Following the Leader in Multiplayer Tabletop Games

James Goodman
Diego Perez-Liébana
Simon Lucas
james.goodman@qmul.ac.uk
diego.perez@qmul.ac.uk
simon.lucas@qmul.ac.uk
Queen Mary University of London
London, United Kingdom

ABSTRACT

In a two-player zero-sum game, players classically want to maximise their chance of winning. When a game has more than two players, using the binary win rate as an objective is no longer such an obvious choice. A player might instead have the objective of doing as well as possible in terms of ranked order, or in maximising their score. We investigate the impact of different game-agnostic objectives in several popular tabletop games, and whether it can be better to use the game score as a proxy for winning. We find that the games considered largely fall into two groups. In one it is helpful to focus just on one's own score during the game, and then shift to beating opponents only in the end-game. In the other, larger, group it is better to 'Follow the Leader' and constantly track one's relative position to the opponents throughout the game.

Using the score as a proxy enables rollout simulations in Monte Carlo (Tree) Search to be terminated early, and we investigate the optimal length of these. Games in which long-term planning is required, or when the full score is only known at the end of the game benefit from a full rollout, while games with adversarial counter-moves benefit from a short rollout length.

CCS CONCEPTS

• **Computing methodologies** → *Multi-agent planning*; **Heuristic function construction**; **Game tree search**.

KEYWORDS

tabletop games, heuristic, MCTS, multi-player

ACM Reference Format:
James Goodman, Diego Perez-Liébana, and Simon Lucas. 2023. Following the Leader in Multiplayer Tabletop Games. In *Foundations of Digital Games 2023 (FDG 2023), April 12–14, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3582437.3582449

1 INTRODUCTION

When playing Chess, Go, Checkers, or any two-player zero sum game, the standard objective of a player, whether human or AI, is to win, or draw if that proves impossible. When a game has more than 2 players this simple objective becomes muddier if the game has some ordinal ranking of final positions. It seems reasonable that a player prefers to come 2^{nd} over 3^{rd}. But what if they spot an opportunity in the game that gives them a slim chance of winning instead of taking second place, but which will relegate them to last position if it fails? A 'Hail Mary' pass of this type is clearly a good call in a 2-player game, but whether this is true with more players is more contextual on the current position and the chosen objectives of the players. (A 2-player game where a draw is possible shares partly in this dilemma if comparing a 'safe' draw to a remote-odds chance of victory; but in this case the decision is usually straightforward given the symmetry between winning and losing. The key difference in a multi-player game is that 'losing' is not a singleton set.)

Most work on agents to play games uses the convention of +1 for a Win and -1 for a Loss as the objective the agent seeks to optimise. We investigate the impact of changing the heuristic function used by Monte Carlo Tree Search agents in a suite of ten tabletop games with more than two players. This heuristic function is used to value states reached in search, and hence is the objective the agent is seeking to optimise. Tabletop board and card games usually have a concept of score, with the highest scoring player at the end winning. The score of the game has some potential advantages over the win/loss result:

(1) It provides more information than the 1-bit of win/loss (or 1.5 bits of win/loss/draw), and allows for distinctions between losing players, so that second place is valued more than third.
(2) It provides a signal earlier in the game than the very end, and can be used to value a non-terminal state.
(3) It addresses a common AI pathology of acting randomly due to lack of any reward signal once the game is clearly lost. Agents will still act to minimise the scale of this loss.

A downside of optimising score is that it may not in fact help a player to win the game. In a game like Poker the 'score' is direct financial gain and the true objective, while winning is arguably of no (direct) interest. This does not hold generally for tabletop games played without this financial incentive, and here score is only a proxy for winning or placing higher up the end-game ranking. For example, in Dominion good players deliberately avoid increasing their score in the early game so they can maximise their ability to get even more points in the end-game. In this case a short-term focus on score will lead to poor play. Many tabletop games have been critiqued as being 'multiplayer solitaire' [26], meaning that the multiple players in the game do not interact strongly, and good play can result from just focusing on just one's own score. For games with this characteristic the additional information of using the Score over a binary win/loss could be beneficial.

For any game it is possible to construct or learn a game-specific heuristic function that estimates the likelihood of a win from the current state. Here we are interested in what *game-agnostic* heuristics work well across many different games[1] and are useful when the resources for this game-specific work are not available.

We ask the following questions:

(1) Is using the final score a better general objective than the binary win/loss rate in these games in terms of actually winning?
(2) Does using the score as an objective lead to a higher score, i.e. does targeting this actually optimise the score in the final result?
(3) Using a score enables a rollout to be terminated early with a computational saving, whereas using the win rate requires rollout to the end of the game. How quickly should we terminate, given that score is only a proxy for winning?
(4) Does the effect of a heuristic vary with player count, and is any effect noticeably different for two-player games, in which there is no benefit of coming second?

We do not (yet) ask the question about the behavioural differences of players with different objective functions. This is future work. Using automated AI agents to playtest games could be much enhanced with agents of different behaviours that better match the variety of human objectives, and could also inject variety into AI opponents on digital implementations of multiplayer games.

There is relatively little work on game-agnostic heuristics suitable for multi-player games, and we make three main contributions. Firstly we experimentally show that in general across 10 different games it is better to use a heuristic that is based on the score, but crucially with a bonus for also winning. Secondly we show that simple heuristics that amalgamate winning with the attained score are generally better than using either on its own. Thirdly we show that the best performing heuristic splits these games into two groups. One in which there are adversarial counter-moves or actions that directly damage one opponents, and the other in which players focus mainly on their own score and position and can pay less attention to opponent actions.

2 BACKGROUND

2.1 Tabletop Games and TAG

TAG is a framework for the implementation of modern Euro-style board and card-games [10]. These games are of research interest both because of their high popularity, and because they often have high levels of hidden information, stochasticity and support more than two players. Using TAG has the advantage that different algorithms can be tested on a variety of very different games to see how they generalise, and what types of game they may be suited to.

[1]This agnosticism is specifically within the galaxy of tabletop board and card games with more than 2 players, admittedly a subset of the wider games universe.

The 10 games used in this work are listed below, for more detailed descriptions see https://boardgamegeek.com/:

- Colt Express (2014). Players plan a partially observable sequence of actions to rob a train. These plans are executed in a second phase, with players robbing train passengers and each other. Actions in the plan need to be executed in the light of the current situation, not necessarily the one anticipated when planning.
- Dots and Boxes (1889). Players take turns to connect adjacent dots on a 7×5 grid. A player scores 1 point for completing the fourth line around one square, and goes again.
- Diamant (2005). A simultaneous-move game. Players decide in successive rounds stay in or leave a cave. The fewer players that stay, the more treasure each gets. Those that leave divide up treasure left over from previous rounds, but cannot participate in future rounds. If the cave collapses, then all players still inside lose their collected treasure.
- Dominion (2008). A Deck-building card game in which a player first needs to build an 'engine', and then use this to gain victory points. Each game uses 10 out of 25 card types which interact in different ways. The experiments here use the set recommended for a player's first game.
- Exploding Kittens (2015). Players are knocked out if they draw an Exploding Kitten. Cards can be played to peek at and manipulate cards in the draw deck. The winner is the last one standing.
- Love Letter (2012). A game of role deduction. Players target opponents to gain information or knock them out. The highest surviving card wins the round, and the game ends when one player has won 5 rounds.
- Poker (1810). Uses classic Texas Hold'em rules.
- Sushi GO! (2013). Simultaneous-move set collection. After playing a card to their tableau, all players pass their remaining hand clockwise, so that ultimately the game becomes perfect information. After 7 cards have been played, the value of the sets in a player's tableau is scored.
- Uno (1971). Players match suits and numbers to discard all cards from their hand before their opponents. Points are scored for the value of cards held by opponents.
- Virus (2015). Players play cards from their hand to construct a healthy body of four organs cards, and use virus cards to infect the organs of other players. The score is the number of healthy organs, and four are required to win.

