Skill Depth in Tabletop Board Games

James Goodman, Diego Perez-Liebana, Simon Lucas
Game AI Research Group
Queen Mary University of London
james.goodman, diego.perez, simon.lucas@qmul.ac.uk

***Abstract*—There are well-established methods for rating the relative skill of players such as Elo or TrueSkill ratings. This is not the case for rating games by their relative difficulty, or the level of 'skill' required to play them well. Previous work has proposed *skill-traces* as an answer to this question, which use data from games played between agents with progressively higher computational budgets to estimate the difficulty of the game (or *skill-depth*). We try to improve on previous work by expanding the algorithmic space considered and that this can radically change the ratings of some games. We then propose a new parameterised model for the level of skill a game requires and test this on a suite of multiplayer tabletop board games, concluding that the parameters can be usefully interpreted and provide a slightly better fit to human-estimates.**

I. Introduction

It is hopefully not controversial to state that Chess is a game of skill, that Tic-Tac-Toe has very little skill involved, and that Snakes and Ladders has none, with the outcome dependent purely on dice results unaffected by any player decision. 'Skill' here meaning that a player with better knowledge of the game and rules, and/or able to think through the consequences of different actions will win, on average, more games than a less well endowed player.

Measuring the skill of a player in terms of their relative win rate is widely accepted, and is the basis of Elo, TrueSkill and other rating systems [1], [2]. How can we measure the skill of a game rather than a player? One approach is to see how performance in the game changes as we adjust the computational capacity of agents. If a game has a level of skill then increasing this capacity should increase win rate, score, or other game-appropriate performance metric.

This idea is developed in Lantz et al. 2017. They consider a *skill ladder*, conjecturing that a game can be quantified in terms of its *depth*, or number of rungs on the ladder [3]. A game like Tic-Tac-Toe should have a short ladder, as a beginner rapidly learns a few key heuristics and then plateaus, with a guaranteed draw. Games like Chess or Go should have very long skill ladders, as lifetimes of investment in learning the game pay off with ever greater performance.

This work is inspired by this general approach, and builds in particular on Browne 2022 [4]. This uses Monte Carlo Tree Search (MCTS) with increasing budgets to estimate the skill ladder of classic tabletop games in the Ai Ai framework [5]. We agree with the general principle and develop further some of the specifics of implementation and interpretation:

1) Stochasticity and/or imperfect information in a game changes the observed rewards of skill. A common design goal is to introduce randomness so that even a novice player has a chance of victory [6], [7]. In poker, and many other card games, a player may have a sequence of very lucky card deals and do well almost regardless of underlying skill-level. This means that a 'good' player may win only 60% of games against a 'poor' player, for some fixed definitions of 'good' and 'poor'. In deterministic games such as Chess, we expect a 'good' player to win closer to 100% of games against a 'poor' player. Elo will rate the 'good' player in the stochastic game lower than in the deterministic game. Comparing skill-ladders across games needs to account for this. Browne 2022 includes one stochastic game (Can't Stop, also included here), and no games with imperfect information.
2) The impact of the number of players. At a basic level this requires the win rate to be adjusted for the number of players in a game. We are also interested in how the skill level of a game changes with the player count.
3) The space of measuring algorithm is important. Browne 2022 uses MCTS with a fixed set of standard parameterisations. However, if this vanilla MCTS is poor at a particular game (as we show later to indeed be the case) then the skill-estimate it produces will be poor. A better estimate of the skill level can be obtained from a larger algorithmic space, and tuning to the game in question.
4) Browne 2022 measures the win rate in 1000 games against an opponent with half the computational budget to give a direct *ladder* of 16ms vs 32ms, 32ms vs 64ms and so on. This may not statistically distinguish skill differences when the change in win rate is small. We extend this to a *grid*, of 16ms vs 32ms, 16ms vs 64ms, and so on for all pairwise budgets. A difference between 16ms and 32ms might not be statistically resolvable, but that between 16ms and 64ms/128ms might be.

Our contributions are to extend the analysis of [4] to a set of 16 varied tabletop board games with imperfect information and more than 2 players, and using a grid of data as just described. We expand the algorithmic space from vanilla MCTS and show how this improves the skill trace evaluation. Finally we propose a three-parameter model of the depth of skill in a game, fit this to the grid data and suggest how these parameters can usefully be interpreted.

979-8-3503-5067-8/24/$31.00 ©2024 IEEE

II. Previous Work

A common feature in the literature is to quote the size of the state space as a measure of the algorithmic challenge a game presents. Thompson 2000 suggested a game's strategic 'depth can actually be measured by recording the results of games and determining how many distinct "levels" there are: if the players in class 1 all lose regularly to the players in class 2, who lose to players in class 3, etc., up to class n, then the value of n measures the depth of the game' [8].

The idea of a *skill ladder* is introduced by [6] as a sequence of skills or heuristics that a beginner at game starts to learn in sequence. The more such skills, or 'rungs' on the ladder, the deeper the game. Lantz et al. 2017 suggest constructing the skill ladder using AI agents by determining the increase in computational resource required to make progress towards optimal performance [3]. One operationalisation of this is to define a 'rung' as the difference in resource that leads to a fixed change in performance, such as a 60% mean win rate over a player at the previous, lower rung [5]. The computational resource to be restricted is open. This approach has been used in Yahtzee and single-player puzzle games with restrictions on either the number of neurons in the hidden layer of a neural network, or the length of training time of a policy through reinforcement learning [9], [10]. The MCTS time budget has been used in [4], [5].

