**인공지능**

PROJECT REPORT : Four Stone Game Agent



제출일자 : 12월 13일

축산식품생명공학과

200912526 김명준

소프트웨어학과

201411191 김건수

스마트ICT융합공학과

201713074 임예린

1. **목표**

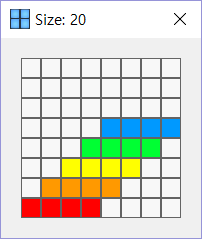
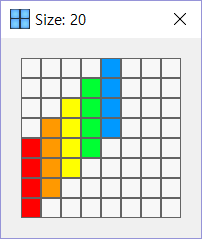
* Four Stone 게임에서 현재 놓인 돌의 배치를 보고 다음에 둘 돌의 위치를 판단하는 에이전트 작성
* 둘 돌의 위치판단은 수읽기를 통한다.
* Min-Max 알고리즘을 적용해 검색할 노드의 수를 줄이고 최대한 깊은 레벨까지 탐색한다.
* 적절한 평가함수를 정의해서 사용한다.
* 평가함수의 값에 휴리스틱 분석을 적용해 적절한 우선순위를 적용한다.

1. **인터페이스**

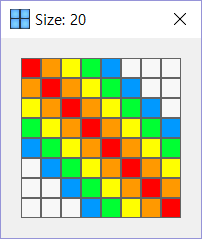
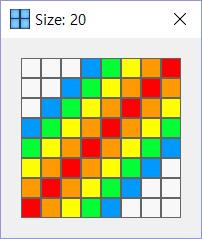
* 돌을 둔 화면을 출력한다.
* re.sgf파일에서 돌의 배열 정보를 읽어 화면에 출력하고, 이를 다시 an.sgf파일에 저장한다.
* 따라서 에이전트는 an.sgf파일을 읽어서 수읽기를 통해 다음에 둘 돌의 위치정보를 추가해 re.sgf에 저장하면 된다.

1. **에이전트**
   1. **접근전략**

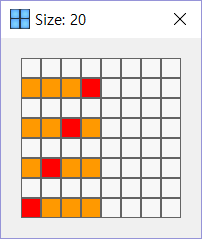
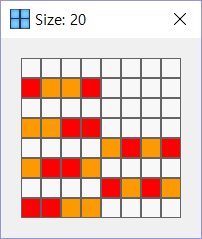
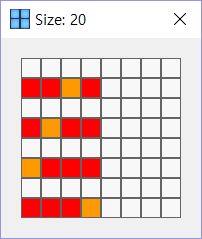
* 돌이 놓여있는 전체 맵의 총 점수를 통해 유불리를 판단한다.
* 게임의 목적은 돌을 4개이상 연결하여 승리하는 것이다.
* 따라서 내가 4개가 될 가능성이 있다면 해당 4개의 칸은 +점수, 상대가 4개가 될 가능성이 있다면 해당 4개의 칸은 -점수를 나타낸다.
* 내 점수와 상대 점수의 총 합이 현재 맵의 전체 점수가 되며, 이 점수가 +라면 내가 이기고 있다는 것이고, 반대로 -라면 내가 지고 있다는 것이다.
* 돌을 두어 이길 수 있는 형태는 가로 4개, 세로 4개, 주대각선 4개(slash), 부대각선 4개(back-slash)이다.
* 8 by 8 크기의 정사각형 형태의 맵에서 플레이하므로, 가로·세로 5개 위치로써 총 40개씩, 대각선 총 25개씩이 가능한 승리위치이다. 또한 4개의 칸 중 내 돌이 많이 채워질수록 승리할 확률이 높아지며, 상대의 돌이 하나라도 포함되면 승리할 확률이 0이 된다.
* 따라서 각 방향의 4개칸을 검사해 상대 돌이 없을 경우, 채워진 내 돌의 개수에 따라 점수를 부여해 평가할 수 있다.

**Figure 1. 가로 가능성 Figure 2. 세로 가능성**

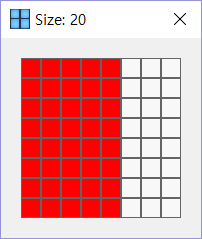
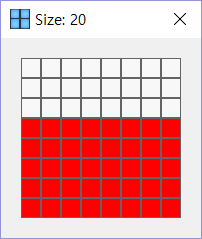
 

**Figure 3. 주대각선 가능성 Figure 4. 부대각선 가능성**

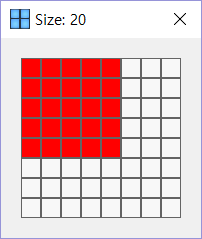
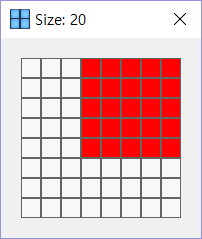
  

