Searching by learning: Exploring artificial general intelligence on small board games by deep reinforcement learning
Wang, H.

Appendix A

A.1 Symbols

Table A.1: Notations

-	Type	Description	Ref.
γ	$0 \leq \gamma \leq 1$	the discount factor of $max_{a'}Q(s', a')$	Eq. (2.1)
α	$0 \leq \alpha \leq 1$	the learning rate of Q-learning	Eq. (2.2)
ϵ	$0 \leq \epsilon \leq 1$	ϵ-greedy for exploration and exploitation	Eq. (2.3)
l	$\mathbb{N}^+$	match number used to control decaying speed of ϵ	Eq. (2.3)
d	$\mathbb{N}^+$	dimension of action space	Eq: (3.1)
$\mathbf{p}$	$\mathbb{R}^d$	policy provided by the neural network	Eq: (3.1)
π	$\mathbb{R}^d$	improved estimate policy after performing MCTS	Eq: (3.1)
v	$\mathbb{R}$	state value prediction	Eq: (3.1)
z	$\{-1, 0, 1\}$	real game end reward	Eq: (3.1)
λ	$0 \leq \lambda \leq 1$	a weight to balance policy and value loss function	Eq: (4.1)
β	$\mathbb{R}$	a weight number to balance $U(s, a)$ and $U_{rave}(s, a)$	Eq: (5.4)
L	$\mathbb{N}^+$	the length of the reward list	Eq: (7.1)
τ	$0 \leq \tau \leq 1$	a ratio to locate game length threshold in reward list	Eq: (7.1)
r_τ	$\mathbb{N}^+$	threshold reward of game length to judge win or loss	Eq: (7.1)

A.2 Abbreviations

Table A.2: Abbreviations

Abb.	Full Name
AI	Artificial Intelligence
AGI	Artificial General Intelligence
MCS	Monte Carlo Search
MCTS	Monte Carlo Tree Search
GGP	General game playing
RAVE	Rapid Action Value Estimation
GDL	Game Description Language
DQN	Deep Q-networks
GM	Game Manager
TCP/IP	Transmission Control Protocol/Internet Protocol
UCT	Upper Confidence bound applied to Trees
AMAF	All Moves As First
RHEA	Rolling Horizon Evolutionary Algorithm
P-UCT	Policy-Upper Confidence bound applied to Trees
HPC	High Performance Computing
TSP	Travelling Salesman Problems
BPP	Bin Packing Problems
R2	Ranked Reward
MDP	Markov Decision Process
NRPA	Nested Rollout Policy Adaptation

A.3 Algorithms

Algorithm 8 Time Limited Monte Carlo Search Algorithm

```
function MonteCarloSearch(time_limit)
    get legal actions set A of current state s
    get next states set S' where s' ∈ S'
    z(s')=0, count(s')=0
    while time_cost ≤ time_limit do
        for each s' in S' do
            outcome(s')←random simulation from s' to game end.
            z(s')+=outcome(s')
            count(s')+=1
        selected_action← getActionFromStates(s, arg max_{s'∈S'} z(S')/count(S'))
    return selected_action
```

Algorithm 9 QM-learning Enhancement

function QMPlayer(current state s, learning rate α, discount factor γ, Q table: $Q(S, A)$)
  for each match **do**
    if s terminates **then**
      for each (s, a) from end to the start in current match record **do**
        R(s,a) $=s'$ is terminal state? getGoal(s', myrole) : 0
        Update $Q(s, a) \leftarrow (1-\alpha)\, Q(s, a) + \alpha\ (\ R(s, a) + \gamma\ max_{a'} Q(s', a'))$
    else
      if ϵ-greedy is enabled **then**
        selected_action = Random()
      else
        selected_action = SelectFromQTable()
        if no s record in $Q(S, A)$ **then**
          MonteCarloSearch(time_limit)
      performAction(s, selected_action)
  return $Q(S, A)$

Algorithm 10 Neural Network Based MCTS with Only Rollout Simulation Value

function ROLLOUT(s, f_θ)
    Search(s)
    $\pi_s \leftarrow$normalize($Q(s, \cdot)$)
    return π_s
function SEARCH(s)
    Return game end result if s is a terminal state
    if s is not in the Tree **then**
        Add s to the Tree, initialize $Q(s, \cdot)$ and $N(s, \cdot)$ to 0
        Get $P(s, \cdot)$ and $v(s)$ by looking up $f_\theta(s)$
        Get result $v(s)$ by performing random rollout until the game ends
        return $v(s)$
    else
        Select an action a with highest UCT value
        $s' \leftarrow$getNextState(s, a)
        $v \leftarrow$Search($s\prime$)
        $Q(s, a) \leftarrow \frac{N(s,a)*Q(s,a)+v}{N(s,a)+1}$
        $N(s, a) \leftarrow N(s, a) + 1$
    return v;

