論文2003-40CI-6-9

컴퓨터 바둑에서 String 안정도와 Group 영역에 의한 정적분석

(Static Analysis of String Stability and Group Territory in Computer Go)

朴 鉉 秀 *, 李 斗 漢 *, 金 恒 俊 **

(Hyun Soo Park, Doo Han Lee, and Hang Joon Kim)

요 약

우리는 컴퓨터 바둑에서 String의 안정도를 휴리스틱하게 정의하고 Group의 영역을 분할한다. String 안정도의 요소는 눈(eye, E), 유사눈(eye-like, EL), 특수눈(special-eye, SE), 확장가능점(extension-point, EX), 공배수(liberty, L), 그리고 연결가능점(connection-point, CP)이다. String의 안정도(stability)는 완전한 삶(Complete Alive, C), 삶(Alive, A), 모름(Unsettled, U), 위험(Danger, D), 그리고 죽음(Killed, K)의 5단계로 휴리스틱하게 정의한다. Group은 String과 연결 링크 점에 의해 만들어지고 Group 영역(Territory)을 가진다. Group의 영역 분할은 String과 String사이에 마름모, 한 칸, 날 일자, 그리고 두 칸의 연결 링크 점과 String의 안정도에 의해 획득되어진다. 우리는 실험을 통하여 제안한 방법과 프로기사의 형세판단을 비교하였다. 결과로 평균 오차가 8.7로 나타났다.

Abstract

We define a string stability heuristically and divide the board into group territory in computer Go. Elements of string stability are eye(E), eye-like(EL), special-eye(SE), extension-point(EX), liberty(L) and connection-point(CP). A string stability have 5 levels that are complete alive, alive, unsettled, danger and killed level. A group is made strings and link-points and have the territory. Territory division of a group is acquired by strings stability and link-points which are marym-mo, hankan, nalil-ja, and twokan between string and string. We compare our method with the result of evaluation of professional player. As a result, the mean error is 8.7.

Keywords : computer go, group, territory, evaluation, eye

Ⅰ. 서 론

컴퓨터 바둑은 흑과 백이 차례로 한 수씩 번갈아 두는 게임이다. 표준 바둑판의 크기는 19×19이고 9×9와 13×13도 사용되어지며, 반드시 먼저 흑이 두는 것으로 시작한다. 바둑 게임의 최종적인 목표는 적보다 보다 넓은 영역을 차지하는 것이다.

컴퓨터 바둑에 비해 체스 게임은 오랜 연구 결과 전문가 수준에 이를 정도로 개발이 되었지만, 컴퓨터 바둑은 지난 12-18년 동안 상당히 발전을 했지만 아직 초급자 수준에 미치는 단계에 있다. 가장 강한 컴퓨터 바둑 프로그램은 대략적으로 아마 추어 5급 수준이라고 평가하며 초반의 정석과 포석은 강하지만 중반의 전투와 수상전(semeai) 그리고 사활(life and death)은

* 正會員, 慶東情報大學 컴퓨터情報通信系列
(Kyundong college of techno-information)
** 正會員, 慶北大學校 컴퓨터工學科
(Kyungpook National University)
接受日字:2003年8月5日, 수정완료일:2003年11月7日

아직도 매우 많은 문제점을 갖고 있다[1].

바둑의 탐색 범위의 방대함은 체스나 다른 게임보다도 그 크기가 매우 크다. 서로 다른 자리에 둘 수 있는 경우 $3^{19\times19} \fallingdotseq 10^{170}$로 이것의 약 1.2%만이 타당하다. 그리고 더 엄격히 말하자면 탐색 범위는 바둑의 룰에 의해 위치 반복을 허락하지 않기 때문에 더 큰 수이다[2]. 따라서 바둑 판 전체의 탐색 보다 지역적 분할을 통한 효율적인 탐색이 무엇보다 중요하다. 이러한 방법으로 효율적인 Pruning 기법과 휴리스틱 탐색기법이 연구가 되어 왔으며 그 외에도 다른 많은 방법으로 문제를 해결하고자 하였다. 특히, Knowledge Base에 의한 접근 방법으로 적용되어지는 정석은 매우 높은 수준에 이르렀다. 바둑판에서 얼마만큼 정확한 정보를 획득하는가에 따라 문제 접근 방식이 달라지며 효율적 문제 해결도 가능해진다. 즉, 공격의 시점인가 혹은 방어의 시점인가? 아니면 정석을 선택하여야 하는가? 혹은 사활이 걸린 문제인가? 아니면 영역을 넓여야 하는가? 아니면 안정적으로 끝내기를 해야하는가? 축이 걸렸는가? 패가 반드시 필요한가? 등 매우 많은 문제에 있어 상대의 수에 대해 자신의 수를 찾아야 한다.

그러므로 문제의 범위를 축소하는 것은 해를 찾기가 용이하다. 본 논문은 이러한 문제의 범위를 축소하기 위해 바둑판에 놓인 가장 기본적인 단위인 string의 안정도와 group 영역을 의미 있게 분할한다. 바둑판의 영역 분할을 위해서 string 안정도를 휴리스틱하게 정의한다. 휴리스틱한 방법은 경험적 지식을 기반으로 최적의 해를 구하지는 못하지만 문제의 해를 얻을 수 있는 방법중의 하나이다. 안정도의 요소를- 눈(eye, E), 유사눈(eye-like, EL), 특수눈(special-eye, SE), 확장가능점(extension-point, EX), 공배수(liberty, L), 그리고 연결가능점(connection-point, CP)- 이용하여 5단계로 휴리스틱하게 정의한다. 그리고 string과 string 사이에 마름모, 한 칸, 날 일자, 그리고 두 칸의 연결 링크 점을 이용하여 group을 만든다. group의 영역(territory)은 안정도와 연결 링크 점에 의해 획득되어진다. 궁극적으로 string의 안정도와 group 영역의 획득에 의한 정적분석이 프로기사의 형세판단과 비교하여 타당한 방법임을 보이고자 한다.

Ⅱ. 관련 연구

바둑에서 정적분석은 string의 안정도와 group의 영역에 대한 부분으로 안정성(safety)에 관련하여 전술적 탐색에 의해 향상된 휴리스틱 평가함수에 의해 개산되어진다. 완전 탐색에 의한 완벽한 함수의 구현은 문제의 범위확대와 이웃한 group의 사활과 직접적으로 관련이 있으므로 매우 어려운 문제로 확대 될 수 있다. 그러므로 정적인 분석에 의해 돌(stones)의 안정성과 string의 안정성 그리고 영역의 안정성 평가가 선행되어지든지 아니면 어느정도의 전술적 탐색에 의해 안정도 평가가 이루어지는 방식으로 연구가 진행되어졌다.

