적자(pixel tracer)의 시작위치 정보인 S가 보존이 된 후, 추적자는 다음의 두 가지 규칙을 따르면서 오른쪽 또는 왼쪽으로 이동을 한다. 즉 추적자인 p가 물체의 픽셀에 속한 경우에는 왼쪽(L)으로 이동하고, 그렇지 않은 경우에는 오른쪽(R)으로 이동한다. 이후 추적자가 시작픽셀에 최초의 진입방향과 같게 되면 추적 알고리즘은 종료를 한다.

Algorithm 1 Square Tracing Algorithm(STA)

```
1: procedure STA
2:   T(p,d) ← S(p,d)
3:  do
4:     if p = object then T(p,d) ← T(p_L,d_L)
5:     else T(p,d) ← T(p_R,d_R)
6:  while T(p,d) ≠ S(p,d)
```

[Fig. 5] Procedure of STA[8]

한 예로 [Fig. 6](a)의 회색부분은 주어진 물체의 픽셀들이 되며, [Fig. 6](b)는 정사각형 추적 알고리즘을 적용한 추적 경로를 보여주고 있다. ①과 같이 물체의 픽셀을 만나게 되면 ②와 같이 왼쪽으로 이동하며, ②와 같이 물체의 픽셀을 못 만나게 되면 ③과 같이 오른쪽으로 이동을 한다. 만약 ㉑과 같이 물체의 시작픽셀 S를 만나면서 S에 대한 진입방향이 같게 되면 추적 알고리즘은 종료가 된다.



(a) (b)

[Fig. 6] (a) Initial pixels. (b) Contour-following sequence of STA

3.2.2 무어의 이웃 추적

무어의 이웃 추적(MNT)은 특정 픽셀을 중심으로 이웃해 있는 픽셀들을 시계방향으로, 즉 8-방향(하(D)→좌하(DL)→좌(L)→좌상(UL)→상(U)→우상(UR)→우(R)→우하(DR))로 연결성을 추적하는 것이다.

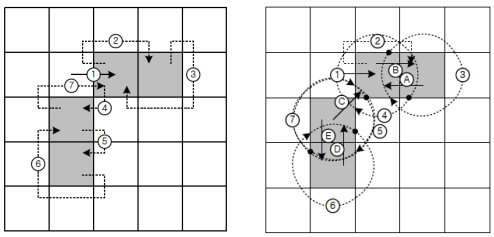
Algorithm 2 Moore-Neighbor Tracing(MNT)

```
1: procedure MNT
2:   T(p,d) ← S(p,d)
3:  do
4:     if p_D = object then T(p,d) ← T(p_D,d_D)
5:     else if p_DL = object then T(p,d) ← T(p_DL,d_L)
6:     else if p_L = object then T(p,d) ← T(p_L,d_U)
7:     else if p_UL = object then T(p,d) ← T(p_UL,d_U)
8:     else if p_U = object then T(p,d) ← T(p_U,d_R)
9:     else if p_UR = object then T(p,d) ← T(p_UR,d_R)
10:    else if p_R = object then T(p,d) ← T(p_R,d_D)
11:    else if p_DR = object then T(p,d) ← T(p_DR,d_D)
12:    else STOP
13: while T(p,d) ≠ S(p,d)
```

[Fig. 7] Procedure of MNT[8]

한 예로 [Fig. 8](a) 내의 ①과 같이 추적자가

물체의 픽셀을 만난 경우에 ②와 같이 시계방향으로 탐색을 하여 인접한 픽셀을 찾게 된다. 이후 ⑦과 같이 물체의 시작픽셀을 만나면서 시작픽셀에 대한 진입방향이 같게 되면 추적은 종료가 된다.



(a) (b)

[Fig. 8] Contour-following sequence of MNT (a) and the RSA (b)

3.2.3 방사상 수색 알고리즘

방사상 수색 알고리즘(RSA)은 MNT와 매우 유사한 방법으로 MNT와 같이 8-방향 탐색이 아닌, 방향벡터를 기준으로 시계방향(우상(UR)→우(R)→우하(DR)→하(D)→좌하(DL)→좌(L)→좌상(UL)→상(U))으로 탐색하는 방식이 된다.

