General Win Prediction from Agent Experience

Raluca D. Gaina
Queen Mary University of London
London, UK
Email: r.d.gaina@qmul.ac.uk

Simon M. Lucas
Queen Mary University of London
London, UK
Email: simon.lucas@qmul.ac.uk

Diego Perez-Liebana
Queen Mary University of London
London, UK
Email: diego.perez@qmul.ac.uk

***Abstract*—The question of whether the correct algorithm is used for the problem at hand usually comes at the end of execution, when the algorithm's ability to solve the problem (or not) can be verified. But what if this question could be answered in advance, with enough notice to make changes in the approach in order for it to be more successful? This paper proposes a general agent performance prediction system, tested in real time within the context of the General Video Game AI framework. It is solely based on agent features, therefore removing potential human bias produced by game-based features observed in known games. Three different models can be queried while playing the game to determine whether the agent will win or lose, based on the current game state: early, mid and late game feature models. The models are trained on 80 games in the framework and tested on 20 new games, for 14 variations of 3 different methods. Results are positive, indicating that there is great scope for predicting the outcome of any given game.**

I. Introduction

Many researchers have approached the problem of General Video Game Playing in the last years, partly encouraged by the proliferation of benchmarks like the General Video Game AI (GVGAI; [1]) framework, the Arcade Learning Environment (ALE; [2]) and others. In the different studies, authors employ either a single or a combination of techniques in order to tackle this complex problem, in which an agent must be able to play any within a collection of games. Known examples are Mnih et al. [3] work on the Atari framework or the several times winner of the GVGAI competition, YOLOBOT [4], respectively.

A recent survey in the uses of GVGAI for research and education [5] highlights a commonality on multiple studies: not only most approaches rarely surpass 25% of victories across the set of games tested (with just some approaches reaching 50% in particular games and settings), but also the victory count is accumulated in specific games. Some games in the framework are too hard for any of the GVGAI agents developed so far, and it remains an open question as to whether they are too hard for a general approach.

This is something that may be expected a priori, but as D. Ashlock et al. [6] suggest in their work, a hyper-heuristic approach or a portfolio of agents should be able to overcome this problem. However, finding the right approach for the right game, especially if that game is unknown, is a hard challenge.

Several previous studies suggest clustering games using game features or performance of agents on them [6], [7], [8]. In general, this clustering can be used to select which agent, from a collection of different techniques, should be used to play the game at stake. This is a reasonable approach, but little thought has been put so far into analysing if the algorithm should be changed once the game has already started. The technique used in a particular game may need to be discarded in favour of another one, either because the choice was wrong in the first place, or because the game conditions have changed.

In fact, it is common for a human who is playing a game to have a certain intuition about how well are they doing mid way through it. A player in Space Invaders can see, before losing the game, that the presence of too many aliens close to the ground is a bad sign. Having most pellets still to be eaten in Pac-Man with no power pills left in the level can also be an indication of the likely (negative) outcome of the game. Our interest is to see if it is possible to give this ability to a general agent, allowing the possibility of changing technique before it is too late in the game.

The use of game features, however, poses an additional problem: including the number of pills or aliens as features is a very specific approach. In fact, even considering GVGAI terms (as presence of Non-Player Characters, portals, resources, etc.) is not general enough. This does not only tailor the methods to GVGAI (which may be hard to avoid when working with a specific framework), but also to the games the algorithm designer has seen in the past. Other features, however, can be more resilient to this bias, such as agent-based features [9]: decisiveness of action selection, speed of convergence to a recommendation or analysis of the fitness landscape.

The work presented in this paper explores the idea of designing a game outcome predictor. In particular, we propose building predictors that only focus on agent-based features, in order not to bias the prediction towards already seen games. The question this paper tries to answer is if it is possible to train a model solely on agent experiences, so it is able to estimate the probability of victory at the current state for any game within the GVGAI framework.