Algorithm 11 Neural Network Based MCTS with Only RAVE Value

function RAVE(s, f_θ)
    Search(s)
    $\pi_s \leftarrow$normalize($Q_{rave}(s, \cdot)$)
    return π_s
function SEARCH(s)
    Return game end result if s is a terminal state
    if s is not in the Tree **then**
        Add s to the Tree,
        Initialize $Q(s, \cdot)$, $N(s, \cdot)$, $Q_{rave}(s, \cdot)$ and $N_{rave}(s, \cdot)$ to 0.
        Get $P(s, \cdot)$ and $v(s)$ by looking up $f_\theta(s)$
        return $v(s)$
    else
        Select an aciton a with highest UCT_{rave} value
        $s\prime \leftarrow$getNextState(s, a)
        $v \leftarrow$Search($s\prime$)
        $Q(s, a) \leftarrow \frac{N(s,a)*Q(s,a)+v}{N(s,a)+1}$
        $N(s, a) \leftarrow N(s, a) + 1$
        $N_{rave}(s_{t_1}, a_{t_2}) \leftarrow N_{rave}(s_{t_1}, a_{t_2}) + 1$
        $Q_{rave}(s_{t_1}, a_{t_2}) \leftarrow \frac{N_{rave}(s_{t_1},a_{t_2})*Q_{rave}(s_{t_1},a_{t_2})+v}{N_{rave}(s_{t_1},a_{t_2})+1}$
        $\triangleright$ where $s_{t_1} \in VisitedPath$, and $a_{t_2} \in A(s_{t_1})$, and for $\forall t < t_2, a_t \neq a_{t_2}$
    return v;

Algorithm 12 Neural Network Based MCTS with Rollout Simulation and RAVE Value

function RoRA(s, f_θ)
  Search(s)
  $\pi_s \leftarrow$normalize($Q_{rave}(s, \cdot)$)
  return π_s
function Search(s)
  Return game end result if s is a terminal state
  if s is not in the Tree **then**
    Add s to the Tree,
    Initialize $Q(s, \cdot)$, $N(s, \cdot)$, $Q_{rave}(s, \cdot)$ and $N_{rave}(s, \cdot)$ to 0.
    Get $P(s, \cdot)$ and $v(s)$ by looking up $f_\theta(s)$
    Get result $v(s)$ by performing random rollout until the game ends
    return $v(s)$
  else
    Select an aciton a with highest UCT_{rave} value
    $s\prime \leftarrow$getNextState(s, a)
    $v \leftarrow$Search($s\prime$)
    $Q(s, a) \leftarrow \frac{N(s,a)*Q(s,a)+v}{N(s,a)+1}$
    $N(s, a) \leftarrow N(s, a) + 1$
    $N_{rave}(s_{t_1}, a_{t_2}) \leftarrow N_{rave}(s_{t_1}, a_{t_2}) + 1$
    $Q_{rave}(s_{t_1}, a_{t_2}) \leftarrow \frac{N_{rave}(s_{t_1},a_{t_2})*Q_{rave}(s_{t_1},a_{t_2})+v}{N_{rave}(s_{t_1},a_{t_2})+1}$
    $\triangleright$ where $s_{t_1} \in VisitedPath$, and $a_{t_2} \in A(s_{t_1})$, and for $\forall t < t_2, a_t \neq a_{t_2}$
  return v;

Algorithm 13 Neural Network Based MCTS with Neural Network and Rollout Simulation Value

```
function WRo(s, f_θ)
    Search(s)
    π_s ←normalize(Q(s, ·))
    return π_s
function Search(s)
    Return game end result if s is a terminal state
    if s is not in the Tree then
        Add s to the Tree, initialize Q(s, ·) and N(s, ·) to 0
        Get P(s, ·) and v(s)_network by looking up f_θ(s)
        Get result v(s)_rollout by performing random rollout until the game ends
        v(s) = (1 − weight) ∗ v_network + weight ∗ v_rollout
        return v(s)
    else
        Select an action a with highest UCT value
        s′ ←getNextState(s, a)
        v ←Search(s′)
        Q(s, a) ← (N(s,a)∗Q(s,a)+v) / (N(s,a)+1)
        N(s, a) ← N(s, a) + 1
    return v;
```

Algorithm 14 Neural Network Based MCTS with Neural Network, Rave and Rollout Simulation Value

function WRoRa(s, f_θ)
    Search(s)
    $\pi_s \leftarrow$normalize($Q(s, \cdot)$)
    return π_s
function Search(s)
    Return game end result if s is a terminal state
    if s is not in the Tree **then**
        Add s to the Tree,
        Initialize $Q(s, \cdot)$, $N(s, \cdot)$, $Q_{rave}(s, \cdot)$ and $N_{rave}(s, \cdot)$ to 0.
        Get $P(s, \cdot)$ and $v(s)_{network}$ by looking up $f_\theta(s)$
        Get result $v(s)_{rollout}$ by performing random rollout until the game ends
        random rollout path added to $VisitedPath$
        $v(s) = (1 - weight) * v_{network} + weight * v_{rollout}$
        return $v(s)$
    else
        Select an aciton a with highest UCT_{rave} value
        $s' \leftarrow$getNextState(s, a)
        $v \leftarrow$Search($s\prime$)
        $Q(s, a) \leftarrow \frac{N(s,a)*Q(s,a)+v}{N(s,a)+1}$
        $N(s, a) \leftarrow N(s, a) + 1$
        $N_{rave}(s_{t_1}, a_{t_2}) \leftarrow N_{rave}(s_{t_1}, a_{t_2}) + 1$
        $Q_{rave}(s_{t_1}, a_{t_2}) \leftarrow \frac{N_{rave}(s_{t_1},a_{t_2})*Q_{rave}(s_{t_1},a_{t_2})+v}{N_{rave}(s_{t_1},a_{t_2})+1}$
        $\triangleright$ where $s_{t_1} \in VisitedPath$, and $a_{t_2} \in A(s_{t_1})$, and for $\forall t < t_2, a_t \neq a_{t_2}$
    return v;