David B.Benson은 safe(unconditionally alive)를 그래프 이론적 정적분석에 의해 정의하였다. string의 집합에 속한 영역이 두 개의 서로 다른 vital 영역을 가질 때 string의 집합은 unconditionally alive이다[3]. 그리고 safety와 unconditional life는 동치임을 증명하였다. 이는 매번 패스를 한다고 할 지라도 바둑 룰에 의거하여 alive가 되는 것을 의미하는 것으로 바둑에서 사활의 가장 기본이 되는 것이다. 또한 Landman은 Bargo model을 사용하여 다양한 eye-spaces의 값을 계산하였다[4]. 그리고 Popma and Allis는 'X life'를 제안하였는데 X번 패싱 후에도 여전히 alive할 수 있는 것을 의미한다[5]. Wolf는 사활 문제를 트리 탐색을 사용하여 해결하려고 연구하였다[6]. Martin Müller는 Benson의 unconditional safety에 덧 붙여 번갈아 가며 두는 것에 대해 정의하였으며 또한 Chain 형태의 문제를 제기하여 Chain으로 확장하여 정의를 하였다[7]. 그리고 Müller는 그의 프로그램인 Explorer에 정확한 평가부분과 휴리스틱 영역 평가로 구분하여 평가하여 전체판의 평가를 하였다. 정확한 평가는 safe territory, 정적인 룰에 의한 semeai 해결 그리고 Endgame Areas에 thermograph의 mast value를 사용하였으며, 휴리스틱 영역 평가는 흑과 백에 대한 dividers와 potential deviders로 나타내고 zones을 safe, potential과 threatend으로 구분하였다. 그리고 points에는 near, junction 그리고 far away로 구분하였다. 그리고 전체판 평가는 정확한 평가 부분에서 각 로컬의 평균이 계산되고 더해진다. 판의 나머지 부분은 휴리스틱하게 평가되는데 각 형태의 points의 수가 먼저 계산되어진다. 이 수에 weight factor(black points: +1, white points: -1, Potential territory: ±1/2, near point:±0.2, junction and far-away points: 0)를 곱해서 전체 스코어가 나온다[10].

또한, Ken Chen과 Zhixing Chen은 Go Intellect와

HandTalk에서 트리 검색의 효과를 대략적으로 낼 수 있는 방법으로 group의 내부와 둘러싸고 있는 상황에 대해 정적 분석하였다. 이들은 적군의 죽은 돌을 가지거나 혹은 가지지 않은 eye-spaces를 연구하여 바둑의 게임트리의 기하급수적 폭발을 해결하기 위한 기초를 제공하려고 했다. 눈은 empty points와 적의 dead stones이 될 수 있으며 바둑판의 첫줄과 첫줄 이상에서의 eye-point의 분류(False eye-point, Half eye-point, Full eye-point)를 달리 하고 eye-regions의 상황에 따라 눈의 수를 휴리스틱하게 정의하였다.

실제 바둑에서 형세판단법은 정밀계산법, 비교상쇄법 그리고 개산법의 세 가지 방법이 있다. 형세판단은 미완성된 영역의 가치와 세력의 잠재력 그리고 수 읽기, 전체적 안목과 계산 능력등 고도의 테크닉이 요구된다. 그러나 컴퓨터 바둑에서는 종반의 형세판단은 매우 높은 수준이라 볼 수 있지만 중반과 초반의 형세에서 다소 정확하지 못한 면이 있다. 이는 종반일 경우 영역이 확정되어지고 세력이나 수 읽기가 줄어드는 점이 특징이라 볼 수 있다. 그러므로 본 논문에서는 중반의 group 영역의 획득과 프로기사의 형세판단과 차이를 비교한다.

Ⅲ. 안정도(string stability)

바둑의 반면을 형성하는 가장 기본적인 요소를 점(BoardPoint)으로 정의한다[11]. 반면에 존재하는 점은 색(color) 속성을 가지는데 공배(empty), 흑(black) 그리고 백(white)으로 stone_color로 표현되고, 돌(stone)들은 놓여진 순서인 수순에 의해 번호를 매길 수 있는데 이를 id로 표현한다. 공배(empty)일 경우 흑돌의 영향력과 백돌의 영향력을 받을 수 있는데 이러한 영향력 값은 value에 각각 저장한다. 그리고 각 돌들은 string의 번호로 string_id로 표현된다. group 번호는 group_id로 표현한다. BoardPoiint를 아래와 같이 표현한다.

```
class BoardPoint{
 private:
 int stone_color; //돌 색: empty, black, white
 int id; //수순
 int value; //영향력 값
 int string_id; // 소속 string 번호
 int group_id; // 소속 group 번호
 ......
 public:
 void SetStoneColor(int,int,int);
 void SetId(int,int,int);
 void SetStringId(int,int,int);
 void SetGroupId(int,int,int);
 void SetValue(int,int,int,int);
 int GetStoneColor(int,int);
 int GetId(int,int);
 int GetStringId(int,int);
 int GetGroupId(int,int);
 int GetValue(int,int,int);
 void Init_Pan(int,int);
 void InitStringNumber(int,int);
 void Pan_Input(int , int , int);
 void Stone_Check();
 void Dont_Put();
 void Capture_String();
 void Inf_Change_Check();
 int PutStoneOk(int x, int y);
 ......
} BoardPoint[PAN_SIZE][PAN_SIZE];
```

본 논문에서는 string의 안정도를 휴리스틱한 방법을 사용하여 정의하고자 한다. string의 정의는 Benson[1]과 Kierulf[9]에 따라 같은 색의 연결된 돌들이라고 한다.

string은 바둑 표현에 있어 가장 기본적 요소이고 liberty를 잃게 되면 그 string은 잡히게(captured) 되고 바둑판에서 제거되어야 한다. 동일한 색을 가진 이웃한 돌들을 <그림 1>에서 string의 예로 보여준다.

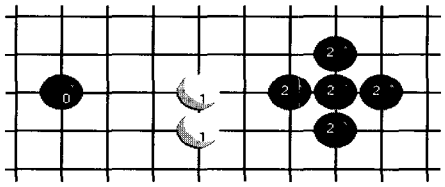

그림 1. string의 예
Fig. 1. Example of strings.

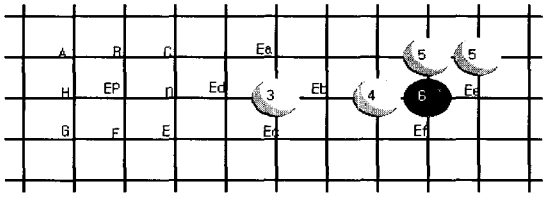

그림 2. 점(EP)과 이웃점(A-H) 그리고 liberty(Ea-Ef)
Fig. 2. A point(EP), adjacent point(A-H) and liberty (Ea-Ef).

liberty는 돌(stone)에 인접한 공배를 의미하는데 <그

림 2>은 한 점(EP)에 이웃한 점과 돌(stone)의 liberty를 보여 준다. liberty는 string의 돌에 인접한 공배이며, 여기서 Ea - Ed가 인접한 공배이며 우측의 예는 Ee와 Ef가 공배이다. 이러한 liberty들은 string의 안정도에 매우 중요한 영향을 미친다. Wilcox는 5개의 liberty를 가지면 전술적으로 안정적이라고 휴리스틱하게 제안하였다.

string의 정의에 따라 반면에서 string을 표시한 것이 <그림 1>이다.

