Searching by learning: Exploring artificial general intelligence on small board games by deep reinforcement learning

Wang, H.

Chapter 7

Ranked Reward Reinforcement Learning

7.1 Introduction

In previous chapters, we mainly test our approaches on relatively small two-player board games. This chapter will however deal with single-player games and form thereby a bridge to combinatorial optimization.

In recent years, the interest in combinatorial games as a challenge in AI has increased after the first AlphaGo program [16] defeated the human world champion of Go [74]. The great success of the AlphaGo and AlphaZero programs [10, 16, 17] in two-player games, has inspired attempts in other domains [35, 89]. So far, one of the most challenging single player games, Morpion Solitaire [116] has not yet been studied with this promising deep reinforcement learning approach.

Morpion Solitaire is a popular single player game since 1960s [27, 116], because of its simple rules and simple equipment, requiring only paper and pencil. Due to its large state space it is also an interesting AI challenge in single player games, just like the game of Go challenge in two-player turn-based games. Could the AlphaZero self-play approach, that turned out to be so successful in Go, also work in Morpion Solitaire? For ten years little progress has been made in Morpion Solitaire. It is time to take up the challenge and to see if a self-play deep reinforcement learning approach will work in this challenging game as a few works on applying reinforcement learning are studied to deal with combinatorial tasks [117, 118, 119].

AlphaGo and AlphaZero combine deep neural networks [68] and MCTS [7] in a self-play framework that learns by curriculum learning [95]. Unfortunately, these approaches can not be directly used to play single agent combinatorial games, such as Travelling Salesman Problems (TSP) [120] and Bin Packing Problems (BPP) [121], where cost minimization is the goal of the game. To apply self-play for single player games, Laterre et al. proposed a Ranked Reward (R2) algorithm. R2 creates a relative performance metric by means of ranking the rewards obtained by a single agent over multiple games. In two-dimensional and three-dimensional bin packing R2 is reported to out-perform MCTS [122]. In this chapter we use this idea for Morpion Solitaire. Our contributions can be summarized as follows:

1. We present the first implementation[1] of Ranked Reward AlphaZero-style self-play for Morpion Solitaire.
2. On this implementation, we report our current best solution, of 67 steps (see Fig 7.2).

This result is very close to the human record, and shows the potential of the self-play reinforcement learning approach in Morpion Solitaire, and other hard single player combinatorial problems. Our result is even more remarkable, because it has been achieved in a tabula rasa setting, that is starting only with the knowledge [123] about the rules of the game and not encoding strategies and tactics of human players.

This chapter is structured as follows. After giving an overview of related work in Sect. 7.2, we introduce the Morpion Solitaire challenge in Sect. 7.3. Then we present how to integrate the idea of R2 into AlphaZero self-play in Sect. 7.4. Thereafter, we set up the experiment in Sect. 7.5, and show the result and analysis in Sect. 7.6. Finally, we conclude this chapter and discuss future work.

7.2 Related Work

Recent successes of AlphaGo series spark the interest of creating new self-play deep reinforcement learning approaches to deal with problems in the field of game AI, especially for other two player games [30, 31, 33, 64].

However, for single player games, self-play deep reinforcement learning approaches are not yet well studied since the approaches used for two-player games can not directly be used in single player games [122], since the goal of the task changes from

[1]Source code: https://github.com/wh1992v/R2RRMopionSolitaire

winning from an opponent, to minimizing the solution cost. Nevertheless, some researchers did initial works on single games with self-play deep reinforcement learning [124]. The main difficulty is representing single player games in ways that allow the use of a deep reinforcement learning approach. In order to get over this difficulty, Vinyals et al. [125] proposed a neural architecture (Pointer Networks) to represent combinatorial optimization problems as sequence-to-sequence learning problems. Early Pointer Networks achieved decent performance on TSP, but this approach is computationally expensive and requires handcrafted training examples for supervised learning methods. Replacing supervised learning methods by actor-critic methods removed this requirement [126]. In addition, Laterre et al. proposed the R2 algorithm through ranking the rewards obtained by a single agent over multiple games to label win or loss for each search, and this algorithm reportedly outperformed plain MCTS in the bin packing problem (BPP) [122]. Besides, Feng et al. recently used curriculum-driven deep reinforcement learning to cope with hard Sokoban instances [127]. The key of R2 is a progressing (like curriculum learning style) ranked reward list.