Algorithm 15 Rolling Horizon Evolutionary Algorithm

function RHEA(s, $time\_limit$)
    Set up population of n valid action sequences of length l: $A_{n,l}$
    for all $A_{i<n}$ **do** Evaluate(A_i)
    repeat
        new action sequence A_j = mutate one randomly chosen action sequence by changing every move with a small random chance
        $f(A_j)$ = Evaluate(A_J)
        add A_J to population
        remove A_i with worst $f(A_i)$ from population
    until $time\_cost \geq time\_limit$
    return first action of best sequence in population
function EVALUATE(A_i)
    repeat
        Play action sequence in A_i
        Get result $success, game\_steps$ by performing random rollout until the game ends
    until $repetitions \geq 2$
    compute fitness $f(A_i)$ from average success probability with sequence length penalty (line 117)
    return $f(A_i)$

A.4 Elo Computation

In this dissertation, like AlphaZero series papers did, a whole history Bayesian Elo computation [82] is also employed to present the relative competence of playing the game of different trained models instead of a win or loss rate. In this section, a full computation process will be described in detail based on the Bayesian Elo computation system (called Bayeselo) provided on github [131].

Bayeselo is a free software tool to compute Elo ratings. It receives a file containing game records written in PGN (Portable Game Notation) format [132], and produces a rating list [131]. Therefore, a full process can be simply described in Fig. A.1.

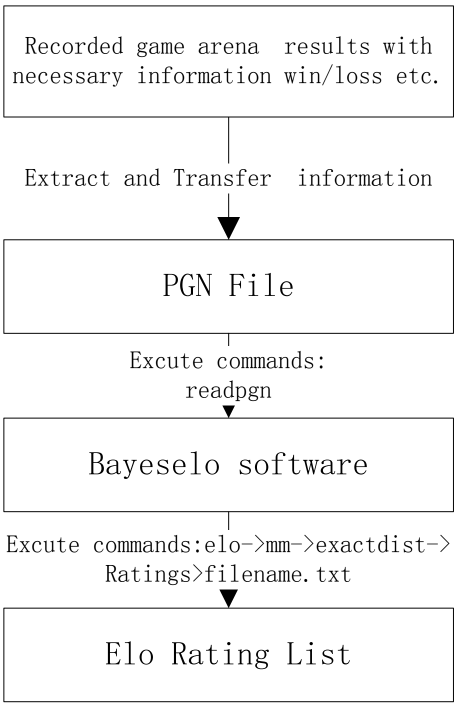

Figure A.1: a full Bayesian Elo Computation Process.

An example of part of a PGN file (arena_othello_final.pgn) generated based on win/loss results recorded during AlphaZero-like self-play arena competition is shown as Fig. A.2.

```
......
[Event "arena_othello_final.pgn"]
[Iteration "926"]
[Site "liacs server, Leiden"]
[Round "6"]
[White "bestmodel_mcts_rave_run2"]
[Black "bestmodel_mcts_rave_run5"]
[Result "1-0"]
Here are detailed game moves for [Iteration "926, round6"]

[Event "arena_othello_final.pgn"]
[Iteration "926"]
[Site "liacs server, Leiden"]
[Round "7"]
[White "bestmodel_mcts_rave_run2"]
[Black "bestmodel_mcts_rave_run5"]
[Result "0-1"]
Here are detailed game moves for [Iteration "926, round7"]

[Event "arena_othello_final.pgn"]
[Iteration "926"]
[Site "liacs server, Leiden"]
[Round "8"]
[White "bestmodel_mcts_rave_run2"]
[Black "bestmodel_mcts_rave_run5"]
[Result "0-1"]
Here are detailed game moves for [Iteration "926, round8"]

[Event "arena_othello_final.pgn"]
[Iteration "926"]
[Site "liacs server, Leiden"]
[Round "9"]
......
```

Figure A.2: Small Part of PGN file arena_othello_final.pgn. The file contains much such format iterative arena competition information. Each pair of White and Black players played 20 rounds.

An example of elo rating list generated by operating Bayeselo system with PGN file (arena_othello_final.pgn) as input is shown as follows. See Fig. A.3. The figures of elo ratings in this dissertation are visualized based on such elo rating lists.