Not all games have an inherent score and/or ranking. For example in Chinese Checkers the first player to get all 10 of their pegs across the board wins, and all other players lose equally. Such games with one-winner, many-losers, formally fall outside out target subset. In the games used here, Exploding Kittens is a knock-out game in which the winner is the last player standing. In this case we *can* still give each player an inferred score as the number of players knocked out before them, so the score is the position in the final end-game ranking.

2.2 MCTS

Monte Carlo Tree Search (MCTS) [4, 6, 7] has been used in many games. It searches the forward game tree by sampling. On each iteration four steps are followed:

(1) Selection. Select an action to take from the current node. If all actions have been selected at least once then the best one is picked using the Upper Confidence for Trees equation [18]:

$$J(a) = Q(a) + K\sqrt{\frac{\log(N)}{n(a)}} \tag{1}$$

The action a with largest $J(a)$ is selected. N is the total number of visits to the node; $n(a)$ is the number of those visits that took action a; $Q(a)$ is the mean score for all visits to the node that took action a; K controls the trade-off between exploitation, and exploration choosing actions with few visits so far. This step is repeated down the tree until a node is reached with previously untried actions.

(2) Expansion. Pick one of the untried actions (using an expansion policy, which may be random), and expand this, creating a new node in the game tree.

(3) Rollout. From the expanded node, take actions using a rollout policy (which may be random) for a number of steps, or to the end of the game, to obtain a final score.

(4) Back-propagation. Back-propagate this final score up the tree through all nodes visited this iteration. Each node records the mean score of all iterations that take a given action from that node as $Q(a)$ that will affect future Selection steps. Once the time budget has been used the action at the root node with either the highest score or most visits is executed.

Most of the games have hidden information, and only Dots and Boxes is a full perfect information game. The hidden information in the game is obfuscated by the TAG game engine, and not visible to the MCTS agent when it makes a decision. Throughout this work we use Information Set MCTS (IS-MCTS) which constructs the tree in terms of information sets [8]. Information Sets are defined by the sequence of visible actions to reach the current state, an 'open-loop' approach [25]. In this formulation, with no differential visibility of actions and hence tree transitions, Single/Multiple Observer IS-MCTS variants are identical [8].

The games also have a very wide range of scores, from a maximum of over 5000 in Colt Express, to just 4 in Virus. To standardise across all games, K in the UCT Equation (1) is kept at 1, and the score, $Q(a)$, automatically scaled to the range [0, +1] based on the min/max scores observed. This also avoids the need to tune K for each heuristic as well as each game, as these can have very different scales, for example a win/loss heuristic is always in -1, 0, +1.

Two of the games used, Diamant and Sushi Go, have all players moving simultaneously. We use Sequential UCT for this (with IS-MCTS), in which each player takes their move, and then assumes that the other players can condition their moves on this. This can lead to conservative play, but has been successful in many games [34].

2.3 Previous Work

The first uses of MCTS always rolled out to the end of a game, and back-propagated a +1 for a win and -1 for a loss. This was one of the key advantages of the method over minimax search, which required an often quite sophisticated state heuristic function to evaluate states [3, 6, 7].

The first use of a heuristic function was to decide when a rollout was won/lost and computational resources could be saved by early termination [20, 36]. These still back-propagated a +1/-1 for presumed win/loss. Early termination has been found generally useful in other works, although in 2-player games [2, 16, 19]. The first back-propagation of the result of a heuristic function with early termination as far as we can tell is Ramanujan and Selman, 2011, with a hand-crafted heuristic in Mancala [27]. More recent work has sometimes found an extreme case of no rollouts works best, and heuristic evaluation of the state at expansion has been used, for example in General Game Playing [22] and Go [30].

Here we are concerned with useful game-agnostic heuristic functions that do not need to be developed/tuned for each game, and are useful out-of-the-box for a range of games. Hence we do not review the extensive literature on crafting game-specific heuristics.

Other than winning or losing, most games include a 'score' of some sort, and this is a natural alternative as a reward signal to back-propagate. Kloetzer et al. 2007 mix the final score with win/loss in 2-player Amazons, which is similar to our final conclusions[17], but in a 2-player game with rollout to game-end. The final score with full rollouts has also been used in 3/4-player Spades and Hearts [32]. The difference in score between 2 teams in Spades has been used, with termination at the end of a hand once the result is known [35]. Pepels et al. 2014 try an interesting approach of using the score in the guise of a control variate (amongst other metrics) as a qualitative bonus in Amazons and other games [24].

The score difference (to the score of the other team) has been used as well as win/loss in the 4-player/2-team card game Scopone [9]. This is the same as our 'Leader' heuristic introduced in Section 3, and was found to provide a small but significant improvement in the win rate. In the 7-player game '7 Wonders', the win rate with rollout to the end of the game was retained, with the score used only to bias the selection policy in the tree [28]. Similarly the simple win rate was used as the objective in MCTS applied to 4-player Settlers of Catan, with heuristics used to bias search and rollout [33]. In 4-player Kingdomino [11] compare three different heuristics using flat Monte Carlo (MC) search (to game-end), and find that using the player score, or score relative to the winner, perform better in terms of both win rate and score than a binary win/loss reward. The authors also use MCTS, but only with the binary win/loss reward, and this performs poorly against flat MC. Tests of using the final score in Go suggested that a binary win/loss led to better performance [12]. In the 2-player BlokusDuo, a heuristic that processes the score through a logistic function so that it is on the same range as a 0/1 loss/win has been shown to work well [29].

Other game-agnostic heuristics are possible. A set of 13, many specific to grid-based counter games, have been investigated in the Ludii framework [31]. In the GVGAI framework, game-agnostic heuristics such as exploring the map, or interacting with as many sprites as possible, have been used [14]. These examples are agnostic within the genres of games in those frameworks, but do not generalise in most cases to the tabletop board and card games in TAG.

Reinforcement Learning (RL) is a different algorithmic paradigm to the planning of MCTS with a forward model, but also has to address the question of what objective training should maximise. There is not space for a full review of objectives used in RL, but it has often been necessary to reduce the range of the reward signal to stabilise learning, as in Atari where the score change is often truncated to +/- 1 point [23]. In Go, the binary win/loss is most commonly used [30], although the score (difference) in final stones has been used to generalise across games with different komi (handicap) settings [37]. Outside this scenario, using the Score reduced the performance as measured in terms of matches won.