In addition to TSP and BPP, Morpion Solitaire has long been a challenge in NP-hard single player problems [27]. In brief, the goal in Morpion Solitaire is to maximally extend a given geometrical structure performing certain legal moves, and given a set of rules. Previous works on Morpion Solitaire mainly employ traditional heuristic search algorithms [116]. Cazenave created Nested Monte-Carlo Search and found an 80 moves record [128]. After that, a new Nested Rollout Policy Adaptation algorithm achieved a new 82 steps record [29]. Thereafter, Cazenave applied Beam Nested Rollout Policy Adaptation [129], which reached the same 82 steps record but did not exceed it, indicating the difficulty of making further progress on Morpion Solitaire using traditional search heuristics.

It is interesting to develop a new approach, applying (self-play) deep reinforcement learning to train a Morpion Solitaire player. The combination of the R2 algorithm with the AlphaZero self-play framework could be a first alternative for above mentioned approaches.

7.3 Morpion Solitaire

Morpion Solitaire is a single player game played on an unlimited grid. The rules of the game are simple. There are 36 black circles as the initial state (see Fig 7.1). A move for Morpion Solitaire consists of two parts: a) placing a new circle on the paper so that this new circle can be connected with four other existing circles horizontally, vertically or diagonally, and then b) drawing a line to connect these

five circles (see action 1, 2, 3 in the figure). Lines are allowed to cross other lines (action 4), but not allowed to overlap. There are two versions: the Touching (5T) version and the Disjoint (5D) version. For the 5T version, it is allowed to touch (action 5, green circle and green line), but for the 5D version, touching is illegal (any circle can not belong to two lines that have the same direction). After a legal action the circle and the line are added to the grid. This chapter focuses on the 5D version.

The best human score for the 5D version is 68 moves [27]. A score of 80 moves was found by means of Nested Monte-Carlo Search [128]. In addition, [29] found a new record with 82 steps, and [129] also found a 82 steps solution. It has been proven mathematically that the 5D version has an upper bound of 121 [28].

7.4 Ranked Reward Reinforcement Learning

AlphaZero self-play achieved milestone successes in two-player games, but can not be directly used for single player cost minimization games. Therefore, the R2 algorithm has been created to use self-play for generic single player MDPs. R2 reshapes the rewards according to player's relative performance over recent games [122]. The pseudo code of R2 is given in Algorithm 7.

Following AlphaZero-like self-play [34], we demonstrate the typical three stages as shown in the pseudo code. Since self-play in Morpion Solitaire MCTS is too time consuming due to the large state space. Thus, we rely on the policy directly from f_θ without tree search (line 6). For stage 3, we directly replace the previous neural network model with the newly trained model and let the newly trained model play a single time with MCTS enhancement (line 15). The R2 idea is integrated (see line 9 to line 11). The reward list B stores the recent game rewards. According to a ratio τ, the threshold of r_τ is calculated. We then compare r_τ to the game reward r_T to reshape the ranked reward z according to Equation 7.1.

$$z = \begin{cases} 1 & r_T > r_\tau \\ -1 & r_T < r_\tau \\ random(1, -1) & r_T = r_\tau \end{cases} \tag{7.1}$$

where r_τ is the stored reward value in B indexed by $L \times \tau$, L is the length of B, τ is a ratio parameter set to control the index of r_τ in B.