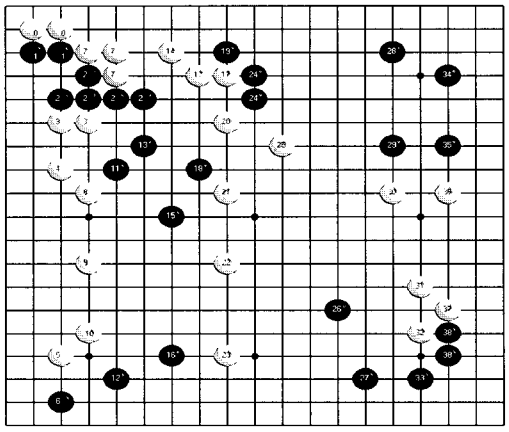

그림 3. 반면에서 strings
Fig. 3. strings on the board.

본 논문에서는 string의 안정도(stability)를 5 level로 구분한다. string의 안정도(stability)는 휴리스틱한 방법으로 정의를 한다. 안정도(stability)는 완전한 삶(Complete Alive, C), 삶(Alive, A), 모름(Unsettled, U), 위험(Danger, D), 그리고 죽음(Killed, K)의 레벨로 정의한다. 이러한 구분은 안정도를 이루는 요소들의 값에 의해 이루어진다. 이 요소들은 string의 눈(eye, E)의 개수, 유사 눈(eye-like, EL)의 개수, 특수 눈(special-eye, SE)의 개수, 확장가능점(extension-point, EX)의 개수, 공배(liberty, L)의 개수 그리고 연결가능점(connection-point, CP)의 개수에 의해 정의되어진다. 공배수(liberty, L)는 string에 인접한(adjacent) 공배(dame)의 개수를 의미하는 것으로 공배가 0이면 완전히 죽은 string으로 바둑판에서 제거되어진다. 그러므로 공배가 적다는 것은 매우 위험한 상황으로 직접적으로 사활에 영향을 미치는 요소이다.

Definition 1.: C(s,element) = number of element of s is sting_id.

<그림 2>에서 string 3번의 공배는 Ea-Ed으로 4개로 C(3,L) = 4이고 string 6번의 공배는 Ee와 Ef로 2개로 C(6,L) = 2이다. 그림에서 만 본다면 string 6은 공배수가 0에 가깝기 때문에 매우 위험한 상태이며, 축머리에 아군이 없으면 축으로 잡힐 수 있는 경우이다. 그러므로 공배수가 적은 이웃한 적을 공격하는 것은 유리한 것이고 반대로 아군의 공배가 적으면 다른 조치가 필요할 것이다. <그림 2>에서 눈(eye, E)는 임의의 공배(EP, dame)에 인접한 공배(B,D,F,H)가 모두 동일한 string이라면 이러한 공배를 눈이라 정의한다. 눈은 삶에 있어 가장 기본이 되는 요소로 두 눈이 있으면 무조건 삶이다. 그러므로 눈은 사활문제나 수상전에서는 삶의 기본이 되고, game tree를 이용할 경우 마지막 leaf node에서의 평가 항목이 될 것이다. 아래는 임의의 점-posit이 눈인지 아닌지에 대한 판단 알고리즘이다.

```
int IsEye(int posit, int string_id) // posit is EP point. and string_id is the number of string id.
{ int x, y;
 int count;
 x = posit /100; y = posit % 100;
 count = 0;
 if ( y > 0){ if(BoardPoint[x][y-1].GetStringId(x,y-1) == string_id) count++;}
 else if ( y == 0) count++;
 if ( x < 18){ if (BoardPoint[x+1][y].GetStringId(x+1,y) == string_id) count++; }
 else if ( x == 18) count++;
 if ( y < 18){ if (BoardPoint[x][y+1].GetStringId(x,y+1) == string_id)  count++;}
 else if ( y == 18) count++;
 if ( x > 0){ if (BoardPoint[x-1][y].GetStringId(x-1,y) == string_id) count++;}
 else if ( x == 0) count++;
 if ( count == 4 ) return 1;
 else return 0;// not eye.
}
```

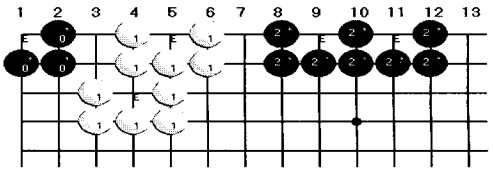

그림 4. 눈의 예
Fig. 4. Example of eyes.

아래의 <그림 4>에서 C(0,E) = 1, C(1,E) = 2 그리고 C(2,E) = 2이다. 그러므로 string 1과 2의 경우 어떠한 경우라도 바둑 룰에 의해 죽지 않는 완전한 삶의 에이다.

유사눈(eye-like, EL)은 <그림 2>에서 공배(EP) 점에서 8방향의 점(A-H)들 중에 모두 적군이 아니고, 눈이면 count를 1 증가시키고, 적이 못 놓는 자리 일 경우 count를 1 증가시키고, 아군이면 count를 1증가 시켜서 count가 6이상이면 이 공배를 유사눈(eye-like, EL)이라 한다. 유사눈(EL)을 구하는 알고리즘을 아래와 같이 구현하였다.

```
int IsEyelike(int posit, int string_id)
{int count;
 count = 0;
 if ( is posit a coner point ) count = 5; // if coner points
 if ( is the posit edge point ) count = 3;// if edge points
 if ( one more points of adjacency points are enemy points
) retrun 0; // this EP is not EL
 while( adjacent points exist ) {
 if (a adjacent point is eye) count++;
 if (a adjacent point is Dontputposition) count++;
 if (a adjacent point is our forces) count++;
 }
 if (count >= 6 ) return 1;//this EP is EL
 else return 0;//not EL
}
```

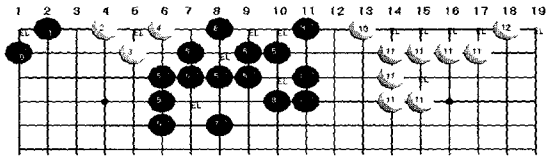

그림 5. 유사눈(eye-like)의 예
Fig. 5. Example of eye-like.

<그림 5>에서 C(0,EL) = C(1,EL) = C(2,EL) = C(3,EL) = C(4,EL) = C(7,EL) = C(10,EL) = 1 이고 C(5,EL) = C(11,EL) = 5 그리고 C(6,EL) = C(12,EL) = 2이다. 이러한 유사눈(EL)은 언제든지 눈(E)으로 되기 용이하지만, 유사눈(EL)이 덩어리로 되어 있으면 유사눈(EL) 덩어리들의 개수와 각 덩어리에 속한 유사눈(EL)의 개수가 중요하다. 11번 string의 EL 중에서 변에 위치한 EL 덩어리들은 일반적 유사눈(EL)과는 다르게 처리할 필요성이 있는데 유사눈(EL) 덩어리들의 개수와 덩어리에 속한 개수를 표현하는 정의를 아래와 같이 한다.

Definition 2.: Lump(s,EL) = number of lump of s.
s is a string_id.

Definition 3. : Clump(EL,n) = number of EL of n.
n is a coordinate value(x*100+y) and EL.