An alternative view is that skill depth is in the eye of the beholder and is dependent upon the agent playing the game [11]. Different games require different skills, and the ladder for a game that depends on deep planning, such as Chess, may not be meaningfully comparable with a ladder for a game like Poker that depends on probabilistic analysis and social deception. A linked idea is to identify a subset of games which best distinguish between different algorithms [12]. Given a set of algorithms/agents, each of which has a different core skill, a ladder could be constructed for each skill to give a more multidimensional perspective.

Outside of the ladder approach, the skill depth of a game has also been measured by the average depths of action rules generated from human play-traces [13]. This is dependent on the skill of the players from whom the traces are acquired. Relative Algorithm Performance Profiling [14] measures the 'quality' of a game by the range of performance of different algorithms (in 1-player games). This is a relatively crude metrics and very sensitive in practice to the performance of the least-skilled, often random, agent. Single-player puzzle games have also been analysed using relative entropy and defining their difficulty by the number of bits of information needed to guide a player to a solution [15].

Browne 2022 is the work on which we build most directly [4], and is now outlined in more detail. This modifies the ladder approach to fit a curve to the results of games between agents with different budget levels. He investigates 31 games in the Ai Ai framework and proposes a Skill Trace estimate for a game that has two components. Firstly a regression line is fit to the relative win rates to predict y, the relative win rate

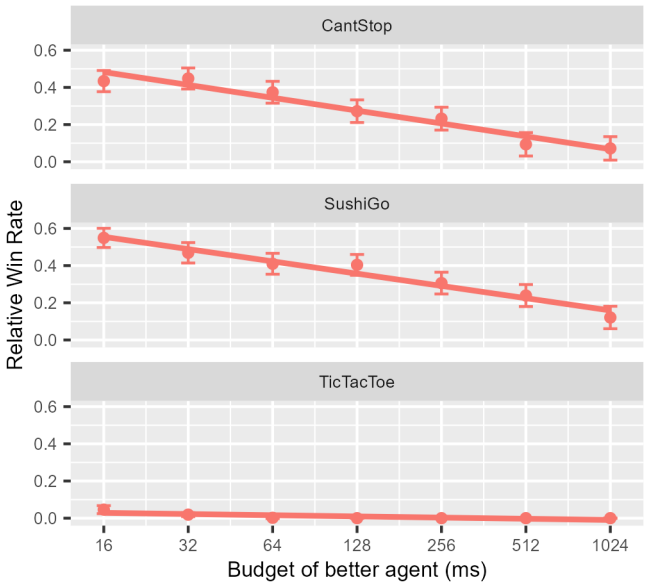

Fig. 1: Example of Skill Trace (ST) Ladders for three 2-player games, using the technique of Browne 2022. Explanation in Section II. A game has a high ST if the area under the line is large, and the projected win rate at 2048ms (against an opponent with 1024ms) is high.

of the next, untried, higher budget agent. This will be in [0, 1] with 0 being a 50% win rate in a 2-player game, and 1 being a 100% win rate. Secondly the win rates of the better budget in each tournament are squared and averaged as an approximation of the area under the curve, AUC. The Skill Trace (ST) is given by:

$$ST = y + (1 - y)\text{AUC} \tag{1}$$

Together this ensures the Skill Trace of a game is in [0, 1]. An ST of 1 implies an agent with budget $2X$ will always win 100% of games against an opponent of budget X. A game like Tic-Tac-Toe will have $y = 0$, as once the budget passes a certain threshold all games will be drawn.

In Figure 1 Tic-Tac-Toe reaches this threshold in under 40ms of MCTS budget, Can't Stop reaches this shortly after 1 second, and Sushi Go (if we extrapolate) at 2-4 seconds. The contribution of AUC means the ST is not 0, and will be larger the longer it takes to each this threshold. Tic-Tac-Toe therefore has a lower ST than Can't Stop, even though they have the same projected y of about 2 seconds.

Browne 2022 then regress the ST calculated for each game against the 'Complexity' rating of the games on BoardGameGeek[1] (BGG) as a surrogate for inherent game skill-depth. This compiles the votes of thousands of hobbyist gamers on a scale of 1 to 5, with 1 being the simplest possible game and 5 being highly complex and involved. Tic-Tac-Toe is rated at 1.29, Chess at 3.66. All the games analysed are 2-

[1] www.boardgamegeek.com

Parameter	Values	Default
K	0.01, 1.0, 100	1.0
ϵ	0.03, 0.1, 0.3	-
Selection	UCT, UCB-Tuned, Regret Matching	UCB
Rollout policy	Random, MAST	Random
MAST γ	0.0, 0.5	-
Rollout length	0, 3, 10, 30, 100, 300, 1000	1000
Tree type	Standard, MultiTree, SelfOnly	Standard
Heuristic	WinOnly, WinPlus, Ordinal, Score, ScorePlus, Leader	WinOnly

TABLE I: MCTS Parameters that define the algorithmic space.

player games with perfect information. Some have stochastic elements.

III. Algorithmic Space

A single algorithm will not perform equally well in all games. The No Free Lunch Theorem makes this point theoretically [16], and multiple results show this empirically with even a single algorithm requiring much parameter-tuning to the problem at hand. MCTS initially gained its foothold in Game AI research with levels of success in Go not achieved by minimax-search approaches; vice versa MCTS is poor at Chess, for which minimax search works well [17].

We hypothesize that using a single set of algorithmic settings in MCTS may lead to incorrect estimation of the skill depth of a game. Using a wider space of possible algorithms can select the correct tool for each game, and the scarce computational resource is then allocated optimally.

