Heuristic Sampling for Fast Plausible Playouts

Cameron Browne and Fabio Barbero
Department of Data Science and Knowledge Engineering
Maastricht University
Maastricht, The Netherlands
cameron.browne@maastrichtuniversity.nl, fabio.barbero@student.maastrichtuniversity.nl

***Abstract*—This paper proposes Heuristic Sampling (HS) for generating self-play trials for games with a defined state evaluation function, with speeds comparable to random playouts but game length estimates comparable to those produced by intelligent AI agents. HS produces plausible results up to thousands of times faster than more rigorous methods.**

***Index Terms*—AI, Board Games, Game Length, Ludii**

I. Introduction

This short note addresses a simple and practical question: *what is the fastest way to reliably estimate the average length of games that will occur between humans playing a given rule set?* Game length – along with *balance* and *decisiveness* – has emerged as a key indicator of abstract board game quality in automated game evaluation [1], [2]. Overly short game lengths indicate that the rule set may be trivial, flawed or subject to easily obtained winning strategies, while excessive game lengths indicate that the rule set may be poorly tuned, reward stalling play, or simply not converge to a result.

Both extremes are usually warning signs of serious problems in a rule set (with some exceptions), and while game length alone is not sufficient to estimate the quality of a game, it provides a useful filter for quickly identifying flawed cases.

A. Context

This question is important for the Digital Ludeme Project, which aims to reconstruct missing rules of ancient board games based on historical evidence and automated evaluations of potential rule sets that fit the known details [3]. The Ludii general game system [4] implements over 1,000 *ludemes* (game-related concepts) that can be combined in different ways to define new rule sets, providing many thousands – or millions – of plausible rule sets to test for each reconstruction task. It is imperative to have fast and reliable game evaluation metrics, which ideally operate at the millisecond level.

Potential rule sets are evaluated through automated self-play trials between agents [2]. Random playouts provide the speed required but produce unrealistic games involving poor move choices that no intelligent player would make, hence can give misleading results. Intelligent playouts by AI agents give more realistic results but are orders of magnitude slower to generate and too slow for practical purposes.

This research is funded by the European Research Council as part of the Digital Ludeme Project (ERC Consolidator Grant #771292) led by Cameron Browne at Maastricht University's Department of Data Science and Knowledge Engineering.

Specifically, we want a method that produces an *observed mean game length* L_{obs} that approximates the *expected mean game length* L_{exp} of a given rule set within a certain tolerance. For the purposes of this exercise, a value of L_{obs} between $L_{exp}/2$ and $2 \times L_{exp}$ is deemed to be acceptable.

In this paper, we present a simple method called Heuristic Sampling that produces playouts with plausible game lengths not far off those produced by intelligent play but with speeds comparable to random play.

II. Heuristic Sampling

Heuristic Sampling (HS) applies when a heuristic state evaluation function is available. It involves successively choosing for each state S_i a high-reward action a from the set of available actions A_i and applying it without lookahead search. This may be described as a form of tournament selection in which the maximal element is chosen from a randomly selected subset of the available actions for each state.

We denote a HS search as $HS_{1/n}$ where $1/n$ is the fraction of available actions to sample at each state. Specifically:

```
1. for each state $S_i$
2.     generate the set of available actions $A_i$
3.     if $|A_i| == 1$ return $A_{i_0}$
4.     randomly select $max(2, \lceil\frac{|A_i|}{n}\rceil)$ actions from $A_i$
5.     for each selected action $a$
6.         apply $a$ to $S_i$ to give $S_{i_a}$
7.         if $S_{i_a}$ is a winning state then return $a$
8.         else if $S_{i_a}$ is a losing state then ignore $a$
9.         else evaluate $S_{i_a}$ with heuristic function
10.    return action $a$ with highest seen heuristic estimate
```

HS implicitly handles winning "mate-in-one" [5] or "decisive" [6] moves, if such moves are among the randomly chosen subset for their state. It also encourages greater exploration of the game state space than traditional search methods, by not following the same line of optimal play every time. This can prove beneficial when the aim is to exercise rule sets for evaluation purposes rather than outright playing strength.

A. Same-Turn Continuation

In Ludii, a *turn* is a consecutive sequence of actions by the same player [4]. For example, making a line of three in Nine Men's Morris allows the mover to then capture an enemy piece (Fig. 1); both actions are part of that player's turn.