II. Related work

There is extensive literature on extracting various AI gameplay measures. Traditionally, these methods are predominantly used in the area of Procedural Content Generation in order to assess the quality of a level or game created automatically.

Liapis et al. [10] create models of player types called "procedural personas", which then they use to automatically generate levels of a roguelike puzzle game. For this purpose, they identify several features that the evolved agents will focus on: the number of monsters they kill, the number of treasures

collected or reaching the exit of the level. Using these different personas to automatically play-test levels, the authors are able to generate interesting levels which highlight agent strengths.

Some researchers focus more on the area of human-computer interaction and how measures extracted from game-play can be used in predicting various aspects characterizing automatically generated games (engagement, frustration and challenge in [11]; or human enjoyment when playing against different ghost teams in the game Ms Pac-Man [12]). The content and game-play features highlighted by Shaker et al. [11] in the platformer game Super Mario Bros are directly applicable to AI game-play as well as humans: number of enemies, number and width of gaps in the level, enemies placement, boxes, power-ups and events triggered during play.

Isaksen et al. [13] define several metrics characterising dice games: win bias (the difference between the probabilities of player A winning a dice battle and player A losing the battle), tie percentage (the probability of a tie in a given battle) and closeness (how much the result of the battle centered around a tie). Volz et al. [14] evaluate how close the game ended as well in the card game Top Trumps with the objective of automatically balancing the game.

However, the features explored by these authors are game-specific and applying these methods to other domains is not straightforward. When designing games, Browne and Maire [15] looked at 57 different criteria in judging an evolved game split into 3 categories, intrinsic, viability and quality. The authors use general game-playing agents to test their games, which are written in the Ludi Game Description Language. Most of the quality features analysed are resultant from AI game-play, such as depth, drama, decisiveness or uncertainty. Some of these metrics, where possible to translate to single-player games, were adapted for our study.

Several works move away from the area of PCG and instead focus on extracting measures of player behaviour to specifically tune game-playing agents or perform a deeper performance analysis than the typical win rate investigation. Khalifa et al. [16] used features from human game-play data to tune a human-like Monte Carlo Tree Search (MCTS) player. Their features mostly focused on actions, such as action repetition, change frequency or pauses, with an additional map exploration metric. The authors applied the features extracted from human data to tune a Monte Carlo Tree Search agent on 3 different games in the General Video Game AI framework (GVGAI), with mixed results.

More general measures for better analysis are depicted by Volz et al. in [9]. Their prototype implementing the measures for live game-playing agent analysis also uses the GVGAI framework, allowing for a general application of the method on several different games. Some of these metrics, such as decisiveness or action entropy, were included in this study, excluding multi-player or comparison metrics.

Some researchers use such metrics for machine learning tasks. For example, Bontrager et al. [7] cluster the games in the GVGAI framework based on the performance of several agents submitted to the corresponding competition. In this case, the performance of an agent is simply characterized by the win rate, which is shown to differ between the players. The authors signify that some agents possess skills useful in certain tasks, while other agents lack or make up for them in different ways.

Mendes et al. [8] used this conclusion to construct a hyper-heuristic agent. The authors extracted several game features (number and type of NPCs, resources available, map dimensions and number and types of other sprites) and used a classification method to determine which AI agents, selected from a subset of GVGAI competition entries, achieve highest win rates when specific game features or combination of features are present in a new game tested. The algorithm then decides which agent to query for a solution depending on the recommendation of the classifier (a Support Vector Machine and a Decision Tree). The agent selected will play the entire game with no changes.

A similar approach was employed by Horn et al. in [17] for AI hybrid evaluation (excluding the hyper-heuristic construction step). They propose a game difficulty estimation scheme based on game features (NPC types, puzzle elements, pathfinding requirements or traps). These are arguably more open to human bias, as each metric is evaluated manually. Although the game difficulty features identified do not correspond to agent win rates, the authors carry out an analysis which gives a deeper insight into reasons for agent performance levels.