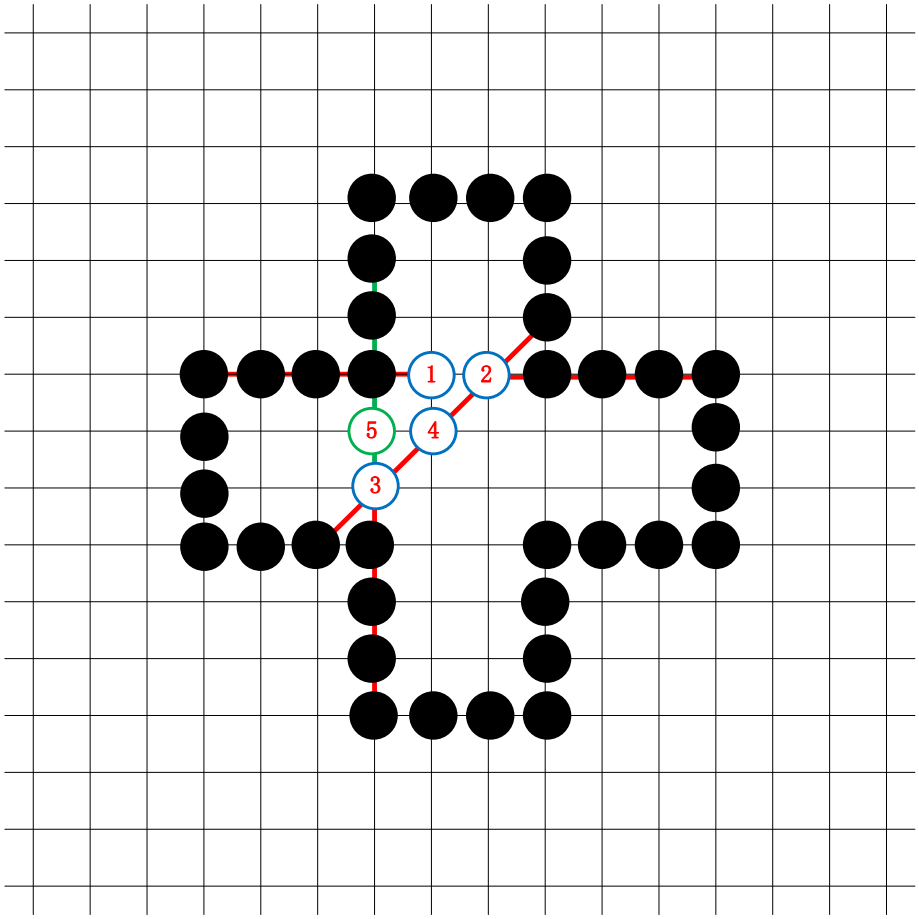

Figure 7.1: Moves Example: Moves 1, 2, 3, 4 are legal moves, move 5 is illegal for the 5D version, but legal for the 5T version. Move 1 is the first move, Move 2 is the second move, and so on.

Algorithm 7 Ranked Reward Reinforcement Learning within AlphaZero-like Self-play Framework

function RANKEDREWARDREINFORCEMENTLEARNING
  Initialize f_θ with random weights; Initialize retrain buffer D and reward list B
  for iteration=1, ..., I **do** ▷ self-play curriculum of I tournaments
    for episode=1,..., E **do** ▷ stage 1, self-play tournament of E games
      for t=1, ..., T', ..., T **do** ▷ play game of T moves
        $\pi_t \leftarrow$ perform MCTS based on f_θ or directly get policy from f_θ
        a_t =randomly select on π_t before T' or $\arg\max_a(\pi_t)$ after T' step
        executeAction(s_t, a_t)
      Calculate game reward r_T and store it in B
      Calculate threshold r_τ based on the recent games rewards in B
      Reshape the ranked reward z following Equation 7.1 ▷ Ranked Reward
      Store every (s_t, π_t, z_t) with ranked rewards z_t $(t \in [1, T])$ in D
    Randomly sample minibatch of examples (s_j, π_j, z_j) from D ▷ stage 2
    Train a new model $f_{\theta'}$ based on f_θ and examples
    Play once with MCTS enhancement on $f_{\theta'}$ ▷ stage 3
    Replace $f_\theta \leftarrow f'_\theta$
  return f_θ;

7.5 Experiment Setup

We perform our experiments on a GPU server with 128G RAM, 3TB local storage, 20 Intel Xeon E5-2650v3 cores (2.30GHz, 40 threads), 2 NVIDIA Titanium GPUs (each with 12GB memory) and 6 NVIDIA GTX 980 Ti GPUs (each with 6GB memory). And the code of framework interface is based on [65].

The hyper-parameters of our current R2 implementation are as much as possible equal to previous work. In this work, all neural network models share the same structure as in [34]. The hyper-parameter values for Algorithm 7 used in our experiments are given in Table 7.1. Partly, these values are set based on the work reported in [32] and the R2 approach for BPP [122]. T' is set to half of the current best record. m is set to 100 if using MCTS in self-play, but 20000 for MCTS in stage 3. Due to that MCTS needs too much computation in our setting, we do not use MCTS to enhance model in self-play (get policy from f_θ directly), but we use MCTS once for every 10 iterations in stage 3. Furthermore, as there is an upper bound of the best score (121), we did experiments on 16×16, 20×20 and 22×22 boards respectively. Training time for every algorithm is about a week.

Table 7.1: Default Parameter Settings

Parameter	Brief Description	Default Value
I	number of iterations	100
E	number of episodes	50
T'	step threshold	41
m	MCTS simulation times	20000
c	weight in UCT	1.0
rs	number of retrain iterations	10
ep	number of epochs	5
bs	batch size	64
lr	learning rate	0.005
d	dropout probability	0.3
L	length of B	200
τ	ratio to compute r_τ	0.75

7.6 Result and Analysis

As we mentioned above, the best score for Morpion Solitaire of 82 steps has been achieved by Nested Rollout Policy Adaptation (NRPA) in 2010. The best score achieved by human is 68. Our first attempt with limited computation resources on a large size board (22×22) achieved a score of 67, very close to the best human score. The resulting solution is shown in Fig 7.2.