More formally, if we have a space of algorithms, $\mathcal{A}$, and a game g, then each algorithm, $a \in \mathcal{A}$, will have a different performance, J, in g at any given resource level, b, i.e. $J = f(g, a, b)$, hopefully with this function $f(\cdot)$ being non-decreasing in b. There is no *a priori* reason for the same $a \in \mathcal{A}$ to be optimal at all budget levels and we define 'optimal' performance, J_{opt} for a game at a given budget to be:

$$J_{opt}(b, g) = \max_{a \in \mathcal{A}} J(g, a, b) \tag{2}$$

We seek to plot this J_{opt} for increasing b, and hence estimate the skill depth of a game. Within the framework of *anytime* algorithms, a budget constraint of computational budget per decision is used. This is not the only possibility. The memory usage during decision making could be another valid choice, or if reinforcement learning were used then training time and/or the number of parameters in the network would be more natural choices. An *anytime* restriction keeps the comparison to a single computational bottleneck without suggesting that this is the only important or interesting one.

To measure the performance, J, the win rate of an agent a_1 with budget b_1 is compared against agent a_2 with lower budget $\frac{b_1}{N}$. Note that a_2 may be different to a_1, with each being the best point in $\mathcal{A}$ for their respective budgets. This is the same approach as used in [4], in which $N = 2$, and $|\mathcal{A}| = 1$. The algorithmic space used is outlined below and in Table I.

- **Selection** is the tree selection policy used. UCT is the standard UCB for Trees algorithm [18], UCB-Tuned and Regret Matching use the selection rules of the same names [19], [20].
- **K** is the exploration constant in the UCB equation $M(s, a) = Q(s, a) + K\sqrt{\frac{2 \log N(s)}{n(s,a)}}$. ϵ is used for Regret Matching, and is the probability of taking a random exploration action in the tree.
- **Rollout policy** is either to take Random actions, or use Move-Average Sampling (MAST) to define a soft-max rollout policy over the average values of previous rollouts containing the action [21]. **Rollout length** is the number of actions to take after leaving the tree. **MAST** γ is the discount applied to the MAST action estimates from the previous decision; 0.0 preserves no memory, while higher values inherit action estimates from previous moves.
- **Tree type** is how players are modeled in the tree. 'Standard' uses the normal MCTS approach of modeling all players in a single tree. 'MultiTree' has one tree for each player [22], and 'SelfOnly' just models the player's own actions, with other players assumed to act randomly.
- **Heuristic** is the function used to value a terminal state, or the state at the end of a rollout. These can give very different ranges of values, and all values are scaled to a range of [0, 1] by monitoring the min/max reward values observed. This scaling ensures that the **K** values have consistent interpretations.
  - WinOnly returns +1/0.5/-1 for a Win/Draw/Loss respectively, and 0 if the game is not over.
  - Score returns the raw game score; ScorePlus adds a +/-50% bonus for winning/losing.
  - Leader returns the difference between the player's score and that of the best other player. A +/-50% bonus is added for winning/losing [23].

IV. Games

The 16 games used are listed in Tables II. There is insufficient space to describe all of them in detail, and details are available at www.BoardGameGeek.com, or tabletopgames.ai. Three of the games (Connect 4, Dots And Boxes and Can't Stop) are in the corpus used in [4]. Five of the games are perfect information, the remainder are all imperfect information.

V. Model of skill depth

A few axioms are the starting point of our model for skill depth in a game.

1) As a player's skill improves, their win rate against fixed opponents will go up.
2) Two players of equal skill will, *a priori*, win 50% of games (counting a draw as half a win, and with players playing from each position an equal number of times in case of first-player advantage).
3) As the budget advantage over opponents tends to infinity, player win rate will tend to a plateau level. This level may be lower than 100% in games with stochasticity.
4) There will be diminishing returns from additional skill. If doubling the skill/budget leads to an increase of $X\%$

in the win rate, then doubling it again will lead to an increase of $Y \leq X\%$.

5) Ultimately a game may exhibit *perfect play*, in which an agent with budget Ω is never (on average) beaten by a player with budget $Z > \Omega$. In Tic-Tac-Toe Ω is about 40ms based on Figure 1.

A first natural model is the logistic curve, shifted to the origin, in Equation (3). This has two parameters M and r.

$$S_{adj} = M\left(1 - \frac{2}{1+e^{rx}}\right) = M\left(1 - \frac{2}{1+e^{r\log_2 \frac{B}{b}}}\right) \quad (3)$$

S_{adj} is the adjusted win rate, with 0 meaning a 50:50 win rate in a 2-player game, +1 meaning a 100% win rate and -1 a 100% loss rate. See Equation (5) later for adjustments for more than 2 players.

$x = \log_2 \frac{B}{b} \in \mathbb{R}^+$, where B is the budget of the better agent, and b that of the less good agent. Equation 3 is monotonically increasing with B (Axiom 1). Log_2 is used as x then takes integer values given the budget increases by powers of 2. For agents with equal budgets, $\frac{B}{b} = 1$, and $S_{adj} = 0$ (Axiom 2).

M is the maximum achievable performance ($M \leq 1.0$) which is approached asymptotically as $x \to \infty$ (Axiom 3). M will be 1 if sufficient additional budget will lead to a 100% win rate. As discussed in the Introduction, with stochasticity and imperfect information it is expected that M will be some intermediate value between 0 and 1 for games in which a lucky deal of cards can allow a weak player to win the occasional game against a much better opponent.

r is a measure of the speed with which performance increases towards M as the budget is increased (Axiom 4). High r means a performance plateau is rapidly reached (high diminishing returns, suggestive of a short skill ladder), low r means progress is slow and performance keeps increasing with budget (low diminishing returns, and a long skill ladder).