<그림 5>에서 Lump(11,EL) = 2이며 Clump(EL, 1400) = 4가 되어진다. 그리고 Lump(5,EL) = 4이고 Clump(EL,801) = 1 이고 Clump(EL,900) = 2이다.

특수눈(special-eye, SE)는 <그림 2>에서 공배점(EP)에서 8방향의 점들 중에 눈(E)이면 count를 1 증가시키고, 적이 못 놓는 자리 일 경우 count를 1 증가시키고, 아군이면 count를 1증가 시켜서 count가 6이상이고, Angle point(A,C,E,G)중 적군 돌이 1개 이하이면 특수눈(SE)이다. 단, 바둑판에서 coner points((0,0),(18,18), (18,0),(0,18))나 edge point((0,y)(x,0),(18,y),(x,18),0<=x,y<=18)의 공배점(EP)의 경우에는 적군 돌이 존재하지 않아야 한다. 특수눈(SE)을 구하는 알고리즘을 아래와 같이 구현하였다.

```
int IsSpecialEye(int posit, int string_id)
{int count;
 count = 0;
 if ( is posit a coner point ) count = 5;
 if ( is the posit edge point ) count = 3;
 if ( adjacent points are enemy and are not angle points )
return 0; // this EP is not SE.
 if ( adjacent points are a coner point or a edge point and
enemy ) return 0; // this EP is not SE
 if ( two more angle points are enemy ) return 0; // this
EP is not SE.
 while( adjacent points exist ) {
 if (a adjacent point is eye) count++;
 if (a adjacent point is Dontputposition) count++;
 if (a adjacent point is our forces) count++;
 }
 if (count >= 6 ) return 1;//this EP is EL
 else return 0;//not EL
}
```

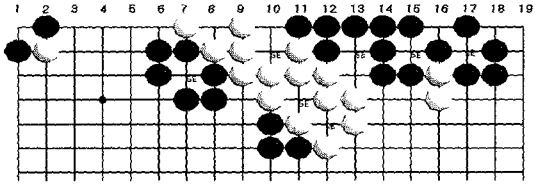

그림 6. 특수눈(SE)의 예
Fig. 6. Example of special-eye(SE).

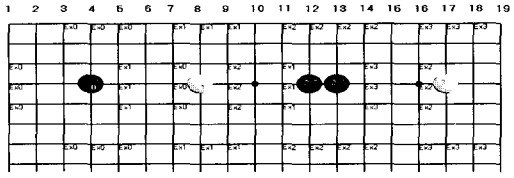

그림 7. 확장점(EX)의 예
Fig. 7. Example of extension-point(EX).

<그림 6>은 특수눈(SE)의 예를 보여 준다. 좌표(0,0)은 (1,1)이 인접한 흑돌과 다르게 백돌이고 즉, 적군이므로 특수눈(SE)가 될 수 없다. 이와 같은 이유로 (0,10)의 점도 인접한 점(0,11)이 흑돌이므로 특수눈(SE)가 될 수 없다. 이러한 특수눈(SE)는 거의 눈에 가까운 형태로 되어 있다.

확장점(extension-point, EX)은 string이 두 칸 벌림이나 눈목자 벌림 위치의 공배 점으로 string과 확장점 사이의 링크에 인접한 점에는 적의 돌이 없어야 한다. <그림 7>은 string의 확장점(EX)을 보여 준다. C(0, EX) = C(1,EX) = 12이며, C(2,EX) = 14이고, C(3,EX) = 9이다. 이러한 확장점(EX)는string이 위험한 상태일 때 도주로 정보로 이용 될 수 있다. 또한 초기 바둑판에서는 자신의 영역을 최소한으로 늘이는 정보로 이용가능하다. 이러한 확장점(EX)은 string의 안정도에도 중요한 요소이다.

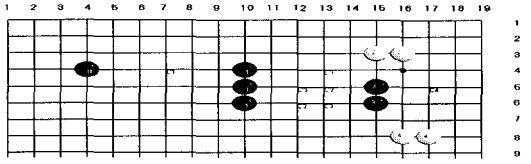

그림 8. 연결점(CP)의 예
Fig. 8. Example of connection-point(CP).

연결점(connection-point, CP)은 마름모(Marym-Mo, MM), 한 칸(HanKan, HK), 날일자(Nalil-Ja, NJ), 혹은 두 칸(TwoKan, TK) 위치에 둘 때 아군의 다른 string과 마름모(MM), 한 칸(HK), 날일자(NJ) 그리고 두 칸(TK)으로 연결되어 group을 형성하게 될 수 있는 점을 의미한다. 이러한 연결점(CP)이 많으면 주변에 아군이 많으므로 위험의 정도가 낮아지며 적군에 대응하기가 용이하다. 반대로 연결점(CP)는 상대 적의 string과 string의 연결점을 선점하여 끊는 점으로 이용될 수 있다. string의 안정도가 낮은 상태이면 강력한 연결이 필요하므로 마름모(MM) 연결점이나 한 칸(HK) 연결점을 사용하는 것이 유리할 것이고 string의 안정도가 높지만 적의 공격을 받고 있으면 날일자(NJ) 연결점이나 투 칸(TK) 연결점을 이용하여 연결하는 것이 유리하다.

```
void FindConnectNum(int posit)
{
if(Is posit connected our force when put a marymmo position)
 add connection point to list of CP
if(Is posit connected our force when put a hankan position)
 add connection point to list of CP
if(is posit connected our force when put nalilja position)
 add connection point to list of CP
if(Is posit connected our force when put twokan position)
 add connection point to list of CP
}
```

<그림 8>은 연결점들을 보여 주고 있다. string 0와 string 1과의 연결점은 c1이고 string1과 string3과의 연결점은 c3이다. 그리고 string 2와 string4의 연결점은 c4이다. 즉, C(0,CP) = 1이고 C(1,CP) = 6이고 C(3,CP) = 5이다. 그리고 C(2,CP) = C(4,CP) = 1이다.

아래는 string을 클래스로 정의한 부분이다. string 클래스는 string의 안정도(stability), 눈의 개수, 눈의 좌표, 특수눈의 개수, 특수눈의 좌표, 유사눈의 개수, 유사눈의 좌표, 색, 돌의 개수, 돌의 좌표, 공배의 개수, 공배의 좌표, 이웃한 우리편 스트링의 개수 등의 정보를 유지하고 이에 따른 멤버 함수를 가진다.

```
class TGString{
private:
 int Stability;                 // 스트링의 안정도
 int NoOfEye;                   //눈의 개수
 TStoneList *Eye;               //눈의 좌표들
 int NoOfSpecialEye;            //특수눈의 개수
 TStoneList *SpacialEye;        //특수눈의 좌표들
 int NoOfEyeLike;               //유사눈의 개수
 TStoneList *EyeLike;           //유사눈의 좌표들
 int Color;                     //색
 int NoOfStone;                 //스트링에 속한 돌의 수
 int NoOfEmpty;                 //스트링에 속한 공배의 수
 TStoneList *Member;            //스트링에 속한 돌들의 좌표
 TStoneList *Empty;             //스트링에 속한 공배의 좌표
 int NoOfEscapePoint;           //스트링이 도망 좌표 개수
 TStoneList *EscapePoint;       //스트링이 도망 좌표
 int Status;                    //스트링의 상태
 int SizeOfArea;                //스트링의 영역
 TStoneList *Areapoint;         //스트링의 영역 좌표
 int NoOfOurArmy;               //이웃한 우리편 스트링 개수
 TStoneList *OurArmy;           //이웃한 우리편 스트링 좌표
 int group_id;                  //스트링이 속한 그룹의 번호
```

```
 int NoOfConnect;          //이웃한 우리편과 연결점의 수
 TStoneList *ConnectPoint; //이웃한 우리편과 연결점
 int NoOfExtend;           //영역을 확장할 수있는 점의 수
 TStoneList *ExtendPoint;  //영역을 확장 점 좌표
 int SizeOfInfluence;      //영향력의 크기
 TStoneList *InfluencePoint; //영향력에 속한 점 좌표
public:
 void SetStability(int stringid, int stability);
 void SetGroupId(int StringId, int GroupId);
 ....
 int GetStability(int);
 int GetGroupId(int StringId);
 ....
 };// end of string class
```

