A NOTE ON THE RECENT EMPIRICAL EVALUATION OF RBG AND LUDII

JAKUB KOWALSKI, MAKSYMILIAN MIKA, JAKUB SUTOWICZ, AND MAREK SZYKUŁA

Abstract. We present an experimental comparison of the efficiency of three General Game Playing systems in their current versions: Regular Boardgames (RBG 1.0), Ludii 0.3.0, and a Game Description Language (GDL) propnet. We show that in general, RBG is currently the fastest GGP system. For example, for chess, we demonstrate that RBG is about 37 times faster than Ludii, and Ludii is about 3 times slower than a GDL propnet. Referring to the recent comparison [*An Empirical Evaluation of Two General Game Systems: Ludii and RBG*, CoG 2019], we show evidences that the benchmark presented there contains a number of significant flaws that lead to wrong conclusions.

1. Introduction

In this note, we are referring to three General Game Playing (GGP) standards described in the literature: Regular Boardgames (RBG) [6] – a novel formalism describing via regular expressions deterministic perfect information games played on arbitrary graph boards; Ludii [10] – created mainly for modelling traditional strategy games and charting their historical development, a descendant of Ludi [1]; and the classic Stanford's GGP [3] based on Game Description Language (GDL) [9], which uses first-order logic to encode transition systems of all deterministic perfect information games in a very knowledge-free way. The most efficient implementations of GDL reasoners are based on Propositional Networks (propnets) [13].

The benchmark recently published in [11] contains a number of significant flaws skewing the results in favour of Ludii. Despite our requests, the authors did not share the Ludii version used for that experiment nor they turned into discussion about its correctness. Nevertheless, an analysis of the results and of the public Ludii versions provides strong evidence for the errors.

In this note, we show a fair benchmark, demonstrating that RBG in its first version is generally the fastest GGP system. Then, through a detailed analysis, we point out the errors of the previous benchmark [11], explaining the differences in the results. Finally, we describe how to independently reproduce our experiment.

The most important error is that the comparison was made between games with significantly different rules, i.e., Ludii under the same name had much simpler and easier for computing rules (e.g., omitting special cases, splitting complicated moves into multiple turns). An extreme example how the flaws affected the conclusions is the case of chess. The authors presented that Ludii is apparently 12,000 times faster than reasoning in GDL. However, a proper benchmark reveals that the current version of Ludii is 3 times slower than a GDL propnet for chess, apart from the existing bugs in the Ludii's chess rules.

Institute of Computer Science, University of Wrocław, Wrocław, Poland
E-mail addresses: jakub.kowalski@cs.uni.wroc.pl, mika.maksymilian@gmail.com, jakubsutowicz@gmail.com, msz@cs.uni.wroc.pl.

2. Our benchmark

Table 1 presents a benchmark, comparing the performance between the same games (with a few marked exceptions, with rather minor influence on the performance, explained later). By the same games we mean the games with isomorphic game trees induced by the rules. This assumption is required to ensure fairness when comparing efficiency of game reasoning for games encoded via different formalisms.

The computations were made on a single core of Intel(R) Core(TM) i7-4790 @3.60GHz of a computer with 16GB RAM. The gcc version was `gcc (Ubuntu 7.4.0-1ubuntu1 18.04.1) 7.4.0`, and the Java version was `Java(TM) SE Runtime Environment (build 12.0.2+10)`.

Each test of RBG compiler took at least 10 minutes, not counting preprocessing. Each GDL propnet result is the average time of 10 runs lasting 1 minute, not counting preprocessing (averaging is a proper practice here, because of non-deterministic propnet construction, cf. [13]). Each Ludii test was performed via the dedicated menu option. Although theoretically Ludii 0.3.0 has a benchmark option callable from command-line, it is left completely undocumented, hence, currently, we have treat the menu option as the only official possibility admitted by the authors.

In all cases, the old version RBG 1.0 of the reasoning engine was used. Some additional games were added in RBG 1.0.1 to match their Ludii equivalents (see Subsection 2.2).

