韓國IT서비스學會誌
第16卷 第4號
2017年 12月, pp.137-148

Journal of Information Technology Services
https://doi.org/10.9716/KITS.2017.16.4.137

기계학습 알고리즘 기반의 인공지능 장기 게임 개발

장명규* · 김영호* · 민동엽* · 박기현* · 이승수* · 우종우**

Development of Artificial Intelligence Janggi Game based on Machine Learning Algorithm

Myeonggyu Jang* · Youngho Kim* · Dongyeop Min*
Kihyeon Park* · Seungsoo Lee* · Chongwoo Woo**

■ Abstract ■

Researches on the Artificial Intelligence has been explosively activated in various fields since the advent of AlphaGo. Particularly, researchers on the application of multi-layer neural network such as deep learning, and various machine learning algorithms are being focused actively. In this paper, we described a development of an artificial intelligence Janggi game based on reinforcement learning algorithm and MCTS (Monte Carlo Tree Search) algorithm with accumulated game data. The previous artificial intelligence games are mostly developed based on mini-max algorithm, which depends only on the results of the tree search algorithms. They cannot use of the real data from the games experts, nor cannot enhance the performance by learning. In this paper, we suggest our approach to overcome those limitations as follows. First, we collects Janggi expert's game data, which can reflect abundant real game results. Second, we create a graph structure by using the game data, which can remove redundant movement. And third, we apply the reinforcement learning algorithm and MCTS algorithm to select the best next move. In addition, the learned graph is stored by object serialization method to provide continuity of the game. The experiment of this study is done with two different types as follows. First, our system is confronted with other AI based system that is currently being served on the internet. Second, our system confronted with some Janggi experts who have winning records of more than 50%. Experimental results show that the rate of our system is significantly higher.

Keyword : Janggi Game, Reinforcement Learning, MCTS(Monte Carlo Tree Search) Algorithm, Artificial Intelligence Software, Machine Learning

Submitted : July 19, 2017 1st Revision : October 22, 2017 Accepted : October 28, 2017

* 국민대학교 컴퓨터공학부
** 국민대학교 컴퓨터공학부 교수, 교신저자

1. 서 론

최근 알파고(AlphaGo)의 등장으로 인공지능의 연구가 활성화되고 있으며 그중에서도 기계학습 분야의 연구가 가장 활발하게 이루어지고 있다(Park, 2016; Kim et al., 2016). 그러나 실제 많은 사람이 사용하는 게임 소프트웨어에서는 아직 인공지능의 개념 또는 기계학습 알고리즘이 적용되어 개발된 게임들을 찾기가 쉽지 않거나 초보적인 수준에 그치고 있다(Mandziuk, 2010; Millington and Funge, 2009). 특히 기계학습 알고리즘이 적용된 사례를 찾기 쉽지 않은 이유는 학습을 위해 충분히 정형화된 데이터가 확보되어 있지 않고, 학습 데이터가 충분하더라도 알고리즘이 훈련된 데이터에 의존적이기 때문에, 신규 데이터의 경우 처리 과정에 오류가 발생할 가능성이 존재한다. 또한, 알파고에서 적용된 딥 러닝(Deep Learning) 방식을 게임에 적용할 경우에는 추가적인 하드웨어 없이 알고리즘만으로는 시스템의 성능과 학습 시간을 극복하는 데 한계를 가지고 있다.

본 연구에서는 이러한 문제점들을 극복하기 위하여, 학습 데이터가 비교적 충분히 존재하고, 수많은 실전 대국을 통하여 해당 데이터들의 신뢰도가 어느 정도 확보된 장기 기보들을 수집하여 강화학습과 Monte Carlo Tree Search(MCTS) 알고리즘(Browne et al., 2012)을 적용하는 인공지능 장기 게임을 개발하고자 한다. 장기와 같은 보드게임은 인공지능 분야에서 알고리즘의 성능을 가시적으로 확인할 수 있고 게임 종류에 따라 다양한 복잡도를 가지고 있어 알고리즘 수행 시간과 복잡도의 관계를 확인하기 위한 테스트를 진행할 수 있어 연구 목적으로 많이 사용되어 왔다(Kulsinskas et al., 2015; Spencer and Oommen, 2015; Robilliard and Fonlupt, 2015). 또한, 장기에 관한 연구는 국내에서 보다, 중국에서 비교적 활발히 이루어지고 있는데, 주로 MIN/MAX 알고리즘과 MCTS를 변형하거나 DB화하여 경로를 미리 찾아 탐색 시간을 단축하는 방식을 사용하였다(Fan et al., 2010; Wu and Tao, 2016). 학습의 방식도 일부 사용하였으나 대부분 알고리즘을 이용한 학습이 아니라 개발된 AI끼리 지속적인 대국을 통해 패배한 경로를 다시 가지 않도록 구현하는 방식(Ong et al., 2007)을 사용하였다. 이러한 연구는 정교한 알고리즘의 성능으로 가시적인 성과를 보이고 있으나, 기존의 트리탐색의 범위를 벗어나지 못하는 제한점이 있다. 또한 알고리즘의 범위 내에서만 처리되기 때문에 시스템의 성능을 스스로 향상시키지 못하는 제한점이 있다.

