General Game Playing Hyper-Agents for Ludii

Nicholas Thompson
College of Science and Engineering
Flinders University
Adelaide, Australia
thom1135@flinders.edu.au

Matthew Stephenson
College of Science and Engineering
Flinders University
Adelaide, Australia
matthew.stephenson@flinders.edu.au

***Abstract*—This paper explores the viability and effectiveness of hyper-agent approaches for the Ludii general game system. These hyper-agents utilise trained machine learning models to predict the optimal sub-agent and heuristics for previously unseen board games, based on automatically detectable game parameters (ludemes and concepts). Several hyper-agents based on portfolio and ensemble design approaches were implemented within the Ludii system. Each hyper-agent was trained on 460 games with known sub-agent and heuristic performances, with evaluations being performed on a representative set of 50 new games. Our best performing hyper-agent approach demonstrated a statistically significant win-rate improvement over all of the individual sub-agents utilised in its training corpus.**

***Index Terms*—General Game Playing, Hyper-Agents, Board Games, Ludii**

I. Introduction

The development of artificial intelligence (AI) agents for game playing is an area of research whose broad goal is to explore the capabilities of AI in complex strategic decision-making and long-term planning problems [1], [2]. With well-defined parameters and objectives, games provide excellent environments both for benchmarking agent performance and developing novel search algorithms [3]. Techniques developed for game playing agents can often be adapted to real-world domains, and real-world environments can be virtually simulated within digital game environments [4].

Specialised agents developed for mastery of a single game have seen impressive results against expert level human players. IBM's Deep Blue, a chess playing agent, exemplified this with its defeat of chess grandmaster and reigning world champion Garry Kasparov in 1997 [5]. Google DeepMind's AlphaGo and its successors have defeated professional Go players [6], with the most recent iteration, MuZero, mastering Chess, Go and Shogi [7]. Despite their impressive performance, specialised agents find much of their success in domain or game specific knowledge, hindering their performance in games outside of their training domain [8].

Rather than focusing on an individual or limited set of games, General Game Playing (GGP) agents are developed with the goal of matching and exceeding human performance across a varied range of games, with limited prior knowledge of the game's rules, mechanics and environments [2], [9]. Research into general game playing has become increasingly popular in the academic AI community, due its potential to provide insight into the broader challenge of artificial general intelligence [10]. One promising approach to developing general game playing agents involves the use of hyper-agents, which combine multiple specialised sub-agents or lower level algorithms in an attempt to leverage each of their individual strengths [11]. Hyper-agents are known to work best in scenarios where multiple sub-agents are available, and where each sub-agent performs differentially well across various scenarios within the same domain [12].

This research explores two hyper-agent approaches, each based on trained machine learning models, known as the portfolio and ensemble techniques. A portfolio agent implements a selection mechanism to change its applied sub-agent based on the game it is presented with. An ensemble agent instead utilises a voting system, where each sub-agent casts a vote for their preferred move before an arbitrator selects the most popular. Several hyper-agents based on these two techniques were developed and evaluated for the Ludii general game system, one of the largest collections of board games for AI research purposes. Our hyper-agents utilise automatically extractable features based on the pieces, rules and mechanics of the game, in order to select the optimal sub-agent(s) for selecting appropriate moves. Our results demonstrate the effectiveness of hyper-agent approaches for Ludii, providing a statistically significant performance improvement when compared to all other baseline agents.

The rest of this paper is structured as follows. Section II covers prior work related to General Game Playing and Hyper-Agents. Section III describes the implemented hyper-agent approaches and training process. Section IV defines the evaluation process, including how a diverse set of test games was selected. Section V presents the experimental results and discusses our findings. Section VI provides our overall conclusions and suggestions for future research.

II. Background

A. General Game Playing Systems

General game playing systems aim to provide a universal framework and language that can be used to describe and play a large range of games in a structured way. Developed in 2005, the Game Description Language (GDL) was one of the first general game frameworks for academic research [13]. GDL represents its games as state machines, with rules to specify state transitions, and distinctions made between initial states,

goal states, and terminal states. More recent general game playing systems such as General Video Game AI (GVGAI), Regular Boardgames (RBG), and Ludii, were created to offer a wider range of game types, along with more efficient play-outs and streamlined game description languages [14].

The General Video Game AI (GVGAI) framework allows for the creation of simple arcade-style video games using the Video Game Description Language (VGDL), which provides a high level representation of a game's mechanics and level layout [15], [16]. This framework has been the subject of much research over the past decade, with several associated competitions focusing on both game playing agents and content generation [17]. Regular Boardgames (RBG) is a formal game description language designed for deterministic, complete-information games [18]. RBG describes games using regular expressions that define legal moves and transitions in a compact, low-level format. This makes RBG particularly well suited for combinatorial game solving and AI competitions, where performance optimisation is crucial [19].

However, out of all the current general game playing systems, the one that arguably offers the largest and most diverse collection of games is Ludii. Ludii is a general board game system that describes its games in terms of keywords called ludemes [20], [21]. A ludeme is a term that describes a single game element, as defined in the Ludii Game Description Language (LGDL), and can be combined to describe a large set of highly varied games [22]. Ludii also allows for the automatic detection of more abstract game concepts, often arising from distinct combinations of ludemes within a specific rule context, allowing for clearer conceptual distinction between games [23]. The high number and variety of games available within Ludii, along with this corpus of automatically detectable game features (ludemes and concepts), makes it an ideal framework for evaluating hyper-agent game playing techniques.