Table A.3: An example of Generated Elo Rating List by Bayeselo

Rank	Name	Elo	+	-	games	score	oppo.	draws
1	bestmodel_mcts_rave_rollout_run6	117	22	22	960	62%	-2	0%
2	bestmodel_weight_mcts_rave_rollout_run5	102	21	21	960	57%	-2	0%
3	bestmodel_weight_mcts_rave_rollout_run2	100	21	21	960	58%	-2	0%
4	bestmodel_weight_mcts_rollout_run5	97	22	21	960	58%	-2	0%
5	bestmodel_mcts_rave_run1	86	21	21	960	58%	-2	0%
6	bestmodel_weight_mcts_rave_rollout_run8	81	21	21	960	55%	-2	0%
7	bestmoldel_pi_v_run1	77	22	21	960	61%	-2	0%
8	bestmodel_weight_mcts_rave_rollout_run3	71	21	21	960	54%	-1	0%
9	bestmodel_mcts_rave_rollout_run2	62	21	21	960	57%	-1	0%
10	bestmodel_weight_mcts_rave_rollout_run1	54	21	21	960	53%	-1	0%
11	bestmodel_weight_mcts_rave_rollout_run6	53	21	21	960	52%	-1	0%
12	bestmodel_mcts_rave_run4	53	21	21	960	54%	-1	0%
13	bestmodel_weight_mcts_rave_rollout_run4	52	21	21	960	52%	-1	0%
14	bestmodel_weight_mcts_rollout_run3	44	21	21	960	52%	-1	0%
15	bestmodel_weight_mcts_rollout_run8	39	21	21	960	51%	-1	0%
16	bestmoldel_pi_v_run4	39	21	21	960	56%	-1	0%
17	bestmodel_weight_mcts_rave_rollout_run7	36	21	21	960	50%	-1	0%
18	bestmodel_weight_mcts_rollout_run7	34	21	21	960	51%	-1	0%
19	bestmodel_mcts_rave_run7	32	21	21	960	51%	-1	0%
20	bestmodel_weight_mcts_rollout_run2	30	21	21	960	51%	-1	0%
21	bestmodel_mcts_rave_rollout_run5	28	21	21	960	52%	-1	0%
22	bestmodel_mcts_rave_run2	21	21	21	960	51%	0	0%
23	bestmodel_weight_mcts_rollout_run4	20	21	21	960	49%	0	0%
24	bestmodel_weight_mcts_rollout_run1	20	21	21	960	50%	0	0%
25	bestmoldel_pi_v_run6	18	21	21	960	53%	0	0%
26	bestmodel_mcts_rollout_run3	17	21	21	960	53%	0	0%
27	bestmodel_mcts_rave_rollout_run7	15	21	21	960	51%	0	0%
28	bestmodel_mcts_rollout_run1	11	21	21	960	52%	0	0%
29	bestmodel_mcts_rave_run8	10	21	21	960	49%	0	0%
30	bestmodel_weight_mcts_rollout_run6	8	21	21	960	48%	0	0%
31	bestmodel_mcts_rollout_run2	7	21	21	960	52%	0	0%
32	bestmodel_mcts_rave_rollout_run8	6	21	21	960	49%	0	0%
33	bestmodel_mcts_rave_run5	5	21	21	960	49%	0	0%
34	bestmodel_mcts_rave_run6	4	21	21	960	48%	0	0%
35	bestmodel_mcts_rave_rollout_run3	2	21	21	960	50%	0	0%
36	bestmoldel_pi_v_run7	-2	21	21	960	51%	0	0%
37	bestmodel_mcts_rollout_run6	-6	21	21	960	49%	0	0%
38	bestmoldel_pi_v_run3	-7	21	21	960	51%	0	0%
39	bestmodel_mcts_rollout_run5	-8	21	21	960	49%	0	0%
40	bestmodel_mcts_rave_rollout_run4	-10	21	21	960	48%	0	0%
41	bestmodel_mcts_rollout_run7	-11	21	21	960	49%	0	0%
42	bestmodel_mcts_rave_rollout_run1	-17	21	21	960	48%	0	0%
43	bestmoldel_pi_v_run8	-17	21	21	960	49%	0	0%
44	bestmodel_mcts_rollout_run8	-43	21	21	960	45%	1	0%
45	bestmoldel_pi_v_run5	-65	21	21	960	44%	1	0%
46	bestmodel_mcts_rave_run3	-66	21	22	960	41%	1	0%
47	bestmoldel_pi_v_run2	-100	21	22	960	41%	2	0%
48	bestmodel_mcts_rollout_run4	-109	22	22	960	38%	2	0%
49	randomplayer	-988	124	204	960	0%	21	0%

Bibliography

[1] Crevier, D.: AI: the tumultuous history of the search for artificial intelligence. Basic Books, Inc. (1993)

[2] Kurzweil, R.: The singularity is near: When humans transcend biology. Penguin (2005)

[3] Searle, J.R., et al.: Minds, brains, and programs. The Turing Test: Verbal Behaviour as the Hallmark of Intelligence (1980) 201–224

[4] Hodson, H.: Deepmind and Google: the battle to control artificial intelligence. The Economist (2019)

[5] Grace, K., Salvatier, J., Dafoe, A., Zhang, B., Evans, O.: When will AI exceed human performance? Evidence from AI experts. Journal of Artificial Intelligence Research **62** (2018) 729–754