These works are, however, based on game features as defined by human knowledge on the existing data set. This paper proposes a game win predictor based solely on agent experiences, aiming to remove potential human bias resultant from designing features seen on known games.

III. Background

A. General Video Game AI

The domain chosen for this study is the General Video Game AI (GVGAI) framework [18], which allows for general video-game playing agent testing on a wide range of different games. The diversity in game features, difficulty and different agent performance is showcased in the previous section, which highlights it as an appropriate environment for general agent testing. As opposed to other frameworks focused on the area of General (Video) Game Playing, GVGAI does not make the game ruleset available to the AI agents. Instead, the information given contains the current game state with the sprites present in the level, the avatar the agent is controlling, the action set available, the history of events up until that game tick and the current game score. Additionally, in the planning tracks, agents have access to a non-deterministic Forward Model (FM), which they may use to simulate possible future states. All information is offered to the agents through Java objects. All games in the framework are real-time, giving 1 second initialization time and 40 milliseconds decision time at every game tick. There are 100 single-player grid-physics games available in the framework as of March 2018, all of which are used in this study for a large scale experiment.

B. Game-playing agents

This section describes the 3 methods that the game-playing agents used in this study are based on.

1) Random Search (RS): This agent uniformly samples at random action sequences of length L within the allocated budget and chooses for play the first action in the best solution found. In order to evaluate a sequence, the FM is used to simulate through the actions, in turn, until the end of the game or the end of the sequence is reached. The value of the final state is computed using a heuristic (see Equation 1, where H^+ is a large positive integer number and H^- is a large negative integer number), this becoming the value of the solution.

$$f = score \; + \; \begin{cases} H^+, \; if \; loss \\ H^-, \; if \; win \end{cases} \tag{1}$$

2) Rolling Horizon Evolutionary Algorithm (RHEA): This agent is one of the promising methods in the domain of General Video Game Playing, as showcased in [19]. In its vanilla form, it randomly initializes a population of size P with individuals of length L, which it then evolves over several generations by applying various evolutionary techniques, such as uniform crossover and uniform 1-bit mutation. One individual represents a sequence of actions, which is evaluated similarly to the RS procedure: the FM model is used to simulate through the sequence of actions and the final state is evaluated with the same heuristic described in Equation 1. The first action of the best individual found at the end of the evolution is selected.

Previous work is used to select the agents employed in this study, the best of each being chosen. As a result, vanilla RHEA [20], EA-MCTS [21] and EA-Shift [19] form the subset of RHEA variations. EA-MCTS employs a different seeding method, using the solution provided by a Monte Carlo Tree Search agent (awarded half the thinking budget) to generate its initial population. EA-Shift focuses on Monte Carlo roll-outs added at the end of individual evaluation, as well as a shift buffer applied for population management (the population is not discarded and reinitialized at every game tick, but instead shifted to the left and new random actions are added at the end of each individual). Additionally, we add EA-All for completeness, which combines EA-Shift with EA-MCTS.

3) Monte Carlo Tree Search (MCTS): This agent is the most dominant method in GVGAI, many competitors choosing it as the basis for their entry. A comprehensive survey of MCTS techniques can be found in [22]. MCTS builds an asymmetric tree to make its choices, relying on statistics gathered from several simulated play-throughs. Each iteration that adds to the tree statistics begins by navigating down the tree using the tree policy (Upper Confidence Bound for Trees with an exploration constant of $\sqrt{2}$ for the agents used in this study, aiming to balance between exploration and exploitation). When it finds a node not yet fully expanded, a new child of this node is added to the tree, by selecting a new action to play from this game state. A Monte Carlo simulation (or roll-out) is run from the newly added child until the end of the game or a depth L is reached. The final state is evaluated with the same heuristic from Equation 1 and the value is backed up the tree, updating all nodes visited during this iteration. In our implementation, the nodes only store statistics and not the actual game states, the FM being used to simulate through the tree at every iteration. A selective window containing W iterations is used for analysis as the equivalent of RHEA population size P. The most visited action at the end of the process is selected for play.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2}$$

C. Classification

Due to the high variety of the games in the GVGAI framework and the low overall performance of the general agents (most games remain too difficult to be solved), as highlighted in the literature review, the $F1$-Score (see Equation 2) will be reported as to the quality of the classifiers employed in this study. This represents the harmonic average between precision and recall, 1 signifying the best value and 0 the worst. It is meant to be a better measure of classifier quality than accuracy when there is an imbalance in data (in this case, a majority of games resulting in a loss, see Table I) [23].

