TDNTupleAgt.docx TODO.docx notes-WK-RubiksCube.docx

FUTURE WORK FOR GAME LEARNING – WK/10/2020

The following describes ideas for student projects and student theses in the context of GBG, see https://github.com/WolfgangKonen/GBG and https://blogs.gm.fh-koeln.de/ciop/research/gbg-general-board-games/. It is an internal document.

OPEN PROJECTS

Note that the sections *New Agents* and *New Games* contain **several** open projects in their lists. Some of them are further worked out in the sections below.

NEW AGENTS

- **GBG-based agents for Ludii**
  - Reinforcement-Learning GBG agents for Ludii, game Othello *(at the moment assigned)*
  - GBG agents for Ludii and other games
  - Perfect- or strong-playing agents for Ludii
- **AlphaGo-Zero-inspired GBG agent** *(at the moment assigned)*
- **CNN-like agent**
- **Java-Python bridge for DL agents** (DL: deep learning)
- Language-independent agent: An agent like in microRTS, “Creating non-Java AIs”, which can be in any non-Java language and corresponds with a server via socket connections
- Agent Balancing or **Game Balancing through Tournament Selection**.
- **Replay Experience Buffer** (a buffer which collects data from the latest episodes to have good data and enough data for training).

NEW GAMES

Games of special interest for GBG:

- **Investigations on the learnability of Sim in GBG**
- **Investigations on the learnability of Nim3P in GBG**
- **EWN, EinStein würfelt nicht!**
- **Rubik’s Cube**
- **Othello vs. Edax**
- Shut The Box, https://en.wikipedia.org/wiki/Shut_the_Box, a 2-, 3-, …, N-player nondeterministic dice game. Drawback: There is perhaps not much strategy to learn.
- **Poker**, https://de.wikipedia.org/wiki/Poker, in der Texas Hold’em-Variante – because it can be played with 2 to up to 10 or 14 players and is quite different from other board games. See also recent DeepStack-arXiv-publication. Drawback: quite complex rules and needs long tournaments to determine winner. *(at the moment assigned)*
- **Blackjack** – similar to Poker, but a bit simpler
- Are there other interesting N-player games with N>2? The problem is that some games with N>2 do not have any longer a true winning strategy, it is not possible to say: “This is the best move in that state”.

A baseline aspect for gaming:

- Client-server Framework for GBG

Further 2-person games (if students are interested):

- Mühle, https://de.wikipedia.org/wiki/Mühle_(Spiel)
- Checkers (Dame), https://de.wikipedia.org/wiki/Dame_(Spiel)
- Qubic (3D-TicTacToe)
- Abalone
- LoA (Lines of Action)
- Yavalath (the first computer-generated game)
- … and perhaps other interesting games from Ludii.

GBG-BASED AGENTS FOR LUDII

REINFORCMENT-LEARNING GBG AGENTS FOR LUDII

How strong are GBG agents compared to other general-game-playing agents?

Ludii (https://ludii.games/) is an interesting game platform, developed by Cameron Brown and his team, with a large variety of games and with a number of different tree-based agents (MCTS, Alpha-Beta, no pre-training). It offers the possibility to play with a user-supplied JAR-agent against the Ludii agents. Ludii does not offer the possibility to train **reinforcement learning (RL) agents**. It would be interesting to see, how RL agents for Connect Four or Othello, that were trained in GBG and packed into a JAR, compare with the built-in Ludii agents on the Ludii games Connect Four or Othello. The central research question is: Are general-purpose RL agents (with pre-training) stronger than general-game-playing agents (with no pre-training)?

Work steps:

- Start with a simple Java GBG agent (Random or MCTS) and wrap it into a Ludii JAR in order to get familiar with the Ludii interface.
- Build RL Java agents according to the Ludii interface. "Inside" they should host the proper GBG agents. Possibilities: TD n-tuple or Sarsa n-tuple (both pre-trained in GBG). Needs a translation of GBG states and actions to Ludii states and actions.
- Make a competition of these agents against the various Ludii agents on the specific games Connect Four, Othello, Nim.