좌표의 값은 x*100+y값으로 정수 값으로 표현되어진다.

Ⅳ. 그룹과 영역(group and territory)

그룹(group)은 string들의 연결성과 협공 그리고 전략적 차원에서 중요한 요소이다. group은 string들간의 link에 의해 구성되어지며, string의 안정도에 매우 많은 영향을 받는다. 그룹(group)은 string과 string의 링크(link, L) 점에 의해 생성이 되어지는데 마름모, 한 칸, 날 일자, 그리고 두 칸 연결이 가능할 때 이러한 연결 공배 점을 링크 점으로 정의한다. 예를 들면, <그림 9>에서 마름모 연결 링크 점으로 연결되어지는 string으로는 0과 1 그리고 1과 2가 있으며, 한 칸 연결 링크 점으로 연결되어지는 string 으로 6과 7이 있다. 그리고 날 일자 연결 링크 점으로는 연결되어지는 string으로 2와 3 그리고 5와 6이 있다. 두 칸 연결 링크 점으로는 7과 8이 연결이 되어진다.

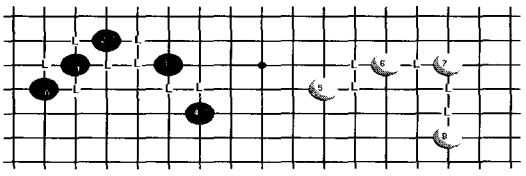

그림 9. 연결점(L)의 예
Fig. 9. Example of link-points(L).

이러한 링크 점을 사용하여 group이 형성되어지는데 string 0,1,2,3,4는 하나의 group이 되어진다. 또한 string 5,6,7,8은 또 다른 하나의 group이 되어진다.

영역(territory)은 바둑 게임의 최종 목표인 큰 영역을 가진 자가 이긴다는 원칙에 의거 매우 중요한 요소이다. 또한 삶의 기본 요소인 눈(eye)을 포함하고 있으므로 사활 요소에도 중요한 요소이다.

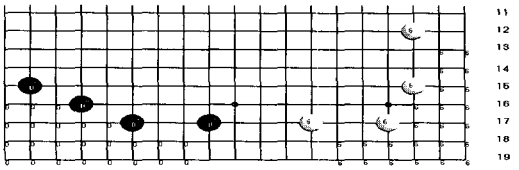

그림 10. 그룹영역(group territory)의 예
Fig. 10. Example of group territory.

<그림 10>에서 group 0의 영역(territory)일 경우 0을 표시하여 영역이 표시되어지며, group 6의 영역(territory)일 경우 6을 표시하여 영역을 나타내었다. 그리고 각 그룹 영역의 크기는 C(0,A) = 24이고 C(1,A) = 24이다. group의 영역(territory)를 찾기 위해 먼저 string들간의 link 점 들을 찾고 string과 변과의 link도 찾는다. 그리고 link 점들로부터 양 끝점을 찾는다. 양 끝점 중에 적은 값에서 큰 값으로 증가 시켜 영역(territory)을 찾는다.

<그림 11>은 반면에서 그룹영역(group territory)의 예를 보여 주는 것으로 각 그룹의 번호를 그 영역과 링크에 표시를 하였다.

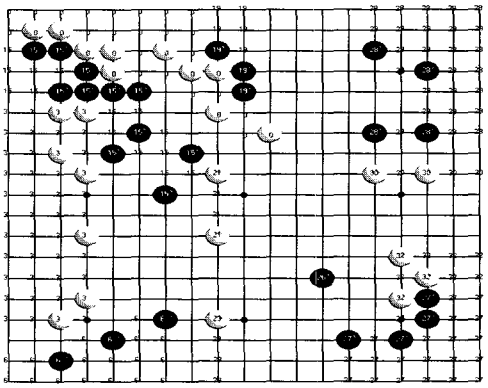

그림 11. 그룹영역(group territory)의 예
Fig. 11. Example of group territory.

Ⅴ. 안정도(stability)

string의 안정도(stability)는 휴리스틱한 방법으로 정의를 한다. 안정도는 크게 5레벨로 구분하며 완전한 삶(Complete Alive, C), 삶(Alive, A), 모름(Unsettled, U), 위험(Danger, D), 그리고 죽음(Killed, K)의 레벨로 정의한다. 그리고 완전한 삶(C)은 6개 레벨로 다시 세분

화하여 정의한다. 예를 들어 상수 값 CA(100)는 눈(E)의 개수가 2개 이상이거나 확장가능점(EX)이 12개 이상이거나 유사눈(EL)과 특수눈(SE)의 합이 10개 이상이거나 영역의 크기가 12이상일 경우이다. 즉, 눈이 2개 이상이면 무조건 사(Unconditional alive)는 기본적 규칙을 적용하였고 확장점(EX)이 12개 이상이라는 것은 거의 연속적으로 3 수 이상 두지 않으면 잡히기 어려운 상태이다. 그리고 유사눈(EL)과 특수눈(SE)의 합이 10개 이상이고 유사눈(EL)의 덩어리가 3이상이라는 것은 유사눈(EL)일 경우 덩어리 형태로 구성될 경우 변에서 최소 7개 이상이 존재하면 살 수 있다. 그리고 덩어리가 3개 이상이라는 의미는 유사눈(EL)이 각각 떨어져서 3개를 형성하고 있으므로 흑백이 번갈아 두는 바둑에서 먼저 둘 경우 눈 2개를 완벽하게 만들 수 있다. 그리고 영역(territory)의 크기가 12개 이상이라는 의미는 삶의 충분한 영역을 휴리스틱하게 정의하였다.

표 1. string 안정도의 휴리스틱 분류
Table 1. Heuristic classification of string stability.