B. Hyper-Agents

This section covers the two main design approaches for creating hyper-agents, namely that of portfolio and ensemble techniques.

1) Portfolio: Portfolio agents are arguably the simplest form of hyper-agent, whereby a single sub-agent is selected to decide which move to make in any given game state. Portfolio agents have been previously demonstrated to be effective for several video games [12]. In 2017 a portfolio agent was developed using the Angry Birds AI competition framework, following the observation that agents from previous years had varied performances across different levels [24]. The highest score that each agent achieved for each level was recorded after a series of prior competition runs, along with key features of each level, to provide the portfolio agent with a set of training data. After training machine learning models to predict the best agent for any given level based on these features, the resulting portfolio agent was able to outperform all prior competition agents across 80 previously unseen levels.

An exploration of portfolio search optimisation for general strategy game playing found that portfolio approaches for Stratega, a general strategy games framework, were also highly effective [25]. This analysis reviewed portfolio improvements to decision-making for real-time scenarios, in which the portfolio search modelled increasingly optimised moves for both agent and opponent units across the six potential actions that each unit could take. This approach is also noted as a common implementation for unit micromanagement in other real-time strategy games [26].

2) Ensemble: Ensemble agents employ a slightly different approach to utilising sub-agents. Rather than selecting the best agent for a particular scenario, ensemble agents instead poll all their sub-agents and often proceed with the most popular move. A 2019 study evaluated an ensemble decision system for the General Video Game AI competition [11], in which the ensemble agent allowed each sub-agent to vote for their desired move before a separate arbitrator made the final decision. The authors observed that the ensemble agent was consistently outperformed by individual sub-agents in each game, even though these agents were also included part of the ensemble. This highlights one of the potential limitations with the equal polling system used by most ensemble agents, in that while they can be very good at achieving a very consistent performance across many games they are rarely the best choice (i.e., "a jack of all trades, but master of none").

Another 2022 study describes a weighted ensemble agent created to play the game Werewolf [27]. Werewolf is a social communication game in which players must utilise incomplete information, deception, and deduction to identify secret impostors (i.e., werewolves). This study also highlights a potential issue with regular ensemble agents, namely that of each sub-agent having an equal weighting in move selection polls, that reduces the effective contribution of high performing sub-agents. To address this, each sub-agent was assigned a weight based on its overall performance during training, such that the importance of its vote when polled would be adjusted accordingly. While this ensemble agent performed well in certain setups, the results from this study make it difficult to draw conclusions about the effectiveness of a weighted ensemble approach, since the top sub-agent still outperformed the ensemble agent in most games.

C. Hyper-Agents for Ludii

In 2021, a preliminary study was conducted to investigate whether machine learning models could be used to predict the performance of general game playing heuristics for Ludii, based on the ludemes used in each game's description [28]. This study explored the performance of several agent heuristics across 695 games within Ludii, with their results indicating that a game's ludemes could be used to reliably predict heuristic performance. This work also suggested potential venues for further research, including into the development of a full hyper-agent for Ludii capable of selecting optimal agents and heuristics for new games.

With its large repository of games, and an existing collection of baseline game playing agents, the Ludii game system provides the ideal environment for the development of general game playing hyper-agents. At the time of writing, the Ludii system does not yet have any hyper-agent implementations, and results sourced from the Ludii database indicate strong differential performance between agents across the wide variety of its games [29]. If a hyper-agent approach could be developed that is capable of identifying correlations and patterns between each agent's performance and Ludii's general game features, optimal agents could be selected to play novel games, with the potential for improved zero-shot general game playing performance.

III. Methodology

A. Hyper-Agent Design

Based on the previously described portfolio and ensemble approaches, three potential hyper-agent designs were proposed.

1) Portfolio Agent: This agent operates by using a trained machine learning model to predict the best performing sub-model for the provided game. This selection process occurs at the start of the game, after which the selected sub-agent will be used to make all future moves.

2) Ensemble Agent: This agent operates by allowing all available sub-agents to select a desired move, after which a simple majority vote is used to determine which move to make. Any ties between multiple moves with the highest number of votes are decided through random selection. This process is repeated for every move that needs to be made.

3) Weighted Ensemble Agent: This agent operates by first using a trained machine learning model to predict the expected performance (i.e., win-rate) of each sub-agent for the provided game. This prediction process occurs at the start of the game, in order to determine a respective weighting for each sub-agent. Afterwards, this agent operates the same as the Ensemble agent, except that selected moves are instead determined through a weighted vote process rather than a simple majority (i.e., agents with a higher predicted performance have greater voting power).