This model assumes that regardless of the starting point, doubling the budget will give the same performance improvement in violation of Axiom 5. To cater for this a third parameter, β, is added to given Equation (4). The change in Equation (4) from (3) is highlighted in **red**.

$$S_{adj} = \frac{M}{\mathbf{1} + \frac{\mathbf{b}}{\beta}}\left(1 - \frac{2}{1+e^{r\log_2 \frac{B}{b}}}\right) \quad (4)$$

If r controls the rate of diminishing returns against a fixed opponent, β can be thought of as the 'second order' effect. Against a fixed poor opponent (low b) we reach a plateau of $M1$, while against a better fixed opponent (high b) we can only reach a plateau $M2 < M1$. Figure 2 illustrates the difference that β makes as the quality of the opponent changes.

As $\beta \to \infty$, Equation (4) simplifies to (3). A game with a large β *will* have the same M for all b, and will not exhibit diminishing returns. This can be interpreted as a game with high skill depth where investing more cognitive resources provides a consistent benefit with no threshold at which perfect play is achieved. Low β is indicative of a game like Tic-Tac-Toe in which perfect play is achieved at a low skill level.

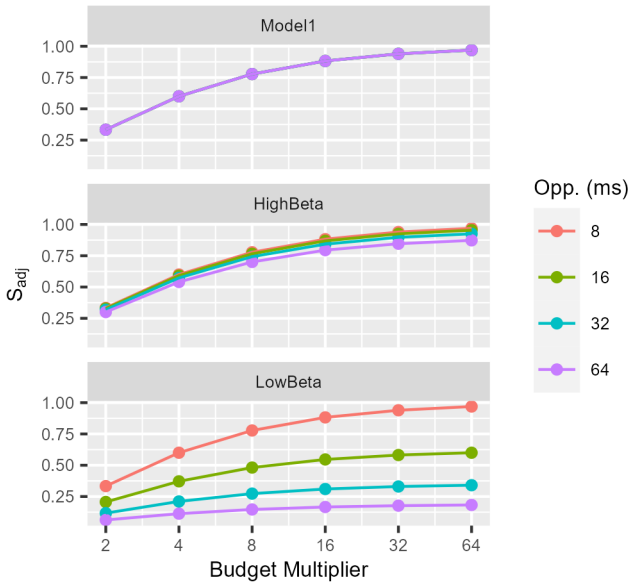

Fig. 2: Example of theoretical fits for Model 1 (Equation 3, top, $M = 1, r = 1$), and then Model 2 (Equation 4) high beta (middle, $\beta = 500$) and low beta (bottom, $\beta = 5$). Model 1 assumes that an agent with budget $2x$ will beat an agent with budget x by a fixed percentage for all x, and the line of adjusted win rate to budget multiplier is the same for all fixed opponents. Model 2 fulfils Axiom 5, and perfect play is approached as x increases.

We hypothesize that the three parameters of a game from fitting the data to Equations (3) and (4) will provide a more nuanced characterisation of a game than ST alone. In particular, a game with high skill depth should have a high β.

VI. Methodology

Parameter tuning runs find a good set of MCTS parameters for each within the algorithmic space in Table I. Optimal settings may vary both by player count and the computational budget. NTBEA is used for parameter optimisation [24].

This gives a tuned set of parameter settings for each budget and player count. For some games it was found that the same agent was robustly best across all player counts and budgets, while for others this varied. For each game a Skill Ladder is constructed using the methodology in [4], from the win rates of an agent with a budget of $2x$ against one with a budget of x, for $x \in 8, 16, 32, 64, 128, 256, 512$ms. 1000 games are run for each pairing, with player positions rotated, and the same set of random seeds used.

In addition win rates of $4x$ versus x, $8x$ versus x, $8x$ versus $2x$ and so on are calculated. This adds additional experimental time, as 28 tournaments are required instead of 7, but permits a full grid to be constructed and provides better resolution of differences in skill depth. This process was repeated twice. Once using the tuned parameter settings specific to each game (the 'Tuned' dataset), and once using the same default MCTS parameter settings for all games (the 'Classic' dataset). The

default settings (in Table I) correspond to the settings used in [4], with the exception that Open Loop Information Set MCTS is used, with the state redeterminised at the root on each iteration [25]. This is because most of the games have imperfect information, which was not the case in [4]. All experiments were run using the TAG research framework, which allows the same algorithm to be applied to very different games with useful data reporting [26].

Skill Traces are calculated for both Tuned and Classic datasets. These are also rank regressed against the complexity ratings of the games from BoardGameGeek (BGG) . A comparison of these shows how the increase in algorithmic space is important, with results reported in Section VII-A.

For each game the theoretical models of Section V are then fit to the Tuned dataset. This fit used the `nlm` function in base R[2]. L2 regularisation ($\lambda = 0.0001$) was used on M, r and $\log \beta$ to impose a weak prior of low skill and a flat skill ladder. This was found to be necessary as otherwise the high noise in low-skill games could cause overfitting to the parameters. The results are reported in Section VII-B.

A. Multiplayer adjustments

Adjustments are required for games with more than 2-players to compare metrics across different player counts. Firstly, for games with N-players the win rate of a single agent with a budget of $2x$ is measured against N-1 agents with a budget of x. This provides the raw data for the ladder or grid. Secondly the observed win rate must be adjusted to take account of the player count, as an agent with no skill difference will be expected to win 50% of 2-player games, 33% of 3-player games etc.