**Figure 5. 돌1개 가능성 Figure 6. 돌2개 가능성 Figure 7. 돌3개 가능성**

* 1. **구조**
     1. **main.cpp**
* CSetNode.h를 포함한다.
* 메인 함수이다. C++로 작성되어 있다.
* 크게 an.sgf파일에서 현재 배열을 읽어오는 부분, 수읽기 후 배열에 추가하는 부분, 추가된 배열을 re.sgf파일에 쓰는 부분으로 구성되어 있다.
* 이 중, 수읽기 부분에 CSetNode 클래스를 이용한다.
  + 1. **CMyNode.h / CMyNode.cpp**
* 맵을 좌표를 참조해 실제 배열을 생성하는데 사용된다.
* 플레이어 흑백여부, 놓아야 할 위치정보, 점수, 자식노드와 같은 정보를 기록할 수 있다.
  + 1. **CSetNode.h / CSetNode.cpp**
* CSetNode.h에서 CMyNode.h를 포함한다.
* 맵의 배열정보를 넘겨받아 점수를 계산하고 수읽기를 수행하는 함수들로 구성되어 있다.
  1. **기능**
     1. **main()**
* FILE구조체는 an.sgf파일에서 정보를 읽어 char형의 임시 버퍼에 저장한다.
* 구분자를 제외하고 B나 W에 따라 맵을 구성하는 각 노드의 상태를 결정하고, 실제 x, y좌표를 통해 해당하는 2차원 int형 배열 map의 원소에 상태를 부여한다.
* 생성된 map정보를 CSetNode클래스의 evaluate() 함수에 넘긴다.
* evaluate() 함수가 반환한 가로 위치를 토대로 해당 세로줄 내 놓을 수 있는 위치를 bottom to up방식으로 찾는다.
* 찾은 위치를 기존 위치와 함께 re.sgf파일에 기록한다.
  + 1. **CMyNode 클래스**
* 생성자와 자식노드를 생성하는 GenChild() 함수를 갖는다.
* 생성자는 플레이어 여부와 맵 정보를 넘겨받아 탐색할 노드 하나를 생성한다.
* GenChild() 함수는 현재 맵에서 각 세로줄에 놓을 수 있는 각각의 후보를 찾는다. 만약 빈 공간이 남아있어 놓을 수 있다면, 상대가 각 세로줄에 놓았을 경우를 추가해서 후보노드를 생성하고, 이를 vector에 저장해 반환한다.
  + 1. **CSetNode 클래스**
* evaluate() 함수는 맵의 좌표를 넘겨받아 해당하는 노드를 생성하고, 자식노드를 생성해 각 경우의 점수를 비교·판단하여 최종적으로 놓아야 할 가로위치를 반환한다.
  + 만약 다음에 놓아서 내가 이기는 경우가 있다면, 해당 후보의 위치를 반환한다.
  + 만약 다음에 놓지 않고 상대가 놓으면 이기는 경우가 있다면, 해당 후보의 위치를 반환한다.
  + 위 두 경우에 해당하지 않으면 Min-Max 알고리즘을 수행한다.
  + Minimax() 함수는 주어진 depth만큼 반복적으로 ABPruning() 함수를 호출하며 각 후보노드에 점수를 부여한다. 몇 번의 테스트 결과, depth == 6의 값을 사용했다.
  + 최종적으로 지금 당장 놓을 수 있는 후보노드 간의 점수를 비교하여 가장 점수가 높은 위치를 반환한다.
  + 만약 점수가 같다면, 상대방을 막는 경우를 우선시한다. 이는 돌이 3개일 경우, 상대방과 나의 구분없이 같은 점수를 부여했기 때문이며, 이 부분에서 우선순위를 처리한다.
* Minimax() 함수는 알파값과 베타값을 각각 INT\_MIN, INT\_MAX로 초기화하며, 처음 ABPruning() 함수를 호출한다.
* ABPruning() 함수는 실질적인 Min-Max 알고리즘을 수행하는 부분이며, 각 후보노드에 있어 자식노드를 생성하고 알파값 및 베타값과 비교하며 각 노드의 점수를 부여한다.
* CopyMap() 함수는 출력인자로서 copyMap을 받아, 함께 넘겨받은 map으로부터 맵의 위치정보를 복사한다. 이는 맵에 후보노드를 추가해 점수를 계산하는 방식으로 인해 원본 맵의 정보가 훼손됨을 방지하기 위해 복사 맵을 이용하기 위해 사용된다.
* PutInCol() 함수는 주어진 가로위치의 놓을 수 있는 가장 아래 위치에 돌을 놓는 함수이다.
* CanPutInCol() 함수는 주어진 가로위치에 돌을 놓을 수 있는지 여부를 판단해주는 함수이다.
* getScore() 함수는 현재 맵의 점수를 판단하는 함수이다.
  + 가로 총 점수 + 세로 총 점수 + 주대각선 총 점수 + 부대각선 총 점수의 전체 합산점수를 반환한다.
  + 각 방향의 점수계산은 분리된 각 함수에 의해 수행된다.
  + 각 방향의 점수계산 중간 결과가 최대값(내가 이기는 경우)이나 최소값(상대가 이기는 경우)이라면, 계산을 중지하고 해당 값을 반환한다.
* check(\*)Score() 함수는 \*로 나타낸 각 방향에 대해 합산점수를 계산한다.
  + 각 경우는 접근전략 항목의 Figure1~4에서 볼 수 있듯이 이길 수 있는 위치에 대한 점수를 계산한다.
  + 각 경우는 적당한 모서리를 기준으로 4개칸의 시작점부터 해당하는 방향으로 연속되는 3개칸을 검사한다.

