“Papers with Code” for Othello

This document describes how to exactly reproduce the experiments in [Scheier2022]

[Scheier2022] Scheiermann, Johannes; Konen, Wolfgang: „AlphaZero-Inspired Game Learning: Faster Training by Using MCTS Only at Test Time”, IEEE Transactions on Games, 2022, doi:10.1109/TG.2022.3206733, preprint available at https://arxiv.org/abs/2204.13307.
Local directory: C:\user\wolfgang\www\Optimierung\AlphaZeroOthello\

Preliminaries

Download GBG from https://github.com/WolfgangKonen/GBG, start or compile it according to https://github.com/WolfgangKonen/GBG/wiki/Install-and-Configure

If not present, build `jartools/GBGBatch.jar`.

Syntax:

```
GBGBatch gameName n agentFile [ nruns maxGameNum csvFile scaPar0 scaPar1
scaPar2 propsName ]
```

propsName codes the properties file name. If not given, it defaults to `props_batch.txt`.

For more information on GBGBatch, see the extensive Javadoc in GBGBatch.java.

The following files are included for reference in this papers-with-code distribution, but they should be normally present within the indicated directories in the actual GBG distribution from GitHub as well:

- experiments/Othello/experim*.sh
- experiments/RubiksCube/experim*.sh
- src/starters/props_batch.txt
- src/starters/GBGBatch.java
- resources/ R_plotTools/*.R

To run the visualization scripts in R_plotTools/, you need to have R ≥ 4.0 installed.

Experiment of Sec. IV.A (Fig. 4)

In Sections IV.A of [Scheier2022] we run the experiments without MCTS wrapping during training. Results are shown in Fig. 4.

The experiments start from so-called *stubs*. These stubs are agents which have all the parameters specified but are not yet trained. These stubs are part of the GBG distribution and found in the relevant subdirectories of GBG/agents/Othello/ with filenames ending in *-stub.agt.zip*.

Overview

The experiment is laid down in experim_Fig04.sh, which is started on UNIX machines from GBG/ via

```
screen ./experiments/Othello/experim_Fig04.sh
```

Step 1: We produce from a given stub 20 trained agents in `$train_out_dir`.

- GBGBatch with n=5 → batch05 → 20 trained agents

Step 2: We evaluate each of the 20 trained agents by evaluating them against Edax, levels 1, …, 9. Additionally, we let an MCTS agent compete 20 times against Edax.

- GBGBatch with n=6 → batch06 → evaluate all agents in `$train_out_dir`
- GBGBatch with n=7 → batch07 → evaluate MCTS 20 times.

Details

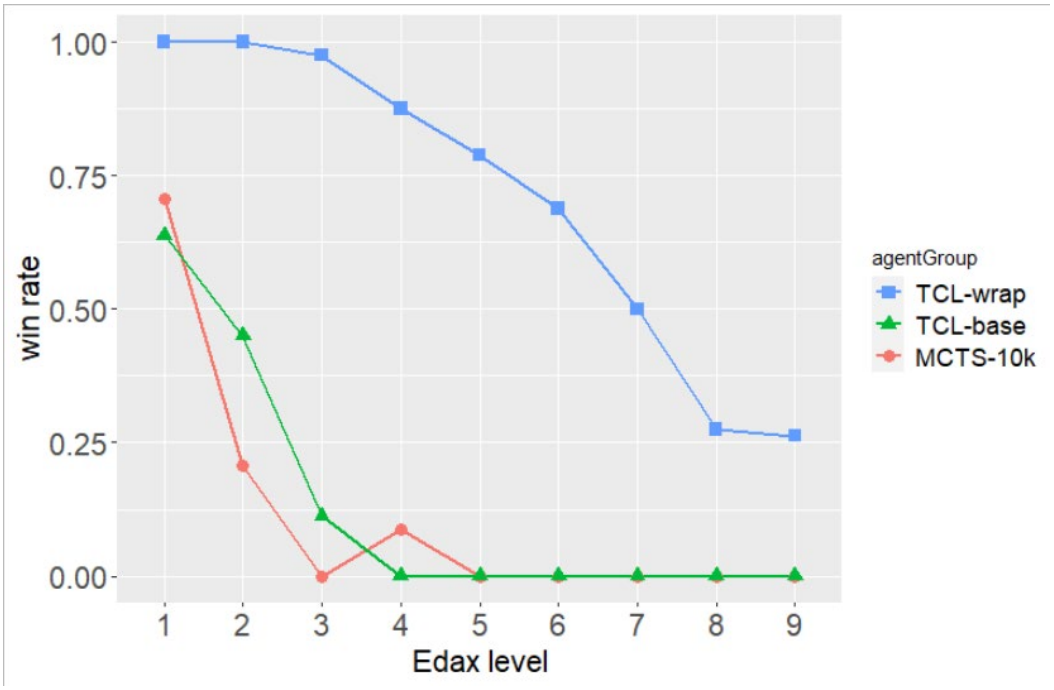
Details Step 1:

- The stub is *TCL4-100_7_250k-lam05_P4_H010-FAm-stub.agt.zip*.
- batch05 calls MCompeteSweep. multiTrainSweepOthello
- The optional property batchSizeArr, which may be set in the properties file `props_batch.txt`, should be commented out for this experiment.
- Results are the trained agents in `$train_out_dir`.
- `$train_out_dir` should be empty or non-existing prior to training (!)

Details Step 2:

- batch06 calls MCompeteSweep.multiCompeteSweepOthello. multiCompeteSweepOthello calls SingleCompetiro.doSingleCompetition which loops over {0, iterMWrap} and depthArr (the array of Edax depth levels).
- The property depthArr in the properties file `props_batch.txt` should be commented out or set to 1 2 3 4 5 6 7 8 9.
- Results are the csv-files as specified in experim_Fig04.sh, located in directory agents/Othello/csv/.

The results can be visualized with multiTrainOthello-20batch.R, which generates a plot resembling Fig. 4 in [Scheier2022]:



The parallel-version script experim_Fig04_parallel.sh distributes the training runs on 5 cores, see Computation Times.

Note that – if training is run on a Unix machine – the evaluation step should be commented out. The trained agents should be transferred to a Windows machine and only there the evaluation step, which requires the execution of executable edax.exe, can be carried out.

Experiment of Sec. IV.E (Fig. 10)

In Sections IV.E of [Scheier2022] we run the experiments with 100 MCTS iterations during training. Results are shown in Fig. 10.

The experiments start from so-called *stubs*. These stubs are agents which have all the parameters specified but are not yet trained. These stubs are part of the GBG distribution and found in subdirectory GBG/agents/Othello/ with filenames ending in *-stub.agt.zip*.

Overview

The experiment is laid down in experim_Fig10.sh (or experim_Fig10_parallel.sh), which is started on UNIX machines from GBG/ via

```
screen ./experiments/Othello/experim_Fig10.sh
```

Step 1: We produce from a given stub 20 trained agents in `$train_out_dir`.

- GBGBatch with n=5 → batch05 → 20 trained agents

Step 2: We evaluate each of the 20 trained agents by evaluating them against Edax, levels 1, …, 9.

- GBGBatch with n=6 → batch06 → evaluate all agents in `$train_out_dir`

Details

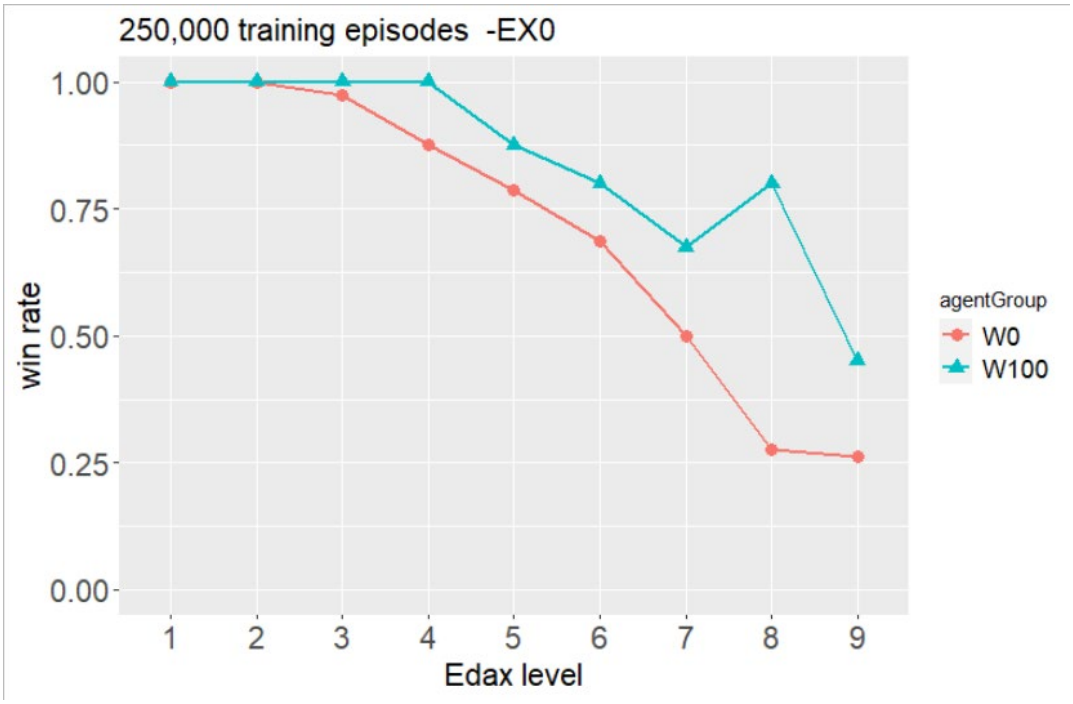
Details Step 1:

- The stub is *TCL4-100_7_250k-lam05_P4_H010-FAm-stub.agt.zip*.
- batch05 calls MCompeteSweep. multiTrainSweepOthello
- The optional property batchSizeArr, which may be set in the properties file `props_batch.txt`, should be commented out for this experiment.
- Results are the trained agents in `$train_out_dir`.
- `$train_out_dir` should be empty or non-existing prior to training (!)

Details Step 2:

- batch06 calls MCompeteSweep.multiCompeteSweepOthello. multiCompeteSweepOthello calls SingleCompetiro.doSingleCompetition which loops over {0, iterMWrap} and depthArr (the array of Edax depth levels).
- The property depthArr in the properties file `props_batch.txt` should be commented out or set to 1 2 3 4 5 6 7 8 9.
- Results are in csv-file `$csv_out_file` as specified in experim_Fig10.sh, located in directory agents/Othello/csv/.

The results can be visualized with multiTrainOthello-Fig10.R, which generates plots resembling Fig. 10 in [Scheier2022]:



W100 is clearly better than W0, making the competition against Edax 8 a win for W100. However, it needs 60x more training time

The parallel-version script experim_Fig10_parallel.sh distributes the training runs on 10 cores, see Computation Times.

Note that – if training is run on a Unix machine – the evaluation step should be commented out. The trained agents should be transferred to a Windows machine and only there the evaluation step, which requires the execution of executable edax.exe, can be carried out.

Experiment H001

This is a new experiment, it is not part of [Scheier2022].

Overview

The experiment is laid down in experim_H001.sh, which is started on UNIX machines from GBG/ via

```
screen ./experiments/Othello/experim_H001.sh
```

Step 1: We produce from a given stub 5 trained agents in `$train_out_dir_H001` and from another stub 5 trained agents in `$train_out_dir_H010`.

- GBGBatch with n=5 → batch05 → 5 +5 trained agents

Step 2: We evaluate each of the 10 trained agents by evaluating them against Edax, levels 1, …, 9.

- GBGBatch with n=6 → batch06 → evaluate all agents in `$train_out_dir_H001` and `$train_out_dir_H010`