If there are N players in a game, with a single higher budget agent, and N-1 lower budget agents, then the expected win rate is 1/N for the higher budget agent and (N-1)/N for the lower budget agent. We standardise with Equation (5) so that these two win rates are equal, and the difference between them is 0.0 if skill makes no difference, and 1.0 if the higher skill agent wins all games.

We adjust the raw win rate of the better agent, p_{raw}, in an N-player game as follows:

$$S_{adj} = p_{raw} * N - (1 - p_{raw}) * \frac{N}{N-1} \quad (5)$$

This gives the desired $\mathbb{E}[S_{adj}] = 0$ if skill has no impact, and $\mathbb{E}[S_{adj}] = 1.0$ if the better agent wins all games.

VII. Results

A. Classic Skill Trace

Table II summarises the results for Skill Trace (ST) for both Classic and Tuned datasets. Differences of more than x4 between the two datasets are highlighted in *red*. In all cases, 2-player games show a higher ST than 3-player versions of the same game (and this reduces further for 4-Players, not reported here). The Spearman rank correlations, ρ, of ST with BGG complexity (and associated p-values) are reported separately for 2-player games and 3-player games. This is because, as is clear in the table, the ST for one game can vary between the two player counts.

Game	BGG	Classic ST 2P	Classic ST 3P	Tuned ST 2P	Tuned ST 3P
Sushi Go	1.16	0.225	0.038	0.189	0.050
Dots + Boxes	1.59	0.644	0.277	0.353	*0.012*
Puerto Rico	3.27	-	0.219	-	0.070
Connect4	1.19	0.287	-	0.282	-
Dominion	2.35	0.330	0.046	0.288	*0.219*
Can't Stop	1.15	0.096	0.014	0.028	0.003
Virus	1.06	0.125	0.048	*0.000*	*0.008*
Colt Express	1.83	0.108	0.039	*0.009*	*0.001*
Stratego	1.85	0.252	-	*0.010*	-
Catan	2.30	-	0.119	-	0.098
Poker	2.43	0.056	0.012	0.087	0.003
Love Letter	1.19	0.020	0.001	0.013	0.002
7 Wonders	2.32	-	0.012	-	0.006
Hearts	1.75	-	0.014	-	0.004
Diamant	1.11	0.001	0.000	0.002	0.000
Tic-Tac-Toe	1.29	0.000	-	0.000	-
Spearman ρ		0.27	0.21	0.36	0.42
p-value		0.40	0.50	0.24	0.16

TABLE II: Skill Trace (ST) for each game for 2/3 players. Not all games support both. BG $\in [1, 5]$ is the complexity rating from www.boardgamegeek.com. The bottom row reports the Spearman rank correlation of ST against BGG.

Games with tuned parameters similar to the default settings such as Connect4 and Tic-Tac-Toe unsurprisingly show the same patterns in both. Others can look quite different, e.g.:

- 2P Virus is a moderate skill game (ST of 0.125) under the Classic settings, but a very low-skill game under Tuned settings (ST of 0.0). A similar collapse in estimated skill occurs in 3P Dots and Boxes.
- 3P Dominion in contrast has an increased skill-level with the Tuned settings (0.22 from 0.05 in Classic).

Virus is a simple family game with a strong 'take that' mechanic that players use to attack other players by playing viral infection cards on their healthy organs. There is little planning involved, and it has the lowest BGG complexity rating of the games at 1.06 (in range 1 to 5). The low skill rating ascribed by the Tuned parameter is much more in line with this expectation than the Classic results. The main differences in the Tuned parameter settings are that the rollout length is very short and uses a score-based heuristic. This is also true of Dots and Boxes. In both these games the change of score 1-2 moves in the future reflects good play (damaging an winning opponent in Virus; scoring a complete Box in Dots and Boxes). Using the default MCTS setting of random rollout to the end of the game leads to a high-variance reward which needs many iterations to resolve the signal from the noise. Using quicker, shorter iterations gives a low-variance (and, for these games at least, reliable) signal that leads to much better play at low budgets. In Virus this rapidly picks up the key tactic of attacking the current winner while protecting one's own organs. In Dots and Boxes this short-term focus picks up

[2]https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/nlm

the key tactic of closing available 3-boxes, without creating any for the opponent to exploit. In 3-Players this suffices, while 2-Player Dots and Boxes has long-term strategy towards the end-game, which accounts for the higher ST here in both cases [27]. There is a qualitative difference between the 2- and 3-player games that is revealed with the Tuned dataset but not the Classic one.

In these games, the default setting is in a poor region of MCTS parameter space and the high ST arises because this poor policy makes solid progress as computational budget is increased. A small shift in MCTS parameter space gives a better policy that with a 32ms budget can easily defeat a Classic 1024ms agent. The agents rapidly reach the plateau of perfect play, and the estimated ST correctly plummets.

Dominion is an example of a similar effect, but where this increases the estimated ST. Dominion is an example of a 'deceptive' game in which scoring points early damages long-term success [28]. The crucial parameter changes are reduced randomness in rollouts with the use of MAST [21] and more strategic opponent modelling with MultiTree MCTS [22].

The rank correlation to BGG complexity ratings using the Tuned dataset is a little better than with the default Classic dataset; ρ increases and the p-values decrease in all cases. However this is still very poor, with p-values of 0.24 and 0.16. Recall that the target BGG complexity ratings are not true skill-depths, but an independent proxy. The fact that Tic-Tac-Toe is rated as 1.29, and 6th most complex of the 12 2-player games in Table II feels unintuitive and is a cautionary point against relying too heavily on the BGG ratings as a gold standard.