Table 1. Efficiency of reasoning in RBG, Ludii, and GDL propnet reasoner for the **flat Monte Carlo** test. The values are the **numbers of playouts per second**.

Game	**RBG 1.0/1.0.1**	**Ludii 0.3.0**	**GDL propnet**
Amazons	569	*n/a*	4
Amazons-split	8,798	3,859	365
Arimaa	0.14*	*n/a*	*n/a*
Arimaa-split	666*	446†	*n/a*
Breakthrough	19,916	3,546	2,735
Chess-fifty move	523*	14†	45
Connect-4	190,171	63,427	45,894
English draughts-split	23,361*	7,111†	3,466
Gomoku-free style	2,430*	26,878	*n/a*
Hex	6,794	10,625	*n/a*
Reversi	8,682	1,312	373
Skirmish	3,989*	636†	237
The mill game	10,102*	2,467†	*n/a*
Tic-tac-toe	526,930	422,836	104,500

* This game code was not originally available in RBG 1.0 and was added later.
† The rules in Ludii differ from the others.

2.1. Games in Ludii 0.3.0.

Unfortunately, Ludii is a closed-source system, which limits debug possibilities for game rules. In multiple cases, the game rules provided with Ludii 0.3.0 differ

from the orthodox ones (official rules, which are encoded in RBG and GDL). Of course, in many cases these are just regular bugs little affecting the efficiency, so mostly harmless for an efficiency comparison. But there are also significant simplifications (as move splitting) and rule omissions, which make the computation easier.

We describe the details of the game rules occurring in both benchmarks, Table 1 and [11].

(1) Amazons:
- The only available version of amazons is with split movements. In the split version, moving a queen and firing an arrow is performed in two consecutive turns, which implies that the branching factor (number of legal moves) is much smaller; this property is crucial for MC-based tests, where in each state all legal moves are computed (see [5] for a comparison using MCTS). To illustrate this issue, we also present both RBG and GDL results for the orthodox (non-split) version.

(2) Arimaa:
- Ludii contains only a split version, where each step is performed in a separate turn. Additionally, finishing a full turn is a move itself, unless there are no more steps.
- The Ludii's version violates the constraint that a turn must make the position changed.
- The pull action is additionally split into two moves, in contrast with the push action, which is not.
- The traps always kill a piece immediately, regardless of the presence of a protecting neighbor. Because of the previous point, it is also not possible to pull a piece while going into a trap.
- The rabbit moves only forwards (does not walk horizontally).
- There is also no ending condition in the rules other than a win. Therefore, Ludii uses its (undocumented and unknown) internal limit.

(3) Chess:
- En-passant does not work correctly.
- There is a number of errors related to double pawn move (e.g. the skipped square is impassible by sliding pieces).
- Promotion is split into two moves.
- It implements the fifty-move rule, which disagrees with common GDL implementations with 200 turn limit.

(4) Chinese checkers:
- The version in Ludii is a split, where each jump is performed in a separate turn, and ending a sequence of jumps is a move itself.
- The above fact makes this game variant absurd from the playing perspective. Indeed, a player can force a draw at almost any moment by jumping over a single piece there and back, until the internal turn limit passes.
- There is no rule about non-leaving the goal area once reached.
- There is no rule about not having a legal move.

(5) Double chess (skirmish):
- It is similar to skirmish (see below). The difference is just the larger board and that promotion is allowed only to queen.

(6) English draughts:
- The version in Ludii is a split, where each single capture is performed as a separate move.

- Within one capture sequence (one full move), it is possible to jump with different pieces. However, a capturing sequence ends when the last used piece has no more captures. Therefore, for instance, if each of pieces A and B can perform two captures, one can jump with piece A, then jump with piece B, and then continue capturing with piece A.
- There is no orthodox ending condition. The game ends when a player has no legal move or the internal turn limit passes.