Prior knowledge:

- Java
- Knowledge in RL helps, but can be also acquired on-the-job

PERFECT- OR STRONG-PLAYING AGENTS FOR LUDII

For some games there exist perfect- or strong-playing agents (programmed, not self-learned). Some of them are included in GBG and used for evaluation purposes against the self-learning agents. The perfect- or strong-playing agents are:

- Nim: Bouton's agent
- Connect Four: Alpha-Beta agent with opening book
- Othello: Edax (https://github.com/abulmo/edax-reversi by Richard Delorme)

Ludii (https://ludii.games/) is an interesting game platform with many games and with a number of different tree-based agents (MCTS, Alpha-Beta, no pre-training). It offers the possibility to play with a user-supplied JAR-agent against the Ludii agents. Ludii does not offer the possibility to play against perfect- or strong-playing game-specific agents. It would be interesting to see, how the above agents compare with the built-in Ludii agents on the Ludii games Nim, Connect Four and Othello. The central research question is: How far from perfect or strong playing (in the context of certain games) are the general-game-playing agents in Ludii ?

Work steps:

- Start with a simple Java GBG agent (Random or MCTS) and wrap it into a Ludii JAR in order to get familiar with the Ludii interface.
- Build Java agents according to the Ludii interface. “Inside” they should host the proper GBG agents: Nim Bouton, Connect Four Alpha-Beta, Othello Edax. Needs a translation of GBG states and actions to Ludii states and actions.
- Make a competition of these agents against the various Ludii agents on the specific games Nim, Connect Four or Othello
- Nim offers a special option for research: It is a scalable game (the number and the size of the heaps can be varied), being probably more complex for self-trained agents in the case of larger heaps. Measure the performance of Ludii agents as a function of number and/or size of heaps.

Prior knowledge:

- Java

ALPHAGO-ZERO-INSPIRED GBG AGENT

AlphaGo Zero (Silver et al, 2017, *Nature*) (local copy) is probably the world’s strongest Go player and also capable of learning other complex games. It is a clever combination of MCTS (Monte Carlo Tree Search) and DNN (Deep Neural Networks) which learns *tabula rasa* (from zero) and solely from self-play.

GBG is a General Board Game playing and learning framework which includes also agents learning solely from self-play. While the complex CNN-based deep neural networks in AlphaGo Zero are beyond reach for simple hardware, the combination “MCTS and neural network” can be carried over to GBG. This might lead already to big improvements when playing games where a certain thinking time is available.

An interesting point to research: Can such an agent beat the strong-playing Othello agent Edax on some higher Edax levels? Currently, the best TD agents are on par with Edax level-1, but they will consistently lose against Edax level-2 and higher, where 21 (!) Edax levels are available.

Interesting side project: There exists a very good tutorial on AlphaGo Zero:

- https://web.stanford.edu/~surag/posts/alphazero.html : Surag Nair, 2017, with pseudo code.
- https://github.com/suragnair/alpha-zero-general: A GitHub with full implementation in Python.

Work steps:

- Understand GBG’s TD n-tuple agent
- Build an AlphaGo-Zero-inspired agent, where MCTS is coupled with a (pre-trained) TD n-tuple network (instead of a DNN).
- Evaluate the new agent with various parameters on various games and compare it with plain TD n-tuple agent and with n-ply wrapped TD n-tuple agent. Evaluation criteria: computation time, win rate.
- Make a special evaluation {Othello, Edax}.
- Optional: Integrate and test the idea from [Silver2017] that the search tree of MCTS is re-used in subsequent time steps of a game episode (re-using the subtree from the relevant child node which becomes the new root node).

Prior knowledge:

- Java
- MCTS and RL knowledge helps, but can be also acquired on-the-job

CNN-LIKE AGENT

GBG has seen a number of successes: The TD n-tuple agent works very well on smaller boards in various games (2048, Connect Four, Nim, smaller Hex boards) But TD n-tuple reaches a barrier when it comes to larger boards (Hex boards 7x7 and higher, Othello). This is probably because the number of weights needed for larger boards grows exponentially.

A weight-saving variant would be to use the main idea from **CNN (Convolutional Neural Networks)**: Define weight-sharing filters or weight-sharing n-tuples, which have trainable weights but share them with filters placed at another position on the board.

