Tackling Morpion Solitaire with AlphaZero-like Ranked Reward Reinforcement Learning

Hui Wang, Mike Preuss, Michael Emmerich and Aske Plaat
Leiden Institute of Advanced Computer Science
Leiden University
The Netherlands
email: h.wang.13@liacs.leidenuniv.nl

***Abstract*—Morpion Solitaire is a popular single player game, performed with paper and pencil. Due to its large state space (on the order of the game of Go) traditional search algorithms, such as MCTS, have not been able to find good solutions. A later algorithm, Nested Rollout Policy Adaptation, was able to find a new record of 82 steps, albeit with large computational resources. After achieving this record, to the best of our knowledge, there has been no further progress reported, for about a decade.**

In this paper we take the recent impressive performance of deep self-learning reinforcement learning approaches from AlphaGo/AlphaZero as inspiration to design a searcher for Morpion Solitaire. A challenge of Morpion Solitaire is that the state space is sparse, there are few win/loss signals. Instead, we use an approach known as ranked reward to create a reinforcement learning self-play framework for Morpion Solitaire. This enables us to find medium-quality solutions with reasonable computational effort. Our record is a 67 steps solution, which is very close to the human best (68) without any other adaptation to the problem than using ranked reward. We list many further avenues for potential improvement.

***Index Terms*—Morpion Solitaire, Ranked Reward, Reinforcement Learning, AlphaZero, Self-play**

I. Introduction

In recent years, the interest in combinatorial games as a challenge in AI has increased after the first AlphaGo program [1] defeated the human world champion of Go [2]. The great success of the AlphaGo and AlphaZero programs [1], [3], [4] in two-player games, has inspired attempts in other domains [5], [6]. So far, one of the most challenging single player games, Morpion Solitaire [7] has not yet been studied with this promising deep reinforcement learning approach.

Morpion Solitaire is a popular single player game since 1960s [7], [8], 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, 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.

Hui Wang acknowledges financial support from the China Scholarship Council (CSC), CSC No.201706990015.

AlphaGo and AlphaZero combine deep neural networks [9] and Monte Carlo Tree Search (MCTS) [10] in a self-play framework that learns by curriculum learning [11]. Unfortunately, these approaches can not be directly used to play single agent combinatorial games, such as travelling salesman problems (TSP) [12] and bin package problems (BPP) [13], 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 outperform MCTS [14]. In this paper 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 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.

This paper is structured as follows. After giving an overview of related work in Sect. II, we introduce the Morpion Solitaire challenge in Sect. III. Then we present how to integrate the idea of R2 into AlphaZero self-play in Sect. IV. Thereafter, we set up the experiment in Sect. V, and show the result and analysis in Sect. VI. Finally, we conclude our paper and discuss future work.

II. Related Work

Deep reinforcement learning [15] approaches, especially the AlphaGo and AlphaZero programs, which combine online tree search and offline neural network training, achieve super human level of playing two player turn-based board games such as Go, Chess and Shogi [1], [3], [4]. These successes spark the interest of creating new deep reinforcement learning approaches to solve problems in the field of game AI, especially for other two player games [16]–[19].

However, for single player games, self-play deep reinforcement learning approaches are not yet well studied since the

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

approaches used for two-player games can not directly be used in single player games [14], since the goal of the task changes from winning from an opponent, to minimizing the solution cost. Nevertheless, some researchers did initial works on single games with self-play deep reinforcement learning [20]. The main difficulty is representing single player games in ways that allow the use of a deep reinforcement learning approach. In order to solve this difficulty, Vinyals et al. [21] 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 [22]. 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) [14]. Feng et al. recently used curriculum-driven deep reinforcement learning to solve hard Sokoban instances [23].

In addition to TSP and BPP, Morpion Solitaire has long been a challenge in NP-hard single player problems. Previous works on Morpion Solitaire mainly employ traditional heuristic search algorithms [7]. Cazenave created Nested Monte-Carlo Search and found an 80 moves record [24]. After that, a new Nested Rollout Policy Adaptation algorithm achieved a new 82 steps record [25]. Thereafter, Cazenave applied Beam Nested Rollout Policy Adaptation [26], 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.

We believe that it is time for 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.

III. Morpion Solitaire

Morpion Solitaire is a single player game played on an unlimited grid. It is a well know NP-hard challenge [7]. The rules of the game are simple. There are 36 black circles as the initial state (see Fig 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). A line is allowed to cross over each other (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. In this paper we are interested in the 5D version.