[6] Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT Press (2018)

[7] Browne, C.B., Powley, E., Whitehouse, D., Lucas, S.M., Cowling, P.I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S.: A survey of Monte Carlo Tree Search methods. IEEE Transactions on Computational Intelligence and AI in Games **4** (2012) 1–43

[8] Fagin, R., Moses, Y., Halpern, J.Y., Vardi, M.Y.: Reasoning about knowledge. MIT press (2003)

[9] Bishop, C.M.: Pattern recognition. Machine learning **128** (2006)

[10] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al.: A general reinforcement learning algorithm that masters Chess, Shogi, and Go through self-play. Science **362** (2018) 1140–1144

[11] Ward, C.D., Cowling, P.I.: Monte Carlo Search applied to card selection in magic: The gathering. In: 2009 IEEE Symposium on Computational Intelligence and Games, IEEE (2009) 9–16

[12] Winands, M.H., Björnsson, Y., Saito, J.T.: Monte-Carlo Tree Search solver. In: International Conference on Computers and Games, Springer (2008) 25–36

[13] Watkins, C.J., Dayan, P.: Q-learning. Machine Learning **8** (1992) 279–292

[14] Fan, J., Wang, Z., Xie, Y., Yang, Z.: A theoretical analysis of deep Q-learning. In: Learning for Dynamics and Control, PMLR (2020) 486–489

[15] Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., et al.: Deep Q-learning from demonstrations. In: Proceedings of the AAAI Conference on Artificial Intelligence. Volume 32. (2018)

[16] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of Go with deep neural networks and tree search. Nature **529** (2016) 484

[17] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al.: Mastering the game of Go without human knowledge. Nature **550** (2017) 354

[18] Gelly, S., Silver, D.: Combining online and offline knowledge in UCT. In: Proceedings of the 24th International Conference on Machine Learning. (2007) 273–280

[19] Finnsson, H.: Generalized Monte-Carlo Tree Search extensions for general game playing. In: Proceedings of the AAAI Conference on Artificial Intelligence. Volume 26. (2012)

[20] Sironi, C.F., Winands, M.H.: On-line parameter tuning for Monte-Carlo Tree Search in general game playing. In: Workshop on Computer Games, Springer (2017) 75–95

[21] Sironi, C.F., Liu, J., Winands, M.H.: Self-adaptive Monte Carlo Tree Search in general game playing. IEEE Transactions on Games **12** (2018) 132–144

[22] Méhat, J., Cazenave, T.: A parallel general game player. KI-künstliche Intelligenz **25** (2011) 43–47

[23] Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., et al.: Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature **575** (2019) 350–354

[24] Gelly, S., Silver, D.: Monte-Carlo Tree Search and rapid action value estimation in computer Go. Artificial Intelligence **175** (2011) 1856–1875

[25] Sironi, C.F., Winands, M.H.: Comparison of rapid action value estimation variants for general game playing. In: 2016 IEEE Conference on Computational Intelligence and Games (CIG), IEEE (2016) 1–8

[26] Couëtoux, A., Milone, M., Brendel, M., Doghmen, H., Sebag, M., Teytaud, O.: Continuous rapid action value estimates. In: Asian Conference on Machine Learning, PMLR (2011) 19–31

[27] Demaine, E.D., Demaine, M.L., Langerman, A., Langerman, S.: Morpion Solitaire. Theory of Computing Systems **39** (2006) 439–453

[28] Kawamura, A., Okamoto, T., Tatsu, Y., Uno, Y., Yamato, M.: Morpion Solitaire 5D: a new upper bound of 121 on the maximum score. arXiv preprint arXiv:1307.8192 (2013)

[29] Rosin, C.D.: Nested rollout policy adaptation for Monte Carlo Tree Search. In: Twenty-Second International Joint Conference on Artificial Intelligence. (2011) 649–654

[30] Wang, H., Emmerich, M., Plaat, A.: Monte Carlo Q-learning for general game playing. arXiv preprint arXiv:1802.05944 (2018)

[31] Wang, H., Emmerich, M., Plaat, A.: Assessing the potential of classical Q-learning in general game playing. In: Benelux Conference on Artificial Intelligence, Springer (2018) 138–150

[32] Wang, H., Emmerich, M., Preuss, M., Plaat, A.: Hyper-parameter sweep on AlphaZero General. arXiv preprint arXiv:1903.08129 (2019)

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

[34] Wang, H., Emmerich, M., Preuss, M., Plaat, A.: Alternative loss functions in AlphaZero-like self-play. In: 2019 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE (2019) 155–162

[35] Wang, H., Preuss, M., Plaat, A.: Warm-start AlphaZero self-play search enhancements. In: International Conference on Parallel Problem Solving from Nature, Springer (2020) 528–542

[36] Wang, H., Preuss, M., Plaat, A.: Adaptive warm-start MCTS in AlphaZero-like deep reinforcement learning. arXiv preprint arXiv:2105.06136 (2021)

[37] Wang, H., Preuss, M., Emmerich, M., Plaat, A.: Tackling Morpion Solitaire with AlphaZero-like ranked reward reinforcement learning. In: 2020 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), IEEE (2020) 149–152