IV. Data set

To obtain the set of agents used to generate the data set, 3 roll-out values L were tested for RS (10, 30, 90); 2 parameter sets were tested for all RHEA variations (P=2,L=8 and P=10,L=14); 3 parameter sets were tested for MCTS (W=2,L=8; W=10,L=10 and W=10,L=14).

All 14 algorithm variations described previously were run on all 100 games publicly available in the GVGAI Framework, 20 times on each of the 5 levels, being given a budget of 900 FM calls. Each run produced 2 log files, recording information about the agent inner processing, as well as its actions played and game scores, at every game step, in addition to the final game results (win/loss, final score and number of game ticks).

Data set and processing scripts are publicly available[1]. On each game, Formula-1[2] points are awarded attending to a ranking determined by win rate. The first 10 ranked entries receive 25 points, second 18, then 15, 12, 10, 8, 6, 4, 2, 1 and 0 for the 11^{th} and below positions. Points across games are summed up for an overall ranking, shown in Table I.

A list of all the features extracted can be found below. Features ϕ_2, ϕ_8, ϕ_9, ϕ_{10}, ϕ_{11} and ϕ_{12} compute averages from the beginning of the game up until the current tick t. Features ϕ_5, ϕ_6, ϕ_{11} and ϕ_{12} rely on the FM. Only *agent features* were used in this study, with the exception of the game score:

- ϕ_1 **Current game score**
- ϕ_2 **Convergence**: Iteration number when the algorithm found the final solution recommended during one tick. A low value indicates quick and almost random decisions.
- ϕ_3 **Positive rewards**: Count of positive scoring events.
- ϕ_4 **Negative rewards**: Count of negative scoring events.

[1]https://github.com/rdgain/ExperimentData/tree/GeneralWinPred-CIG-18

[2]Not to be mistaken with F1 accuracy measure for classifiers.

#	Algorithm	Points	Avg. Wins
1	**10-14-EA-Shift**	1225	26.02 (2.11)
2	**10-RS**	898	24.33 (2.13)
3	**2-8-EA-All**	888	23.95 (1.98)
4	**30-RS**	885	22.49 (2.02)
5	**2-8-EA-Shift**	866	24.54 (2.00)
6	**14-MCTS**	780	24.29 (1.74)
7	**10-14-EA-All**	695	22.66 (2.02)
8	**10-14-RHEA**	664	23.23 (2.08)
9	**10-MCTS**	652	24.01 (1.65)
10	**2-8-EA-MCTS**	621	23.98 (1.73)
11	**10-14-EA-MCTS**	618	23.99 (1.80)
12	**8-MCTS**	594	23.42 (1.61)
13	**90-RS**	457	16.31 (1.67)
14	**2-8-RHEA**	257	18.33 (1.77)

TABLE I: GVGAI-style Formula-1 point ranking of all methods. Type and configuration (roll-out length L if one value, population size P and roll-out length L if two values) are reported, followed by the sum of Formula-1 points across 20 games and the average win rate.

ϕ_5 **Success**: The slope of a line over all the win counts. Win count increases when any solution sees the end of the game with a win, at any point during search. A high value indicates the increase in discovery of winning states.

ϕ_6 **Danger**: The slope of a line over all the loss counts. Loss count increases when any solution sees the end of the game with a loss, at any point during search. A high value indicates the increase in discovery of losing states.

ϕ_7 **Improvement**: The slope of a line resultant from all best fitness values plotted over game tick. A high value indicates good fitness improvement.