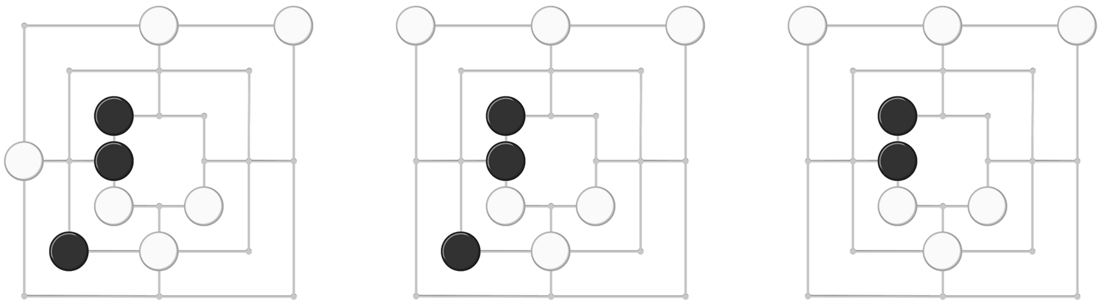

Fig. 1. White makes a line of three in Nine Men's Morris, then captures a black piece to complete their turn.

Same-Turn Continuation (STC) involves repeatedly applying HS to the current state while it is the same player's turn to move, so that the turn's ultimate heuristic estimate is taken at the end of the turn. That is, while ever it remains the same player's turn to move after an action has been applied, then another action from the resulting state should be chosen and applied. The resulting heuristic estimate at the end of the turn should then be used as the value for the turn's first action.

Without STC, only the first action in such chained sequences would be considered, giving poor results for games in which strong moves are contingent upon a (weaker) dependent move being made first. STC is similar in principle to *quiesence search*, in which the search continues beyond its nominal depth for unstable states until a "quiet" state is reached [7].

III. Experiments

The experiments were conducted using the Ludii general game system.[1] The heuristic state estimates for each game are provided by a set of 15 general heuristics implemented for Ludii, with weight vectors learnt per game through gradient descent. The result is that Ludii provides competent but not overly strong heuristic state evaluations for most games.

Experiments were run on the following games:

1) *Tic-Tac-Toe* ($L_{exp} = 9$ plies): Standard game on 3×3 board, game tree complexity $\approx 10^5$.
2) *Connect4* ($L_{exp} = 36$ plies): Standard game on 6×7 board, game tree complexity $\approx 10^{21}$.
3) *English Draughts* ($L_{exp} = 70$ plies): Standard game, 8×8 board, game tree complexity $\approx 10^{31}$.
4) *Nine Men's Morris* ($L_{exp} = 50$ plies): Standard game, nine pieces per player, game tree complexity $\approx 10^{50}$.
5) *Halma* (plies unknown): Two-player game on a reduced 6×6 board, game tree complexity unknown.
6) *Lines of Action* ($L_{exp} = 44$ plies): Standard game on 8×8 board, game tree complexity $\approx 10^{64}$.
7) *Gomoku* ($L_{exp} = 30$ plies): Standard game on 15×15 board, game tree complexity $\approx 10^{70}$.
8) *Chess* ($L_{exp} = 70$ plies): Standard game on 8×8 board using FIDE rules, game tree complexity $\approx 10^{123}$.
9) *Shogi* ($L_{exp} = 115$ plies): Standard game on 9×9 board, game tree complexity $\approx 10^{226}$.

[1]Ludii is available at: http://ludii.games

These games were selected to test the methods over a range of game types and complexities. Expected game lengths L_{exp} were taken from the "Game Complexity" Wikipedia page.[2] Note that these expected game lengths are specified in terms of *plies* whereas Ludii measures game length in terms of *turns*, i.e. consecutive sequences of moves by the same player; these will typically be identical but this is not guaranteed.

A. Playout Methods

The following playout methods were compared:

1) *Random*: Uniformly random playouts.
2) $HS_{1/n}$: Heuristic Sampling with Same-Turn Continuation, for sampling ratios $1/2$, $1/4$ and $1/8$.
3) AB_d: Standard Alpha Beta to depths of $d = 1$, 2 and 3. AB_1 is essentially equivalent to $HS_{1/1}$ without STC.
4) UCT: Standard UCT search with a budget of 1,000 iterations per move using uniformly random playouts and a default exploration constant of $C = \sqrt{2}$.

Each search method was applied to 100 trials of each game where possible (Shogi involved fewer trials due to excessive run times). A cap of 1,000 turns was applied for all trials except for $Random$ search; trials that exceeded this turn limit were abandoned as draws and excluded from the sample.

IV. Results

Table I shows the expected length of each game L_{exp} (in plies) and the observed length L_{obs} (in turns) obtained for each playout method, in addition to the number of completed trials N, minimum and maximum observed lengths, standard deviation (SD), standard error (SE) and average time per trial in seconds.