[38] Wang, H., Tang, Y., Liu, J., Chen, W.: A search optimization method for rule learning in board games. In: Pacific Rim International Conference on Artificial Intelligence, Springer (2018) 174–181

[39] Genesereth, M., Love, N., Pell, B.: General game playing: Overview of the AAAI Competition. AI Magazine **26** (2005) 62–62

[40] Love, N., Hinrichs, T., Haley, D., Schkufza, E., Genesereth, M.: General game playing: Game description language specification. Technical report, Stanford Logic Group Computer Science Department Stanford University (2008)

[41] Kaiser, D.M.: The design and implementation of a successful general game playing agent. In: FLAIRS Conference. (2007) 110–115

[42] Genesereth, M., Thielscher, M.: General game playing. Synthesis Lectures on Artificial Intelligence and Machine Learning **8** (2014) 1–229

[43] Świechowski, M., Mańdziuk, J.: Fast interpreter for logical reasoning in general game playing. Journal of Logic and Computation **26** (2016) 1697–1727

[44] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518** (2015) 529–533

[45] Mehat, J., Cazenave, T.: Monte-Carlo Tree Search for general game playing. Univ. Paris **8** (2008)

[46] Banerjee, B., Stone, P.: General game learning using knowledge transfer. In: IJCAI Proceedings-International Joint Conference on Artificial Intelligence. (2007) 672–677

[47] Hammersley, J.: Monte Carlo methods. Springer Science & Business Media (2013)

[48] Thielscher, M.: The general game playing description language is universal. In: IJCAI Proceedings-International Joint Conference on Artificial Intelligence. Volume 22. (2011) 1107

[49] Watkins, C.J.C.H.: Learning from delayed rewards. PhD thesis, King's College, Cambridge (1989)

[50] Even-Dar, E., Mansour, Y.: Convergence of optimistic and incremental Q-learning. Advances in Neural Information Processing Systems **14** (2001) 1499–1506

[51] Hu, J., Wellman, M.P.: Nash Q-learning for general-sum stochastic games. Journal of Machine Learning Research **4** (2003) 1039–1069

[52] Wiering, M., Van Otterlo, M.: Reinforcement learning. Adaptation, Learning, and Optimization **12** (2012)

[53] Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: A survey. Journal of Artificial Intelligence Research **4** (1996) 237–285

[54] Torrey, L., Shavlik, J.: Transfer learning. In: Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques. IGI Global (2010) 242–264

[55] Vodopivec, T., Samothrakis, S., Ster, B.: On Monte Carlo tree search and reinforcement learning. Journal of Artificial Intelligence Research **60** (2017) 881–936

[56] Méhat, J., Cazenave, T.: Combining UCT and nested Monte Carlo Search for single-player general game playing. IEEE Transactions on Computational Intelligence and AI in Games **2** (2010) 271–277

[57] Cazenave, T., Saffidine, A., Schofield, M.J., Thielscher, M.: Nested Monte Carlo Search for two-player games. In: AAAI. Volume 16. (2016) 687–693

[58] Ruijl, B., Vermaseren, J., Plaat, A., Herik, J.v.d.: Combining simulated annealing and Monte Carlo Tree Search for expression simplification. arXiv preprint arXiv:1312.0841 (2013)

[59] Soemers, D.J., Mella, V., Browne, C., Teytaud, O.: Deep learning for general game playing with Ludii and Polygames. arXiv preprint arXiv:2101.09562 (2021)

[60] Goldwaser, A., Thielscher, M.: Deep reinforcement learning for general game playing. In: Proceedings of the AAAI Conference on Artificial Intelligence. Volume 34. (2020) 1701–1708

[61] Tao, J., Wu, L., Hu, X.: Principle analysis on AlphaGo and perspective in military application of artificial intelligence. Journal of Command and Control **2** (2016) 114–120

[62] Zhang, Z.: When doctors meet with AlphaGo: potential application of machine learning to clinical medicine. Annals of Translational Medicine **4** (2016)

[63] Silver, A.: Leela Chess Zero: AlphaZero for the PC (2019)

[64] Tian, Y., Ma, J., Gong, Q., Sengupta, S., Chen, Z., Pinkerton, J., Zitnick, C.L.: Elf opengo: An analysis and open reimplementation of AlphaZero. arXiv preprint arXiv:1902.04522 (2019)

[65] Nair, S.: AlphaZero General. `https://github.com/suragnair/alpha-zero-general` (2018) Accessed May, 2018.