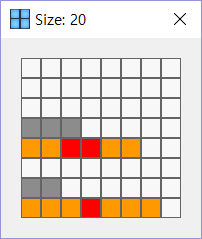
 

**Figure 8. 가로 검사영역 Figure 9. 세로 검사영역**

**Figure 10. 주대각선 검사영역 Figure 11. 부대각선 검사영역**

* + 각 검사영역 내 검사대상은, 접근전략 항목의 Figure 5~7에서 볼 수 있는 돌이 1개, 2개, 3개가 될 수 있는 경우들이다.
  + 이 경우 들에서 내가 이길 수 있는 경우는 +점수, 상대가 이길 수 있는 경우는 -점수를 부여했다.
  + 또한 나와 상대가 섞여 있는 경우는 상호간 block으로 처리하고 계산에서 제외했다.
  1. **실행흐름**
* an.sgf에서 현재 놓인 돌의 위치정보를 획득하여, 이 정보를 기반으로 놓을 수 있는 후보의 각 위치에 대해 노드를 생성하고, 각 노드의 자식노드를 생성해 depth가 0이 되는 지점에서 해당 맵에 대한 평가를 실시한다.
* 맵에 대한 평가는, 상기 Figure 8~11에서 볼 수 있는 검사 영역을 돌며 상대나 내가 이길 수 있는 가능성을 모두 세어 점수화 한 것이며, 이 점수를 토대로 Min-Max 알고리즘을 수행해 가장 좋은 위치를 판단한다.
* 판단한 위치를 현재 놓인 돌의 위치정보와 취합해 이를 re.sgf파일로 쓴다.
  1. **평가함수**
* 이길 확률이 높은 수를 놓는 것이 중요하므로, 이길 가능성의 개수를 이길 확률로서 판단하고 이를 지표로 삼았다.
* 빈 공간을 0, 내 돌을 1, 상대 돌을 2라 하면, 아무것도 놓이지 않은 빈 공간인 0000의 경우를 제외하고 돌이 하나라도 놓인 상태를 가장 낮은 수준의 이길 가능성으로 판단하고 여기에 100점을 부여했다.
* 돌2개와 돌3개의 경우, 각각 돌1개와 돌2개에 비해 높은 점수가 필요하다. 실제 놓인 돌의 수가 보다 적은 수인 경우보다 보다 많은 수인 경우가 이길 확률 자체가 높기 때문이다. 따라서 각각에 대해 단독 점수의 정의가 필요하다.



**Figure 12. 각 돌의 점수계산**

* 돌1개의 경우는 어느 방향으로도 확장이 가능하지만, 2개 이상이 되면 방향이 고정된다. 따라서 한 방향으로만 고려하여, 보다 낮은 돌의 획득가능 최대점수를 기준으로 유불리에 따라 세부적으로 조정했다.
* 돌은 총 1, 2, 3, 4개가 놓일 수 있으며, 이 중 4개의 경우 승리 혹은 패배로써 게임이 종료되므로 고려대상에서 제외한다.
* 현재 놓인 돌의 점수가 아닌, 놓일 가능성이 있는 위치의 점수이므로 1개보단 2개가 유리하며, 2개보단 3개가 불리하다. 전자의 경우 이길 확률 자체가 확정적으로 상승하기 때문이며, 후자의 경우 Min-Max 알고리즘에 의해 3개가 놓이면 상대 입장에선 반드시 막아야 하기 때문에 선택될 확률이 낮아지기 때문이다. 이는 3개짜리 형태를 소수 만드는 것보다 2개짜리 형태를 다수 만들어 상대가 방어할 수 없는 외통수를 만드는 것이 승리에 중요한 점인 것을 반영한 것이기도 하다.
* 돌1개가 가질 수 있는 최대 점수는 400점이다. 따라서 돌2개의 단독점수는 500점으로 정했다.
* 돌2개가 가질 수 있는 최대 점수는 1500점이다. 따라서 돌3개의 단독점수는 1400점으로 정했다.