Timings were taken on a standard consumer machine with six 2.9 GHz $i9$ cores, similar to machines that Ludii users performing reconstruction tasks will typically use. Timings are therefore indicative only and can be multiplied by a factor of around ×10 to estimate their serial equivalents.

Fig. 2 summarizes these results and highlights the performance of each playout method against expected length L_{exp} per game (dotted line). Additional comparisons between HS with and without STC reveal that HS without STC performs almost no better than $Random$ for games such as Nine Men's Morris with chained action sequences per turn.

[2]https://en.wikipedia.org/wiki/Game_complexity

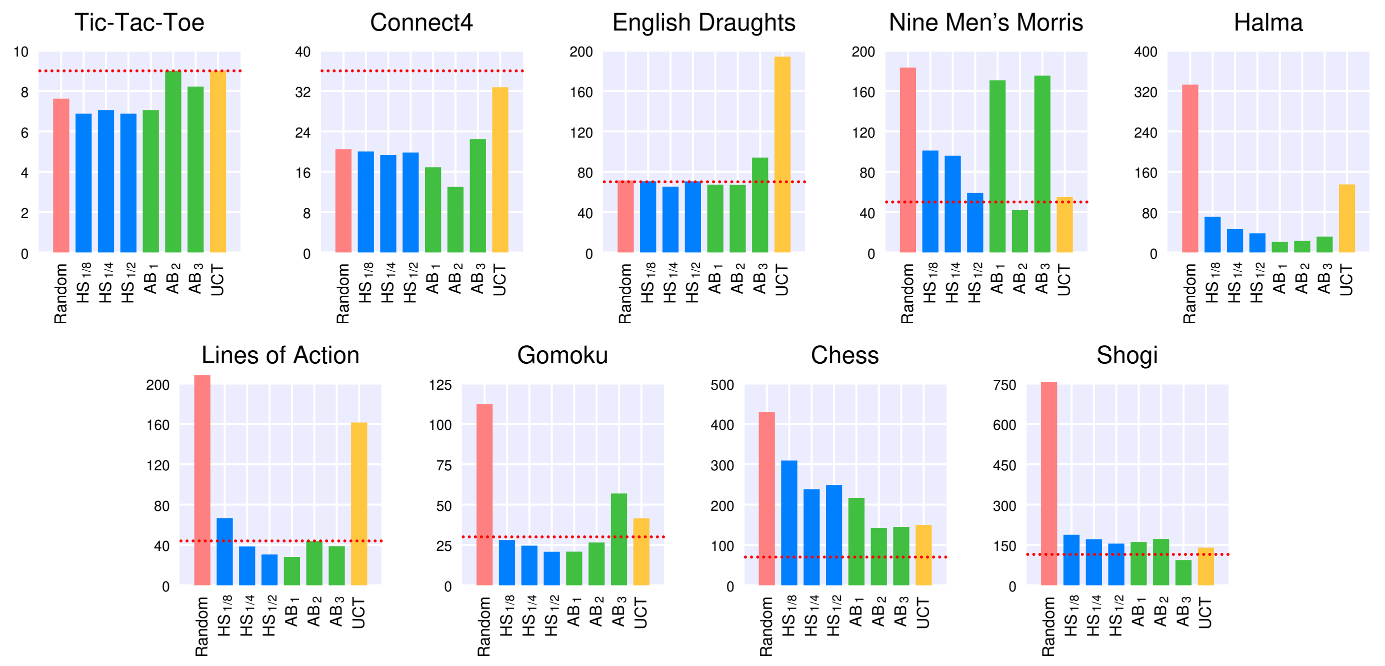

Fig. 2. Mean number of turns per playout for each method applied to each game (dotted lines show expected game length).

V. Discussion

As expected, the stronger searches (AB_3 and UCT) give best results but are thousands of times slower than $Random$ and HS. HS performs almost as well as AB for most cases – sometimes better! – and produces results in the acceptable $L_{exp}/2$ to $2 \times L_{exp}$ range for all games tested except Chess. Chess is unusual in that many if not most games between human players end in a resignation before the game is fully played out and recorded game lengths reflect this truncation.

$HS_{1/2}$ generally outperforms $HS_{1/4}$ and $HS_{1/8}$ but not in all cases. Smaller sampling ratios may be preferable in cases where heuristic state evaluations are expensive to compute.

Note that $Random$ playouts can be much longer than HS playouts due to the undirected nature of purely random play. HS can actually be much faster than $Random$ in such cases!