[66] Chaslot, G.M.J., Winands, M.H., HERIK, H.J.V.D., Uiterwijk, J.W., Bouzy, B.: Progressive strategies for Monte-Carlo Tree Search. New Mathematics and Natural Computation **4** (2008) 343–357

[67] Chaslot, G.M.B., Winands, M.H., van Den Herik, H.J.: Parallel Monte-Carlo Tree Search. In: International Conference on Computers and Games, Springer (2008) 60–71

[68] Schmidhuber, J.: Deep learning in neural networks: An overview. Neural Networks **61** (2015) 85–117

[69] Clark, C., Storkey, A.: Training deep convolutional neural networks to play Go. In: International Conference on Machine Learning. (2015) 1766–1774

[70] Tesauro, G.: Temporal difference learning and TD-Gammon. Communications of the ACM **38** (1995) 58–68

[71] Heinz, E.A.: New self-play results in computer Chess. In: International Conference on Computers and Games, Springer (2000) 262–276

[72] Wiering, M.A., et al.: Self-play and using an expert to learn to play Backgammon with temporal difference learning. Journal of Intelligent Learning Systems and Applications **2** (2010) 57

[73] Van Der Ree, M., Wiering, M.: Reinforcement learning in the game of Othello: learning against a fixed opponent and learning from self-play. In: 2013 IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), IEEE (2013) 108–115

[74] Plaat, A.: Learning to Play. Springer (2020)

[75] Chen, Y., Huang, A., Wang, Z., Antonoglou, I., Schrittwieser, J., Silver, D., de Freitas, N.: Bayesian optimization in AlphaGo. arXiv preprint arXiv:1812.06855 (2018)

[76] Iwata, S., Kasai, T.: The Othello game on an n$\times$ n board is PSPACE-complete. Theoretical Computer Science **123** (1994) 329–340

[77] Buro, M.: The Othello match of the year: Takeshi Murakami vs. Logistello. ICGA Journal **20** (1997) 189–193

[78] Chong, S.Y., Tan, M.K., White, J.D.: Observing the evolution of neural networks learning to play the game of Othello. IEEE Transactions on Evolutionary Computation **9** (2005) 240–251

[79] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)

[80] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

[81] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research **15** (2014) 1929–1958

[82] Coulom, R.: Whole-history rating: A Bayesian rating system for players of time-varying strength. In: International Conference on Computers and Games, Springer (2008) 113–124

[83] Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K., et al.: A racing algorithm for configuring metaheuristics. In: Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation. Volume 2. (2002)

[84] Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: International Conference on Learning and Intelligent Optimization, Springer (2011) 507–523

[85] Moerland, T.M., Broekens, J., Jonker, C.M.: Model-based reinforcement learning: A survey. arXiv preprint arXiv:2006.16712 (2020)

[86] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D.: Deep reinforcement learning that matters. In: Proceedings of the AAAI Conference on Artificial Intelligence. Volume 32. (2018)

[87] Igami, M.: Artificial intelligence as structural estimation: Deep Blue, Bonanza, and AlphaGo. The Econometrics Journal **23** (2020) S1–S24

[88] Li, Y., Fang, Y., Akhtar, Z.: Accelerating deep reinforcement learning model for game strategy. Neurocomputing **408** (2020) 157–168

[89] Segler, M.H., Preuss, M., Waller, M.P.: Planning chemical syntheses with deep neural networks and symbolic AI. Nature **555** (2018) 604–610

[90] Fu, M.C.: AlphaGo and Monte Carlo Tree Search: the simulation optimization perspective. In: 2016 Winter Simulation Conference (WSC), IEEE (2016) 659–670

[91] Matsuzaki, K., Kitamura, N.: Do evaluation functions really improve Monte-Carlo Tree Search? ICGA Journal **40** (2018) 294–304

[92] Allis, L.V.: A knowledge-based approach of Connect-Four. J. Int. Comput. Games Assoc. **11** (1988) 165

[93] Reisch, S.: Gobang ist PSPACE-vollständig. Acta Informatica **13** (1980) 59–66

[94] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)

[95] Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: Proceedings of the 26th Annual International Conference on Machine Learning. (2009) 41–48

[96] Mandai, Y., Kaneko, T.: Alternative multitask training for evaluation functions in game of Go. In: 2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI), IEEE (2018) 132–135

[97] Caruana, R.: Multitask learning. Machine Learning **28** (1997) 41–75

[98] Matsuzaki, K.: Empirical analysis of PUCT algorithm with evaluation functions of different quality. In: 2018 Conference on Technologies and Applications of Artificial Intelligence (TAAI), IEEE (2018) 142–147

[99] Thill, M., Bagheri, S., Koch, P., Konen, W.: Temporal difference learning with eligibility traces for the game Connect Four. In: 2014 IEEE Conference on Computational Intelligence and Games, IEEE (2014) 1–8

[100] Zhang, M., Wu, J., Li, F.: Design of evaluation-function for computer Gobang game system. Journal of Computer Applications **7** (2012) 051

[101] Emmerich, M.T., Deutz, A.H.: A tutorial on multiobjective optimization: fundamentals and evolutionary methods. Natural Computing **17** (2018) 585–609

[102] Justesen, N., Mahlmann, T., Risi, S., Togelius, J.: Playing multi-action adversarial games: Online evolutionary planning versus tree search. IEEE Transactions on Computational Intelligence and AI in Games (2017) 281–291

[103] Coulom, R.: Efficient selectivity and backup operators in Monte-Carlo Tree Search. In: International Conference on Computers and Games, Springer (2006) 72–83