Another problem is the clustering of ST results below 0.02. This is due to the lower overall win rates in these mostly stochastic games compared to those in [4]. The squaring of the win rate makes the AUC component in (2) very small, and there is random noise in y with the slope of the regression line on just 7 points.

The results in this section show that expanding the algorithmic space provides better ST estimates than a fixed algorithm for some games. These results only use the 'ladder' data in line with the methodology of Browne 2022. The next section considers how the broader grid data can provide more information.

B. Model Fitting

Tables III and IV shows the parameters fit to each game. This is only reported here for the Tuned parameter settings based on the results in Section VII-A.

When constructing the model in Section V, M in Model 1 was defined to be the adjusted win rate achievable asymptotically as the computational budget increases. Hence values above 1.0 are nonsensical, and yet Table III has several examples where this is the case. These are twinned with low r values, which defines the rate of increase towards M. The low r means that the model fit only gets anywhere near M well beyond 1024ms and hence the range of the data. In these cases M is not usefully interpretable.

Game	BGG	ST	Model 1		Model 2		
			M	r	M(8)	r	β
Sushi Go	1.16	0.189	0.89	0.56	0.86	0.59	2400
Dots + Boxes	1.59	0.353	1.0	1.1	0.99	1.3	510
Connect4	1.19	0.282	1.0	0.96	1.0	1.1	680
Dominion	2.35	0.288	1.0	1.2	1.0	1.7	200
Can't Stop	1.15	0.028	2.0	0.14	0.68	1.4	6.8
Virus	1.06	0.000	0.06	0.04	0.03	1.1	0.80
Colt Express	1.83	0.009	1.5	0.13	0.49	0.99	14
Stratego	1.85	0.010	1.5	0.12	0.45	1.0	13
Poker	2.43	0.087	0.14	1.2	0.14	1.3	520
Love Letter	1.19	0.013	0.80	0.19	0.35	0.25	78
Diamant	1.11	0.002	0.78	0.08	0.17	1.4	2.2
Tic-Tac-Toe	1.29	0.000	0.17	0.10	0.05	0.59	0.90
Spearman's ρ		0.36	-	-	-	-	0.40
p-Value		0.24	-	-	-	-	0.20

TABLE III: ST and new parameters for 2-Player games. M(8) is reported for Model 2 instead of M, as this is interpretable as the maximum adjustable win rate against an 8ms opponent.

Game	BGG	ST	Model 1		Model 2		
			M	r	M(8)	r	β
Sushi Go	1.16	0.050	0.57	0.34	0.44	0.53	270
Dots + Boxes	1.59	0.012	0.35	0.08	0.13	0.32	2.3
Puerto Rico	3.27	0.070	0.53	0.90	0.52	1.3	180
Dominion	2.35	0.219	0.52	0.89	0.51	0.96	1400
Can't Stop	1.15	0.003	1.0	0.11	0.28	1.1	6.3
Virus	1.06	0.008	0.14	0.11	0.05	0.59	1.5
Colt Express	1.83	0.001	1.0	0.10	0.29	0.60	22
Catan	2.30	0.098	0.70	0.31	0.66	0.33	2900
Poker	2.43	0.003	0.29	0.15	0.10	0.99	16
Love Letter	1.19	0.002	0.31	0.13	0.11	0.66	18
7 Wonders	2.32	0.006	0.62	0.25	0.35	0.70	60
Hearts	1.75	0.004	1.5	0.10	0.42	0.57	27
Diamant	1.11	0.000	0.35	0.08	0.09	0.78	2.1
Spearman's ρ		0.42	-	-	-	-	0.62
p-Value		0.16	-	-	-	-	0.03

TABLE IV: ST and new parameters for 3-Player games.

Introducing the third parameter, β, in Model 2 removes the problem, increasing r and moving M, and $M(b)$, into a reasonable range. The problem with Model 1 is that it tries to fit the same M and r for all opponents and cannot adjust for lower-skill games in which this pattern flattens out against better opponents and perfect play is approached within the available budget.

The example of 2-player Can't Stop in Figure 3 illustrates this. For each opponent from 8ms to 64ms this plots the win rate of increasingly better agents (doubling, quadrupling etc. the budget). By 64ms the opponent is playing a very good game, and multiplying the budget by any given factor is much less helpful than with a lower budget opponent. This is in line with the low β of 6.8 - $M(b)$, the upper performance limit, decreases rapidly as the opponent (b) gets better.

Contrast this with 2-player Sushi Go in Figure 3. Here the strength of the fixed opponent makes no change to the impact of budget increases; doubling the budget always gives the same increase in performance as represented by the very high β. This is interpretable as Sushi Go is a game of high skill as there is no sign of perfect play being approached. Can't

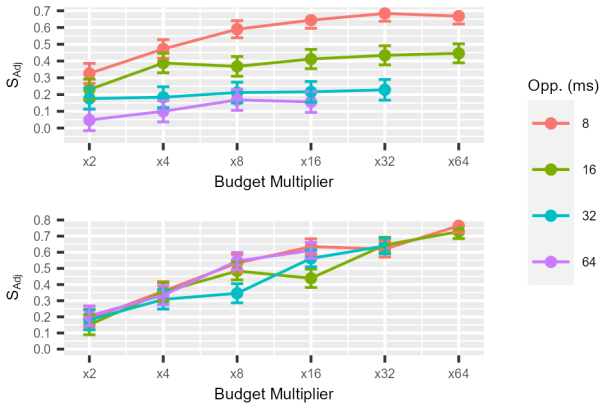

Fig. 3: The empirical data for 2P Can't Stop (top) and Sushi Go (bottom) illustrate games with low and high β respectively. Compare with the theoretical model in Figure 2.