본 연구에서는 이러한 기존 연구들의 장점과 또한 제한점을 분석 하여, 보다 우수한 승률을 얻고자 다음과 같이 접근하고자 한다. 첫째, 단순히 트리탐색 알고리즘에 의존적이 아닌 장기 우승자들의 누적된 기보를 활용한 학습방식을 적용함으로 더 풍부한 실전 결과를 반영할 수 있게 한다. 둘째, 기존 연구들에서 트리구조를 사용한 것과는 달리, 장기의 특성상 같은 상태를 중복시키지 않기 위하여 기보를 그래프 구조로 생성하여 사용하고자 한다. 셋째, 대전을 반복할수록 보다 우수한 승률을 얻기 위하여 보드 상황에 따라 강화 학습알고리즘을 수행하며, 학습된 그래프 상에서 MCTS 알고리즘을 응용하여 합리적인 시간 내에 가장 좋은 다음 수를 선택하게 된다.

본 연구의 실험은 현재 온라인 또는 모바일상에서 서비스 되고 있는 장기 인공지능과 본 연구의 시스템이 대국을 하거나, 또는 온라인상의 장기고수들과 본 연구의 시스템이 대국을 하는 방식으로 알파고의 진행과 유사하게 대국을 하게 된다. 시스템의 성능평가는 이러한 방식으로 대국을 진행한 결과를 통하여 누적된 승률을 산출하여 평가할 수 있다. 또한, 시스템은 대국 시마다 새롭게 그래프를 생성할 필요 없이 객체 직렬화 방식으로 대국의 결과를 저장하여, 지속적으로 대국을 할 수 있게 한다.

2. 관련 연구

2.1 국내외 게임 인공지능 동향

초기의 게임 인공지능은 1950년대 체스 등 보드

게임에서 인간과의 대결을 위한 수준에서 시작되었으며, 1980년대에는 팩맨, 갤러그 등 비디오게임이 등장하면서 사용자의 캐릭터와 대결하기 위한 NPC(Non-Player Character)의 움직임을 구현하기 위해 사용하였다. 1990년대 이후에는 비디오 콘솔 게임, 실시간 전략 시뮬레이션(Real Time Strategy, RTS), 슈팅 게임 등 다양한 게임 분야가 등장하면서 게임 내부 인공지능의 수준이 향상되었고 RPG(Role-Playing Game) 게임에서 사용자에게 미션을 주거나 대화를 하기 위한 NPC까지 등장하였으나 기존 Rule-based System으로 구현된 FSM(Finite State Machine) 수준에 지나지 않았다(Park and Kim, 2013; Cho and Park, 2008).

그러나 최근 게임에 학습 알고리즘이 적용된 사례들이 나타나고 있다. 예를 들어 PVP(Player vs Player) 시스템에서는 대국자의 선호도, 승률, 성향 등을 고려하기도 하고, NC소프트의 '블레이드 & 소울'의 경우 사용자들의 움직임을 학습하여 몬스터들의 움직임에 적용하여 실제 사람이 게임을 하는 듯한 느낌을 받게 하였다(Bang et al., 2016). 또한, 넥슨의 경우 '절차적 콘텐츠 생성'이라는 요소를 개발하여 실시간으로 생성되는 환경에서 게임이 진행되도록 서비스를 제공하고 있다. 인간이 절대 우세라고 예측되던 바둑에서, 구글 딥마인드(Google DeepMind)의 알파고가 완벽하게 승리하면서 학습 알고리즘에 더욱 관심을 가지게 되었다(Siver et al., 2016). 본 연구에서는 우선 게임의 기본이 되는 보드게임의 학습 알고리즘 개발을 통해 성능을 테스트 뒤, 해당 알고리즘을 다른 게임에 적용할 방법을 발전적으로 연구하고자 한다.

2.2 강화학습

강화학습은 현재 상태(State)에서 어떤 행동(Action)을 취하는 것이 최적인지 학습하는 방식이다. 행동 결과에 따라 보상(Reward) 값을 줌으로써 옳은 행동일 경우 플러스 점수를 얻게 되고 잘못된 행동일 경우 마이너스 점수를 주는 방식으로 구현한다. 일반적으로 유한한 상태일 경우 MDP(Markov Decision Process)로 표현이 가능하며 이미 존재하는 상태의 데이터와 존재하지 않는 상태와의 균형을 잡는 데 적합하다. 보드게임과 같이 종료 상태가 존재하고 보상을 주는 조건을 명확하게 설정할 수 있는 경우에 많이 사용되고 알파고의 학습 방식에도 사용되었으며 최근에는 자동차의 자율주행 학습 방식에도 적용되고 있다(Ingram and Mcgregor, 2014).