[104] Ruijl, B., Vermaseren, J., Plaat, A., van den Herik, J.: Combining simulated annealing and Monte Carlo Tree Search for expression simplification. In: Proceedings of the 6th International Conference on Agents and Artificial Intelligence-Volume 1, SCITEPRESS-Science and Technology Publications, LDA (2014) 724–731

[105] Chaslot, G., Bakkes, S., Szita, I., Spronck, P.: Monte-Carlo Tree Search: A new framework for game AI. In: AIIDE. (2008)

[106] Guo, Y., Liu, Y., Oerlemans, A., Lao, S., Wu, S., Lew, M.S.: Deep learning for visual understanding: A review. Neurocomputing **187** (2016) 27–48

[107] Runarsson, T.P., Lucas, S.M.: Coevolution versus self-play temporal difference learning for acquiring position evaluation in small-board Go. IEEE Transactions on Evolutionary Computation **9** (2005) 628–640

[108] Wu, D.J.: Accelerating self-play learning in Go. arXiv preprint arXiv:1902.10565 (2019)

[109] Cazenave, T., Chen, Y.C., Chen, G.W., Chen, S.Y., Chiu, X.D., Dehos, J., Elsa, M., Gong, Q., Hu, H., Khalidov, V., et al.: Polygames: Improved zero learning. ICGA Journal (2020) 1–13

[110] Silver, A.: Fat Fritz 2: The best of both worlds. `https://en.chessbase.com/post/fat-fritz-2-best-of-both-worlds` (2021) Accessed June, 2021.

[111] Rosin, C.D.: Multi-armed bandits with episode context. Annals of Mathematics and Artificial Intelligence **61** (2011) 203–230

[112] Perez, D., Samothrakis, S., Lucas, S., Rohlfshagen, P.: Rolling horizon evolution versus tree search for navigation in single-player real-time games. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation. GECCO '13, New York, NY, USA, Association for Computing Machinery (2013) 351–358

[113] Liu, J., Liebana, D.P., Lucas, S.M.: Rolling horizon coevolutionary planning for two-player video games. In: 2016 8th Computer Science and Electronic Engineering Conference, CEEC 2016, Colchester, UK, September 28-30, 2016, IEEE (2016) 174–179

[114] Gaina, R.D., Couëtoux, A., Soemers, D.J.N.J., Winands, M.H.M., Vodopivec, T., Kirchgeßner, F., Liu, J., Lucas, S.M., Pérez-Liébana, D.: The 2016 two-player GVGAI Competition. IEEE Transactions on Games **10** (2018) 209–220

[115] Gaina, R.D., Devlin, S., Lucas, S.M., Perez, D.: Rolling horizon evolutionary algorithms for general video game playing. IEEE Transactions on Games (2021)

[116] Boyer, C.: Morpion Solitaire. `http://www.morpionsolitaire.com/` (2020) Accessed May, 2020.

[117] Ma, Q., Ge, S., He, D., Thaker, D., Drori, I.: Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning. arXiv preprint arXiv:1911.04936 (2019)

[118] Zhang, W., Dietterich, T.G.: Solving combinatorial optimization tasks by reinforcement learning: A general methodology applied to resource-constrained scheduling. Journal of Artificial Intelligence Reseach **1** (2000) 1–38

[119] James, J., Yu, W., Gu, J.: Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. IEEE Transactions on Intelligent Transportation Systems **20** (2019) 3806–3817

[120] Rego, C., Gamboa, D., Glover, F., Osterman, C.: Traveling salesman problem heuristics: Leading methods, implementations and latest advances. European Journal of Operational Research **211** (2011) 427–441

[121] Hu, H., Duan, L., Zhang, X., Xu, Y., Wei, J.: A multi-task selected learning approach for solving new type 3D bin packing problem. arXiv preprint arXiv:1804.06896 (2018)

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

[123] Wang, H., Schwab, I., Emmerich, M.: Comparing knowledge representation forms in empirical model building. INTELLI 2015 (2015) 184

[124] Moerland, T.M., Broekens, J., Plaat, A., Jonker, C.M.: A0C: AlphaZero in continuous action space. arXiv preprint arXiv:1805.09613 (2018)

[125] Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks. In: Advances in Neural Information Processing Systems. (2015) 2692–2700

[126] Bello, I., Pham, H., Le, Q.V., Norouzi, M., Bengio, S.: Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940 (2016)

[127] Feng, D., Gomes, C.P., Selman, B.: Solving hard AI planning instances using curriculum-driven deep reinforcement learning. arXiv preprint arXiv:2006.02689 (2020)

[128] Cazenave, T.: Nested Monte-Carlo Search. In: Twenty-First International Joint Conference on Artificial Intelligence. (2009)

[129] Cazenave, T., Teytaud, F.: Beam nested rollout policy adaptation. In: ECAI. (2012)

[130] Vilalta, R., Drissi, Y.: A perspective view and survey of meta-learning. Artificial intelligence review **18** (2002) 77–95

[131] Coulom, R.: Bayesian Elo computation system. `https://github.com/ddugovic/BayesianElo` (2018) Accessed May, 2018.

[132] Edwards, S.J., et al.: Standard portable game notation specification and implementation guide. http://www.saremba.de/chessgml/standards/pgn/pgncomplete (1994)