(7) International draughts:
- Similarly to English draughts, the version in Ludii is a split.
- It inherits the same bugs as English draughts.
- The rules that a player must capture the largest possible number of pieces, and the pieces are removed from the board only after the full capturing move, are omitted.
- Promotion occurs right after arriving at the last rank as in English draughts, in contrast to the rule that it should occur only when a sequence of captures ends there.

(8) Skirmish: The version of skirmish chess variant used in Ludii does not match any available GDL skirmish. The GDL/RBG embeds standard rules of chess except the king becomes a regular piece. The Ludii's version is simpler due to the following:
- There is no castling nor en-passant.
- Pawn promotion is split into two moves.
- Also, there is no turn limit in the game rules, thus the internal limit must be used.

(9) The mill game:
- This is a split version. Placing or moving a piece is performed in a separate turn than capturing.
- The Ludii version omits the rule that an opponent's piece cannot be captured if it is in a mill, unless all his pieces are in the mill.
- There is no flying phase when the player has only 3 pieces left.

The remaining compared games from 0.3.0 version: breakthrough, connect-4, gomoku-free style, hex, reversi, and tic-tac-toe, are encoded, according to our knowledge, without any errors.

2.2. **Games in Regular Boardgames 1.0.1.** Since the Ludii authors invented their own rules for many games, there were no equivalent versions available in RBG 1.0. However, we made an effort to encode some of these variants (not copying the bugs, however). They have been included to the benchmark for a rough comparison. Due to the corruption, uncertainty, and the lack of debug possibilities of the rules of games present in Ludii, an exact comparison for such games is impossible or would require too much effort.

The new games are a part of RBG 1.0.1 and these cases are marked with star in Table 1; the other game codes are exactly those from RBG 1.0. The names of the games from our comparison directly match the filenames in the RBG repository, i.e., in the order from Table 1: `amazons.rbg`, `amazons-split.rbg`, `arimaa-split.rbg`, `breakthrough.rbg`, `chess-fiftyMove`, `connect4.rbg`, `englishDraughts-split.rbg`, `gomoku-freeStyle.rbg`, `hex.rbg`, `reversi.rbg`, `skirmish.rbg`, `theMillGame-split.rbg`, `ticTacToe.rbg`.

2.3. **Games in GDL.** Whenever an equivalent GDL code is available, we also compared with the result from a GDL propnet (described in [13]).

We additionally modified the existing GDL code for chess (the fastest version, which originally implements 200 turn limit) to match with the fifty-move rule. Also, we fixed the GDL orthodox version of amazons for the terminal condition, which was wrong.

The GDL codes used in the experiment were: amazons – `amazons_fixed.kif` from [4], amazons-split – `amazons_10x10.kif` from [12], breakthrough – `breakthrough.kif` from [12], chess-fifty move – `chessFiftyMove` from [4], connect-4 – `connectFour.kif` from [12], English draughts-split – `englishDraughts.kif`, reversi – `reversi.kif` from [12], skirmish – `skirmishZeroSum.kif` from [12], tic-tac-toe – `ticTacToe.kif` from [12].

3. Comparison with the other benchmark

We provide an explanation why the results from [11] are much different than those in our proper experiment, leading to an opposite conclusion. We show that that benchmark contains significant flaws of different kind. The issues make an impression that the Ludii reasoning engine is generally faster than the others.

(1) The most significant issue is that most of the compared games do not have same rules in all the three GGP systems. Only 5 out of 14 games were correctly matched with their equivalents. The differences in the other games are significant, often simplifying the computation in favor for Ludii. We give a detailed analysis below.
(2) In two cases, connect-4 and reversi, the RBG result was miscalculated. According to our benchmark, the reported values are respectively at least about 2 and 4 times smaller than it should be, and they fall just below the Ludii results for these games. The differences certainly are much too large to be explained by hardware differences, which is of similar performance. It is difficult to say whether the other RBG results were computed correctly. In general, in our test we obtained larger values for RBG and smaller values for Ludii with respect to those from [10].
(3) The results for GDL were obtained on a different hardware than those for RBG and Ludii. Indeed, exactly the same GDL results were reported before in [10], where they were obtained with a slower processor. Furthermore, the result for chess was produced using the GGP-Prover instead of a propnet, despite the fact that there was available an efficient chess implementation working well under a propnet (`chess_200.kif` from [12]).