본 연구에서는 기보를 이용한 학습 과정에서 다음 식 (1)과 같이 MDP State Value Function 학습 알고리즘을 장기에 맞게 적용하여 각 상태의 가치를 조절하고자 한다(Feinberg and Shwartz, 2002). 수식에서 R은 해당 State의 기댓값이며 r_0, $r_1, \cdots, r_n$은 현재 State부터 다음에 나오는 Step 별 각 State의 가치를 의미한다. γ은 discount factor 값으로 Step에 따라 변화하는 보상, 패널티의 값을 조정하기 위한 값이다. r의 범위는 장기의 특성상 번갈아 가며 차례가 나타나기 때문에 r_0, r_2, …, r_{2n}(초의 차례)와 r_1, r_3, …, r_{2n-1}(한의 차례)(n은 자연수)와 같이 1씩 건너뛰며 적용한다.

$$R = r_0 + \gamma r_1 + \cdots + \gamma^n r_n = \sum_{t=0}^{n} \gamma^t r_t \qquad (1)$$

2.3 MCTS 알고리즘

MCTS 알고리즘은 기존의 MIN MAX 알고리즘의 한계를 극복하기 위한 접근 방식이다. MIN MAX 알고리즘은 각 상태에서 나올 수 있는 모든 다음 상태를 계속 이어나가며 지정된 깊이까지 모든 상태를 만든 뒤 계층별로 가장 가치가 높은 값/가장 가치가 낮은 값을 번갈아 가며 선택하여 현재 상태에게 가장 유리한 수를 찾는 방식이다(Rivest, 1987). 그러나 MIN MAX 알고리즘은 모든 경우의 수를 판단해야 하기 때문에 깊이를 깊게 보거나 바둑과 같이 경우의 수가 커질 경우 실행 시간이 길어지기 때문에 성능 향상을 시키는데 한계를 가지고 있었다. MCTS 알고리즘은 기본적으로 다음의 4가지 단계들을 반복하게 되는데, 우선 특정

다음 상태들을 무작위로 일부 선택(Selection)하여 해당 상태들의 하위 상태들을 확장(Expansion)하고 확장된 경로를 시뮬레이션(Simulation) 한 뒤 결과에 대해 역전파(Backpropagation)를 통해 다음 상태 중 가장 좋은 수를 선택하는 방식을 사용함으로써 MIN MAX 알고리즘의 한계를 극복하였다.

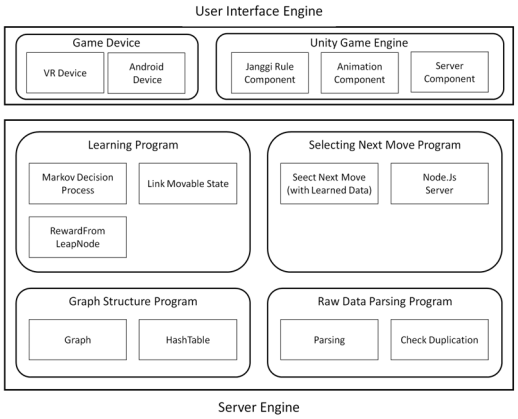
본 연구에서는 학습 과정과 수를 선택하는 일련의 과정에서 MCTS의 각 과정을 나누어 필요한 부분에 적용함으로써 각 상태의 가치를 조절하고 수를 선택할 수 있도록 적용한다.

3. 시스템 설계

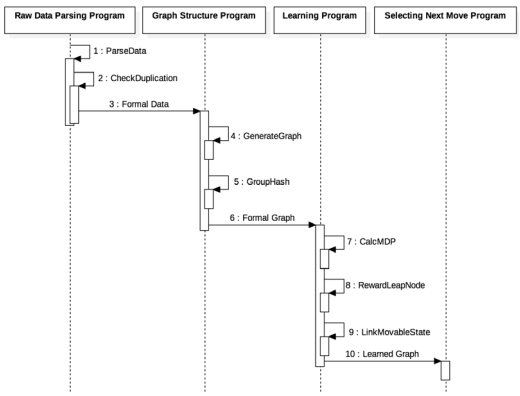
본 연구의 시스템은 <Figure 1>과 같이 크게 서버 엔진, 사용자 인터페이스 엔진으로 구성된다. 서버 엔진은 다수의 세부 모듈로 구성되며 시스템의 주 알고리즘을 수행하게 된다. 인터페이스 엔진은 게임의 결과를 제어하고 2D 또는 3D 환경에서 게임을 진행할 수 있도록 VR기기와 Unity 게임 엔진을 포함한다. 시스템의 전체적인 흐름은 <Figure 2>와 같이 진행되며 완성된 학습 그래프와 사용자 인터페이스 엔진 간의 통신 방식으로 시스템이 작동한다. 각 모듈의 세부기능은 다음과 같다.