B. Sub-Agent Performance Prediction

The Ludii general game system includes a handful of baseline general game playing techniques that can be treated as sub-agents for our developed hyper-agent. Short descriptions and references for each of the five sub-agent techniques used in our available agent pool are defined in Table I. As well as existing agents, Ludii also provides a set of automatically detectable features for each game, comprising 598 ludeme and 511 concept values. Each ludeme feature is a binary value, representing whether a corresponding ludemic keyword is present in the game's description. Concepts can be either binary or numerical values, and are determined based on specific ludemic patterns and arrangements within the game description (e.g., the number of players, or whether pieces are captured by jumping over them). Note, for this paper we only consider the 511 "compilation" concepts that are available directly at the start of the game, without requiring costly playouts to be run.

TABLE I
Set of sub-agents used in the Ludii hyper-agent implementations

Agent	Description
Alpha-Beta	Alpha-Beta tree pruning algorithm [30]
UCT	Monte Carlo Tree Search (MCTS) algorithm, with Upper Confidence Bounds applied to Trees [31]
MAST	MCTS with Move-Average Sampling [32]
MC-GRAVE	MCTS with the Generalised Rapid Action Value Estimation heuristic [33]
Progressive History	MCTS with progressive bias using the history heuristic [34]

In order to utilise our proposed portfolio and weighted ensemble agents, we first need to train machine learning models capable of predicting the performance of each sub-agent for a provided game. For this purpose, we sourced data from the Ludii database[1] (v1.3.12), which provided both the ludemic and concept feature values for all games in Ludii, but also agent performance results for a large (although not complete) number of games. Out of the 1109 games listed in the Ludii database with both agent and ludeme result, 460 also include expected win-rates for all sub-agents. These games were therefore utilised to train our hyper-agent machine learning models.

1) Portfolio Agent (best agent prediction): In order to predict the best performing agent for a given game, we can either use a classification or regression model. A single classification model can be trained on each of the 460 labelled games, to directly predict the single best performing agent. However, we can also utilise a suite of multiple regression models, each trained to predict the win-rate for a specific sub-agent. When required to predict the best agent, we can use these models to predict the win-rate for all sub-agents and simply return whichever has the highest value. In order to determine which approach and model would perform best, we trained and evaluated several machine learning techniques using 10-fold-cross-validation, see Table II, calculating the average difference between the win-rates of the predicted and the actual best sub-agent (i.e., regret) as our comparison metric.

The regret metric was selected over an alternative measure of accuracy to better reflect the actual cost of making a suboptimal prediction, since the selected sub-agent will still produce a win-rate across a set of games. In addition to the candidate models, a dummy (naive) regressor and classifier were also evaluated, providing a baseline point of comparison for model performance. In this case, the approach that achieved

[1] https://ludii.games/download.php

TABLE II
TRAINED MODEL PERFORMANCE FOR PREDICTING BEST AGENT

Algorithm	Regret
Random Forest Regressor	**5.22**
Gradient Boosting Regression	**5.78**
Random Forest Classifier	**5.78**
Multi-layer Perception Classifier	**7.19**
Decision Tree Classifier	**9.55**
Support Vector Regression (sigmoid)	**12.99**
Dummy Regressor (mean)	**12.99**
Dummy Classifier (most frequent)	**12.99**
Gaussian Naive Bayes	**14.14**
Linear Regression	**19.03**

TABLE III
TRAINED MODEL PERFORMANCE FOR PREDICTING BEST HEURISTIC

Algorithm	Regret
Random Forest Regressor	**7.70**
Gradient Boosting Regression	**8.10**
Random Forest Classifier	**8.86**
Multi-layer Perception Classifier	**10.38**
Decision Tree Classifier	**13.68**
Gaussian Naive Bayes	**14.06**
Support Vector Regression (sigmoid)	**14.71**
Dummy Regressor (mean)	**16.55**
Dummy Classifier (most frequent)	**16.55**
Linear Regression	**16.76**

the lowest average regret was the suite of multiple random forest regressor models.

2) Weighted Ensemble Agent (best agent prediction): Unlike the portfolio agent, the weighted ensemble agent requires that the win-rates of each sub-agent be predicted individually to provide a suitable weighting for move selection. Thankfully, we can utilise the same Random Forest Regressor models as the portfolio agent to provide this weighting. This essentially means that when using either the portfolio or weighted ensemble hyper-agents, the trained random forest regressor models are run at the start of the game to provide a win-rate estimate for each sub-agent. In the case of the portfolio agent this is used to select the sub-agent with the highest predicted win-rate to play the entire game, while for the weighted ensemble agent these predicted win-rates will provide the respective sub-agent voting weights when conducting move polls.

3) Heuristic Prediction: One issue that also needs to be addressed is that of appropriate heuristic selection, specifically for the Alpha-Beta sub-agent. This agent relies on suitable heuristics being provided to evaluate intermediate game states, and its performance drops significantly if these are not available. To account for this, we also trained additional machine learning models to predict the best performing Ludii heuristic, using a similar approach that of predicting the best sub-agent. The results for these heuristic prediction models are presented in Table III, showing that using a suite of random forest regressor models was once again the best performing approach. These models will be run at the start of each game to determine the heuristic that Alpha-Beta agent will use to select its moves (if chosen by the hyper-agent). Further details on all available Ludii game heuristics are provided in [28].