3 EXPERIMENTS

For each game we ran a round robin tournament between MCTS agents that differed only in the heuristic used to value a state at the end of a rollout. Five heuristics were used:

- Win. +1 for a Win, -1 for a Loss. A Draw, or any non-terminal state is valued at zero.
- Score. Uses the raw game score.
- Score+. Uses the raw game score, but then adds a 50% bonus if the player wins (in a terminal state), and subtracts 50% if the player loses (in a terminal state). The purpose of this bonus is to incentivise winning and avoiding a preference for a higher score even if this means losing the game.
- Leader. This uses the difference between the player's game score and that of the current highest scoring *other* player. This is positive if winning, and negative otherwise. This is multiplied by 150% if the game state is terminal.
- Default. The games implemented in TAG all have a 'default' heuristic hand-crafted by the game's implementer. These vary in sophistication, but are +1 for a Win, -1 for a Loss, with non-terminal states valued within that range, so are enriched versions of 'Win Only'.

As outlined in Section 2.3, MCTS as originally implemented ran rollouts to the end of a game, and then back-propagated the win/loss. Previous work in tabletop games has shown that tuning MCTS parameters leads to stronger performance if the rollout is terminated early, say after 30 or 100 actions [13]. In long-running games this trades-off the benefit of more iterations and reduced variance in the reward signal against a potential bias in the reward signal received [19].

For each game, the rollout length was tuned to give the highest win rate using a heuristic. Tuning was over the set 0, 1, 3, 7, 10, 20, 30, 50, 100, 200, 300 using the NTBEA optimisation algorithm [21] with 500 games per run. To avoid over-fitting to a specific scenario, the rollout was tuned for each of the Score, Leader and Score+ heuristics, player counts of 2 and 4, and for 40ms and 200ms computational budgets. The final reported figure in Table 1 is the mode of the combined distributions, although in practice there were no major differences for a given game across these scenarios.

The final tuned values are shown in Table 1. Four games marked as 'Any' returned flat distributions across rollout lengths between

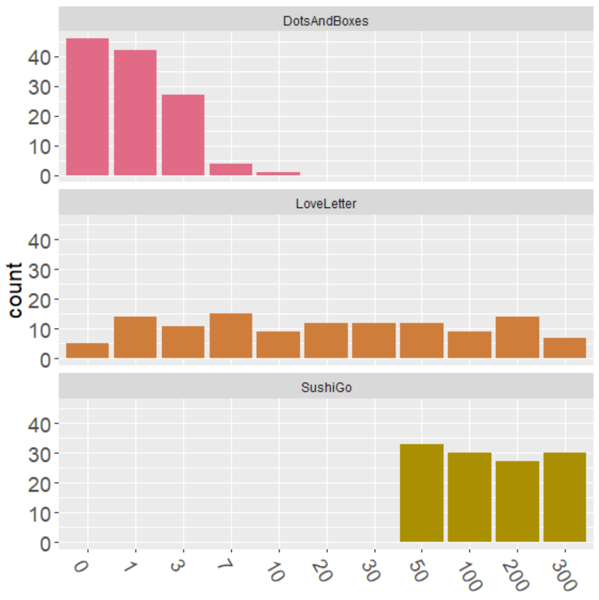

Figure 1: The results of optimising rollout length for three games. The x-axis is the rollout length. The Count on the y-axis is the number of independent optimisation runs that returned that value. Each individual run had a budget of 500 trials.

Game	Rollout	Entropy	2P	3P	4P
Dots + Boxes	0	1.2	82	82	82
Uno	7	1.7	1800	1400	1300
Virus	10	2.1	30	40	65
Sushi Go	100	1.4	60	80	90
Colt Express	200	1.8	80	130	190
Dominion	300	1.6	180	250	260
Diamant	Any	2.3	45	75	110
Expl. Kittens	Any	2.3	80	80	80
Love Letter	Any	2.4	90	160	180
Poker	Any	2.4	15	30	50

Table 1: Optimal rollout length tuned for each game. The 'Any' games have no preferred rollout length, with similar performance for all values from 0 to 300. The average length of a game in actions for 2P/3P/4P (P = players), and the entropy of the distribution of rollout lengths from tuning are shown for reference.

0 and 300; i.e. any given optimisation run would return a given value more or less at random. Reporting a single tuned value as 'optimal' can be misleading if it is the result of one optimisation run; it is equally important to know when the value of the parameter is relatively unimportant to avoid putting undue weight on this 'optimal' value. Therefore Table 1 includes the entropy of the final distributions, with a minimum of 0.0 for all runs producing the same recommendation, and a maximum of 2.4 for a completely flat

distributions across all 11 values. Figure 1 shows distributions of the optimised values for three example games to clarify this; Dots and Boxes has an optimal rollout between 0 and 3 (with a mode of 0), while Sushi Go! requires a rollout length of more than 50, which effectively means rolling out to the end of the game (see Table 1 for average game lengths). Performance in Love Letter is not significantly affected by the rollout length. For the four games with flat distributions, we somewhat arbitrarily use a rollout of 30 actions in later experiments.

For each game a round robin tournament was then run between the five different heuristics for each combination of:

- Player count of 2, 3, 4 or 5 (as permitted by the rules).
- MCTS budget of 40ms, 200ms and 1000ms per decision.

A rank between 1 and 5 is given to each heuristic in each tournament based on win rate. This is distinct from the rank an agent achieves in any individual game. The top rank goes to the heuristic that won most games in the tournament, and the 5th rank to the one that won fewest. Section 4 looks at a number of cuts of this data:

- The overall ranking of the heuristics. Are any dominant?
- By player count across all games. Is there a difference between 2-players and more than 2-players?
- The ranking of heuristics for each game. Can games be grouped into clusters?
- A deep-dive into the full results for one game in each cluster identified in the previous step.

The ranking of the five agents is used instead of the win rate for two reasons. Firstly it makes it easy to compare a tournaments with different numbers of players, which have different mean win rates (50% for 2-players, 33% for 3-players, etc.). Secondly, it equally weights all games in the averages. Using the win rate means a game where the winner has a 20% lead over other agents will swamp games in which the lead is only 2-5%. Win rates *are* reported in the game-specific results in sections 4.4.2 to 4.4.3.

One concern was that by terminating early the 'Win' heuristic, which only provides any information if a terminal state is reached, is handicapped compared to the others. To test this a set of tournaments was also run between the tuned agents, and agents using the four game-agnostic heuristics and rollout to game-end. The results of these (not reported here in detail) confirmed that using a full rollout for the games with short tuned rollout length always led to very poor performance, regardless of heuristic. For the other games, where the rollout length was unimportant or already tuned to be long, then there was little difference in the results, and no change to the order of heuristic performances reported in Table 5.