3.1. **Differences in game rules.** The only fully correctly matched games between RBG and Ludii were breakthrough, connect-4, hex, reversi, and tic-tac-toe. The same issue concern matching the GDL versions, as they correspond with RBG.

The authors did not share their Ludii version used for that experiment, thus it is impossible to check directly what exactly game rules were used there. However, there are strong indirect evidences for inaccuracies.

- First, correct rules of the concerned games in a proper variant, fully corresponding to those in RBG 1.0, are not present in any of the published Ludii versions. Thus, it would be unbelievable that they existed in the past and were used for the benchmark.
- Second, the current Ludii 0.3.0 version exhibits similar performance compared to the results from [11]. This shows that in terms of efficiency, there are only minor differences between these versions, which are not meaningful for the conclusions. There are two exceptions, chess and the mill game, for which the public version is significantly slower; this suggests that a much simpler ruleset must had been used.

Note that in our experiment (Table 1), we have used the same version RBG 1.0 as in [10]. A few new game variants were added, because otherwise we could not provide any meaningful comparison with Ludii for them. They roughly demonstrate that the situation in that benchmark would be different if the games were more correctly matched. We have also found a few minor mistakes

in the game codes of RBG 1.0, which were fixed. Nevertheless, they had unnoticeable impact on performance, except a small difference for gomoku and the mill game.

(1) Amazons, arimaa, Chinese checkers, English draughts, international draughts, and the mill game: First, in Ludii, these were split versions, which were compared with orthodox rules versions in RBG. The importance of this mismatching is clearly visible in our benchmark (see Table 1). We included both orthodox and split versions of amazons and arimaa in RBG, which use equivalent rules up to the split moves. The split versions in Ludii indeed work faster than the orthodox versions in RBG, but under the proper matching, it is the opposite. It should not be surprising, since the point of a MC test is to compute all legal moves for every visited game state. In the extreme case of arimaa, splitting reduces branching factor from over 200,000 (the pure flat-MC test does not merge distinct legal moves leading to the same game state) to about 20 (with increased number of turns up to 4); however, the paper reported its 6,490 speedup over RBG without any meaningful analysis for possible reasons. Furthermore, amazons-split code, which was available in RBG 1.0, was ignored.

    The second reason is that, except amazons, these games had incorrect and simplified rules, as their corresponding versions in the public releases have. The simplification is especially influential in the case of arimaa and international draughts, where Ludii's versions omit costly parts of rules (e.g., preventing from repeating the position, restricting captured opponent's pawns to those out of mill).
(2) Chess and double chess: Most likely the used version was a simpler variant of chess rules. This conclusion is supported by the results, as the reported performance of chess is more close to that of skirmish in our benchmark. Furthermore, this kind of simplification seems to be a common practice, as for double chess Ludii contains only the skirmish version; despite that, in the paper, it was compared with orthodox double chess in RBG.
(3) Gomoku: The Ludii's version was the *free style gomoku*, in contrast with the *gomoku standard* in RBG and GDL. Gomoku standard was added to Ludii in 0.3.0. Indeed, in this particular case this was only a minor difference.

4. Reproduction of the results

4.1. **RBG.** While the procedure below may look a bit complicated, it is to ensure that the same version RBG 1.0 is used, with the exception of the added games.