Work steps:

- Elaborate on the concept which is only sketched here.
- Research similar literature
- Option 1: Implement CNNs in Java: The library DeepLearning4J may be something to look at.
- Option 2: Work with a Java-Python bridge and use the possibility to build flexible CNN architectures from TensorFlow, Keras or similar.
- Test the agents on Hex 7x7 and higher or on Othello

Prior knowledge:

- Java [, Python, TensorFLow, Keras]
- CNN and neural networks in general

JAVA-PYTHON BRIDGE FOR DL AGENTS

In the current GBG framework, the neural nets are shallow or very simple networks. If we want to use deep-learning (DL) techniques (either deep neural networks, CNNs or other), it might be desirable to do the time-consuming training with the help of TensorFlow/Keras in Python. This would require a bridge between Java and Python.

Work steps:

- Design one (or perhaps several alternative) concept(s) for Java-Python bridge: Which tasks to hand over, with which granularity?
- First a concept for one specific game, but later with the possibility to work for arbitrary games
- Implement these concepts in GBG
- Needs probably a Replay Experience Buffer to hand over the data for training
- Test them on various games, measure speed, evaluate quality.

Prior knowledge:

- Java, Python
- DL and DNN (CNN as an option) in TensorFlow and Keras

CLIENT-SERVER FRAMEWORK FOR GBG

Currently, GBG is a stand-alone program. Several aspects would be better in a client-server architecture

- A server offers a set of game-competitions and user may connect to such competitions and let their AI agents play from a remote client
- A server offers a set of games to be played, a set of pre-trained agents. The user may play from a client against a selected agent in a selected game. Server may record the games played → tournament, analysis, hall of fame, ….

This leads to the following goals for a thesis project *Client-Server Framework for GBG*:

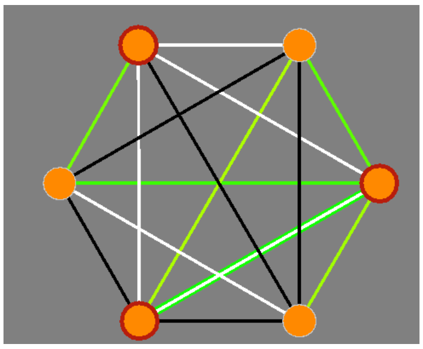
- Make a concept for such a client-server framework. What parts on client, what on server?
- Make a migration path from the current GBG software to the client-server-GBG. Are serious changes in GBG software needed? Can we keep both versions, standalone and CS in one program or do we have to fork?
- Implement and test CS framework
- A starting point can be the Sim(Hexi) CS implementation: https://www.dbai.tuwien.ac.at/proj/ramsey/source.htm

INVESTIGATIONS ON THE LEARNABILITY OF SIM IN GBG

Sim (https://en.wikipedia.org/wiki/Sim_(pencil_game) or https://de.wikipedia.org/wiki/Sim_(Spiel)) is an interesting game for several reasons:

(a) It is simple to learn,
(b) it has quite a complex winning strategy,
(c) it is a scalable game (# of nodes),
(d) it has a large number of symmetries (in fact, exponentially growing with # of nodes),
(e) it can be played with 2 or 3 players (3 separate players or 1-vs-2 variant).

Sim is now available in GBG, but topics around the **learnability of Sim** do need further research:



- Which agents with which parameters can learn to play best: in 6-, 7-, …, 17-nodes Sim and with 2 or 3 players (17 nodes would be the first game without a tie in the 3-player variant).
- Symmetries normally make a game simpler to learn. But Sim has so many symmetries (720 for 6-node Sim, and exponentially rising with higher node numbers) that the normal strategy for symmetries: "Take all of them" does not work any longer. What are better strategies?
- Is there a lower node number than 17 where the 1-vs-2 variant has a clear winner?

INVESTIGATIONS ON THE LEARNABILITY OF NIM3P IN GBG

Nim3P is now available in GBG. Nim3P is a **3-player variant** of the well-known game Nim.
Nim (https://de.wikipedia.org/wiki/Nim-Spiel, https://en.wikipedia.org/wiki/Nim) is an interesting game for several reasons:

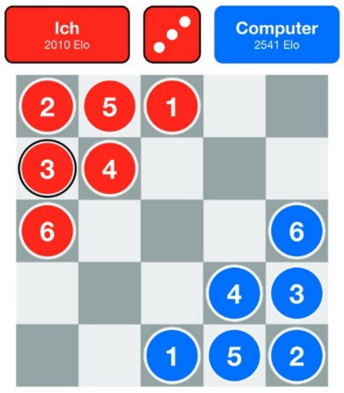
(a) it is a scalable game (# of heaps, size of heaps),
(b) it is simple to learn, if # and size of heaps is small, but more difficult to learn, if they are not,
(c) it has a large number of symmetries (in fact, exponentially growing with # of heaps),
(d) it can be played with 2 players (normal Nim) or 3 players (Nim3P).