4 RESULTS

4.1 Overall

Table 2 averages ranks across all games for the five heuristics, both in terms of their win rate and final score. It shows that Win rate performs poorly, using the raw Score is better, but that it is better still to use Score with a bonus/penalty for winning/losing (either Leader, or Score+). Wilcoxon paired signed rank tests for each heuristic against the Leader result across all tournaments show that Leader and Score+ are much better than the other heuristics in wining games (p-Values < 0.0001). The evidence for a difference between Leader and Score+ heuristics is less strong at 0.04, and Section 4.3 takes this analysis down to individual games. Both Score and Score+, with their focus on personal score, do just as well as the Leader heuristic in aggregate at getting a high score despite being less good at winning.

Heuristic	Win Rank	pValue	Score Rank	pValue
Win	3.9	0.000	4.2	0.000
Score	3.1	0.000	**2.4**	0.235
Score+	2.6	0.039	**2.4**	0.441
Leader	**2.2**	-	**2.3**	-
Default	3.2	0.000	3.7	0.000

Table 2: Mean rank of each heuristic across all tournaments for win rate and score, with 1 being best and 5 worst. The p-Values are for difference to the Leader rank with a Wilcoxon signed rank test.

	2P		3P		4P		p-Values 2/3P	
Heuristic	W	S	W	S	W	S	W	S
Win	3.5	4.0	4.0	4.1	4.3	4.2	0.068	0.368
Score	3.4	3.0	2.9	2.5	2.4	2.4	0.086	0.746
Score+	2.9	2.6	**2.4**	2.5	2.5	2.4	0.301	1.000
Leader	**1.9**	**2.3**	**2.5**	**2.2**	**2.3**	**2.2**	0.092	0.560
Default	3.3	3.1	3.2	3.6	3.4	3.8	-	-

Table 3: Mean rank of heuristic across all tournaments by player count. W is the Win Rate rank, and S is Score. pValues are from Wilcoxon tests for a change from 2P to 3P.

The poor performance of 'Win' is expected for early rollout termination as these will only reach a reward signal in the later game, leading to initial random moves. However, it is still a poor performer when all rollouts go to game-end, only achieving a mean rank of 3.5 if the tournaments are run with a full rollout in all cases.

There is little difference between the Win and Score rankings of the heuristics, although using only the Score performs relatively better at maximising the Score (mean rank of 2.7), than winning games (mean rank 3.1). The Score+ and Leader heuristics both do better at winning games, and are just as good at scoring well. These averages are over all ten games and exhibit a wide variation at the individual game level as discussed in Section 4.3. Despite this the general conclusion remains that Score+ and Leader are better than Score, which is better than Win.

4.2 Player and Budget effects

Table 3 shows the mean rank achieved by the heuristics across all tournaments by player count. All of the games will take up to 4 players, but only 8 accept 5 players. To ensure comparability across player count figures, Table 3 only goes up to 4P.

To compare the relative performance from 2-players to 3-players, Table 2 includes the p-Values from Wilcoxon paired signed rank tests on the change in ranking of each heuristic from a 2-player environment to a 3-player one (N=30). This is not a powerful test

	40ms		200ms		1000ms		p-Values	
Heuristic	W	S	W	S	W	S	W	S
Win	4.2	4.5	3.8	4.1	3.6	3.9	0.010	0.011
Score	2.9	2.1	3.2	2.6	3.1	2.6	0.120	0.031
Score+	2.5	**2.2**	2.5	2.5	2.8	2.7	0.067	0.028
Leader	**1.9**	**2.2**	**2.3**	**2.3**	**2.5**	**2.3**	0.009	0.508
Default	3.5	4.0	3.2	3.5	3.0	3.5	-	-

Table 4: Mean rank of heuristic across all tournaments by budget. W is the Win Rate rank, and S is Score. pValues are for a change from 40ms to 1000ms.

as the majority of the rank changes are zero, and the p-Values show only a weak level of evidence that Leader, Win and Score vary in win rate averaged across all games, with Win and Leader *relatively* better at 2P, and Score with 3+P. For comparison, the same tests for a change in the Win ranking from 3P to 4P give consistent p-Values in the range (0.4, 1.0) compared to (0.01, 0.24) from 2P to 3P. In aggregate the Leader heuristic remains the best performer across the games for any player count.

Table 4 presents the same data split by the computational budget allowed for each decision by MCTS. The same broad conclusions apply. There is evidence that the Win heuristic becomes better on average as the budget increases as hypothesized (p-value 0.01), but the Leader heuristic remains superior across the budgets used.

4.3 Game-level effects

Table 5 gives the average rank achieved by each heuristic for each game. This shows the range in quality of the game-specific 'Default' heuristic, which performs well only in Diamant and Love Letter, while the Uno, Colt Express and Exploding Kittens default heuristics do comprehensively badly despite being much more complex than simply looking at the score. A hand-crafted heuristic may be quite poor in practice, and needs to be evaluated and tuned against simpler options.

The Win heuristic is of clear benefit only in Diamant. This has the smallest branching factor of all games, as each decision is always a binary one of stay or go. This means that the tree reaches much greater depths, and rollouts frequently reach the end of the game.

Overall the best performers are Score+ and Leader, which between them are best at winning in all games except for Diamant (Win) and Dominion (Score); and best at getting the highest score in all games except Dominion.

This game-level analysis provides more resolution on the difference between Score+ and Leader, which were not distinguishable when averaging across all games in Table 2. The differences in actual win rate between Score+ and Leader is significant in all games except Exploding Kittens in Table 2, but in one group Score+ is better than Leader, in the other it is worse.

Colt Express, Dots and Boxes, Diamant, Uno, Love Letter and Virus form one group, with Leader better than Score+. Dominion, Poker and Sushi Go form a group where the reverse is true. Exploding Kittens shows no significant difference between Score+ and Leader in Table 2. Exploding Kittens is a simple game in which the outcome is very random, with relatively little room for strategic

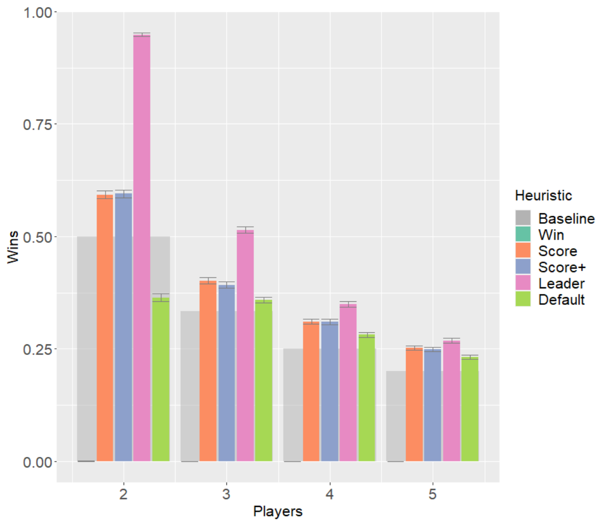

Figure 2: Dots and Boxes: Win rates with a 40ms budget

player decisions; all heuristics, except for the hand-crafted one, are tightly bunched in mean rank.