ϕ_8 **Decisiveness**: Shannon entropy (SE) over the number of times each of the possible actions was recommended (it was the first action of a solution in the final population or analysis window). In all cases of distribution-based features, a high value suggests actions of similar value; the opposite shows some to be dominating.

ϕ_9 **Options exploration**: SE over the number of times each of the possible actions was explored (it was the first action of a solution at any time during search). A low value shows an imbalance in actions explored.

ϕ_{10} **Fitness distribution**: SE over fitness per action.

ϕ_{11} **Success distribution**: SE over win count per action.

ϕ_{12} **Danger distribution**: SE over loss count per action.

The full feature file (processing all games and algorithms for global classifiers) took approximately 2.5 hours to generate, from 26GB of raw metrics data split over 281.4k files (Dell Windows 10 PC, 3.4 GHz, Intel Core i7, 16GB RAM, 4 cores).

Figure 1 shows the pairwise correlation between the features extracted (using the Pearson correlation coefficient), in a comparison between the early (first 30% of game ticks) and late (last 30% of game ticks) phases of the games. Differences are small, but they do exist. An aspect worth highlighting is the higher correlations in the bottom right corner in the late game phase versus the early game phase (i.e. the success distribution appears to increase correlation with all other features).

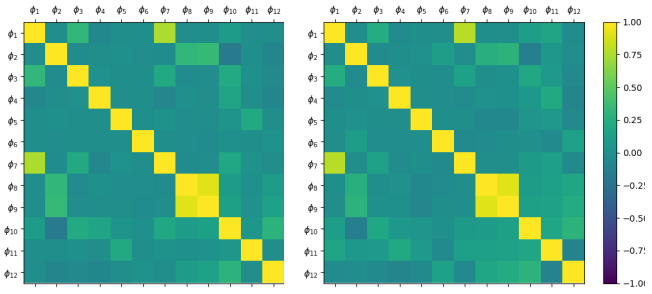


Fig. 1: Feature correlation early game (left, 0-30% of all games) and late game (right, 70-100% of all games)

Another interesting positive correlation that only appears in the late game phase is that between the sense of danger and the convergence, suggesting agents take longer to settle on their final decision when surrounded by possible losses. The case of one action appearing to be dominating leads to a persistent negative correlation between convergence and fitness distribution. This suggests that agents are unlikely to change their decision if one action is deemed significantly better than the rest and try a less promising move.

V. Predictive Models

For the purpose of these experiments, all data sets were randomly split 80/20 in training/test subsets.

The study presented in this paper aims to build several classifier models from agent features extracted, which would predict a win or a loss during play of a new game. We also show that the system is robust enough to handle new agents with significantly different play styles as well.

It takes approximately 10 seconds to process a full feature file and split the data into train and test, another 10 seconds to train a global model on a full feature file (or 1 minute if cross validation is used). Predicting the outcome of 28000 instances takes approximately 1 minute, the equivalent of 2.26ms per instance. As the data used in this study is publicly available, adapting the methods to different problems or agents would only involve extracting the relevant features from the newly introduced agents or problems.

A. Baseline

The baseline model all our classifiers are compared against is a simple rule based predictor incorporating human knowledge. In classic arcade games and most GVGAI games, gaining score is a good thing and often means the player is on the right path to winning if they increase their score. This idea is implemented as described in Equation 3, which compares the count of positive scoring events recorded to the count of negative events. This classifier's performance on the test set is shown in Table II, where it can be observed that it reaches an F1-Score of only 0.59 despite a high precision (0.70). This model will be referred to as R_g in the rest of this paper.

$$\hat{y} = \begin{cases} win & if\ \phi_3 > \phi_4 \\ lose & otherwise \end{cases} \quad (3)$$

	Precision	Recall	F1-Score	Support
Loss	0.83	0.52	0.64	20500
Win	0.35	0.70	0.46	7500
Avg / Total	**0.70**	**0.57**	**0.59**	**28000**

TABLE II: Global rule based classifier report. Global model tested on all game ticks of all instances in the test set.