Nim3P is briefly covered in the recent publication [Konen2020b] of our research group.
It is not yet fully researched which variants and reward schemes of Nim3P are meaningful such that such a 3-player Nim has a clear winning strategy.
Topics around the **learnability of Nim3P** need further research:

- What variants and reward schemes of Nim3P allow for a clear winning strategy?
- Which agents from GBG can learn to play which scalable Nim3P games well? With which agent-specific parameters?

- Symmetries normally make a game simpler to learn. But Nim and Nim3P has so many symmetries (n! for a game with n heaps) that the normal strategy for symmetries: “Take all of them” does not necessarily work any longer. What are better strategies?

EWN (EINSTEIN WÜRFELT NICHT)



EWN, EinStein würfelt nicht!
(https://de.wikipedia.org/wiki/EinStein_w%C3%BCrfelt_nicht,[1]): EWN is a quite interesting game, not so easy, not too complex, non-deterministic, but nevertheless strategic. It is said that there exist 3- and 4-player-variants with teams (needs to be researched).

Work steps:

- Implement the basic 2-player version of EWN in GBG
- Research literature on 3- and 4-player variants
- Test various agents: Which agent with which parameters works best on EWN?
- Optional: Implement and test 3- and 4-player variants

RUBIK’S CUBE

Rubik’s cube (https://de.wikipedia.org/wiki/Zauberw%C3%BCrfel) is so famous that it probably does not need to be introduced. The state space complexity 4.3*10^{19} is enormous.

Can a computer program **learn** to solve Rubik’s cube? – Ideas might be time-reverse reinforcement learning, exploitation of color symmetries. Relatively open and complex task. There exist some recent research papers ([McAleer2018] = https://arxiv.org/pdf/1805.07470.pdf, [Lapan2019]) which claim that they have solved it, but the paper is not very clear about all the details. It might be an interesting task to check if their algorithm is understandable and to see if one can reproduce their results in GBG.

Even more interesting is the paper [Agostinelli2019], which has solved various cube aspects.

A first step might be to start with a 2x2x2 cube (which is already complex enough). A beta version of 2x2x2 cube is available, but it has not yet learned to solve largely disordered cubes.

Interesting side project: Integrate a nice **cube visualization** into the project. A good starting point may be **CubeTwister** from http://www.randelshofer.ch/, which offers Java and JavaScript code for various cubes.

Work steps:

- Understand the existing cube implementation in GBG.
- Understand the representation idea of [McAleer2018] [Agostinelli2019] (the 20*24 array) and translate this idea to the 2x2x2 cube.
- Make a concept whether the algorithm [McAleer2018] [Agostinelli2019] can be implemented in GBG.
- If so, implement it for the 2x2x2 cube.
- Optional: Extend to 3x3x3 cube.

[1] See also http://computerschach.de/Files/2005/EinSteinen%20in%20Jenas%20langer%20Nacht.pdf: nice article with background info on EWN and commented game episode – in German.

- Optional: Integrate a nice 3D cube visualization (e.g. from http://www.randelshofer.ch/).
- Evaluate various agents (and representations): How good are they in solving disordered cubes?

Current status:

- The 2x2x2 cube is now basically solved in GBG, but the 3x3x3 cube is still open. And so is the 3D cube visualization.

OTHELLO VS. EDAX

Othello, https://en.wikipedia.org/wiki/Reversi, a medium-complexity game, is interesting, because it is still unsolved.

Othello is now available in GBG. It comes with a very strong playing Othello-specific agent Edax (https://github.com/abulmo/edax-reversi by Richard Delorme). Although Othello is available, there is need for further research. All Edax-levels above level 1 are very tough to beat. Currently, none of the generic agents (trainable RL-agents or MCTS agents) can beat Edax beyond level 1. Can we improve on that? Can we design an agent trainable by self-play which reliably beats Edax at some of the higher levels 2, 3, 4, …,15? – Might be tackled via **AlphaGo-Zero-inspired GBG agent**.