The best human score for the 5D version is 68 moves [8]. A score of 80 moves was found by means of Nested Monte-Carlo search [24]. In addition, [25] found a new record with 82 steps, and [26] also found a 82 steps solution. It has been proven mathematically that the 5D version has an upper bound of 121 [27].

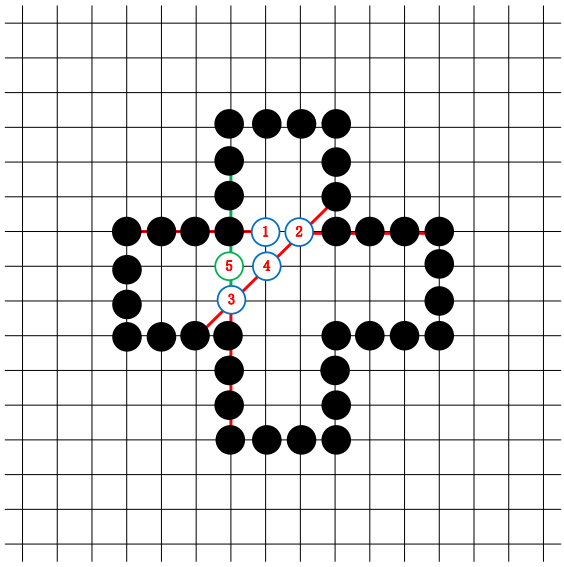

Fig. 1. Moves Example: Moves 1, 2, 3, 4 are legal moves, move 5 is illegal for the 5D version, but legal for the 5T version.

IV. 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 [14]. The pseudo code of R2 is given in Algorithm 1.

Following AlphaZero-like self-play [28], we demonstrate the typical three stages as shown in the pseudo code. For 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 we 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 α, the threshold of r_α is calculated. We then compare r_α to the game reward r_T to reshape the ranked reward z according to Equation 1.

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

where r_α is the stored reward value in B indexed by $L \times \alpha$, L is the length of B, α is a ratio parameter.

Algorithm 1 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 ▷ with or without tree search enhancement
        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_α based on the recent games rewards in B
      Reshape the ranked reward z following Equation 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 $f_{\theta'} \leftarrow f_\theta$
    Play once with MCTS enhancement on $f_{\theta'}$ ▷ stage 3
    Replace $f_\theta \leftarrow f'_\theta$
  return f_θ;

V. 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).

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 [28]. The hyper-parameter values for Algorithm 1 used in our experiments are given in Table I. Partly, these values are set based on the work reported in [29] and the R2 approach for BPP [14]. 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. 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 I
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
α	ratio to compute r_α	0.75

VI. RESULT AND ANALYSIS

As we mentioned above, the best score for Morpion Solitaire of 82 steps has been achieved by Nested Rollout Policy Adaptation 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 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.

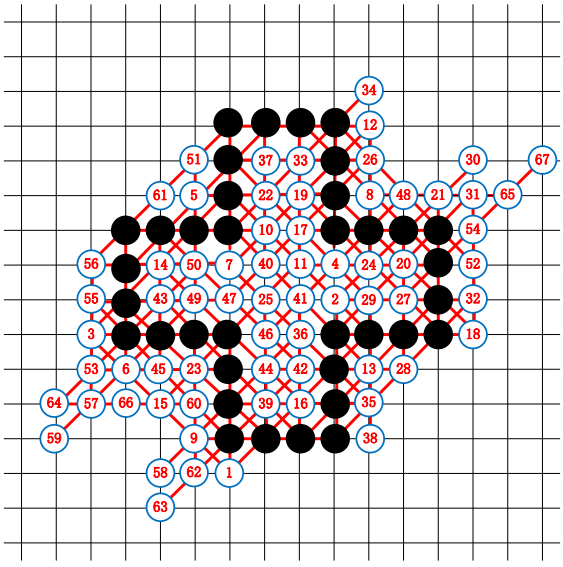

Fig. 2. Detailed Steps of Our Best Solution

VII. Conclusion and Outlook

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.

Our first results give us reason to believe that there remain ample possibilities to improve the approach by investigating the following aspects:

1) Parameter tuning: such as the Monte Carlo simulation times. Since good solutions are sparse in this game, maybe more exploration is beneficial?
2) Neural Network Design: It is reported that Pointer Networks perform better on combinatorial problems. A next step could be to also make the neural network structure deeper.
3) Local Optima: Through monitoring the reward list B, we can adjust in time by enlarging more exploration once it gets stuck in a locally optimal solution.
4) Computation resources and parallelization: enhanced parallelization may improve the results.