Stop is a game of lower skill, with perfect play approached at relatively low budgets. These plots of the empirical data are good matches with the theoretical plots in Figure 2 for low and high β games.

Sushi Go is a simple game has a BGG complexity rating of only 1.16. This is difficult to reconcile with the high β in Table III, and the relatively high ST in Table II. This may indicate there are hidden depths to the game, or, more likely in our opinion, that it is tractable to the increasing depth of search of an MCTS algorithm with perfect card recall in a way that is less interesting to target human players.

A third example of 3P Virus in Figure 4 shows an low-skill stochastic game. The data has noisy fluctuations around a close to 0% adjusted win rate. Without a small amount of L2 regularisation when fitting the model parameters this can lead to overfitting with arbitrarily large value for β and M.

One clear advantage of the model over ST is that M and β in Model 2 help distinguish between the skill depth and stochasticity of a game. Poker and Love Letter both have high levels of uncertainty in the outcome due to the random shuffle of a deck of cards; many games can be won by a weak player who at least knows enough to exploit a good hand. This feature of the game is encapsulated in a low M, and β can be high if there is a consistent reward to higher skill even if M is low. With ST these two effects are merged, and both games have a low ST because the win rate can never approach 100% as it can in more deterministic games. There remain some inconsistencies in β in Tables III and IV, with the value of 3-player Poker being much smaller than 2-player Poker; but this is a step in the right direction.

Rank correlation fits to the BGG ratings on β improve on the Tuned ST in Section VII-A, with ρ values both increasing, and corresponding p-values decreasing, especially for 3-player games. The issues with noise for low levels of ST have been largely addressed by fitting the three parameter model to 28 data-points, versus a 2-parameter model to 7 points. The issues

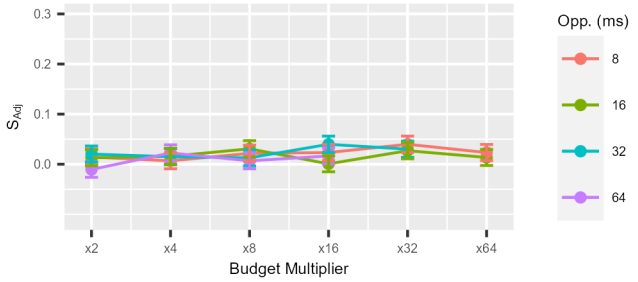

Fig. 4: An example of a low skill game, 3P Virus. The adjusted win rates fluctuate around 0% and 'perfect play'.

with BGG ratings as a proxy for skill-depth are unaffected.

VIII. Conclusion

We applied the Skill Trace (ST) methodology of [4] to a new set of games with imperfect information and more than 2 players. Extending the algorithmic space beyond a single set of default parameters by tuning for each game individually makes a significant change in the rating of some games where the defaults are in a poor area of parameter space for that game. This does not however improve the overall match to human estimates of game skill depth, using BGG complexity ratings as a proxy.

We then constructed a three-parameter model based on 5 axioms of expected game outcomes with changing skill level. This provides a slightly better match to human estimates and helps to disentangle aspects of the skill of a game:

- The maximum win rate a good player can achieve against a less skilled opponent (M). Games in which stochasticity is important, such as the deal of cards in Hearts or Poker are expected to have low M, while games with perfect information such as Connect 4 should have M of one. A low M does not mean a game has no skill depth, but that a better player can only be detected over a large number of games so that the stochastic effects balance out.
- The skill depth of the game (β). A small β indicates a game with rapidly diminishing returns of additional cognitive effort. We expect a game like Chess or Go to have $\beta \to \infty$, so doubling the computational effort gives a constant improvement in performance against *any* opponent. Separating this from M is helpful in games like Poker which can have low M and high β.
- The rate at which we approach M (r). The larger r is, the greater the impact of doubling the budget in performance terms. A small r means there is a small difference in performance for doubling the budget, and we need to multiply by a much larger factor to see an improvement, even if asymptotically the same M level will be reached. This is less directly interpretable compared to β or M.

A key weakness in these metrics in that they are in the context of MCTS, and just measure the effect of increasing the computational resources in this specific algorithmic context. MCTS does not seek to model the way a human plays a

game; and nor is it the best algorithm to play these games. Counterfactual Regret Minimization (CFR) methods will likely be better at Poker than the Information Set MCTS we use [29]; and in some perfect information games with shallow traps, such as Dots and Boxes, minimax search may be better. Future Work can expand this algorithmic space further, in particular with counterfactual regret-based techniques. This may also resolve the noted discrepancy on the lower β value for Poker with 3 players compared to 2 players.

Restricting a different resource, for example a memory limit on the number of expanded nodes, is also a natural further step to see if this gives a different rating of the games. What lies behind a high win rate, the important skills of the player, may vary between games. Some games will reward memory skills to recall cards played, others reward cognitive/planning ability or psychological understanding of other players (a form of opponent modelling). In many games all of these, and more, are important, and a low level of skill in one might be offset by a high-level in another. The approach takes no account of intransitivities in game-play, as another interesting characteristic of game skill-depth; nor can it currently help distinguish between the relative importance of memory/planning/bluffing/probabilistic understanding related skills in a given game.

One interesting idea would be to build on the work of [12], which used information-theoretic analysis to find a subset of single-player GVGAI games that best distinguished between different game-playing algorithms. Each of these games could represent a different set of skills, and be used to distinguish algorithms by the skills they are good at. The corresponding algorithms could then be used to derive skill measurements for games that disentangle the relative importance of these different skills.

