TODOs for GBG-Framework – WK/05/2017

From TR-GBG.pdf (to be deleted there):

* The n-tuple agent developed for C4 (Connect Four) needs to be ported to GBG.
* Allow only trained agents to be saved.
* Clarify: Is the parameter data flow safe, if we issue a ’play’ or ’compete’

for 2 agents of same type but with different parameters?

Solved items:

* OK Bug fix: The current LoadAgent version ‘hangs’ if an agent with incompatible serialID is read (TTT agents TDS & TD-NTuple).
* OK Bug fix in TDAgent.java: We needed to replace the hard-coded   
  private int inpSize[] = { 6, 6, 10, 19, 13, 19, 0, 0, 0, 9 };  
  by the flexible m\_feature.getInputSize(m\_feature.getFeatmode()).
* OK TD-pars tab: Constrain ‘Feature set’ (= value for featmode) to the set of allowed featmode values for the current game (m\_feature.getAvailFeatmode()) and set the initial value to a sensible value 🡪 KG fixed this and generated a pull request
* OK: BUG1 + BUG2 fix for MCAgent: In some versions of Hex, the MC agent does not choose the best action, but **instead the worst** action (!!) This happens for (3x3, 1.), (4x4, 2.), (5,5, 1.), … and so on. The same behavior for TTT (3x3, 1.) – This was due to 2 errors: (1) in RandomSearch there could be actions on newSob even if newSob is already a game-over-state (BUG1) and (2) newSob.getGameScore has to be called with the referingState sob to get the sign right in every case (BUG2)
* OK Improved TR-GBG around game score and game value: Made it simpler!
  + Concentrate Chap. 3.3 on so.getGameScore() and so.getGameScore(sob). Explain the difference in pictures. Give examples showing explicit values for both functions.
  + Move pa.getGameScore(so) to Chap. 3.4 (Game Value), because it is a game value, not a game score (think about renaming it to pa.getGameValue).
  + Move the mind-buggling complicated so.getGameValue() and pa.estimateGameValue(so) to appendix.
* OK XArenaFuncs::multiCompete writes Arena.comp.csv now to agents/<gameName> (and not to root of GBG)
* OK XArenaFuncs::multiCompete has the evaluator mode hard-wired (9). This might fail for other games where the evaluator mode 9 is not present. Extend it that it fetches static Evaluator.getDefaultEvalMode().
* OK Why has XArenaFuncs::multiCompete the evaluator part excluded in case of agent MCTS? – This was due to a wrong (incomplete) implementation of MCTSAgentT::getScore (the part with nextActionScore, same as in MCAgent, was missing). -- Additionally, there was another bug in both MCAgent and MCTSAgentT: Both would have problems when getScore(sob) was called with an already game-over sob (as it regularly happens inside Evaluator9). Then MCTSAgentT resulted in a NullpointerException, since the tree of a root state with game-over condition is never expanded in treePolicy. The fix is to check in getScore(sob) for sob.isGameOver(). If true, return directly sob.getGameScore(sob). Now both bugs are fixed and MCTS is correctly evaluated in multiCompete.
* OK Improve Evaluator chapter in TR-GBG.
* OK Write for TR-GBG.pdf an appendix on N-tuples (with figures): Explain for what the several functions in XNTupleFuncs are needed.

Open items:

* Prepare TR-GBG.pdf for publication on CIplus server.
* Saving (serializing) of things (logs, agents) is still not safe, if something changes in the classes being serialized. Is it possible (with minimal effort) to read older versions of serialized objects as well?
* During TDNTupleAgt-training: Replace the Minimax-evaluation by proper general evaluation (replace JFreeChart plot title)
* Hex: extend to logging with subdirs 🡪 KG
* Types.ACTIONS: replace enum (which needs to be extended whenever we need more ACTIONS) by some class construct
* (OK) Think about the game logic for so.getGameScore(referingState) in 3-player games 🡪 see [notes\_MCTS.docx](file:///C:\WUTemp\FH-MassenDaten\svnSoma\trunk\doc\CaseStudies.d\201314.d\CIG2014\MCTS.literature\notes_MCTS.docx), Chap. “Getting The Score For n-Player Games”. – Still some thinking about intermediate game scores to do.
* If MCTS has several actions with the same value (all are a ‘Win’), then it does not take the shortest path to victory. Instead it takes a random among all winning moves. 🡪 KG will think about an add-on to value function which breaks ties in favor of a shorter path.
* An Evaluator object is currently constructed in several places (multiCompete, multiTrain, train, and taskState TRAIN, menu item ‘Quick Evaluation’ (fct evaluate)), often with different objects (different evaluator modes), and with modes tied to game TicTacToe. Make the modes as simple as possible, then generalize it in such a way that it is for arbitrary games. Perhaps with user-adjustable mode selectors in the ‘Other pars’ tab, perhaps with sensible defaults set in the files defining constants.
* Some elements of measurement in train, multiTrain, compete are still TicTacToe specific or at least 2-player specific. Generalize them to arbitrary 2-player games and later to arbitrary n-player games.
* Bugs in Hex (for KG):
  + When playing a human-agent game, then the scores of the agent are not shown during play (they are however shown when replaying it with the game log)
  + When replaying a human-agent game with the game log, then any ADVANCE will move two plies foreward (one agent, one human ply). When replaying an agent-agent game, it is correctly only one ply per ADVANCE.