구분	상수값		휴리스틱 조건
완전한 삶 (C)	100	CA	C(s,E)>=2 or C(s,EX)> 12 or (C(s,EL)+C(s,SE)>=10 and Lump(s,EL)>=3) or territorysize>=12
	101	CB	C(s,E)==1 and {C(s,EX)>=10 or ((C(s,EL)+C(s,SE)>=9) and Lump(s,EL)>=3 and C(s,SE) >= 3) or territorysize >= 11 or C(s,CP) >=12}
	102	CC	C(s,E) == 1 and {C(s,EX) >= 8 or ((C(s,EL)+C(s,SE) >= 8) and Lump(s,EL) >=3 and C(s,SE) >= 3) or territorysize >= 10 or C(s,CP) >=11}
	103	CD	C(s,E) == 1 and {(C(s,EX) >= 6 or ((C(s,EL)+C(s,SE) >= 7) and Lump(s,EL) >=2 and C(s,SE)>= 2) or territorysize >=11 or C(s,CP) >=10}
	104	CE	C(s,E) == 1 and {(C(s,EX) >= 4 or ((C(s,EL)+C(s,SE) >= 6) and Lump(s,EL) >=2 and C(s,SE) >= 2) or territorysize >=8 or C(s,CP) >=9}
	105	CF	C(s,E) == 1 and {C(s,EX) >= 2 and ((C(s,EL)+C(s,SE) >= 5) and Lump(s,EL) >=2 and C(s,SE)>= 2) or territorysize >= 7 or C(s,CP) >=8}
삶 (A)	200	AA	C(s,E) == 1 and {C(s,EX) >= 1 or ((C(s,EL)+C(s,SE) >= 4) and Lump(s,EL) >=2 and C(s,SE)>= 1) or territorysize >= 6 or C(s,CP) >=7}
	201	AB	C(s,E) == 0 and {(C(s,EX) >= 3 and C(s,L) >= 8) or C(s,CP) >= 6 or (C(s,EL)+C(s,SE) >= 7 and Lump(s,EL) >=2)} and territorysize>=10
	202	AC	C(s,E) == 0 and {(C(s,EX) >= 2 and C(s,L) >= 8) or C(s,CP) >=5 or (C(s,EL)+C(s,SE) >= 6 and Lump(s,EL) >=2)} and territorysize> 9
	203	AD	C(s,E) == 0 and {(C(s,EX) >= 2 and C(s,L) >= 7) or C(s,CP) >=4 or (C(s,EL)+C(s,SE) >= 5 and Lump(s,EL) >=2)} andr territorysize>=8
	204	AE	C(s,E) == 0 and {(C(s,EX) >= 2 and C(s,L) >= 6) or C(s,CP) >=3 or (C(s,EL)+C(s,SE) >= 4 and Lump(s,EL) >=2)} and territorysize>=7
	205	AF	C(s,E) == 0 and {(C(s,EX) >= 2 and C(s,L) >= 5) or (C(s,EL)+C(s,SE) >= 4 and Lump(s,EL) >=2)} and territorysize>=6
	206	AG	C(s,E) == 0 and {(C(s,EX) >= 2 and C(s,L) >= 4) or (C(s,EL)+C(s,SE) >= 4 and Lump(s,EL) >=2)} and territorysize>=6

보통 6개 이상이면 사는 모양이 될 수 도 있겠지만 둘러싸고 있는 적의 모양에 따라 완전히 살았다고 정적으로 평가하기는 어렵다. 그래서 충분히 크기가 큰 영역을 완전한 삶의 기준으로 정의한다.

삶(A)은 7개 레벨로 세분화하여 정의한다. 예를 들어 상수 값 AA(200)는 눈(E)의 개수가 1개이고 확장점(EX)이 1개 이상이거나 유사눈(EL)과 특수눈(SE)의 합이 4이상, 유사눈(EL)의 덩어리가 2이상, 특수눈(SE)이 1이상이거나 영역의 크기가 6이상이거나 연결점(CP)가 7이상이다.

모름(U)은 4개 레벨로 세분화하여 정의한다. 예를 들

표 2. string 안정도의 휴리스틱 분류(계속)
Table 2. Heuristic classification of string stability.

구분	상수값		휴리스틱 조건
모름 (U)	300	UA	{C(s,EX) >= 3 and (C(s,L) >= 5 or C(s,EL) >= 4 or C(s,SE) >= 3) and territorysize<=9} or C(s,CP) >=8
	301	UB	{C(s,EX) >= 3 and (C(s,L) >= 4 or C(s,EL) >= 3 or C(s,SE) >= 2) and territorysize<=9} or C(s,CP) >=7
	302	UC	{C(s,EX) >= 2 and (C(s,L) >= 4 or C(s,EL) >= 3 or C(s,SE) >= 2) and territorysize<=9} or C(s,CP) >=6
	303	UD	{C(s,EX) >= 2 and (C(s,L) >= 3 or C(s,EL) >= 3 or C(s,SE) >= 2) and territorysize<=9} or C(s,CP) >=5
위험 (D)	400	DA	{C(s,EX) >= 1 and (C(s,L) >= 7 or C(s,EL) >= 5 or C(s,SE) >= 3) and territorysize<=9} or C(s,CP) >=4
	401	DB	{C(s,EX) >= 1 and (C(s,L) >= 6 or C(s,EL) >= 4 or C(s,SE) >= 2) and territorysize<=9} or C(s,CP) >=3
	402	DC	{C(s,EX) >= 1 and (C(s,L) >= 5 or C(s,EL) >= 3 or C(s,SE) >= 1) and territorysize<=9} or C(s,CP) >=2
	403	DD	C(s,EX) >= 1 and (C(s,L) >= 4 or C(s,EL) >= 2) and territorysize<=9
	404	DE	C(s,EX) >= 1 and (C(s,L) >= 3 or C(s,EL) >= 1) and territorysize<=9
	405	DF	C(s,EX) == 0 and (C(s,EL) >= 4 or C(s,SE) >= 3) and territorysize<=9
	406	DG	C(s,Ex) == 0 and (C(s,EL) >= 3 or C(s,SE) >= 1) and territorysize<=9
	407	DH	territorysize >= 6 or C(s,L) >= 6
죽음 (K)	500	KA	C(s,EX) == 0 and C(s,SE) == 0 and C(s,E) == 0 and C(s,EL) >= 2 and C(s,L) >= 5 and territorysize>5
	501	KB	C(s,EX) == 0 and C(s,SE) == 0 and C(s,E) == 0 and {C(s,EL) >= 2 or C(s,L) >= 6 or territorysize>5}
	502	KC	C(s,EX) == 0 and C(s,SE) == 0 and C(s,E) == 0 and C(s,EL) >= 1 and C(s,L) >= 6 and territorysize<=5
	503	KD	C(s,EX) == 0 and C(s,SE) == 0 and C(s,E) == 0 and C(s,EL) >= 1 and C(s,L) >= 5 and territorysize<=5
	504	KE	C(s,EX) == 0 and C(s,SE) == 0 and C(s,E) == 0 and C(s,EL) >= 1 and C(s,L) >= 4 and territorysize<=5
	505	KF	C(s,EX) == 0 and C(s,SE) == 0 and C(s,E) == 0 and C(s,EL) >= 1 and C(s,L) >= 3 and territorysize<=5
	506	KG	C(s,EX) == 0 and C(s,SE) == 0 and C(s,E) == 0 and C(s,EL) >= 0 and C(s,L) >= 3 and territorysize<=5
	507	KH	C(s,EX) == 0 and C(s,SE) == 0 and C(s,E) == 0 and C(s,EL) >= 0 and C(s,L) >= 2 and territorysize<=5
	508	KI	C(s,EX) == 0 and C(s,SE) == 0 and C(s,E) == 0 and C(s,EL) >= 0 and C(s,L) >= 1 and territorysize<=5

어 상수값 UA(300)는 확장가능점(EX)이 3이상이고 공배(L)가 5이상이거나 유사눈(EL)이 4이상이거나 특수눈(SE)이 3이상 그리고 영역의 크기가 9이상이거나 연결점(CP)이 8이상이다.