3.1 서버 엔진

서버 엔진은 크게 기보 분할 프로그램, 그래프 구조 생성 프로그램, 학습 프로그램, 다음 수 선택 프로그램 4개로 구분된다. 기보 분할 프로그램은 기보 텍스트 파일을 입력받아 1개의 파일을 1개의 경기로 분할하며, 그래프 구조 생성 프로그램은 기보 파일을 이용해 그래프 구조를 생성한다. 학습 프로그램은 기보들의 통계, 움직임 등을 이용해 학습 알고리즘을 수행하게 되며, 다음 수 선택 프로그램은 학습에 기반을 두어 먼저 수를 선택하고 학습된 경우가 없을 때는 MIN MAX 알고리즘을 수행한다.



〈Figure 1〉 System Architecture



〈Figure 2〉 System flow Diagram

3.1.1 기보 분할 프로그램

기보 분할 모듈에서는 학습을 위해 수집된 기보 파일을 학습을 위해 필요한 데이터만 남기고 각각의 파일로 우선 분할한다. 만약 기보 내에 해설이 있거나 형식에 오류가 발생한 경우도 처리할 수 있도록 한다.

기보는 <Figure 3>과 같이 장기 협회에서 지정된 정식 규격을 사용하여 상차림(마와 상의 위치), 승자, 히스토리가 기록되어 있다. 예를 들어 '79졸78마장군'과 같은 기록은 (7, 9) 위치에 있는 졸이 (7, 8) 위치에 있는 마를 잡은 후 장군 이벤트를 발생시킨 상황이다. 본 연구에서 개발된 프로그램을 수행한 뒤에는 '0 79 78 j h 1'로 표기되며 공격 순서, 전 위치, 후 위치, 움직인 말의 종류,

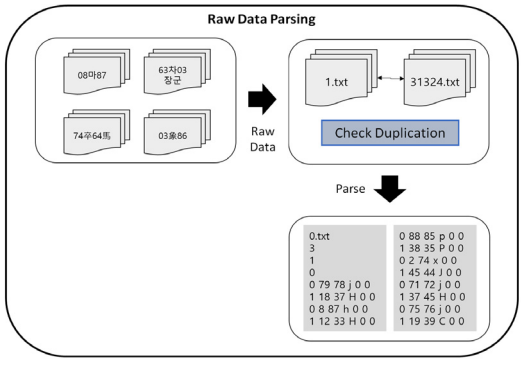
잡힌 말의 종류, 장군 여부 순으로 표기하여 개발에 사용하기 편리하게 변형한다.

모든 기보가 오류 없이 분할되면 1.txt부터 시작하여 숫자를 증가시켜 파일로 저장한다. 또한, 학습 과정에서 중복된 기보가 발생할 경우 학습을 위한 통계에 문제가 발생할 것으로 판단되어 동일한 기보가 존재할 경우 삭제 기능을 구현한다. 전체적인 기보분할의 처리 과정은 <Figure 4>와 같다.

```
[초차림 "마상마상"]
[한차림 "마상마상"]
[대국결과 "초 기권승"]

1. 79졸78 2. 41병42 3. 02마83 4. 17마36 5. 82포85
6. 11차21 7. 85포81 8. 21차24 9. 71졸72 10. 12마33
11. 81포85 12. 32포35 13. 07마86 14. 18상46 15. 08상76
16. 13상41 17. 01차61 18. 24차54 19. 06사96 20. 25장15
21. 61차67 22. 47병57 23. 67차68 24. 14사25 25. 85포05
26. 49병59 27. 95장06 28. 59병69 29. 68차63 30. 15장14
```

〈Figure 3〉 Example of Janggi Game Data

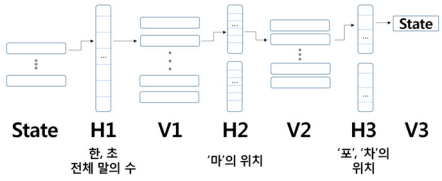


〈Figure 4〉 Data Parsing Process

3.1.2 그래프 구조 생성 및 해쉬 알고리즘

그래프는 인공지능 게임 트리 이론을 반영하며, 그래프로 모델링 하는 이유는 바둑과 달리 장기/체스의 경우 게임 상태에 순환이 발생할 수 있으므로 트리가 아닌 그래프를 사용한다. 그래프의 각 노드는 중복 검사와 접근을 위한 키, 공격자, 부모 노드, 자식 노드, 말의 개수, 각 말의 가치, 통계치 등의 정보를 포함한다. 부모 노드는 이전 상태에서 어떤 말을 움직여서 현재 상태로 올 수 있는 모든 노드를 뜻하며, 자식 노드는 현재 상태에서 어떤 말을 움직여서 나올 수 있는 모든 노드를 의미한다.

그래프의 중복 검사는 수백만 개에 대해 진행되어야 하므로 선형적으로 찾기에는 제한이 있다. 따라서 본 연구에서는 해쉬(Hash) 알고리즘을 적용함으로써 속도를 향상시킬 수 있도록 한다. 본 연구의 해쉬 알고리즘은 <Figure 5>와 같이 3번의 단계를 거친다. 첫째, 해쉬 테이블(Hash Table) H1에서 초/한의 남아있는 전체 말의 개수를 통해 벡터 V1을 선택한다. 둘째, 해쉬 테이블 H2에서 초/한의 '마'의 위치를 이용해 벡터 V2를 선택한다. 셋째, 해쉬 테이블 H3에서 초/한의 '포'와 '차'의 x, y 위치의 합을 이용해 마지막으로 벡터 V3 노드를 찾는다. 만약 해당 노드가 존재하지 않을 경우에는 노드를 새로 만들고, 연결해주어 그래프 내부에 포함시킨다. 전체 알고리즘은 <Figure 6>과 같다.