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.

Acknowledgment

Hui Wang acknowledges financial support from the China Scholarship Council (CSC), CSC No.201706990015.

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[2] A. Plaat, *Learning to Play: Reinforcement Learning and Games*. Springer Verlag, Heidelberg, New York, 2020.

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, no. 7676, p. 354, 2017.

[4] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[5] M. H. Segler, M. Preuss, and M. P. Waller, "Planning chemical syntheses with deep neural networks and symbolic ai," *Nature*, vol. 555, no. 7698, pp. 604–610, 2018.

[6] H. Wang, M. Preuss, and A. Plaat, "Warm-start alphazero self-play search enhancements," *arXiv preprint arXiv:2004.12357*, 2020.

[7] C. Boyer, "Morpion solitaire," http://www.morpionsolitaire.com/, 2020, accessed May, 2020.

[8] E. D. Demaine, M. L. Demaine, A. Langerman, and S. Langerman, "Morpion solitaire," *Theory of Computing Systems*, vol. 39, no. 3, pp. 439–453, 2006.

[9] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[10] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[11] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 41–48.

[12] C. Rego, D. Gamboa, F. Glover, and C. Osterman, "Traveling salesman problem heuristics: Leading methods, implementations and latest advances," *European Journal of Operational Research*, vol. 211, no. 3, pp. 427–441, 2011.

[13] H. Hu, L. Duan, X. Zhang, Y. Xu, and J. Wei, "A multi-task selected learning approach for solving new type 3d bin packing problem," *arXiv preprint arXiv:1804.06896*, 2018.

[14] A. Laterre, Y. Fu, M. K. Jabri, A.-S. Cohen, D. Kas, K. Hajjar, T. S. Dahl, A. Kerkeni, and K. Beguir, "Ranked reward: Enabling self-play reinforcement learning for combinatorial optimization," *arXiv preprint arXiv:1807.01672*, 2018.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[16] Y. Tian, J. Ma, Q. Gong, S. Sengupta, Z. Chen, J. Pinkerton, and C. L. Zitnick, "Elf opengo: An analysis and open reimplementation of alphazero," *arXiv preprint arXiv:1902.04522*, 2019.

[17] H. Wang, M. Emmerich, M. Preuss, and A. Plaat, "Analysis of hyper-parameters for small games: Iterations or epochs in self-play?" *arXiv preprint arXiv:2003.05988*, 2020.

[18] H. Wang, M. Emmerich, and A. Plaat, "Monte carlo q-learning for general game playing," *arXiv preprint arXiv:1802.05944*, 2018.

[19] ——, "Assessing the potential of classical q-learning in general game playing," in *Benelux Conference on Artificial Intelligence*. Springer, 2018, pp. 138–150.

[20] T. M. Moerland, J. Broekens, A. Plaat, and C. M. Jonker, "A0c: Alpha zero in continuous action space," *arXiv preprint arXiv:1805.09613*, 2018.

[21] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Advances in neural information processing systems*, 2015, pp. 2692–2700.

[22] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio, "Neural combinatorial optimization with reinforcement learning," *arXiv preprint arXiv:1611.09940*, 2016.

[23] D. Feng, C. P. Gomes, and B. Selman, "Solving hard ai planning instances using curriculum-driven deep reinforcement learning," *arXiv preprint arXiv:2006.02689*, 2020.

[24] T. Cazenave, "Nested monte-carlo search," in *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[25] C. D. Rosin, "Nested rollout policy adaptation for monte carlo tree search," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[26] T. Cazenave and F. Teytaud, "Beam nested rollout policy adaptation," 2012.

[27] A. Kawamura, T. Okamoto, Y. Tatsu, Y. Uno, and M. Yamato, "Morpion solitaire 5d: a new upper bound of 121 on the maximum score," *arXiv preprint arXiv:1307.8192*, 2013.

[28] H. Wang, M. Emmerich, M. Preuss, and A. Plaat, "Alternative loss functions in alphazero-like self-play," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2019, pp. 155–162.

[29] ——, "Hyper-parameter sweep on alphazero general," *arXiv preprint arXiv:1903.08129*, 2019.