C. Ludii Hyper-agents

Each of the three hyper-agent approaches (portfolio, ensemble and weighted ensemble) were implemented natively into Ludii. This was achieved by integrating the machine learning models described above for agent and heuristic selection with the Ludii system's game object, which provides ludeme and concept feature data at the start of each game. The time required to predict agent and heuristic win-rates using these machine learning models was largely negligible, and can be considered as part of the agent's setup time. A general overview of how our hyper-agents are initialised and play games in Ludii is as follows:

1) **Feature Extraction:** Ludeme and concept feature data is extracted from the Ludii game object.
2) **Agent & Heuristic Prediction:** Trained random forest regressor models are used to predict the win-rate for each sub-agent and heuristic, based on the extracted ludeme and concept features.
3) **Sub-agent initialisation:** All sub-agents are initialised, with Alpha-Beta using the heuristic with the highest predicted win-rate from step 2.
4) **Hyper-Agent Move Selection:** Whenever required to make a move, the hyper-agent uses one of the following approaches:
    a) **Portfolio Agent:** Always uses the move returned by the sub-agent with the highest predicted win-rate from step 2.
    b) **Ensemble Agent:** Each sub-agent receives an equal vote for deciding the next move to make.
    c) **Weighted Ensemble Agent:** Each sub-agent receives a weighted vote for deciding the next move to make, with the strength of its vote equal to its predicted win-rate from step 2.

Agents in Ludii are typically given a fixed time limit to make each move (i.e., thinking time). While agents are typically restricted to a single computational thread, our described ensemble and weighted ensemble approaches allow for efficient parallelisation between sub-agents when multiple threads are available. We therefore also implemented two alternative versions for each these ensemble hyper-agents, one where the available thinking time is split evenly between each sub-agent (non-parallel) and another where each agent is provided the full thinking time (parallel). These parallelised ensemble agents are arguably not a fair comparison, as other agents could also potentially take advantage of multiple threads if allowed to do so, but we still include them as a point of comparison for if all ensemble sub-agents were permitted the full thinking time for deciding their moves.

IV. EXPERIMENTS

The following experiment was developed to evaluate the performance of our proposed hyper-agents, compared to that of

each baseline agent within Ludii. Each agent would play a representative set of 50 previously unseen games, with each game being played 50 times to establish a reliable win-rate measure. The agent being evaluated would always take the role of player 1, with any additional players populated with standard UCT agents. Each agent was provided with one second of thinking time per move, and each game had a maximum turn limit of 1000 moves per agent. This maximum turn limit was imposed to prevent stalemated games from going on forever, and would result in an automatic draw if exceeded. The result of each game was recorded as either a 0 (loss), 1 (win) or $1/N$ (draw), where N is the total number of remaining active players. The evaluated agents included the five sub-agents specified in table I, the portfolio agent, the regular ensemble agent, the weighted ensemble agent and a random agent. Parallel implementations of the ensemble agents, with each sub-agent provided the full thinking time, were also evaluated. All experiments were run using several Docker containers, each with 2x Intel(R) Xeon(R) Gold 5318Y CPU (2.10GHz) and 16GB RAM. Full source code and experiment results for this paper are available at.[2]

A. Evaluation Game Selection

To properly evaluate the effectiveness of our presented hyper-agent approaches, we need to compare their performance against that of the sub-agents for a previously unseen set of games. Of the 1109 games available in the Ludii database, 460 were present in the portfolio model training dataset, leaving a potential 649 games that could be used for evaluation purposes. Given that the necessary computation time required to perform a full evaluation of each agent across all 649 available test games would be very prohibitively costly, we instead opted to select a smaller representative set of 50 evaluation games.

This representative set of 50 evaluation games was selected using a "maximal spread" process, that iteratively selected games with the highest minimum cosine distance from any previously selected evaluation games:

1) Initialise the set of selected evaluation games, S, with an arbitrary first game.
2) For each subsequent selection, choose the game g^* that maximises the minimum distance from all previously selected evaluation games:

$$g^* = \arg \max_{g \in G \setminus S} \min_{s \in S} d_(g, s)$$

where:

- G is the full set of available evaluation games.
- S is the subset of already selected evaluation games.
- $d_(g, s)$ is the distance between game g and game s.

[2]https://github.com/thom1135/ludii-hyper-agents

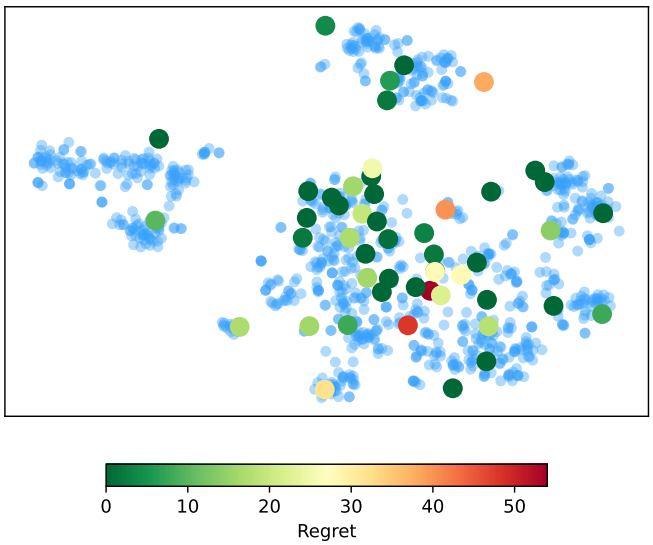

Fig. 1. t-SNE visualisation of the relative spread of all 1109 available games in Ludii, based on their ludeme and concept values. Overlaid is a heat map illustrating the regret associated with sub-agent selection for our portfolio agent during evaluation, where higher regret values indicate suboptimal sub-agent choices.