Another surprise is that AB_d can produce worse game length estimates than AB_{d-1} in several cases. This may be due to the "Odd-Even Effect" in AB search, in which odd search depths (terminating on the owner's ply) can give overly optimistic results, while even search depths (terminating on the opponent's ply) can give overly pessimistic results.[3]

VI. Conclusion

The results of this experiment suggest that Heuristic Sampling with Same-Turn Continuation can produce playouts with acceptably plausible game lengths with the speed of random playouts – and often faster – thus offering a potential approach for testing large numbers of reconstructed rule sets for obvious flaws within a reasonable time. HS produced results generally comparable to more rigorous search methods but up to thousands of times more quickly.

Future work will include evaluating HS over a wider range of games and looking more closely at exactly *how* reliable its game length estimates are for identifying flawed rule sets. Further, the question of what constitutes the range of desirable game lengths for a given rule set remains open.

We will also investigate the use of HS as the playout policy in a Monte Carlo Tree Search framework, to see whether the improved playout quality (over uniformly random playouts) can produce superior playing strength with fewer iterations.

[3]https://www.chessprogramming.org/Odd-Even_Effect

Acknowledgment

Thanks to the Ludii team (Éric Piette, Matthew Stephenson and especially Dennis Soemers) for help developing the Ludii framework in which these experiments were run. The Ludii source code is provided as open source under a CC BY-NC-ND 4.0 licence (https://github.com/Ludeme/Ludii).

References

[1] I. Althöfer, "Computer-Aided Game Inventing," Friedrich-Schiller Univ., Faculty Math. Comp. Sci., Jena, Tech. Rep., 2003.
[2] C. Browne, "Automatic Generation and Evaluation of Recombination Games," Ph.D. dissertation, Queensland University of Technology, 2009.
[3] C. Browne, "Modern Techniques for Ancient Games," in *IEEE Conf. Comput. Intell. Games*. Maastricht: IEEE Press, 2018, pp. 490–497.
[4] É. Piette, D. J. N. J. Soemers, M. Stephenson, C. F. Sironi, M. H. M. Winands, and C. Browne, "Ludii – The Ludemic General Game System," in *Proc. ECAI 2020*. IOS Press, 2020, pp. 411–418.
[5] R. J. Lorentz, "Improving Monte-Carlo Tree Search in Havannah," in *Proc. Comput. and Games, LNCS 6515*, Kanazawa, 2010, pp. 105–115.
[6] F. Teytaud and O. Teytaud, "On the Huge Benefit of Decisive Moves in Monte-Carlo Tree Search Algorithms," in *Proc. IEEE Symp. Comput. Intell. Games*, Dublin, 2010, pp. 359–364.
[7] D. Beal, "A Generalised Quiescence Search Algorithm," *Artificial Intelligence*, vol. 43, pp. 85–98, 1990.

TABLE I
MEAN NUMBER OF TURNS PER GAME, PER PLAYOUT METHOD.