Algorithm 3 Radial Sweep Algorithm(RSA)

```
1: procedure RSA
2:   T(p_{i-1}) ← S(p,d) and find T(p_i) using MNT
3:   d = P_iP_{i-1}
4:   do
5:     if p_UR = object then T(p) ← T(p_UR)
6:     else if p_R = object then T(p) ← T(p_R)
7:     else if p_DR = object then T(p) ← T(p_DR)
8:     else if p_D = object then T(p) ← T(p_D)
9:     else if p_DL = object then T(p) ← T(p_DL)
10:    else if p_L = object then T(p) ← T(p_L)
11:    else if p_UL = object then T(p) ← T(p_UL)
12:    else if p_U = object then T(p) ← T(p_U)
13:    else STOP
14:    T(p_{i-1}) ← T(p_i), T(p_i) ← T(p), d = P_iP_{i-1}
15:  while T(p,d) ≠ S(p,d)
```

[Fig. 9] Procedure of RSA[8]

한 예로 초기에 [Fig. 8](b) 내의 ①과 같은 시작 추적자에서 ②와 같은 무어의 이웃 추적을 이용하여 후속픽셀을 구한다. 이후 현재 추적자 위치 P_i에서 선행 추적자 위치 P_{i-1}간의 Ⓐ와 같은 방향벡터인 $\overrightarrow{P_iP_{i-1}}$을 구한 후, 이 벡터를 ③과 같이 시계방향으로 회전시켜 후속픽셀을 지속적으로 찾아내는 방식이 된다. 이후 Ⓔ와 같은 방향벡터를 이용하여 ⑦과 같이 시계방향으로 회전시켜 물체의 시작픽셀을 만나면서 진입방향이 같게 되면 알고리즘은 종료가 된다.

3.2.4 Theo Pavlidis 알고리즘

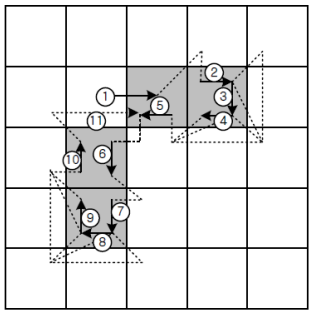
Theo Pavlidis 알고리즘(TPA)은 현재 추적자 위치로부터 좌상(UL), 상(U), 우상(UR) 방향에 위치한 물체의 픽셀을 순차적으로 탐색해 나가는 방식이다. 해당 방향에 물체의 픽셀이 하나도 없으면 추적자의 방향은 우측(R)으로 90도 회전을 한다.

Algorithm 4 Theo Pavlidis' Algorithm(TPA)

```
1: procedure TPA
2:   T(p,d) ← S(p,d)
3:   do
4:     if p_UL = object then T(p,d) ← T(p_UL, d_L)
5:     else if p_U = object then T(p,d) ← T(p_U, d)
6:     else if p_UR = object then T(p,d) ← T(p_UR, d)
7:     else T(p,d) ← T(p, d_R)
6:   while T(p,d) ≠ S(p,d)
```

[Fig. 10] Procedure of TPA[8]

예를 들어 추적자 ①이 좌상(UL)의 픽셀을 확인 후 위쪽(U)에 있는 픽셀을 만나게 되면 ②와 같이 추적자의 방향이 변하지 않게 된다. 추적자 ②와 같이 물체 내 픽셀들을 발견하지 못하면 ③과 같이 추적자의 방향은 오른쪽으로 90도 회전한다. 또한 추적자 ⑤에 대해 좌상(UL)에 픽셀이 있는 경우에는 왼쪽픽셀을 경계선에 포함하고, 추적자의 방향은 ⑥과 같이 왼쪽으로 변경이 된다. 추적자 ⑩에 대해 ⑪과 같이 물체의 시작픽셀을 만나면서 진입방향이 같게 되면 추적은 종료가 된다.



[Fig. 11] Contour-following sequence of the TPA

4. 연구방법 및 결과

4.1 연구방법

[Fig. 1](a)와 같은 바둑에서의 사활문제를 해결하기 위해 세력함수와 외곽선 추적 알고리즘을 이용하여 흑백 돌들에 대한 경계선을 찾고자 했다. 세력함수 및 외곽선 추적 알고리즘의 컴퓨터 적용을 위해 [Table 1]과 같은 컴퓨터 환경 하에서 C++ 소스프로그램을 작성하여 실험을 실시하였다.