The distance between two games was calculated using the cosine similarity between the two feature vectors (defined by the set of ludemes and concepts) that describe each game:

$$d_{\cos}(\mathbf{g}_i, \mathbf{g}_j) = 1 - \frac{\mathbf{g}_i \cdot \mathbf{g}_j}{\|\mathbf{g}_i\| \|\mathbf{g}_j\|}$$

where:

- $\mathbf{g}_i \cdot \mathbf{g}_j = \sum_k g_{i,k} g_{j,k}$ is the dot product of the two game vectors.
- $\|\mathbf{g}_i\| = \sqrt{\sum_k g_{i,k}^2}$ is the magnitude of the vector.

Cosine similarity was chosen over the Euclidean distance because of the high dimensionality of the feature data [35]. Prior to the pairwise distance calculations, a bi-symmetric log transform was performed on the concept feature values as described in [36]. This operation was executed with the intention of minimising the impact of concepts with large numerical ranges, while still preserving the relative significance of concepts with binary values.

A t-distributed stochastic neighbour embedding (T-SNE) visualisation of all 1109 Ludii games is shown in Figure 1, based on each game's feature vector (ludeme and concept values). The large coloured dots represent the 50 games selected for evaluation, while the solid blue dots indicate potential evaluation games that were not selected. The faded blue points indicate games available in the training set. Using this visualisation we can see that our evaluation game selection process provides a fairly well distributed 50 game representation of the full game set.

V. Results

Performance (win-rate) results for each agent, averaged across all 50 evaluation games, are presented in Table IV. To

TABLE IV
AVERAGE AGENT WIN-RATES ACROSS ALL EVALUATION GAMES.

Agent	Win-Rate (%)	Standard Error
Parallel Weighted Ensemble	65.04	0.92
Parallel Ensemble	61.36	0.94
Portfolio	60.28	0.94
Progressive History	52.72	0.97
Weighted Ensemble	49.35	0.97
MAST	47.55	0.97
MC-GRAVE	47.41	0.96
UCT	46.70	0.96
Ensemble	44.75	0.96
Alpha-Beta	43.24	0.95
Random	8.46	0.51

TABLE V
FREQUENCY OF SUB-AGENT SELECTION BY PORTFOLIO AGENT DURING EVALUATION, SUB-AGENT PERFORMANCE IN TRAINING DATA, AND AVERAGE REGRET WHEN SUB-AGENT SELECTED.

Agent	Selected (%)	Training Set (%)	Regret
Alpha-Beta	46.0	51.9	13.0
MC-GRAVE	20.0	8.9	5.8
Progressive History	14.0	8.0	10.7
UCT	12.0	14.3	18.0
MAST	8.0	13.3	16.3
Random	0.0	3.54	N/A

confirm whether any differences in win-rate were statistically significant, pairwise independent samples t-tests with Holm-Bonfferroni correction were conducted between all agent pairs, see Table VI. Table V shows how frequently each sub-agent was selected by our portfolio agent during evaluation, along with the frequency with which each sub-agent scored the highest in the training data and the average regret obtained when selecting this sub-agent during evaluation. Lastly, the colour of the 50 evaluation game dots in Figure 1 represents the average regret obtained by our portfolio agent when playing each of these games, with red dots indicating games where the selected sub-agent performed especially poorly.

These results demonstrate that our implemented hyper-agents perform well in the Ludii system, with the portfolio agent approach outperforming all non-parallelised agents with a statistically significant difference. This improvement serves as further validation of the ability of our random forest regression models to accurately predict agent and heuristic win-rates for each game. While both parallel ensemble implementations performed better than any of the other agents, their non-parallel counterparts saw limited success. Both of the non-parallel ensemble agents underperformed compared to the baseline Ludii sub-agents, although the weighted ensemble agent did provide a statistically significant improvement over the regular (i.e., non-weighted) ensemble agent.

A. Discussion

1) Alpha-Beta Heuristics: The sub-agent with the highest average win-rate within the original Ludii training dataset was Alpha-Beta (Table V), while the results from our evaluation experiment saw Progressive History perform better, with a statistically significant difference (Table VI). One reason for this could be that there were several games within our evaluation set that the Alpha-Beta sub-agent did not support, likely due to incompatible heuristics, resulting in a 0% win-rate. As previously mentioned, many of the Alpha-Beta agent results in the Ludii database are based on playouts using handcrafted heuristics, which is potentially skewing Alpha-Beta's overall win-rate higher in the training data. To ensure the results in this experiment were applicable to general game playing with no domain specific knowledge, our Alpha-Beta sub-agent instead had to rely on the heuristics predicted by our random forest models. Despite the fact that Alpha-Beta's win-rate was likely inflated in the training data, the sub-agent predictions were still reliable enough overall to produce a statistically significant improvement in our portfolio agent's win-rate compared to the baseline Ludii sub-agents.