〈Figure 5〉 Hash Algorithm Process

procedure *stateHashGrouping(current state S)*
Hash H1, H2, H3
Vector V1, V2, V3
if *H1 contains key(CHO p`ieces of S, HAN pieces of S)* **then**
H1 generates V1
if *H2 contains key(V1, MA's position of CHO, MA's position of HAN)* **then**
H2 generates V2
if *H3 contains key(V2, x position's sum of PO, y position's sum of PO, x position's sum of CHA, y position's sum of CHA)* **then**
H3 generates V3
if *S exists in V3* **then**
use of update state in graph

〈Figure 6〉 Hash Algorithm

3.1.3 학습 프로그램

학습 프로그램은 생성된 그래프와 기보 파일을 이용하며, 학습은 다음 2번의 과정을 통해 이루어진다. 첫째, 각 노드에서 움직임에 따라 가치를 조정하고, 둘째, 전체 게임에서 승/패에 따라 강화학습 방식을 적용한다. 학습 적용 전 각 말 가치의

초기 값은 <Table 1>과 같이 장기 협회에서 지정한 기물 점수를 적용한다.

〈Table 1〉 Points for Janggi objects

Object	chariot	cannon	horse	elephant	guard	soldier
Score	13	7	5	3	3	2

첫 번째 학습 방식은 <Figure 7>의 알고리즘과 같이 만약 상대방 말(horse)을 잡은 경우에는 한 단계 전 움직인 말에 대해 +0.01%, 두 단계 전 움직인 말에 대해 -0.01%를 적용한다. 또한, 장군(king)이 나온 경우에는 한 단계 전 움직인 말에 대해 +0.005%, 두 단계 전 움직인 말에 대해 -0.01%를 적용하며, 단순히 움직인 경우에는 +0.0001%를 적용한다. 한 수 쉼이 나온 경우에는 해당 노드에 접근하기 전 상대방이 좋은 수를 두었다고 판단하여 움직인 말에 대해 +0.001%를 적용한다.

두 번째 학습 방식은 <Figure 8>과 같이 가장 하위 노드에서 승/패를 판단한 뒤 0.5의 Discount 값을 이용하여 강화학습 중 MDP State Value Function을 이용한다. 승자일 경우에는 현재 노드 점수를 현재 노드 점수+(하위 노드 점수×0.5)의 점수를 보상 적용하고, 패자일 경우에는 현재 노드 점수를 현재 노드 점수-(하위 노드 점수×0.5)의 점수를 페널티 적용한다. 적용에 있어 -2를 한 이유는 그래프에서 초와 한이 번갈아 차례가 나오기 때문이다.

```
procedure valueEvaluation(current state)
    if piece is moved then
        increase weight for piece
    if take an opponent piece then
        increase weight for piece
        decrease weight for the oppenent piece
    if make the check state for the opponent king then
        increase weight for piece
        decrease weight for the oppenent piece
    if oppenent take rest then
        increase wegiht piece
end evaluate current state's value
```

〈Figure 7〉 Value Evaluation Algorithm

```
procedure rewardFromLeafNode(one game record Sn)
    for i = N to 1 do (N is leaf node)
        if Sn is winner then
            Si-2 ← Si-2 + (Si × discount factor)
        else Sn is loser then
            Si-2 ← Si-2 - (Si × discount factor)
    end for
end reward function
```

〈Figure 8〉 Reward Computation Algorithm

3.1.4 다음 수 선택 프로그램

다음 수 선택 프로그램은 학습된 그래프 결과를 바탕으로 실제 상황이 주어졌을 경우 가장 유리한 수를 판단하는 기능을 수행한다. 즉, 현재 보드의 상황과 공격자 정보 2가지 정보를 입력받으면 현재 상황과 정확히 일치하는 학습 결과가 있는지 그래프를 탐색하여 다음과 같이 진행한다.

- 만약 현재 보드의 상황과 그래프의 학습결과가 일치한다면, 학습된 결과 중 가장 유리한 수를 출력한다.
- 만약 일치하는 노드가 존재하더라도 하위 노드들의 점수의 최댓값이 음수가 나올 경우 해당 노드를 따라갈 경우 좋지 않은 결과가 나온다고 판단하여 따라가지 않도록 설정한다.
- 만약 일치하는 하위노드가 없지만, 전체 그래프 중에 접근할 수 있는 노드가 있다면 해당 노드들 중 가장 유리한 수를 선택한다.
- 만약 일치하는 노드가 없을 경우 MIN/MAX 알고리즘을 사용하며 알고리즘의 깊이는 말의 개수에 따라 다르게 적용하여 좋은 수를 찾아낼 수 있도록 한다. 트리의 깊이(depth)는 다수의 테스트를 통해 30초 이내로 다음 수를 출력할 수 있도록 깊이를 조절한다.