(1) The RBG 1.0 version is available at [7].
(2) To get the games marked with star in Table 1, get RBG 1.0.1 from [8] and extract these games into `games` directory.
(3) Run `make.sh` from `scripts`.
(4) A single test can be run from `rbg2cpp` directory by:

    ```
    make simulate_[game] SIMULATIONS=[playouts]
    ```

    where `[game]` is a game name from `games` directory. The number of playouts should be large enough to process for at least a few seconds. For example, let:

    ```
    make simulate_chess-fiftyMove SIMULATIONS=10000
    ```

    In the output, it can be found the number of playouts performed in specified time in milliseconds, which is the total test time without preprocessing, e.g, `performed 10000 plays in 19038 ms.`

4.2. **Ludii.** To test Ludii, download it from [2]. Having installed Java, run the Ludii player assigned to one processor core on an idle system to ensure than only one is used and to prevent migration, which gives a more stable computation; this is optional, since the results do not differ much. On a linux system, let:

```
taskset -c 0 java -jar Ludii-0.3.0.jar
```

To test a game, load it first with `File/Load Game` and then use `Game/Time Random Playouts`.

5. Conclusions

We have performed a fair benchmark of the efficiency of three GGP systems in their current versions available at the time of writing this note.

The RBG dominates in all except two games: gomoku and hex, for which Ludii is faster. Taking into account tic-tac-toe, where both systems have similar results, it is visible that Ludii is well optimized for this kind of games, where legal moves are actions filling one empty square on the board. For several games in our benchmark, the version in Ludii does not encode proper rules, which also can simplify computation. E.g., in arimaa, there is no checking whether the position is the same, and improper movement of rabbits and trap logic make random playouts shorter in average.

Concerning the benchmark from [10], there are many arguments against its correctness, and if done properly, the conclusions would be different. Indeed, according to our benchmark, even if the unpublished Ludii version was, supposedly, twice faster than the public versions, that does not change the overall situation.

Acknowledgments

We thank Chiara F. Sironi for sharing the GDL propnet code and for helping with using it.

References

[1] C. Browne and F. Maire. Evolutionary game design. *Computational Intelligence and AI in Games, IEEE Transactions on*, 2(1):1–16, 2010.

[2] C. Browne, E. Piette, M. Stephenson, D. JNJ Soemers, and W. Crist. Ludii Portal. `https://ludii.games/downloads.php`, 2019.

[3] M. Genesereth, N. Love, and B. Pell. General Game Playing: Overview of the AAAI Competition. *AI Magazine*, 26:62–72, 2005.

[4] M. Gunther and S. Schiffel. Dresden General Game Playing Server – Games. `http://ggpserver.general-game-playing.de/ggpserver/public/show_games.jsp`, 2019.

[5] J. Kloetzer, H. Iida, and B. Bouzy. The monte-carlo approach in amazons. In *Proceedings of the Computer Games Workshop*, pages 185–192, 2007.

[6] J. Kowalski, M. Mika, J. Sutowicz, and M. Szykuła. Regular Boardgames. In *AAAI Conference on Artificial Intelligence*, pages 1699–1706, 2019.

[7] J. Kowalski, M. Mika, J. Sutowicz, and M. Szykuła. Regular Boardgames 1.0 – source code. `https://github.com/marekesz/rbg/releases/tag/v1.0`, 2019.

[8] J. Kowalski, M. Mika, J. Sutowicz, and M. Szykuła. Regular Boardgames 1.0.1 – source code. `https://github.com/marekesz/rbg/releases/tag/v1.0.1`, 2019.

[9] N. Love, T. Hinrichs, D. Haley, E. Schkufza, and M. Genesereth. General Game Playing: Game Description Language Specification. Technical report, Stanford Logic Group, 2006.

[10] E. Piette, D. JNJ Soemers, M. Stephenson, C. F. Sironi, M. H. M. Winands, and C. Browne. Ludii-the ludemic general game system. *arXiv preprint arXiv:1905.05013*, 2019.

[11] E. Piette, M. Stephenson, D. JNJ Soemers, and C. Browne. An Empirical Evaluation of Two General Game Systems: Ludii and RBG. In *IEEE Conference on Games (CoG)*, 2019.

[12] S. Schreiber. Games – base repository. `http://games.ggp.org/base/`, 2016.

[13] C. F. Sironi and M. H. M. Winands. Optimizing Propositional Networks. In *Computer Games*, pages 133–151. Springer, 2017.