[Table 1] Experimental environment

Desktop	
CPU	Intel(R) Core(TM) i7-5500U CPU @2.40 GHz
Memory	6.00GB
System	64 bit OS, x64 based processor
OS	Microsoft Window 10
Compiler	C++ in Microsoft Visual Studio 2019

4.2 연구결과

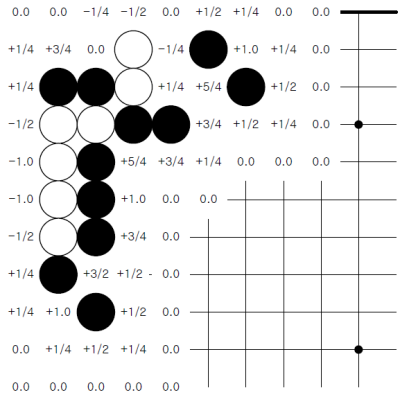
4.2.1 세력함수를 활용한 가상의 영역

본 논문에서는 [Fig. 1](a)와 같은 사활문제에 대해 식 (3)과 같은 간단한 세력함수를 이용하여 흑백간의 가상의 영역을 생성시켰다.

$$F(d) = F_0 \times 2^{-d} \tag{3}$$

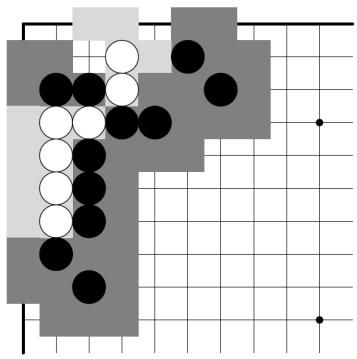
여기서 $d \leq 2$이며 돌 간의 거리가 된다. 참고로 흑돌에 대해서는 $F_0 = +1$, 백돌에 대해서는 $F_0 = -1$이 된다. 또한 $d = |\triangle x| + |\triangle y|$가 된다.

[Fig. 1](a)에 대해 세력함수를 적용하면 [Fig. 12]와 같이 바둑판 내 각 착점에 대해 세력값으로 표시되는 세력지도를 얻을 수 있다.



[Fig. 12] Influence map generated by influence function

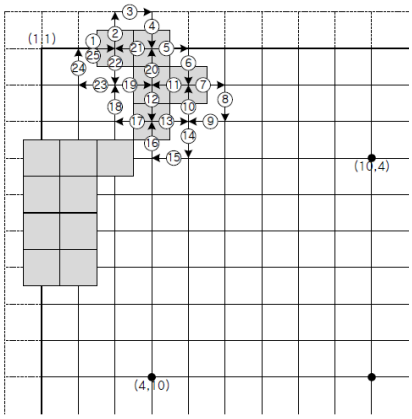
[Fig. 12]의 세력지도를 갖고 향후 외곽선 추적을 위해 흑백간의 가상의 영역을 재 작성해보면 [Fig. 13]과 같다. 참고로 짙은 회색은 흑의 영역이 되고 엷은 회색은 백의 영역이 된다.



[Fig. 13] Territories reconstructed by the influence map

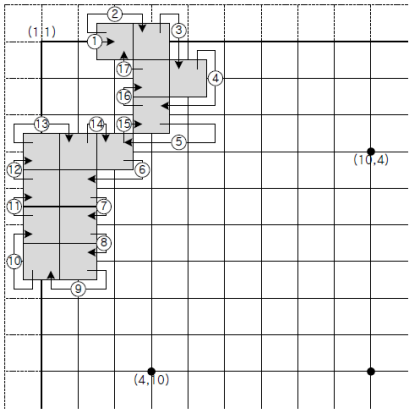
4.2.2 추적 알고리즘을 활용한 경계선

[Fig. 13]에 있는 백돌의 가상 영역에 대해 정사각형 추적 알고리즘을 적용하게 되면 [Fig. 14]에서 보듯이 백돌의 가상영역 전체에 대한 경계선을 찾지 못하는 단점을 알 수 있다.



[Fig. 14] Contour-following sequence of STA for finding White's boundary

반면에 무어의 이웃 추적을 적용하게 되면 [Fig. 15]에서 보듯이 백의 가상영역에 대한 경계선을 모두 찾을 수 있음을 알 수 있다.