	Precision	Recall	F1-Score	Support
Loss	1.00	0.99	0.99	20500
Win	0.97	0.99	0.98	7500
Avg / Total	**0.99**	**0.99**	**0.99**	**28000**

TABLE III: Global AdaBoost classifier report. Global model tested on all game ticks of all instances in the test set.

ϕ_1	0.24	ϕ_2	0.04	ϕ_3	0.08	ϕ_4	0.06
ϕ_5	0.2	ϕ_6	0.1	ϕ_7	0.12	ϕ_8	0
ϕ_9	0.06	ϕ_{10}	0.02	ϕ_{11}	0.02	ϕ_{12}	0.06

TABLE IV: Feature importances extracted from global model. ϕ_x represents a feature and its associated importance.

B. Classifier selection - global model

Seven classifiers (with default hyper-parameters if not specified) were trained and tested for proof of concept and classifier analysis. These are K-Nearest Neighbors (5 neighbours), Decision Tree (5 max depth), Random Forest (5 max depth, 10 estimators), Multi-layer perceptron (1 alpha), AdaBoost-SAMME [24], Naive Bayes and Dummy (simple rule decision making, very poor general performance to be used as another possible baseline). All classifiers are used as per their implementation in the Scikit-Learn Python 2.7.14 library [25].

Cross-validation with 10 folds was used during training to assess performance, the classifiers obtaining 0.95, 1.00, 0.98, 0.96, 1.00, 0.95 and 0.66 accuracy during validation, respectively. Both AdaBoost and the Decision Tree classifier achieved high accuracy values during validation and test (0.99, see Table III for its performance measures) and were deemed equal. AdaBoost was selected as the main classifier for the rest of the experiments presented in this paper.

Feature importances according to AdaBoost can be seen in Table IV. It appears that the game score is most important in distinguishing wins and losses, unsurprisingly, but it is followed close behind by the number of wins seen by the agents, the improvement in fitness and the sense of danger. The decisiveness of the agents is considered to have no impact in deciding the outcome of a game.

C. Model training

All games were split into logical phases for predictions at various points in the games: early game ($0-30\%$), mid game ($30-70\%$) and late game ($70-100\%$). Multiple models were then trained for each of the phases, using agent features based on metrics logged only in the ticks corresponding to each interval. 3 different models resulted, referred to as E_g, M_g and L_g, respectively, in the rest of this paper.

The performance of the models was analysed by testing each on the 20 new games, on their corresponding interval of game ticks. During training with 10-fold cross-validation, they achieve 0.80, 0.82 and 0.99 accuracy, respectively. During test on the new games, they report accuracies of 0.73, 0.80 and 0.99 (0.70, 0.80 and 0.99 F1-Scores), respectively. These results are satisfactory and allow for further exploration.

VI. Live play results

For the experiments in this paper, we simulated live play by extracting agent features from the log files for a range of ticks ($T = \{100 \cdot a : \forall\ a \in [1, 20] : a \in \mathbb{N}\}$), all from the beginning of the game until the current tick tested $t \in T$. Gameplay from all 14 algorithms on the 20 test games (20 plays on each of the 5 levels) was used to compute the final results. Each model was tested on each of the feature files, being asked to predict the game outcome every 100 ticks.

Simulated live play results can be observed in Figure 4. The simple rule based model achieves a high performance in some of the games and it proves better than the trained predictive models (i.e. "Aliens", "Defem", "Chopper", "Eggomania"). As these are games with plenty of scoring events, it is unsurprising that the simple logic of R_g works in these cases. However, there are games where the trained models achieve much better predictions ("Ghost Buster", "Colour Escape" or "Frogs"). The reward gain is not linear in these games, meaning the player need not necessarily be phased by a temporary decrease in score, or too optimistic as a result of score gains.