2) Weak Ensemble Performance: The non-parallel ensemble agents underperformed expectations, bested by the top performing Ludii agents. This is likely, in part, due to the limited thinking time given to each agent within the experiment. With a thinking time of 1 second per turn being split among the five agents within the ensemble, each agent would receive an average of 0.2 seconds to select a move. The same experiment run with increased thinking time per agent would likely yield improved performance for these ensemble implementations. It's important to note here that performance comparisons between the parallel and non-parallel ensemble agents should be done with care, as the non-parallel agents have not been optimised for parallel searches. For an ensemble with five agents, thinking time is increased by a factor of five if each agent is allowed the full thinking time. Despite this potentially unfair advantage, our results do show that both parallel ensemble approaches do perform well, despite the fact each individual agent is not given any additional thinking time compared to the other baseline Ludii agents. Parallelisation is also fairly trivial to implement for ensemble agents, indicating that it is well worth taking advantage of if available.

3) Portfolio Agent Mistakes: Although there are no obvious patterns in the regret heat map presented in Fig. 1, a closer look at the games with the lowest performing portfolio predictions may provide some insight into its limitations. The evaluation games with the highest portfolio agent regret scores were 54 (Agon), 48 (Tara), and 40 (Murus Galicus).

- Agon, also known as Queen's Guard, is a strategy game played on a hexagonal grid in which two players take turns moving their queen and 6 guards, with the goal of reaching the centre of the board in the correct formation. The agent selected for this game was Alpha-Beta, with the material heuristic. This heuristic is likely a key reason for the poor performance, since pieces in this game cannot be captured, leaving material value constant throughout the game. The best performing agent here was MAST, where its ability to refine its strategy dynamically may have allowed it to better recognise positional advantages and navigate the game's unique constraints more effectively.
- Tara is a push-based connection game where players try

TABLE VI

RESULTS OF PAIRWISE INDEPENDENT SAMPLES T-TESTS ASSESSING THE SIGNIFICANCE OF WIN-RATE DIFFERENCES BETWEEN AGENTS. P-VALUES ADJUSTED USING HOLM-BONFERRONI CORRECTION ($\alpha = 0.05$), WITH VALUES BELOW THE ADJUSTED SIGNIFICANCE THRESHOLD SHOWN IN GREEN.

*	AB	M	MG	PH	U	R	PF	E	WE	PW	PE
AB	NA	T:-3.177 P:0.001	T:-3.079 P:0.002	T:-6.988 P:0.000	T:-2.552 P:>0.004	T:32.132 P:0.000	T:-12.710 P:0.000	T:-1.117 P:>0.010	T:-4.494 P:0.000	T:-16.441 P:0.000	T:-13.549 P:0.000
M	T:3.177 P:0.001	NA	T:0.106 P:>0.050	T:-3.778 P:0.000	T:0.626 P:>0.017	T:35.667 P:0.000	T:-9.415 P:0.000	T:2.055 P:>0.005	T:-1.309 P:>0.008	T:-13.077 P:0.000	T:-10.238 P:0.000
MG	T:3.079 P:0.002	T:-0.106 P:>0.050	NA	T:-3.895 P:0.000	T:0.521 P:>0.025	T:35.687 P:0.000	T:-9.548 P:0.000	T:1.954 P:>0.005	T:-1.419 P:>0.007	T:-13.223 P:0.000	T:-10.374 P:0.000
PH	T:6.988 P:0.000	T:3.778 P:0.000	T:3.895 P:0.000	NA	T:4.411 P:0.000	T:40.435 P:0.000	T:-5.597 P:0.000	T:5.847 P:0.000	T:2.465 P:>0.004	T:-9.220 P:0.000	T:-6.411 P:0.000
U	T:2.552 P:>0.004	T:-0.626 P:>0.017	T:-0.521 P:>0.025	T:-4.411 P:0.000	NA	T:34.982 P:0.000	T:-10.065 P:0.000	T:1.430 P:>0.006	T:-1.936 P:>0.006	T:-13.741 P:0.000	T:-10.891 P:0.000
R	T:-32.132 P:0.000	T:-35.667 P:0.000	T:-35.687 P:0.000	T:-40.435 P:0.000	T:-34.982 P:0.000	NA	T:-48.224 P:0.000	T:-33.287 P:0.000	T:-37.261 P:0.000	T:-53.580 P:0.000	T:-49.414 P:0.000
PF	T:12.710 P:0.000	T:9.415 P:0.000	T:9.548 P:0.000	T:5.597 P:0.000	T:10.065 P:0.000	T:48.224 P:0.000	NA	T:11.528 P:0.000	T:8.082 P:0.000	T:-3.606 P:0.000	T:-0.811 P:>0.013
E	T:1.117 P:>0.010	T:-2.055 P:>0.005	T:-1.954 P:>0.005	T:-5.847 P:0.000	T:-1.430 P:>0.006	T:33.287 P:0.000	T:-11.528 P:0.000	NA	T:-3.366 P:0.001	T:-15.228 P:0.000	T:-12.359 P:0.000
WE	T:4.494 P:0.000	T:1.309 P:>0.008	T:1.419 P:>0.007	T:-2.465 P:>0.004	T:1.936 P:>0.006	T:37.261 P:0.000	T:-8.082 P:0.000	T:3.366 P:0.001	NA	T:-11.727 P:0.000	T:-8.901 P:0.000
PW	T:16.441 P:0.000	T:13.077 P:0.000	T:13.223 P:0.000	T:9.220 P:0.000	T:13.741 P:0.000	T:53.580 P:0.000	T:3.606 P:0.000	T:15.228 P:0.000	T:11.727 P:0.000	NA	T:2.794 P:>0.004
PE	T:13.549 P:0.000	T:10.238 P:0.000	T:10.374 P:0.000	T:6.411 P:0.000	T:10.891 P:0.000	T:49.414 P:0.000	T:0.811 P:>0.013	T:12.359 P:0.000	T:8.901 P:0.000	T:-2.794 P:>0.004	NA