[Fig. 15] Contour-following sequence of MNT for finding Black's boundary

참고로 [Fig. 16]은 컴퓨터에 의해 추출된 일련의 백의 경계선 좌푯값들이 되며, [Fig. 17]은 흑의 경계선의 좌푯값들이 된다.

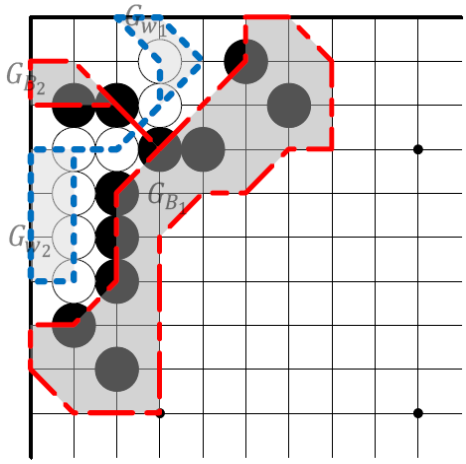
```
Contour-following sequence for white's territory:
(3,1)(4,1)(5,2)(4,3)(3,4)(2,5)(2,6)(2,7)(1,7)(1,6)
(1,5)(1,4)(2,4)(3,4)(4,3)(4,2)(3,1)
```

[Fig. 16] Contour-following sequence of White's boundary

```
Contour-following sequence for black's territory:
(6,1)(7,1)(8,2)(8,3)(8,4)(7,4)(6,5)(5,5)(4,6)(4,7)
(4,8)(4,9)(4,10)(3,10)(2,10)(1,9)(1,8)(2,8)(3,7)(3,6)
(3,5)(4,4)(3,3)(2,3)(1,3)(1,2)(2,2)(3,3)(4,4)(5,3)
(6,2)(6,1)
```

[Fig. 17] Contour-following sequence of Black's boundary

이후 흑백간의 가상영역과 경계선을 그려보면 [Fig. 18]과 같다. 흑의 영역 G_B는 두 개의 소영역인 G_{B_1}, G_{B_2}로 구성되어 있으며, 백의 영역 G_W 역시 두 개의 소영역인 G_{W_1}, G_{W_2}로 구성되어 있다. 즉 $G_B = G_{B_1} \cup G_{B_2}$이 되며 $G_W = G_{W_1} \cup G_{W_2}$가 된다.



[Fig. 18] Boundaries and territories created by MNT

4.2.3 영역간의 둘러싸임

[Fig. 18]에 있는 각 소영역 간의 둘러싸임에 대한 관계를 살펴보면 [Table 2]와 같다.

[Table 2] Correlation between Black and White groups

	G_{B_1}	G_{B_2}
G_{W_1}	○ ×	○ ○
G_{W_2}	○ ×	○ ○

[Fig. 18], [Table 2]에서 보듯이 G_{w_1}은 G_{B_1}, G_{B_2}에 의해 둘러싸여 있고, G_{w_2} 역시 G_{B_1}, G_{B_2}에 의해 둘러싸여 있다. 한편 G_{B_2}는 G_{w_1}, G_{w_2}에 의해 둘러싸여 있으나, G_{B_1}은 백돌에 의해 둘러싸여 있지 않음을 알 수 있다. 결국 [Fig. 1](a)에 있는 사활문제는 흑의 소영역인 G_{B_2}와 백의 영역인 G_W $(=G_{W_1} \cup G_{W_2})$와의 문제가 되어 좌상귀에 있는 흑의 사활문제로 귀결됨을 알 수 있다.

4.2.4 탐색용 게임트리

사활문제 해결을 위한 정확한 수순, 즉 일련의 최상의 수를 찾아내기 위해서는 착수 가능한 모든 착점들을 일일이 뿌리노드로 하여 트리탐색을 실시하여야 한다. 또한 완전탐색(brute-force search)과 같은 간단한 방법을 사용하게 되면 게임트리 내 엄청난 분기로 인해 주어진 시간 내에 문제를 해결할 수 없다[3]. 결국 제한된 뿌리노드 및 자식노드 후보군들을 설정하여, 신뢰상한 트리탐색(Upper Confidence Bounds for Trees: UCT)과 같이 활용(exploitation)이 아닌 탐험(exploration)에 기반을 두는 탐색 방법을 적용해야 된다.