3.2 사용자 인터페이스 엔진

사용자 인터페이스는 유니티 엔진 위에서 구현되며, 서버에서 처리된 결과를 표현한다. 플랫폼은 PC와 Android VR 두 가지로 구현하고 VR을 이용할 경우 Oculus GazePointer 라이브러리를 이용하여 시선 중앙에 커서가 위치할 수 있도록 한다.

유니티에서 제공하는 라이브러리들은 다음의 기능들을 제공하는데, 예를 들면, 장기 말을 이동시키거나 공격 등의 애니메이션을 제어하기 위해 DOTween 라이브러리를 사용한다. 또한, Object를 이동시키는 DOMove, 회전시키는 DORotate, 입력된 방향을 바라보게 하는 DOLookAt 등의 함수를 사용한다. 공격 시에는 2D로 전체 장기판을 관장하는 메인 카메라 이외에 별도의 보조 카메라를 이용하여 장기 말로 착점을 변환하여 공격 애니메이션을 볼 수 있도록 구현한다.

장기 규칙은 사용자가 이동할 말을 선택하였을 때 현재 이동이 가능한 위치를 표시하기 위해 구현하며 서버는 현재 유니티 엔진에서 유지되고 있는 상황과 서버 엔진의 Node.js 서버와 상황을 유지하기 위해 소켓 프로그래밍을 이용하여 구현한다.

4. 시스템 구현

본 연구에서는 장기 게임을 구현하여 학습 알고리즘의 성능을 확인하였고 실험 방식은 현재 서비스 되고 있는 다른 인공지능 및 온라인 장기 사용자들과 대국을 통해 진행하였다.

4.1 시스템 개발 환경

본 연구는 다음 <Table 2>와 같은 환경에서 개발하였다. 개발 언어는 전반적으로 C++를 사용하였으며, 기보의 중복을 검사하는 프로그램은 Python, Unity 엔진에서는 C#, Node.js 서버는 Javascript 언어를 사용하였다.

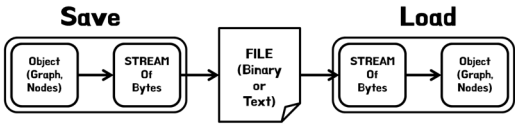
〈Table 2〉 Development Environment

Subject	Content
Operating System	Ubuntu 16.04 LTS
CPU	Intel Core i7 4770 3.4GHz(Quad-Core)
RAM	32GB DDR3 SRAM
Development Language	C++/Python C#/JavaScript
Library	Boost Library Oculus, DOTween
Stack Size	1GB
Virtual Memory Size	48GB

4.2 세부 구현 내용

4.2.1 객체 직렬화

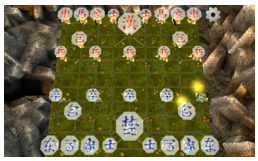
본 연구에서는 생성된 그래프와 학습이 완료된 그래프의 현재 상태를 유지하여 다음 대국에 활용하기 위하여, 전체 그래프를 객체 직렬화(Object Serialization) 방식을 사용하여 저장하였다. 객체 직렬화는 <Figure 9>와 같이 메모리에 있는 정보들을 연속적인 바이트로 변환하고, 변환된 바이트를 다시 원래의 객체로 복원하는 방법을 말한다. 본 연구에서는 이렇게 생성된 구조를 바탕으로 학습 정책과 알고리즘을 테스트하였다.



〈Figure 9〉 Object Serialization Processing

4.2.2 유니티 인터페이스

본 연구의 결과를 사용자 입장의 인터페이스 시스템을 구축하기 위해 유니티 엔진을 이용하여 구현하였다. 처음 게임이 시작되면 상차림을 선택할 수 있고, 움직일 수 있는 말을 선택하였을 때는 <Figure 10>과 같이 2D 상태로 말을 움직일 수 있는 위치를 표시해 준다. <Figure 11>은 말을 이동시켰을 때 3D 애니메이션으로 움직임을 보인 것 이고 VR 기기를 사용하면 입체감 있게 볼 수 있다. 게임이 종료 시에는 <Figure 12>와 같이 다시 경기를 시작하여 상차림을 선택할 수 있도록 하였다.



〈Figure 10〉 Next Move in 2D Environment

〈Figure 11〉 3D Animation Environment

〈Figure 12〉 End of the Game

4.3 실험 결과

4.3.1 해쉬 알고리즘 테스트

본 연구에서는 그래프 생성 과정과 학습 과정, 수 선택 알고리즘 과정 등에서 일치하는 상태 노드를 빠르게 찾기 위해 해쉬 알고리즘을 사용하였다. 아래 <Table 3>과 같이 해쉬 알고리즘을 단계적으로 추가 적용하며 테스트한 결과 해쉬 함수를 사용하지 않았을 때와 비교하여 약 194.28배의 성능 향상 결과를 얻을 수 있었다.