* AB = Alpha-Beta, M = MAST, MG = MC-GRAVE, PH = Progressive History, U = UCT, R = Random, PF = Portfolio, E = Ensemble, WE = Weighted Ensemble, PE = Parallel Ensemble, PW = Parallel Weighted Ensemble

to form a path across the board by pushing stones from the edges. MC-GRAVE was the selected agent for this game, while the best performing agent was UCT. It's possible here that MC-GRAVE's reliance on past playouts is not well suited to the significant board state changes that can result from the push mechanic.

- Murus Galicus is a two player positional strategy game where the goal is to trap an opponent by preventing them from making a legal move. The game includes a stacking mechanic that allows players to create strong defensive formations at the expense of board coverage and mobility. The agent selected here was Alpha-Beta, with the material heuristic, while the best performing agent was MAST once again. The similarity between this finding and the result for Agon could imply that heuristic-driven search methods like Alpha-Beta will under perform when applied with inappropriate heuristics, and that sampling-based techniques like MAST might be more a robust choice for agent selection.

VI. CONCLUSION AND FUTURE WORK

Within this paper we have presented several hyper-agent approaches for the Ludii general game system, capable of utilising a game's ludemes and concepts to select the optimal sub-agent(s). Several of these hyper-agents were shown to produce a consistent improvement in average win-rate across a diverse range of multiple previously unseen games. In particular, our implemented portfolio agent produced a statistically significant performance improvement over each of its constituent sub-agents. Ensemble agents implemented in the Ludii system were outperformed by at least one of their sub-agents overall, though the weighted ensemble agent performed statistically significantly better than its unweighted counterpart. Parallel implementations of these ensemble agents achieved the highest win-rates, though this may be attributable to the effective increase in thinking time they were afforded.

A key limitation of this research is the restricted selection of games used for hyper-agent evaluations. While the results demonstrated statistically significant differences in agent win-rates, the selected set of games did not include enough games from the various game clusters to reveal any meaningful patterns. Future research could address this by incorporating a larger set of evaluation games. Additionally, the non-parallel implementations of both the ensemble and weighted ensemble agents did not perform as well as expected. With the significantly improved results of the parallel implementations, the effect of increased thinking time may provide a compelling avenue of research. Furthermore, instead of an ensemble of agents, future work could explore the combination multiple heuristics into an ensemble, leveraging their complementary strengths to guide decision-making more effectively.

REFERENCES

[1] M. Genesereth and M. Thielscher, *General game playing*. Morgan & Claypool Publishers, 2014.

[2] S. Sharma, Z. Kobti, and S. D. Goodwin, "General game playing: An overview and open problems," in *2009 International Conference on Computing, Engineering and Information*, 2009, pp. 257–260.

[3] M. Fuchs and A. Sudmann, "Games and ai: Paths, challenges, critique," *Eludamos: Journal for Computer Game Culture*, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:219632047

[4] J. Pérez, M. Castro, and G. López, "Serious games and ai: Challenges and opportunities for computational social science," *IEEE Access*, vol. 11, pp. 62 051–62 061, 2023.

[5] M. Campbell, A. Hoane, and F. hsiung Hsu, "Deep blue," *Artificial Intelligence*, vol. 134, no. 1, pp. 57–83, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370201001291

[6] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[7] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver, "Mastering atari, go, chess and shogi by planning with a learned model," *Nature*, vol. 588, no. 7839, p. 604–609, Dec. 2020. [Online]. Available: http://dx.doi.org/10.1038/s41586-020-03051-4

[8] K. Jebari and J. Lundborg, "Artificial superintelligence and its limits: Why alphazero cannot become a general agent," *AI and Society*, forthcoming.

[9] S. Legg and M. Hutter, "Universal intelligence: A definition of machine intelligence," *Minds Mach.*, vol. 17, no. 4, p. 391–444, Dec. 2007.

[10] G. N. Yannakakis and J. Togelius, *Artificial Intelligence and Games*. Springer, 2018, https://gameaibook.org.

[11] D. Anderson, P. Rodgers, J. Levine, C. Guerrero-Romero, and D. Perez-Liebana, "Ensemble decision systems for general video game playing," in *2019 IEEE Conference on Games (CoG)*, 2019, pp. 1–8.

[12] A. Mendes, J. Togelius, and A. Nealen, "Hyper-heuristic general video game playing," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 2016, pp. 1–8.