Within the groups, Diamant stands out as the only game in which 'Win' is the best heuristic, and Dominion as the only game in which 'Score' is best. In Dominion focusing only on one's own score leads to a higher winning rate than looking at score *and also* trying to win!

Both Score+ and Leader incentivise getting a high score *and* winning the game, but differ in the attention they pay to the opponents during the game. Score+ ignores opponents except at the end of the game (if this is within the rollout horizon) when it will try to be ahead of opponents to get the 50% uplift. There is no incentive to stop opponents gaining points during the early or mid-game if this does not affect the player's score.

In contrast the Leader heuristic pays constant attention to opponents through the game, and the reward signal is always a difference between the player's score, and that of another player (the current winner, or the closest competitor). This will incentivise blocking an opponent from gaining points even if there is no direct effect on a player's own score.

Both Leader and Score avoid the problem frequently observed of MCTS agents acting randomly when they can no longer win, for example [9]. The relative performance of these two heuristics groups the games into how 'multiplayer solitaire' the game is, when a player does well by focusing purely on their own position and score [26]. The games in which Leader is dominant are more directly adversarial in play throughout the game. The detailed examples of Dominion and Dots and Boxes below develop this point.

4.4 Game deep dives

This section looks at the results in detail of three games, covering both groups identified in Section 4.3, and the one game (Diamant) in which Win is the best heuristic.

Heuristic	Colt Exp.	Dots+Boxes	Diamant	Dominion	Expl.Kittens	Love Letter	Poker	Sushi Go	Uno	Virus
					Win Rate Rank					
Win	4.5	5.0	**2.1**	4.1	2.5	4.3	4.0	3.3	4.0	5.0
Score	2.4	2.8	4.6	**1.3**	3.0	3.8	2.7	4.1	1.8	3.8
Score+	2.4	3.0	4.0	1.7	**2.4**	3.3	**1.8**	**1.3**	3.0	2.9
Leader	**1.2**	**1.2**	2.9	3.6	2.8	**1.7**	3.8	3.1	**1.2**	**1.2**
Default	4.5	2.9	1.4	4.3	4.2	1.9	2.7	3.2	5.0	2.1
Score+|Leader	***	***	**	**		**	**	*	***	***
					Score Rank					
Win	4.4	5.0	2.8	4.6	2.5	4.4	4.7	4.6	4.0	5.0
Score	**2.0**	2.8	4.1	**1.2**	3.0	3.3	2.0	1.8	1.8	2.8
Score+	**2.0**	3.2	3.2	1.8	**2.4**	3.3	**1.9**	**1.2**	3.0	**1.8**
Leader	**2.0**	**1.2**	**2.2**	3.0	2.8	**1.7**	4.2	2.9	**1.2**	**1.7**
Default	4.6	2.8	2.8	4.4	4.2	2.2	2.2	4.4	5.0	3.8
Score+|Leader		***	**	**		**	***	**	***	

Table 5: Mean Win/Score rank of heuristics across all tournaments by Game. The top half is the mean ranking of a heuristic in win rate, and the bottom half is the mean rank in score over all tournaments for each game (9 or 12). Best performing game-agnostic heuristic for each game in bold. The 'Score+|<name>' lines indicate significance of differences across all tournaments between the Score+ and <name> heuristics using a Wilcoxon signed rank test, with * = p , 0.05, ** = p < 0.01, *** = p < 0.001. The comparison between Score+|Leader breaks the games into two groups, one in which Score+ is better, and one where it is worse.

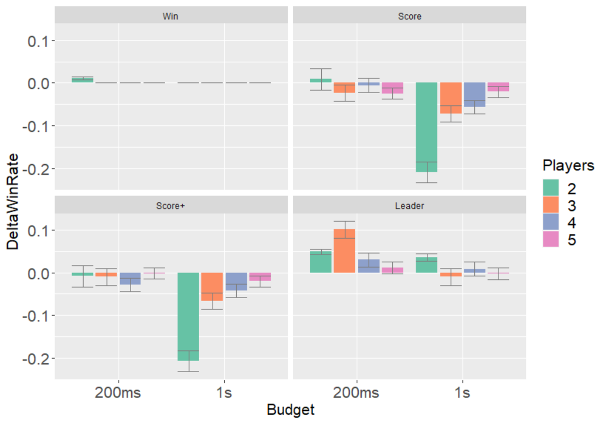

Figure 3: Dots and Boxes: Change to win rate with computational budget. These are changes to the win rate compared to the baseline at 40ms shown in Figure 2.

4.4.1 Dots and Boxes. This is an example from the group of games for which Leader is significantly better than Score+, both in win rate and score.

The number of actions in each game is 82 - the number of edges on a 7x5 grid, all of which must be filled in. This also means that the branching factor in MCTS declines uniformly from 82 in the first move to 2 in the penultimate move.

Figure 2 shows that a simple 'Win' heuristic wins no games, much as in Dominion, and that Leader is significantly better than all other heuristics, but with an advantage that declines sharply beyond 2-players. Score and Score+ are indistinguishable in performance, which suggests that incorporating a win bonus is not relevant to Dots and Boxes.

Dots and Boxes shows the most significant relative impact in performance with increasing budget as shown in Figure 3. As the budget increases, Score and Score+ get progressively worse. Detailed examination of game play reveals that the problem is due to directly adversarial counter-moves. In Dots and Boxes a Score-based heuristic will not realise it is a very bad move to set up a 3-sided box, allowing the next player to immediately score a point and take another turn. Consider the case in mid-game with one move that sets up a 3-box (and the rest do not). All moves except the 3-box moves give a short-term reward of 0, but the move that sets up the box will, once the tree extends back to our move (depth 2 in a 2-player game with the root at depth 0) give an occasional reward of +1.0 in the event that the opponent does not immediately fill in the box. Hence this poor move is encouraged. As the tree grows, MCTS will correctly model the opponent filling in the box to get a score of +1.0, leaving 0.0 to the root player. However, this 0.0 is a lower bound on the estimated value of the move to set up the 3-box, and at least some of the exploration iterations will have returned 1.0. Since the (better) moves that do not set up the 3-box are uniformly valued at 0.0, increasing the budget does not help as no penalty is applied for the gain in the opponent's score if ours is unchanged.

With the Leader heuristic this pathological behaviour does not occur. The sensible moves that avoid the 3-box still return 0.0, and the 3-box move will, as with Score have some transient iterations that give a value of +1.0. However, as the opponent is modelled to fill in the 3-box to get a point, the Leader heuristic will increasingly return a net change of -1.0. This will rapidly set the value of the action to be below 0.0, and the agent will correctly choose one of the other moves.