GAME BALANCING THROUGH TOURNAMENT SELECTION

For some games, the available agents are way too strong against humans. To have a better play experience as a human, a game balancing would be desirable.

Based on the existing Competition Framework for GBG with Elo- and Glicko-rating:

- Generate for game X (many) different agents: select agent types, choose different parametrizations, different wrap levels
- Sort the agents by a tournament on game X to establish the Elo/Glicko rating of all agents.
- This should allow a suitable game balancing (to be worked out)

CLOSED PROJECTS

COMPETITION FRAMEWORK FOR GBG

This project has already been done (Master thesis Felix Barsnick 2019), but **perhaps extensions are possible in one way or the other**. The Tournament System has for example not yet been tested thoroughly on different games.

From ToG-GBG.pdf:

> *A game **competition** is a contest between agents. There are multiple objectives which play a role in such a contest:*
> *a) ability to win a game episode or a set of game episodes, maybe from different start positions or against different opponents (results clearly depend on the nature of the other agents);*
> *b) ability to win in several games;*
> *c) time needed during game play (either time per move or budget per episode or per tournament);*
> *d) time needed for setting up the agent (training), and possibly other objectives in training.*
> *Other objectives are possible as well.*

Competition objectives may be combined, i. e. how is the agent's ability to win if there is a constraint on the play time budget. When running this with different budgets, a curve 'win-rate vs. time budget' can be obtained.

Methods from multi-objective optimization (e. g. Pareto dominance) can be used to inspect and visualize competition results.

This leads to the following goals for a thesis project *Competition Framework for GBG* or *Tournament System*: [DONE]

- Screen existing literature on game competitions
- Make a concept for a competition framework in GBG
- In the case of 1-player games, set up a competition with this rule: Each agent plays one or many episodes alone and the agent achieving the highest overall score wins.
- Round-robin tournaments:
    - selection of agents → GUI
    - result reporting: rank tables vs. score tables
- Include the time aspects:
    - integrate time measurement of agents
    - design a win-rate as a function of time constraints curve
- Multi-objective: Pareto plots: Win-rate vs. time
- Visualization of game-play during competitions
- Elo- and Glicko-rating
- Analyze the competition rules and competition results for existing games and agents: Are the competitions 'fair'? Are there signs for non-transitivity? If so, how to establish a fair winner then?

REALIZED GAMES

- ConnectFour [Thill14, Bagh14], which contains a perfect-playing AlphaBeta agent and a strong-playing TD-n-tuple agent, was ported to GBG [DONE]
- Sim, https://en.wikipedia.org/wiki/Sim_(pencil_game) (2-player variant, 3-player variant), an interesting game for N≥2 players. [DONE]. See also Investigations on Sim in GBG.
- Nim (2 players), Nim3P (3 players).
- Othello (Reversi) [INF-Project Dittmar & Cöln] [DONE]. See also Othello vs. Edax and AlphaGo-Zero-inspired GBG Agent.
- Hex [BA Kevin Galitzki] [DONE]. See also CNN-like Agent.
- 2048 [BA Johannes Kutsch] [DONE].
- TicTacToe

LITERATURE

[Lapan2019] Lapan M.: "Reinforcement Learning to solve Rubik's cube" (2019) https://medium.com/datadriveninvestor/reinforcement-learning-to-solve-rubiks-cube-and-other-complex-problems-106424cf26ff (local copy)

[McAleer2018] McAleer S., Agostinelli F. et al.: "Solving the Rubik's Cube Without Human Knowledge", *arXiv preprint arXiv:1805.07470* (2018) (https://arxiv.org/abs/1805.07470, local copy)

[Konen2020b] W. Konen, S. Bagheri: „Reinforcement Learning for N-Player Games: The Importance of Final Adaptation", accepted for BIOMA'2020, Brussels, 2020. (preprint available here: http://www.gm.fh-koeln.de/ciopwebpub/Konen20b.d/bioma20-TDNTuple.pdf or here: somewhat longer technical report)