Details

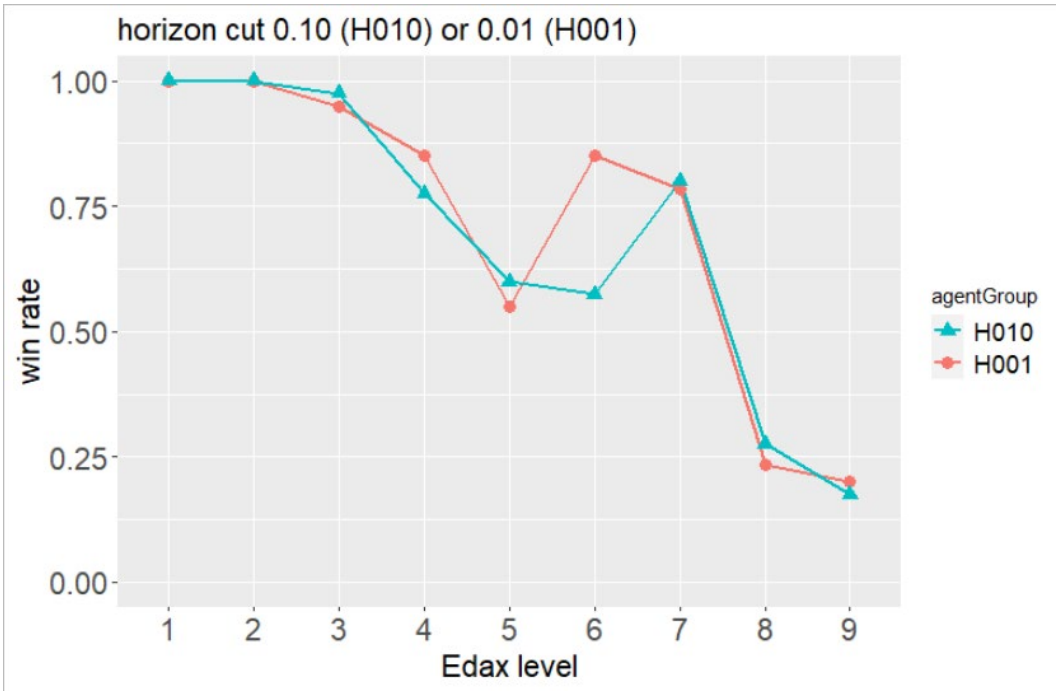
Details Step 1:

- The stubs are *TCL4-100_7_250k-lam05_P4_H010-FAm-stub.agt.zip*. and *TCL4-100_7_250k-lam05_P4_H001-FAm-stub.agt.zip*.
- batch05 calls MCompeteSweep. multiTrainSweepOthello
- The optional property batchSizeArr, which may be set in the properties file `props_batch.txt`, should be commented out for this experiment.
- Results are the trained agents in `$train_out_dir_H001` and `$train_out_dir_H010`.
- These directories should be empty or non-existing prior to training (!)

Details Step 2:

- batch06 calls MCompeteSweep.multiCompeteSweepOthello. multiCompeteSweepOthello calls SingleCompetiro.doSingleCompetition which loops over {0, iterMWrap} and depthArr (the array of Edax depth levels).
- The property depthArr in the properties file `props_batch.txt` should be commented out or set to 1 2 3 4 5 6 7 8 9.
- Results are in csv-file `$csv_out_file_H001` and `$csv_out_file_H010`, as specified in experim_H001.sh, located in directory agents/Othello/csv/.

The results can be visualized with multiTrainOthello-H001.R, which generates this plot:



It seems that H001 and H010 are not statistically different. Therefore, it makes sense to stay with H010 (which was also in use in [Scheier2022], because with 2h training time it has only two thirds of the training time for H001.

The parallel-version script experim_H001_parallel.sh distributes the training runs on 10 cores, see Computation Times.

Note that – if training is run on a Unix machine – the evaluation step should be commented out. The trained agents should be transferred to a Windows machine and only there the evaluation step, which requires the execution of executable edax.exe, can be carried out.

Computation Times

Jan'2023: Training the W100-H010 agent of experim_Fig10.sh for 100 training episodes is measured on laptop CPU to take 160s → estimated training time for 250.000 episodes is 400.000s = 111h = 4.5d. Unclear why it is higher than the 2.5d that we reported for the Fig10-experiment in [Scheier2022].

The standalone CPU is one core of lwivs48 VM (for training) and one core of Intel i7@2.6GHz on laptop (for evaluation)

- experim_Fig04.sh estimated training time 40h, evaluation time **5h** (4h for TCL agents, 1h for MCTS agents). With parallelization on 5 cores, training time is cut down to **8h** (2h for each agent training) with the help of experim_Fig04_parallel.sh.
- experim_Fig10.sh estimated training time 20 x 4.5d = 90 days, which is way too long. experim_Fig10_parallel.sh has a training time of 4.5 days for one agent in each job → **9 days** for the requested two agents per job. Evaluation time: **4h** for 20 TCL agents.
- experim_H001.sh training time 25h (2h for each H010 agent, 3h for each H001 agent), evaluation time **2h**. With parallelization on 10 cores, training time is cut down to **3h** (slowest agent) with the help of experim_H001_parallel.sh.