4.3.2 학습 알고리즘 적용 성능 테스트

본 연구에서 개발한 알고리즘의 성능을 평가하기 위하여 현재 온라인 서비스되는 장기 AI와 테스트를 진행하였다. 테스트 결과 <Table 4>와 같이 무승부를 제외하고 약 59%의 승률을 확인할 수 있었다. 무승부는 게임 대결 시 장기 게임의 특징인 같은 수의 반복이 일어나는 경우 게임을 종료한 경우이다. 또한, 온라인 서비스를 통해 장기 게임 사용자들과 테스트를 진행한 결과 무승부를 제외하고 약 85.4%의 승률을 확인할 수 있었다. 테스트 대상자는 장기 초보자가 아니고 중수 급 이상의 실력자임을 확인하기 위하여 300전 이상, 승률 50% 이상인 경우에만 테스트를 진행하였다.

또한, 본 연구에서 개발한 알고리즘과 단순 MIN/MAX 알고리즘과의 비교 실험을 진행하고자 하였으나 장기의 특성상, 계속해서 같은 수를 반복하는 문제로 인하여 불가능하였고, 또한 상대적인 비교를 위해 같은 온라인 AI를 상대로 테스트를 진행하고자 하였으나 단순 MIN/MAX 알고리즘으로는 온라인 AI와의 대결에서 승률이 저조하기 때문에 비교 대결이 무의미하였다. 반면 학습 알고리즘을 적용한 결과가 훨씬 더 우수한 성능을 가진다는 결과를 간접적으로 확인할 수 있었다.

〈Table 4〉 Performance Improvements After Using Learning Algorithm

Type	Number of Games	Win	Draw	Loss	Winning Rate (Excluding Draw)
vs AI	150	62	45	43	41.3% (59%)
vs user	142	117	5	20	82.4% (85.4%)
Total	292	179	50	63	61.3% (73.9%)

〈Table 3〉 Performance Improvements After Using Hash Algorithm

Test type	Confirmed games	Time	Speed (Games/time)	Speed Improvement (vs 1st test)
Using single array	2,700	4 hours	675	-
Add one movement of 'Cho' and "Han'	15,300	4 hours 30 min.	3,400	5.03 times
Add location of 'horse'	15,300	22 min.	41,727	12.3 times (61.82 times)
Add location of 'cannon', and 'chariot'	15,300	7 min.	131,142	3.14 times (194.28 times)

5. 결론 및 향후 연구

본 연구에서는 학습 알고리즘의 연구를 위해 장기 소프트웨어를 개발하였으며, 연구의 주요 관점은 다음과 같다. 첫째, 단순 트리탐색 알고리즘에 의존하지 않고 장기 전문가들의 누적된 데이터를 활용함으로서 시스템개발에 보다 풍부한 실전결과를 반영할 수 있다. 둘째, 장기 게임의 특성상 같은 상태가 중복시키지 않기 위한 그래프 구조를 사용하였다. 셋째, 대전을 반복 할수록 우수한 승률을 얻기 위해 보드상황에 따라 강화학습 알고리즘과 MCTS 알고리즘을 응용함으로서 합리적 시간 내에 가장 좋은 다음수를 선택할 수 있다. 넷째, 이러한 대량의 데이터는 실시간으로 분석하고 처리하기 어렵기 때문에 해쉬 알고리즘을 다중 처리하여 해결하였다. 다섯째, 또한 학습된 그래프를 객체 직렬화 방식으로 저장하여 게임의 지속성을 제공하게 하였다. 마지막으로, VR기기를 이용해 인터페이스를 개발함으로써 사용자 친화적 인터페이스 기능을 제공하였다.

현재 구현된 시스템은 신경망을 이용한 알파고와 객관적인 성능을 비교하였을 때 부족한 알고리즘의 성능을 보인다. 따라서, 향후 연구에서는 첫째, 신뢰성이 있는 기보를 추가적으로 확보하고, 둘째, 학습 알고리즘을 다양하게 융합하여 접근해 볼 것이며, 셋째, 알파고와 같이 자체 시스템끼리 테스트가 가능하도록 시스템의 성능을 향상할 수 있도록 할 것이다.

References

Bang, J.S., D.C. Lee, S.H. Seo, Y.J. Kim, H.J. Lee, and W.H. Son, “Trends of VR/AR Game Technology”, *Electronics and Telecommunications Trends,* Vol.31, No.1, 2016, 146-156.

(방준성, 이동춘, 서상현, 김용준, 이헌주, 손욱호, “VR/AR 게임기술 동향”, *전자통신동향분석*, 제31권, 제1호, 2016, 146-156.

Browne, C., E. Powley, D. Whitehouse, S. Lucas, P. Cowling, and P. Rohlfshagen, “A Survey of Monte Carlo Tree Search Methods”, *IEEE Transactions on Computational Intelligence and AI in Games*, Vol.4, No.1, 2012, 1-49.