죽음(K)은 9개 레벨로 세분화하여 정의한다. 예를 들면, KA(500)는 확장가능점(EX),눈(E)과 특수눈(SE)은 0이고유사눈이 2이상이고 공배가 5이상이고 내부영역이 5초과이다. 확장 가능점(EX)이 0이라는 말은 적에 의해 완전히 둘러싸인 상태이며 눈(E)과 특수 눈(SE)이 0이고 영역의 크기가 5초과일 경우에 유사 눈(EL)의 개수와 공배(L)의 개수에 의존한 안정도를 정의한다. 안정도에 있어 가장 중요한 눈(E)이 없고 적에 완전히 포위된 상태에서 특수 눈(SE)도 없으며 영역의 크기가 5초과로 유사 눈(EL)과 공배(L)만 가지고 있는 상태이다.

VI. 실험 및 고찰

형세판단에서 볼 때 영역은 중요한 요소 가운데 하나이다. 그러나 세력, 선수, 다음 둘 차례, 수 읽기 등에 의해 보다 정밀한 형세의 계산이 가능하다. 본 연구는 프로기사의 정밀 형세판단과 제안된 group 영역간의 비교로 그 타당성을 고찰한다. <그림 12>에서 기보(Gibo) 8의 프로기사의 형세분석을 보면 백 집은 좌변 40, 우상귀 10, 덤5집 반으로 합계 55집 반이 되고 흑 집은 좌상귀 6, 우상변 15, 우하귀 20집으로 합계 41집으로 백이 14집 우세한 형세이다. 제안된 방법으로 영역분할 된 영역은 백 좌변 27, 우상귀 9으로 36인데 덤 5집반이면 41로 계산되고 흑 영역으로 좌상귀 9, 우하귀 17, 하변 1로 27이 된다. 이 방법에 의해 백이 14집 우세한 형세임을 알 수 있다. 프로기사가 분석한 형세에서 좌변 백 집을 40으로 보고 있는데 이것은 세 칸으로 연결될 수 있는 백의 강한 세력과 흑이 감히 집을 파괴하기 위해 들어갈 수 없음을 계산에 넣었으며, 또한 흑 집으로 우상변 15집으로 계산할 때 흑의 강한 세력을 흑 집으로 넣어 15집으로 계산이 된 것이다. 이러한 세력을 프로기사들은 엄밀히 세력으로 형세 판단에서 집으로 넣어 계산하고 있으나 본 논문에서는 세력에 대한 부분은 제외하고 영역만을 계산하는 것으로 한다. 그리고 <그림 13>은 기보 8의 안정도를 나타낸 것으로 group 6의 안정도는 'KI'로 죽음의 상태를 의미한다. 그리고 group 26은 'AD'로 삶의 상태를 의미하는 것이다.

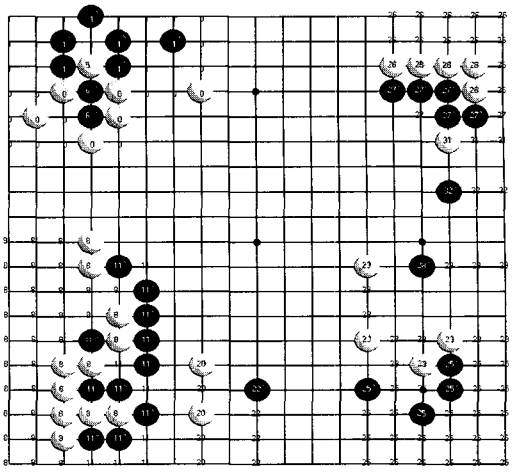

그림 12. 기보 8의 그룹 영역
Fig. 12. Group territory of Gibo 8.

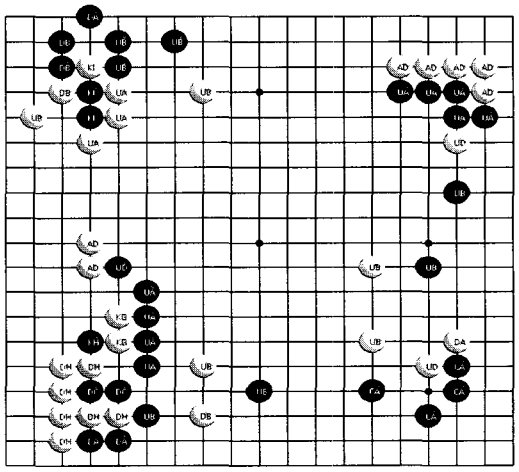

그림 13. 기보 8의 안정도 평가
Fig. 13. Stability evaluation of Gibo 8.

프로기사의 정밀 계산에 세력이 포함되어진 경우를 비교하기 위해, <그림 14>에서와 같은 국면에서는 본 논문에서 제안한 방법으로 group의 영역으로 나누어진다. 프로기사는 흑 집을 계산할 때 좌상귀 18집, 우상귀 9집, 우하변 9집 그리고 하변 20집으로 합계 56집 계산하고 백 집은 좌변 7집, 우하귀 13집 그리고 상변의 세력 전체를 약30집으로 합계 50집으로 계산하여 흑이 6집 우세하고 있다고 형세 분석하였다. 제안한 방법을 사용한 결과로는 좌상귀 14, 우상귀 17, 우하변 9집 그리고 하변 17집으로 흑집 57집이고 좌변 11집, 우하귀 15집, 우상 7집 그리고 좌상 2집으로 백집 35집으로 계

산되어진다. 제안한 방법에는 세력과 선수의 개념을 계산에 포함하고 있지 않다. 이를 프로기사가 사용한 정밀계산에서 세력부분인 30집을 백 집에 더하면 65집이 되어지므로 형세는 8집 백이 우세한 것으로 되어지므로 프로기사의 정밀계산과 14의 오차가 발생되어진다.

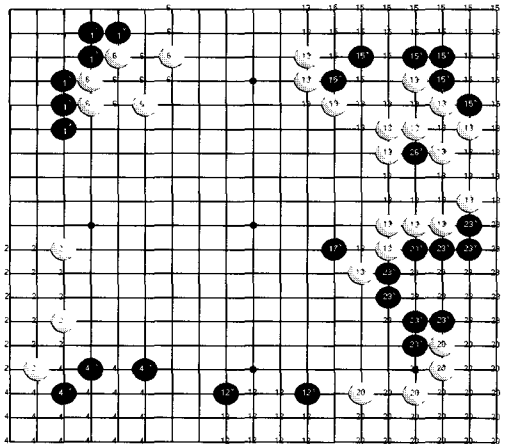

그림 14. 기보 2의 그룹 영역
Fig. 14. Group territory of Gibo 2.