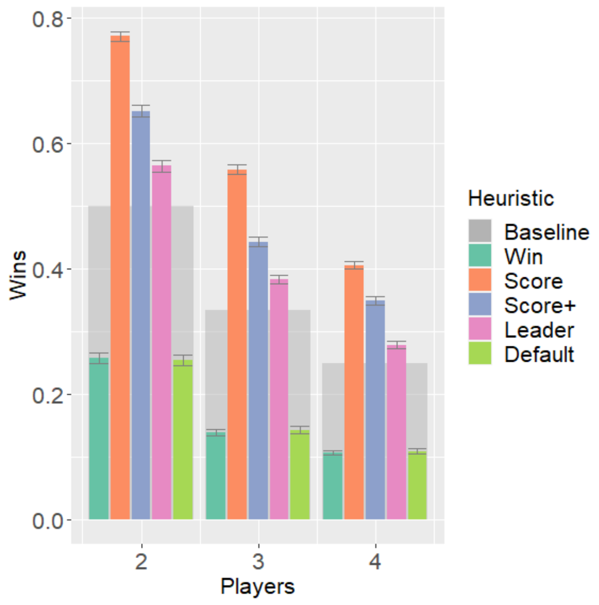

Figure 4: Dominion: Win rates with a 40ms budget. Error bars show 95% confidence intervals on the mean.

The underlying problem for Score and Score+ is the existence of adversarial counter-moves. A bad action on our side can immediately give an opponent an opportunity to score. Similar situations hold for the other games in the group:

- Virus. It is a good move to attack an opponent's healthy organ, reducing their points but not increasing ours directly.
- Colt Express. Like Virus, many moves help one's relative position by robbing or shooting other players - removing points from them, but not increasing one's own score.
- Uno. A player's score is linked to the number of cards they have in hand, the fewer the better. Some actions are good because they cause another player to gain cards into their own hand, even though one's own score is not directly affected.
- Diamant. Player actions interact on each turn decide how the available points are divided. An action to leave can give me a big benefit of picking up gems on the way out, rather than sharing 50% of a contingent reward for staying in the cave. But this allows the other player to gain 100% of that contingent reward, and increase their score by more than you increase yours. The Leader heuristic takes this into account, while Score+ does not.
- Love Letter. Player actions directly attack opponents to knock them out of the current round.

4.4.2 Dominion. This is an example from the group of games for which Score+ is significantly better than Leader, both in win rate and score. Dominion is the most extreme example, as using just the Score leads to a higher win rate.

Figure 4 shows the win rates of the different heuristics for different player counts with the lowest (40ms) budget per decision. An initial point to note is the very poor performance of 'Win'. The other agents are more similar in performance, but for every single tournament (all player counts and all budgets), either Score or

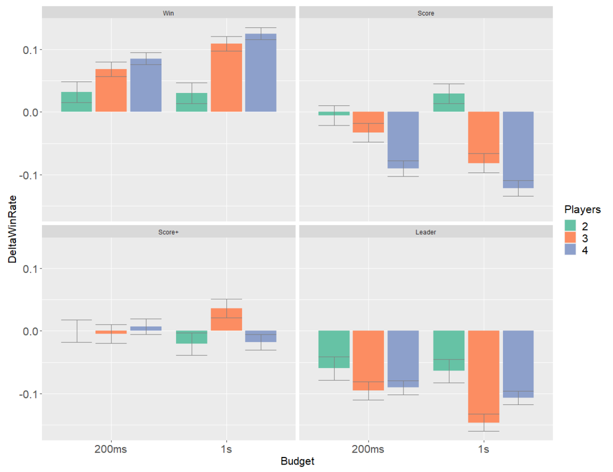

Figure 5: Dominion: Change to win rate with computational budget. Error bars show 95% confidence intervals on the mean difference. This is the change in win rate relative to the 40ms baseline in Figure 4.

Score+ wins, and also gets the highest score. The default heuristic in Dominion is a tuned weighting of nine hand-crafted state features, and the out-performance of these much simpler heuristics is notable. (The default is designed to work well for short rollouts, but even here it underperforms the much simpler Score+.)

One hypothesis is that by trying to out-score an opponent in the short-term, the Leader misses longer-term opportunities of a higher score. However, this cannot by itself be the problem, as both Leader and Score have the same time horizon to the end of the game. This feature of Dominion, that victory points in the early game should be avoided, explains why all heuristics perform best with a rollout to game end. The Leader heuristic tries to maintain a point lead at all times in the game. If all players gain points at about the same rate, then this may give a flat reward signal. It will also have a higher variance than the Score, as it is the difference between two (semi-)independent variables. Which of these effects has more of an impact is an open question.

There is no significant impact of player count in Dominion. In Figure 4 the relative rankings of Win, Score, Score+ and Leader are the same for 2, 3, or 4 players. Figure 5 suggests that Score and Leader do progressively worse for larger budgets and player counts, with Win increasing its performance from a low base.

Dominion has some actions (the Militia cards in the basic set used here) that affect other players, but these do not directly affect the player's score. Player interaction is mostly a race to build an engine to buy the end-game victory cards first. The other games in the Score/Score+ group also show a relative lack of adversarial or opponent-attack moves as seen in the Leader cluster, if not to the same extreme as Dominion:

- Sushi Go! A player cannot directly affect another player's score, although they can indirectly prevent another player gaining points by playing a card that would be useful to them. The better performance of Score+ here suggests this has less of an impact in Sushi Go! than the similar interaction noted in Diamant.

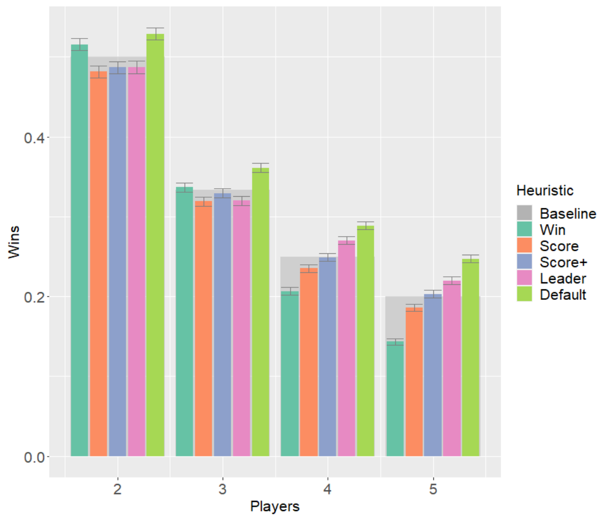

Figure 6: Diamant: Win rates with a 40ms budget

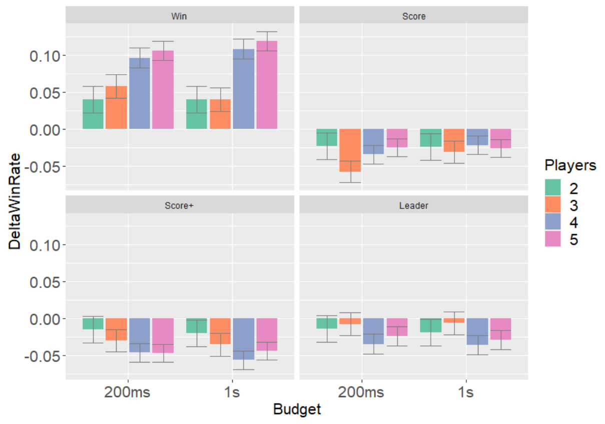

Figure 7: Diamant: Change to win rate with computational budget, compared to the 40ms baseline in Figure 6.