Cho, B.H. and C.J. Park, “Research Trends in Game AI”, *Electronics and Tellecommunications Trends*, Vol.23, No.4, 2008, 115-121.

(조병헌, 박창준, “게임 인공지능 연구동향”, *전자 통신동향분석*, 제23권, 제4호, 2008, 115-121.)

Fan, Y.F., X.J. Bai, R.Y. Lui, and S. Xing, “The Research of Chinese Chess Based on Database with Self Learning”, *Proceedings of the Ninth International Conference on Machine Learning and Cybernetics,* 2010, 319-322.

Feinberg, E. and A. Shwartz, *Handbook of Markov Decision Processes*, Kluwer, Boston, MA, 2002.

Ingram, J. and J. Mcgregor, “A Look at the Future of Autonomous Connected Vehicle Systems”, *Proceedings of International Conference on Software Engineering and Data Engineering*, Vol.23, 2015, 162-177.

Kim, S.W., S.W. Ahn, and H.S. Choo, “Artificial Intelligence of AlphaGo”, SPRi Issue Report, 2016-001, Vol.1, 2016.

(김석원, 안성원, 추형석, “AlphaGo의 인공지능”, SPRi Issue Report, 제2016-001호, 제1호, 2016.)

Kulsinskas, A., C. Balan, N. Bukdahl, and A. Brooks, “Augmentation of Board Games Using Smartphones”, *Lecture Notes in Computer Science*, Vol.2015, No.9177, 2015, 483-492.

Mandziuk, J., *Knowledge-Free and Learning-based Methods in Intelligent Game Playing*, Springer, Berlin, 2010.

Millington, I. and J. Funge, *Artificial Intelligence for Games*, Elsevier Inc., Berlington, MA, 2009.

Ong, C., H. Quek, K. Tan, and A. Tay, “Dis-

covering Chinese Chess Strategies through Co-evolutionary Approaches," *IEEE Symposium on Computational Intelligence and Games,* 2007, 360-367.

Park, D.S., "ICT Convergence Industry Outlook in the Era of Artificial Intelligence," *2017 ICT Industry Outlook Conference*, 2016.

(박대수, "인공지능 시대의 ICT융합 산업 전망", *2017 ICT산업 전망 컨퍼런스*, 2016.)

Park, H.S. and K.J. Kim, "Latest Research Trend of Artificial Intelligence in Games", *Journal of Korean Institute of Information Scientists and Engineers(KIISE)*, Vol.31, No.7, 2013, 8-15.

(박현수, 김경중, "게임 인공지능 최신 연구동향", *정보과학회지*, 제31권, 제7호, 2013, 8-15.)

Rivest, R., "Game Tree Searching by Min/Max Approximation", *Artificial Intelligence,* Vol. 34, No.1, 1987, 77-96.

Robilliard, D. and C. Fonlupt, "Towards Human-Competitive Game Playing for Complex Board Games with Genetic Programming", *Lecture Notes in Computer Science*, Vol.2016, No.9554, 2015, 123-135.

Spencer, P. and J. Oommen, "Novel AI Strategies for Multi-Player Games at Intermediate Board States", *Lecture Notes in Computer Science*, Vol.2015, No.9101, 2015, 33-42.

Silver, D., A. Huang, C. Maddison, A. Guez, L. Sifre, and G. Driessche, "Mastering the Game of Go with Deep Neural Networks and Tree Search", *Nature,* Vol.529, 2016, 484-489.

Wu, G. and J. Tao, "Chinese Chess Algorithm Design and Implementation in the Computer Games", *Proceedings of the 35th Chinese Control Conference*, 2016, 10380-10384.

About the Authors

Myeonggyu Jang (jkorea2@naver.com)

Will graduate from the department of Computer Science, Kookmin University in February 2018, Seoul, Korea, His interests include Artificial Intelligence, Machine Learning, Computer Vision, and Embedded System.

Youngho Kim (tngkr8121@gmail.com)

Will graduate from the department of Computer Science, Kookmin University in February 2018, Seoul, Korea, His interests include Artificial Intelligence, and Embedded System.

Dongyeop Min (dymin01@naver.com)

Will graduate from the department of Computer Science, Kookmin University in February 2018, Seoul, Korea, His interests include Artificial Intelligence, Game, and Embedded System.

Kihyeon Park (ppeaia1@gmail.com)

Will graduate from the department of Computer Science, Kookmin University in February 2018, Seoul, Korea, His interests include Artificial Intelligence, and Embedded System.

About the Authors

Seungsoo Lee (nonestrike@gmail.com)

Will graduate from the department of Computer Science, Kookmin University in February 2018, Seoul, Korea, His interests include Information Security, Network, Artificial Intelligence, and Game.

Chongwoo Woo (cwwoo@kookmin.ac.kr)

Received his PhD in Computer Science from Illinois Institute of Technology in 1991. He is currently a professor of Computer Science, Kookmin University, Korea. His research interests include Artificial Intelligence, Intelligent Agents, Modeling and Simulation, and Machine Learning.