Game	Method	L_{obs}	N	Min	Max	SD	SE	Time per Trial
	$Random$	7.62	100	5	9	0.142	0.278	0.000148s
	$HS_{1/8}$	6.88	100	5	9	0.132	0.259	0.000621s
Tic-Tac-Toe	$HS_{1/4}$	7.05	100	5	9	0.139	0.272	0.000640s
	$HS_{1/2}$	6.88	100	5	9	0.155	0.303	0.000769s
$L_{exp} = 9$	AB_1	7.05	100	5	9	0.185	0.363	0.000583s
	AB_2	9	100	9	9	0.0	0.0	0.000929s
	AB_3	8.22	100	7	9	0.098	0.192	0.00312s
	UCT	9	100	9	9	0.0	0.0	0.0294s
	$Random$	20.46	100	7	37	0.729	1.428	0.000226s
	$HS_{1/8}$	20.02	100	7	42	0.719	1.41	0.000736s
Connect4	$HS_{1/4}$	19.29	100	7	42	0.697	1.37	0.000709s
	$HS_{1/2}$	19.81	100	7	42	0.97	1.90	0.000996s
$L_{exp} = 36$	AB_1	16.88	100	11	29	0.564	1.11	0.00109s
	AB_2	13	100	13	13	0.0	0.0	0.00250s
	AB_3	22.45	100	7	42	1.204	2.36	0.190s
	UCT	32.75	100	17	42	0.601	1.178	0.403s
	$Random$	71.3	100	37	161	3.12	6.12	0.00192s
	$HS_{1/8}$	70.5	100	35	145	2.54	4.99	0.00572s
English Draughts	$HS_{1/4}$	65.2	100	33	197	2.41	4.71	0.00606s
	$HS_{1/2}$	70.5	100	37	139	2.25	4.4	0.00943s
$L_{exp} = 70$	AB_1	67.2	100	43	105	1.76	3.454	0.00948s
	AB_2	67.0	93	47	87	1.64	3.22	0.00784s
	AB_3	94.0	63	74	158	2.042	4.002	0.418s
	UCT	194.0	92	53	879	16.424	32.19	13.9s
	$Random$	183.2	100	56	790	11.801	23.13	0.00432s
	$HS_{1/8}$	101.0	100	36	431	6.22	12.2	0.00176s
Nine Men's Morris	$HS_{1/4}$	95.9	100	36	273	4.57	8.96	0.00215s
	$HS_{1/2}$	58.9	100	30	173	2.44	4.78	0.00349s
$L_{exp} = 50$	AB_1	170.6	100	43	532	10.2	19.9	0.00491s
	AB_2	41.9	100	20	65	0.909	1.78	0.00481s
	AB_3	175.2	84	44	963	19.9	39.1	0.377s
	UCT	54.8	100	31	149	2.22	4.34	6.86s
	$Random$	333.0	100	77	923	15.22	29.83	0.00435s
	$HS_{1/8}$	70.84	100	36	140	2.09	4.10	0.00163s
Halma 6×6	$HS_{1/4}$	46.1	100	26	97	1.19	2.33	0.00135s
	$HS_{1/2}$	37.8	100	20	71	1.011	1.98	0.00133s
L_{exp} not known	AB_1	20.9	100	14	26	0.272	0.532	0.00116s
	AB_2	23.2	45	17	36	0.653	1.28	0.0463s
	AB_3	31.4	23	20	70	2.69	5.27	0.470s
	UCT	134.83	88	29	922	16.9	33.1	45.5s
	$Random$	208.3	100	49	417	7.734	15.158	0.00896s
	$HS_{1/8}$	66.6	100	32	150	2.518	4.94	0.00424s
Lines of Action	$HS_{1/4}$	38.5	100	23	77	1.02	2.00	0.00353s
	$HS_{1/2}$	30.6	100	23	43	0.418	0.82	0.00440s
$L_{exp} = 44$	AB_1	28.1	100	22	38	0.307	0.602	0.00762s
	AB_2	43.8	97	22	163	2.13	4.181	0.200s
	AB_3	38.8	98	23	73	0.778	1.53	1.24s
	UCT	161.0	100	30	518	10.6	20.7	184.4s
	$Random$	112.2	100	52	171	2.64	5.17	0.000353s
	$HS_{1/8}$	28.1	100	10	57	0.862	1.69	0.00360s
Gomoku	$HS_{1/4}$	24.6	100	10	43	0.708	1.39	0.00736s
	$HS_{1/2}$	20.8	100	10	43	0.608	1.19	0.0105s
$L_{exp} = 30$	AB_1	20.9	100	13	45	0.653	1.28	0.0125s
	AB_2	26.5	100	13	81	1.35	2.65	0.141s
	AB_3	56.9	100	11	113	2.73	5.36	2.28s
	UCT	41.47	100	20	70	1.12	2.20	0.904s
	$Random$	429.5	100	54	739	13.2	25.9	0.0243s
	$HS_{1/8}$	309.1	100	5	516	10.7	21.0	0.0317s
Chess	$HS_{1/4}$	238.0	100	16	471	11.1	21.7	0.0376s
	$HS_{1/2}$	248.6	100	16	435	10.8	21.2	0.0800s
$L_{exp} = 70$	AB_1	216.8	100	12	459	10.2	20.1	0.020s
	AB_2	142.3	100	22	501	9.58	18.8	0.564s
	AB_3	144.8	100	24	402	7.92	15.5	6.93s
	UCT	149.8	100	9	532	11.4	22.3	25.7s
	$Random$	756.4	100	100	2501	68.5	134.3	0.172s
	$HS_{1/8}$	187.9	100	75	587	9.5	18.6	0.593s
Shogi	$HS_{1/4}$	171.4	100	31	459	8.80	17.3	1.18s
	$HS_{1/2}$	155.0	100	22	533	9.14	17.9	2.50s
$L_{exp} = 115$	AB_1	161.1	100	38	871	10.8	21.2	5.17s
	AB_2	172.3	96	30	728	10.9	21.3	60.1s
	AB_3	94.0	10	38	114	8.26	16.194	751.5s
	UCT	140.1	10	77	218	14.2	27.7	4,790.9s