Based on these promising results with Ranked Reward Reinforcement Learning we identify areas for further improvement. First, parameter values for the Morpion Solitaire game can be fine-tuned using results of small board games. Especially the parameter $m = 100$ seems not sufficient for large boards. Second, the neural network could be changed to Pointer Networks and the size of neural network should be deeper.

Note that the tuning of parameters is critical; if the reward list B is too small, the reward list can be easily filled up by scores close to 67. The training will then be stuck in a locally optimal solution. As good solutions are expected to be sparsely distributed over the search space, this increases the difficulty to get rid of a locally optimal solution once the algorithm has focused on it.

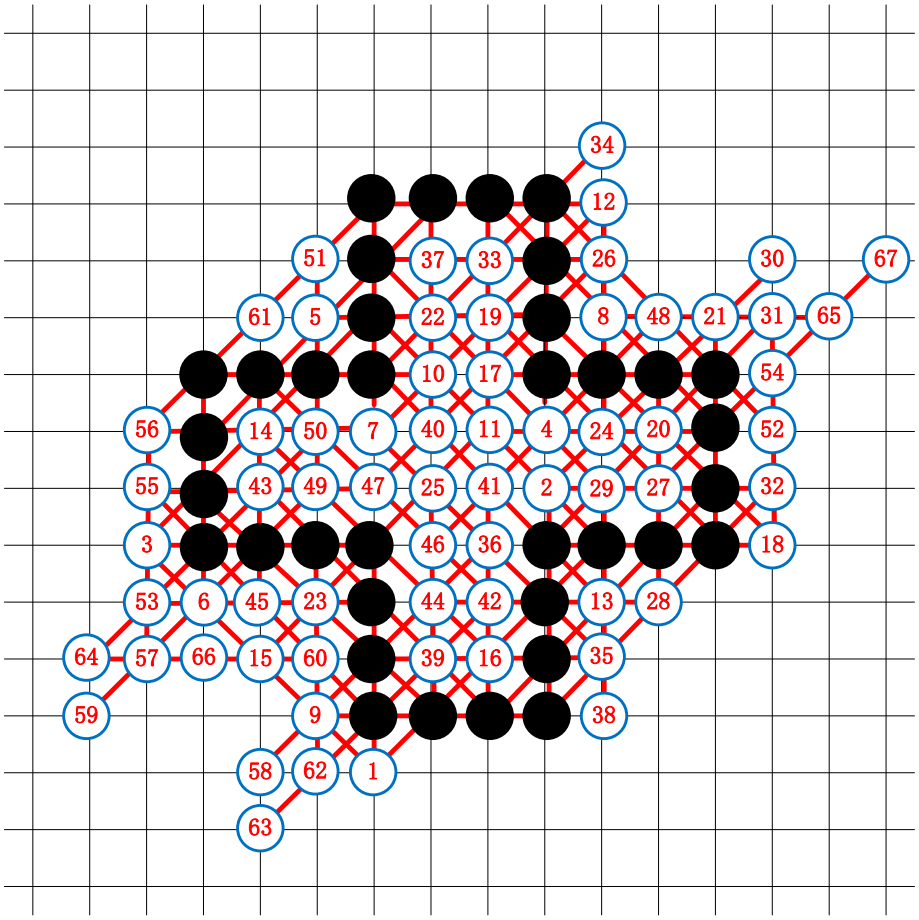

Figure 7.2: Detailed Steps of Our Best Solution

7.7 Summary

In this work, we apply a Ranked Reward Reinforcement Learning AlphaZero-like approach to play Morpion Solitaire, an important NP-hard single player game challenge. We train the player on 16×16, 20×20 and 22×22 boards, and find a near best human performance solution with 67 steps. As a first attempt of utilizing self-play deep reinforcement learning approach to tackle Morpion Solitaire, achieving near-human performance is a promising result.

To summarize, although the problem is difficult due to its large state space and sparsity of good solutions, applying a Ranked Reward self-play Reinforcement Learning approach to tackle Morpion Solitaire is a promising and learns from *tabula rasa*. We present our promising near-human result to stimulate future work on Morpion Solitaire and other single agent games with self-play reinforcement learning.