- Poker. Unlike the other games, Poker has a fixed number of 'points' available represented by the starting chips. During the game these are just reallocated between players, and maximising one's own score will inherently also reduce that of others. Using Leader would double-count a change in score in some cases.

4.4.3 Diamant. This belongs to the group in which Leader is better than Score+, but is also the one game in which the 'Win' heuristic does well.

Partially this is down to the small branching factor and length of a game. In Diamant each decision by a player is a binary choice of 'Stay' or 'Leave' (if they have already left, and other players are still in the cave, they just 'Pass'). As a result with 40/200/1000ms of budget, the tree depth reaches up to 16/19/25 on average. Each game on average has 25 actions per player. Adding in the rollout of 30 actions means that even with 40ms the 'Win' heuristic will return a non-flat reward signal for 90%/60%/45% in a 2/3/4 player game. This explains why the Win performance rapidly degrades in Figure 6 as the player count increases, and also why the performance of 'Win' increases markedly as the computational budget and hence effective search depth increases in Figure 7.

Diamant is one of two games (with Love Letter) in which the hand-crafted 'Default' heuristic does best overall. This is effectively the 'Win' heuristic with a small weighting for the current score added. This keeps the main reward signal, but provides some useful information in the start and mid-game too.

However, the tuning of rollout length for the game did not give a strong preference for a rollout length that reliably gets to the end of a game, as was the case with Dominion, Colt Express and Sushi Go (see Table 1). This suggests that in Diamant, unlike the three games with long rollouts, the relative Score is a good proxy in the early game. This is true in Diamant as gaining points in the short/medium term is not 'deceptive' as it is in Dominion, where this can inhibit a player from gaining points later in the game [1].

Both Sushi Go and Colt Express also have some victory points that are only allocated at the end of the game, and these will only affect decisions if the rollout reaches them. Diamant does not have this feature.

5 CONCLUSIONS

We have tested the impact of the length of rollout and choice of game-agnostic heuristic in a suite of ten different tabletop board and card games. In the Introduction we posed four questions. We are now in a position to address these directly.

5.1 Score or Win Rate?

There is a clear answer in Tables 2 and 5 across the set of games that in general Score is a much better reward signal than Win rate. The advantages of greater information content and an earlier reward signal are shown to practically outweigh any disadvantage from the Score only being a proxy for winning. The one exception is Diamant with its low branching factor discussed in Section 4.4.3.

The best performance in 8 of the 10 games comes from using heuristic objectives that combine the score and winning the game. These split the games into two groups. In one group of games with directly adversarial counter-moves, or moves that attack an opponent, it is best to track the relative score (i.e. the difference between oneself and the best opponent). In the other group it is best to focus on one's own score, plus a bonus for winning. This bonus is required for good performance, so these games are still interactive, but at a more strategic level than planning direct counter-moves as in Chess or Dots and Boxes.

We have a spectrum of games between the purely 'multiplayer solitaire', such as Dominion, with more strategic-only interaction (for which Score+ is the best heuristic), and those that are also tactically interactive (for which Leader is the best heuristic). The relative performance of these heuristics can help identify where a game sits on this spectrum.

5.2 Does a Score objective lead to a higher score?

Optimising the score directly does in general lead to a better score than when just trying to win the game (Tables 2 and 5), and also tends to win more games than trying to optimise the win rate directly! In most games Score is a good proxy for winning the game. However, comparing Score to Score+/Leader in the same tables, it is clear that combining the score and winning in the objective leads in general to better performance in both.

5.3 How long should a rollout be?

If a game has significant victory points that are only scored at the end of the game, requires long-term planning, or if an early score can be deceptive, then an MCTS rollout to the end of the game is advisable. Colt Express, Sushi Go and Dominion are examples.

If the game has directly adversarial counter-moves such as Dots and Boxes or Virus, then a short rollout is preferable.

Surprisingly for many games outside of these categories, the length of rollout is not terribly important in performance.

5.4 Impact of player count

There is only weak evidence for a general effect in tabletop games in Table 3, with 2-player games being slightly more adversarial than 3+ player games, and the Leader heuristic performing better. However in some specific games, such as Dots and Boxes in Figure 2, the effect is very strong with the Leader heuristic (good in adversarial settings) having a much larger advantage in 2-player games than with more players.

5.5 Impact of computational budget

This is an additional fifth question that the data allows us to address. Overall there is not a large sensitivity in the optimal heuristics to use over the 25x range of budgets (40ms to 1000ms) used here. There are some game-specific effects, with the performance of the Win heuristics increasing in Diamant (Figure 7), and in Dominion, albeit from a very low baseline (Figure 5).

6 FUTURE WORK

The 50% score bonus applied on winning used here was ad hoc, and the better performance of the Default heuristic in Diamant, which effectively uses the win rate with a small bonus for Score suggests that this can be improved upon. Our objective was specifically a game-agnostic heuristic that does not require further tuning, but tuning this bonus may help classify games further along a spectrum of 'adversarial' to 'multiplayer solitaire' as discussed in Section 5.1.

These experiments all used a random rollout policy for consistency, and to focus on the effect of the state evaluation heuristic in MCTS. These will be inherently noisy for long rollouts, and using a biased rollout policy, including game-agnostic ones such as MAST or GRAVE may change the optimal rollout lengths [3, 5].

We have also not examined the effect of the different heuristics on game-playing behaviour and the avoidance of random move pathologies. Using different heuristics could be useful in providing different types of player, either for testing of a game, or to provide a variety of AI opponents for a human player [15].

ACKNOWLEDGMENTS

This work was funded by the EPSRC CDT in Intelligent Games and Game Intelligence (IGGI) EP/S022325/1. For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising.