[13] M. Thielscher, "The general game playing description language is universal," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, no. 1, 2011, p. 1107.

[14] Éric Piette, M. Stephenson, D. J. N. J. Soemers, and C. Browne, "An empirical evaluation of two general game systems: Ludii and rbg," 2019. [Online]. Available: https://arxiv.org/abs/1907.00244

[15] D. Perez-Liebana, S. M. Lucas, R. D. Gaina, J. Togelius, A. Khalifa, and J. Liu, *General Video Game Artificial Intelligence*. Morgan & Claypool Publishers, 2019, vol. 3, no. 2, https://gaigresearch.github.io/gvgaibook/.

[16] T. Schaul, "A video game description language for model-based or interactive learning," in *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, 2013, pp. 1–8.

[17] D. Perez-Liebana, J. Liu, A. Khalifa, R. D. Gaina, J. Togelius, and S. M. Lucas, "General video game ai: A multitrack framework for evaluating agents, games, and content generation algorithms," *IEEE Transactions on Games*, vol. 11, no. 3, pp. 195–214, 2019.

[18] J. Kowalski, M. Mika, J. Sutowicz, and M. Szykuła, "Regular boardgames," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 1699–1706.

[19] J. Kowalski, R. Miernik, M. Mika, W. Pawlik, J. Sutowicz, M. Szykuła, and A. Tkaczyk, "Efficient reasoning in regular boardgames," in *2020 IEEE Conference on Games (CoG)*, 2020, pp. 455–462.

[20] M. Stephenson, E. Piette, D. J. Soemers, and C. Browne, "An overview of the ludii general game system," in *2019 IEEE Conference on Games (CoG)*, 2019, pp. 1–2.

[21] Éric Piette, D. J. N. J. Soemers, M. Stephenson, C. F. Sironi, M. H. M. Winands, and C. Browne, "Ludii – the ludemic general game system," 2020. [Online]. Available: https://arxiv.org/abs/1905.05013

[22] D. Parlett, "What'sa ludeme," *Game & Puzzle Design*, vol. 2, no. 2, pp. 81–84, 2016.

[23] Éric Piette, M. Stephenson, D. J. N. J. Soemers, and C. Browne, "General board game concepts," 2021. [Online]. Available: https://arxiv.org/abs/2107.01078

[24] M. Stephenson and J. Renz, "Creating a hyper-agent for solving angry birds levels," in *Artificial Intelligence and Interactive Digital Entertainment Conference*, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:40742810

[25] A. Dockhorn, J. Hurtado-Grueso, D. Jeurissen, L. Xu, and D. Perez-Liebana, "Portfolio search and optimization for general strategy game-playing," 2021. [Online]. Available: https://arxiv.org/abs/2104.10429

[26] D. Churchill and M. Buro, "Portfolio greedy search and simulation for large-scale combat in starcraft," in *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, 2013, pp. 1–8.

[27] M. Khan and C. Aranha, "A novel weighted ensemble learning based agent for the werewolf game," 2022. [Online]. Available: https://arxiv.org/abs/2205.09813

[28] M. Stephenson, D. J. N. J. Soemers, E. Piette, and C. Browne, "General game heuristic prediction based on ludeme descriptions," in *2021 IEEE Conference on Games (CoG)*, 2021, pp. 1–4.

[29] E. P. C. B. Walter Crist, Matthew Stephenson, "The ludii games database: A resource for computational and cultural research on traditional board games," *Digital Humanities Quarterly*, vol. 18, no. 4, 2024.

[30] D. E. Knuth and R. W. Moore, "An analysis of alpha-beta pruning," *Artificial Intelligence*, vol. 6, no. 4, pp. 293–326, 1975. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0004370275900193

[31] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *Machine Learning: ECML 2006*, J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 282–293.

[32] C. F. Sironi, T. Cazenave, and M. H. M. Winands, "Enhancing playout policy adaptation for general game playing," in *Monte Carlo Search*, T. Cazenave, O. Teytaud, and M. H. M. Winands, Eds. Cham: Springer International Publishing, 2021, pp. 116–139.

[33] T. Cazenave, "Generalized rapid action value estimation," in *Proceedings of the 24th International Conference on Artificial Intelligence*, ser. IJCAI'15. AAAI Press, 2015, p. 754–760.

[34] G. M. J.-B. CHASLOT, M. H. M. WINANDS, H. J. V. D. HERIK, J. W. H. M. UITERWIJK, and B. BOUZY, "Progressive strategies for monte-carlo tree search," *New Mathematics and Natural Computation*, vol. 04, no. 03, pp. 343–357, 2008. [Online]. Available: https://doi.org/10.1142/S1793005708001094

[35] S. Xia, Z. Xiong, Y. Luo, WeiXu, and G. Zhang, "Effectiveness of the euclidean distance in high dimensional spaces," *Optik*, vol. 126, no. 24, pp. 5614–5619, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0030402615011493

[36] M. Stephenson, D. Soemers, É. Piette, and C. Browne, *Measuring Board Game Distance*, ser. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Germany: Springer Verlag, Jan. 2023, vol. 13865 LNCS, pp. 121–130.