The methods here are inherently slower than the quick methods in [4]. The main computational overhead is not the calculation of the grid of values between agents instead of a ladder, but the need to tune over the algorithmic space for each game, budget and player count.

Despite these important caveats, this work improves on previous measurement techniques and the models presented provide new insight into the skill of different board games, even if these do not always agree with human judgement.

REFERENCES

[1] R. Herbrich, T. Minka, and T. Graepel, "TrueSkill™: a Bayesian skill rating system," in *Advances in neural information processing systems*, 2007, pp. 569–576.

[2] M. E. Glickman and A. C. Jones, "Rating the chess rating system," *CHANCE-BERLIN THEN NEW YORK-*, vol. 12, pp. 21–28, 1999, publisher: SPRINGER INTERNATIONAL.

[3] F. Lantz, A. Isaksen, A. Jaffe, A. Nealen, and J. Togelius, "Depth in strategic games," in *AAAI Workshops*, 2017.

[4] C. Browne, "Quickly Detecting Skill Trace in Games," in *IEEE Conference on Games (CoG)*, 2022.

[5] S. Tavener, "UCT Skill Ladders," 2020. [Online]. Available: http://mrraow.com/uploads/AiAiReports/uct_skill_ladders.html

[6] G. S. Elias, R. Garfield, and K. R. Gutschera, *Characteristics of games*. MIT Press, 2012.

[7] J. Schell, *The Art of Game Design: A book of lenses*. CRC press, 2008.

[8] J. M. Thompson, "Defining the abstract," 2000. [Online]. Available: http://jnsilva.ludicum.org/TJ/TJ1920/Defining_the_Abstract.pdf

[9] J. Glenn and R. Brunstad, "Automatic Playtesting for Yahtzee," in *2020 IEEE Conference on Games (CoG)*. IEEE, 2020, pp. 760–763.

[10] R. Volkovas, M. Fairbank, J. R. Woodward, and S. Lucas, "Extracting learning curves from puzzle games," in *2019 11th Computer Science and Electronic Engineering (CEEC)*. IEEE, 2019, pp. 150–155.

[11] M. Stephenson, D. Perez-Liebana, M. Nelson, A. Khalifa, and A. Zook, "Game complexity vs strategic depth," 2019, publisher: National Institute of Informatics, Japan.

[12] M. Stephenson, D. Anderson, A. Khalifa, J. Levine, J. Renz, J. Togelius, and C. Salge, "A continuous information gain measure to find the most discriminatory problems for ai benchmarking," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020.

[13] D. Apeldoorn and V. Volz, "Measuring strategic depth in games using hierarchical knowledge bases," in *2017 IEEE Conference on Computational Intelligence and Games (CIG)*. New York, NY, USA: IEEE, Aug. 2017, pp. 9–16. [Online]. Available: http://ieeexplore.ieee.org/document/8080409/

[14] T. S. Nielsen, G. A. Barros, J. Togelius, and M. J. Nelson, "General video game evaluation using relative algorithm performance profiles," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2015, pp. 369–380.

[15] E. Y. C. Chen, A. White, and N. R. Sturtevant, "Entropy as a measure of puzzle difficulty," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 19, 2023, pp. 34–42, issue: 1.

[16] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[17] R. Ramanujan, A. Sabharwal, and B. Selman, "On Adversarial Search Spaces and Sampling-Based Planning." in *ICAPS*, vol. 10, 2010, pp. 242–245.

[18] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *European conference on machine learning*. Springer, 2006, pp. 282–293.

[19] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[20] V. Lisy, "Alternative Selection Functions for Information Set Monte Carlo Tree Search," *Acta Polytechnica*, vol. 54, no. 5, pp. 333–340, 2014.

[21] Y. Bjornsson and H. Finnsson, "CadiaPlayer: A Simulation-Based General Game Player," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 1, pp. 4–15, Mar. 2009. [Online]. Available: http://ieeexplore.ieee.org/document/4804731/

[22] J. Goodman, D. Perez-Liebana, and S. Lucas, "MultiTree MCTS in Tabletop Games," in *2022 IEEE Conference on Games (CoG)*. IEEE, 2022, pp. 292–299.

[23] ——, "Following the Leader in Multiplayer Tabletop Games," in *Proceedings of the 18th International Conference on the Foundations of Digital Games*, 2023, pp. 1–11.

[24] S. M. Lucas, J. Liu, and D. Perez-Liebana, "The N-Tuple Bandit Evolutionary Algorithm for Game Agent Optimisation," in *IEEE Congress on Evolutionary Computation (CEC)*, Rio de Janeiro, Feb. 2018, arXiv: 1802.05991. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8477869

[25] P. I. Cowling, E. J. Powley, and D. Whitehouse, "Information Set Monte Carlo Tree Search," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 2, pp. 120–143, Jun. 2012. [Online]. Available: http://ieeexplore.ieee.org/document/6203567/

[26] R. D. Gaina, M. Balla, A. Dockhorn, R. Montoliu, and D. Perez-Liebana, "TAG: A Tabletop Games Framework," in *Proceedings of the AIIDE workshop on Experimental AI in Games*, 2020.

[27] J. K. Barker and R. E. Korf, "Solving dots-and-boxes," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[28] D. Anderson, M. Stephenson, J. Togelius, C. Salge, J. Levine, and J. Renz, "Deceptive games," in *International Conference on the Applications of Evolutionary Computation*. Springer, 2018, pp. 376–391.

[29] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, "Regret minimization in games with incomplete information," in *Advances in neural information processing systems*, 2008, pp. 1729–1736.