REFERENCES

[1] Damien Anderson, Matthew Stephenson, Julian Togelius, Christoph Salge, John Levine, and Jochen Renz. 2018. Deceptive games. In *International Conference on the Applications of Evolutionary Computation*. Springer, 376–391.
[2] Hendrik Baier. 2015. *Monte-Carlo tree search enhancements for one-player and two-player domains*. Maastricht University.
[3] Y. Bjornsson and H. Finnsson. 2009. CadiaPlayer: A Simulation-Based General Game Player. *IEEE Transactions on Computational Intelligence and AI in Games* 1, 1 (March 2009), 4–15. https://doi.org/10.1109/TCIAIG.2009.2018702
[4] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4, 1 (March 2012), 1–43. https://doi.org/10.1109/TCIAIG.2012.2186810
[5] Tristan Cazenave. 2015. Generalized Rapid Action Value Estimation.. In *IJCAI*, Vol. 15. 754–760.
[6] GMJB Chaslot, Jahn-Takeshi Saito, Bruno Bouzy, JWHM Uiterwijk, and H. Jaap Van Den Herik. 2006. Monte-carlo strategies for computer go. In *Proceedings of the 18th BeNeLux Conference on Artificial Intelligence, Namur, Belgium*. 83–91.
[7] Rémi Coulom. 2006. Efficient selectivity and backup operators in Monte-Carlo tree search. In *International conference on computers and games*. Springer, 72–83.
[8] P. I. Cowling, E. J. Powley, and D. Whitehouse. 2012. Information Set Monte Carlo Tree Search. *IEEE Transactions on Computational Intelligence and AI in Games* 4, 2 (June 2012), 120–143. https://doi.org/10.1109/TCIAIG.2012.2200894
[9] Stefano Di Palma and Pier Luca Lanzi. 2018. Traditional Wisdom and Monte Carlo Tree Search Face-to-Face in the Card Game Scopone. *IEEE Transactions on Games* 10, 3 (Sept. 2018), 317–332. https://doi.org/10.1109/TG.2018.2834618 arXiv: 1807.06813.
[10] Raluca D. Gaina, Martin Balla, Alexander Dockhorn, Raul Montoliu, and Diego Perez-Liebana. 2020. Design and Implementation of TAG: A Tabletop Games Framework. *arXiv preprint arXiv:2009.12065* (2020).
[11] Magnus Gedda, Mikael Z. Lagerkvist, and Martin Butler. 2018. Monte Carlo Methods for the Game Kingdomino. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 1–8.
[12] Sylvain Gelly and David Silver. 2011. Monte-Carlo tree search and rapid action value estimation in computer Go. *Artificial Intelligence* 175, 11 (July 2011), 1856–1875. https://doi.org/10.1016/j.artint.2011.03.007
[13] James Goodman, Diego Perez, and Simon M. Lucas. 2021. Visualising Multiplayer Game Spaces. *IEEE Transactions on Games* (2021). Publisher: IEEE.
[14] Cristina Guerrero-Romero, Annie Louis, and Diego Perez-Liebana. 2017. Beyond playing to win: Diversifying heuristics for GVGAI. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, New York, NY, USA, 118–125. https://doi.org/10.1109/CIG.2017.8080424
[15] Cristina Guerrero-Romero, Simon M Lucas, and Diego Perez-Liebana. 2018. Using a Team of General AI Algorithms to Assist Game Design and Testing. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, Maastricht, Netherlands, 8.
[16] Emil Juul Jacobsen, Rasmus Greve, and Julian Togelius. 2014. Monte Mario: platforming with MCTS. In *Proceedings of the 2014 conference on Genetic and evolutionary computation - GECCO '14*. ACM Press, Vancouver, BC, Canada, 293–300. https://doi.org/10.1145/2576768.2598392
[17] Julien Kloetzer, Hiroyuki Iida, and Bruno Bouzy. 2007. The monte-carlo approach in amazons. In *Proceedings of the Computer Games Workshop*. 185–192.
[18] Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*. Springer, 282–293.
[19] Richard Lorentz. 2016. Using evaluation functions in Monte-Carlo tree search. *Theoretical computer science* 644 (2016), 106–113. Publisher: Elsevier.
[20] Richard J. Lorentz. 2008. Amazons discover monte-carlo. In *International Conference on Computers and Games*. Springer, 13–24.
[21] Simon M. Lucas, Jialin Liu, Ivan Bravi, Raluca D. Gaina, John Woodward, Vanessa Volz, and Diego Perez-Liebana. 2019. Efficient Evolutionary Methods for Game Agent Optimisation: Model-Based is Best. *arXiv preprint arXiv:1901.00723* (2019).
[22] Jacek Mańdziuk and Maciej Świechowski. 2012. Generic heuristic approach to general game playing. In *International Conference on Current Trends in Theory and Practice of Computer Science*. Springer, 649–660.
[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (Feb. 2015), 529–533. https://doi.org/10.1038/nature14236
[24] Tom Pepels, Mandy JW Tak, Marc Lanctot, and Mark HM Winands. 2014. Quality-based rewards for Monte-Carlo tree search simulations. In *ECAI 2014*. IOS Press, 705–710.
[25] Diego Perez Liebana, Jens Dieskau, Martin Hunermund, Sanaz Mostaghim, and Simon Lucas. 2015. Open Loop Search for General Video Game Playing. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference -*

GECCO '15. ACM Press, Madrid, Spain, 337–344. https://doi.org/10.1145/2739480.2754811

[26] Lewis Pulsipher. 2011. The three player problem. In *Tabletop Analog Game Design*. ETC Press.

[27] Raghuram Ramanujan and Bart Selman. 2011. Trade-offs in sampling-based adversarial planning. In *Twenty-First International Conference on Automated Planning and Scheduling*.

[28] Denis Robilliard, Cyril Fonlupt, and Fabien Teytaud. 2014. Monte-Carlo Tree Search for the Game of "7 Wonders". In *Computer Games*, Tristan Cazenave, Mark H. M. Winands, and Yngvi Björnsson (Eds.). Vol. 504. Springer International Publishing, Cham, 64–77. https://doi.org/10.1007/978-3-319-14923-3_5

[29] Kazutomo Shibahara and Yoshiyuki Kotani. 2008. Combining final score with winning percentage by sigmoid function in monte-carlo simulations. In *2008 IEEE Symposium On Computational Intelligence and Games*. IEEE, 183–190.

[30] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (Jan. 2016), 484–489. https://doi.org/10.1038/nature16961

[31] Matthew Stephenson, Dennis JNJ Soemers, Eric Piette, and Cameron Browne. 2021. General Game Heuristic Prediction Based on Ludeme Descriptions. In *Proceedings of the 3rd IEEE Conference on Games*.

[32] Nathan Sturtevant. 2008. AN ANALYSIS OF UCT IN MULTI-PLAYER GAMES. *ICGA Journal* (2008), 14.

[33] István Szita, Guillaume Chaslot, and Pieter Spronck. 2010. Monte-Carlo Tree Search in Settlers of Catan. In *Advances in Computer Games*, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, H. Jaap van den Herik, and Pieter Spronck (Eds.). Vol. 6048. Springer Berlin Heidelberg, Berlin, Heidelberg, 21–32. https://doi.org/10.1007/978-3-642-12993-3_3

[34] Mandy JW Tak, Marc Lanctot, and Mark HM Winands. 2014. Monte Carlo tree search variants for simultaneous move games. In *2014 IEEE Conference on Computational Intelligence and Games*. IEEE, 1–8.

[35] Daniel Whitehouse, Peter I. Cowling, Edward Jack Powley, and Jeff Rollason. 2013. Integrating Monte Carlo Tree Search with Knowledge-Based Methods to Create Engaging Play in a Commercial Mobile Game.. In *AIIDE*.

[36] Mark H. M. Winands and Yngvi Björnsson. 2010. Evaluation Function Based Monte-Carlo LOA. In *Advances in Computer Games*, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, H. Jaap van den Herik, and Pieter Spronck (Eds.). Vol. 6048. Springer Berlin Heidelberg, Berlin, Heidelberg, 33–44. https://doi.org/10.1007/978-3-642-12993-3_4

[37] Zuozhi Yang and Santiago Ontañón. 2020. Are Strong Policies Also Good Playout Policies? Playout Policy Optimization for RTS Games. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 16, 1 (Oct. 2020), 144–150. https://ojs.aaai.org/index.php/AIIDE/article/view/7423 Number: 1.