아래의 <표 3>에서는 기보들에 대한 흑영역을 '+'값으로 백영역을 '-'값으로 표현하고 이들 영역의 차를 표시하였으며 정밀계산 세력이란 프로기사의 정밀 계산할 경우의 세력을 의미한다. 조정영역이란 영역차에 정밀세력을 계산하여 조정한 것으로 프로기사의 세력부분을 제거하여 비교하려 한다. 오차는 정밀계산과 조정영역간의 차를 의미한다. 정밀계산법과 조정영역의 비교 결과 평균오차는 8.7이다. 이러한 오차는 본 논문에서 제안한 방법이 정확한 선수개념, 다음 둘 차례, 세밀한 세력 계산 그리고 수 읽기가 포함되지 않았기 때문에 발생되어지는 것이다.

표 3. 정밀계산법과 group 영역과 비교
Table 3. Comparison of a precision evaluation and group territory.

기보 번호	착점 수	흑영역 (+)	백영역 (-)	영역 차	조정 영역	정밀 세력	정밀 계산	오차
1	60	+47	-62	-15	-5	+10	-4	1
2	57	+57	-35	+22	-8	-30	+6	14
3	79	+49	-45	+4	+4	0	+2	2
4	105	+62	-35	+27	+27	0	+17	10
5	81	+37	-32	+5	+5	0	+11	6
6	71	+48	-38	+10	+10	0	+20	10
7	73	+42	-41	+1	+1	0	-7	8
8	63	+27	-41	-14	-14	0	-14	0
9	72	+28	-60	-32	-32	0	-7	25
10	133	+65	-50	+15	+15	0	+4	11

Ⅶ. 결 론

본 논문은 컴퓨터 바둑에서 string의 안정도를 휴리스틱하게 정의하고 group의 영역(territory)을 분할한다. 분할 된 영역(territory)과 프로기사의 형세판단과 비교 분석하였다.

string 안정도의 요소는 눈(eye, E), 유사눈(eye-like, EL), 특수눈(special-eye, SE), 확장가능점(extension-point, EX), 공배수(liberty, L), 그리고 연결가능점(connection-point, CP)으로 정의하였으며, string 안정도(stability)는 휴리스틱한 방법으로 완전한 삶(Complete Alive, C), 삶(Alive, A), 모름(Unsettled, U), 위험(Danger, D), 그리고 죽음(Killed, K)의 5단계로 정의하였다. group은 string과 연결 링크 점에 의해 만들어지고 group 영역(territory)을 획득한다. group의 영역 분할은 string과 string사이에 마름모, 한 칸, 날 일자, 그리고 두 칸의 연결 링크 점과 string의 안정도에 의해 획득되어진다.

string 안정도와 group 영역에 의한 정적 분석과 프로기사의 형세판단과의 차이는 제안된 방법이 프로기사가 사용하는 정밀한 세력과 선수개념 그리고 수 읽기를 하지 않았으므로 평균 오차가 발생하였다.

우리가 제작 중 컴퓨터 바둑 프로그램은 '큰돌'로 1995년 잉창치배 세계 컴퓨터 바둑대회에 참가 하여 8위를 했으며, 1996년 일본 도쿄에서 열린 제 2회 FOST Cup에서는 단지 2승을 거두어 17위를 하였다. 우리는 제안한 방법을 '큰돌'의 형세판단의 루틴으로 구현하였으며 이전 보다 효율적인 수 선택이 이루어질 것으로 판단된다. 앞으로의 과제는 정밀한 세력 계산 함수의 연구와 선수개념의 적용 그리고 큰 자리의 선점과 대응수에 대한 수 읽기 부분이 연구 과제이다. 그리고 이러한 string 안정도와 group 영역은 다음 후보수 결정에도 매우 중요한 근거로 작용이 되어 질 것이다.

참 고 문 헌

[1] Martin Müller, NTT Communication Sicence

Laboratories, "Computer Go", Artificial Intelligence, 134(1-2):145-179, 2002.

[2] J. Tromp, "On game space size", Message on computer-go mailing list, 1999.

[3] David B.Benson, "Life in the Game of Go", Information Sciences, vol. 10(1976), pp 17-29. Reprinted in Computer Games, Levy, D.N.L. (Editor), Vol. II, pp 203-213, Springer Verlag, New York 1988.

[4] H.Landman, "Eyespace values in Go", in : R.Nowakoski(Ed.), Games of No Chance, MSRI Publications 29, 1996, pp. 227-257.

[5] R. Popma, V. Allis, "Life and death refined", in: J.Herik, V.Allis(Eds.),Heuristic Programming in Artificial Intelligence 3, Ellis Horwood, Chichester, 1992, pp. 157-164.

[6] T.Wolf, "Investigating Tsumego problems with RisiKo", in: D. Levy, D.Beal(Eds.), Heuristic Programming in Artificial Intelligence 2, Ellis Horwood, Chichester, 1991, pp. 153-160.

[7] M.Müller. "Playing it safe: Recognizing secure territories in computer Go by using static rules and search." In H. Matsubara, editor, Game Programming Workshop in Japan '97, pp 80-86, Computer Shogi Association, Tokyo, Japan, 1997.

[8] Ken Chen, Zhixing Chen, "Static analysis of life and death in the game of Go", Information Science 121 (1999) 113-134.

[9] A. Kierulf, K. Chen, and J. Nievergelt. "Smart Game Board and Go Explorer : A study in software and knowledge engineering", Communications of the Association for Computing Machinery, 33(2): 152-167, 1990.

[10] M.Müller. "Position evaluation in computer go", ICGA Journal, 2002.12, pp 219-228,

[11] 이두한, 박현수, 이종철, "컴퓨터 바둑에서 형세판단을 위한 객체지향적 프로그램의 설계 및 구현", 정보과학회논문지(C) 제1권 제2호(95.12)

저 자 소 개

朴 鉉 秀(正會員)

1992년 : 경성대학교 전산통계학과 이학사. 1995년 : 경북대학교 컴퓨터공학과 공학석사. 1997년 : 경북대학교 컴퓨터공학과 박사수료. 1997년~현재 : 경동정보대학 컴퓨터정보통신계열 전임강사. 관심분야는 인공지능, 컴퓨터 바둑

李 阯 漢(正會員)

1987년 : 경북대학교 전자공학과 공학사. 1991년 : 경북대학교 전자공학과 공학석사. 1997년 : 경북대학교 컴퓨터공학과 공학박사. 1994년~현재 : 경동정보대학 컴퓨터정보통신계열 조교수. <주관심분야 : 객체지향데이터베이스, 인공지능, 게임이론>

金 恒 俊(正會員)

1977년 : 서울대학교 전자공학과 공학사. 1979년 : 한국과학기술원 전산학과 공학석사. 1997년 : Shizuoka University 공학 박사. 1979년~현재 : 경북대학교 공과대학 컴퓨터공학과 교수. <주관심분야 : Pattern Recognition , Image Understanding.>