이를 위해 본 실험에서 제시된 [Fig. 18]과 같은 착수 가능한 임의의 착점이 아닌 제한된 착점을 뿌리노드 및 자식노드로 우선적으로 활용하게 되면 트리탐색의 규모와 성능을 획기적으로 개선할 수 있다. 즉 [Fig. 1](a)와 같은 사활문제 해결용 트리탐색을 위해 우선적으로 사용될 뿌리노드 및 자식노드 후보군의 순위는 다음과 같이 고려할 수 있다.

(1) 1순위: G_{B_2}내의 돌들과, G_{B_2}, G_W와 근접해 있는 돌들

(2) 2순위: G_W내의 돌들과, G_W, G_{B_1}와 근접된 돌들이 된다.

이를 활용하게 되면, 향후 사활문제 풀이를 위해 구축될 신뢰상한 트리탐색과 같은 탐색 엔진의 고도화 및 정교화를 기할 수가 있다.

5. 결론 및 제언

본 논문에서는 바둑에서의 흑돌과 백돌간의 경계가 모호한 사활문제를 풀기 위해 세력함수를 이용하여 흑백간의 가상영역을 생성하였으며, 생성된 가상영역에 대한 경계선을 찾고자 외곽선 추적 알고리즘을 적용하였다. 적용결과에 따르면 여러 외곽선 알고리즘 중에 가장 간단한 정사각형 추적 알고리즘은 가상영역의 경계선을 찾는데 문제가 있었으나, 무어의 이웃 추적은 가상영역에 대한 경계선을 정확하게 찾을 수 있음을 알 수 있었다.

또한 추적 알고리즘에 의해 생성된 경계선으로 구분된 각 영역간의 둘러싸임에 대한 관계를 서술하였으며, 향후 사활문제의 최선의 수순을 찾기 위해 트리탐색을 적용 시 획기적으로 게임트리 엔진의 성능 개선 및 규모를 줄일 수 있는 방법을 제시하였다.

REFERENCES

[1] B.D. Lee, “The most promising first moves on small Go boards, based on pure Monte-Carlo Tree Search”, Journal of Korea Game Society, Vol. 18, No. 6, pp. 59-68, 2018.

[2] B.D. Lee, “The Most Promising Sequence of Moves using Monte-Carlo Tree Search on a Small Go Board”, Journal of Korean Society For Computer Game, Vol. 31, No. 3, pp. 121-129, 2018.

[3] B.D. Lee, “Multi-Strategic Learning, Reasoning and Searching in the Game of Go”, PhD thesis, Auckland University, 2005.

[4] N. Sasaki, Y. Sawada, and J. Yoshimura, “A Neural Network Program of Tsume-Go”, Journal of Computer and Games, Springer-Verlag, pp. 167-182, 1999.

[5] XuanXuanGo, “Life-And-Death Problem (Tsume-Go) Solver”, from http://www.xuanxuango.com/solver.htm, 2019.

[6] Sensei's Library, “Influence function”, from https://senseis.xmp.net/?InfluenceFunction, 2019.

[7] A. Zobrist, “Feature Extraction and Representation for Pattern Recognition and the Game of Go”, PhD thesis, University of Wisconsin, 1970.

[8] C.H. Cheng, J.H. Seo, and T.D. Han, “Advanced Contour Tracing Algorithms based on Analysis of Tracing Conditions”, Journal of KIISE, Vol. 33, No. 2, pp. 431-436, 2006.

[9] J.H. Seo, et al., “Fast Contour-Tracing Algorithm Based on a Pixel-Following Method for Image Sensors”, Journal of Sensors, Vol. 16, No. 353, pp. 1-27, 2016.

[10] Y. Batko and V. Dyminsky., “Fast Contour Tracing Algorithm Based on a Backward Contour Tracing Method”, ACIT 2018, 2018.

[11] A.G. Ghuneim. “Contour Tracing Algorithms ", from http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/contour_tracing_Abeer_George_Ghuneim/alg.html, 2019.

이병두(Byung-Doo Lee)

약력 : 1982 한양대학교 원자력공학 학사
1991 서강대학교 정보처리학 석사
2005 Auckland University 컴퓨터공학 박사
현재 용인대학교 컴퓨터학과 조교수

관심분야 : 컴퓨터공학, 인공지능, 컴퓨터바둑