It is interesting to observe that the trained models do not follow the expected curves (E_g being better in the early game phase and then decreasing, M_g showing a spike in the middle of the game and L_g offering good predictions only towards the end of the game). Instead, the early game model appears to have a generally low performance compared to the rest, which can be explained by the limited information available for this particular model. The late game model seems particularly strong in games with very low win rate ("Fireman", for example, in which both E_g and M_g are predicting wins, yet the overall win rate remains at 0% for this game).

It is most interesting to observe the games with close to 50% win rate, "Defem" and "Ghost Buster". High F1-Score values here indicate that the predictors are able to correctly judge both wins and losses equally. And indeed, in both games, the trained models achieve F1-Scores of over 0.8 only half way through the game. Model M_g appears to excel in these situations, meaning that it can recommend the game outcome and possibly the better approach to be used.

It is important to highlight at this point the importance of this great result: the predictor is able to foresee with high reliability, after only a fourth of the game has been played, if the agent is going to win or lose the game. In this case, games that are either won or lost with the same probability as a coin flip. And these are truly general models: trained in *different* games, using only *agent experience* features. This shows a great scope for the system's use within hyper-heuristic methods, as some of the algorithms tested in this study do win at "Defem" and "Ghost Buster". Devising a procedure that determines which is that better method and switches to

	Early-P	Mid-P	Late-P	Total-M
E_g	0.22 (0.72)	0.42 (0.74)	0.49 (0.76)	0.38 (0.74)
M_g	0.29 (0.72)	**0.57 (0.79)**	**0.71 (0.83)**	**0.53 (0.78)**
L_g	0.01 (0.73)	0.05 (0.74)	0.22 (0.76)	0.09 (0.74)
R_g	**0.42 (0.67)**	0.47 (0.61)	0.46 (0.58)	0.45 (0.62)
Total-P	0.24 (0.71)	0.38 (0.72)	**0.47 (0.73)**	

TABLE V: F1-Scores each model per game phase over all games, accuracy in brackets. Each row is a model, each column is a game phase. Highlighted in bold is the best model on each game phase, as well as overall best phase and model.

it when the prediction is a loss is scope for future work, but having a system that indicates if a change should be made is the first step in that direction.

All predictive models were further analysed as to their average quality considering all games. To this extent, table V summarises F1-scores for all models on the different game phases identified. The models are the same as discussed in Section V-C, and they are tested in the same previous test setting, with features extracted from the beginning of the game until the current tick which falls at the half point in each game phase (15%, 50% or 85% of the game ticks).

The results indicate the rule-based model to be giving consistent average performance throughout the game phases, being the best in the early phase with an F1-score of 0.42. In the Mid and Late game phases, model M_g is the best, achieving a 0.57 and 0.71 F1-score, respectively. Overall, the best model is M_g with an F1-score average of 0.53.

It is not surprising that the M_g model is the best in its respective game phase, and it is expected that the prediction quality is generally lower in the Early game phase, when there is less information available and it is harder to judge if the agent's performance is good enough or not. A significant result extracted from the summarised data is that model M_g achieves high (if not the best) F1-scores across all game phases, indicating that the system can identify with high confidence whether the agent is performing well or not and leaving open the possibility of switching approaches appropriately.

VII. Conclusion

This paper presents work in extracting agent features from AI gameplay in a generic setting, using the General Video Game AI framework (GVGAI). Game-specific features are specifically excluded in order to avoid potential bias introduced by human knowledge of already known games. 14 total variations of Rolling Horizon Evolutionary Algorithm, Monte Carlo Tree Search and Random Search are used to generate data on 100 games, playing 20 times each of the 5 levels. Three different models corresponding to early, middle (mid) and late game phases are trained on 80 randomly selected games and tested on the remaining 20 through live play simulation and repeated predictions every 100 game ticks.

The results obtained indicate that models are able to correctly predict in most cases the outcome of the game with sufficient time before the end of the game to make appropriate changes in the method employed. Throughout all experiments, it is apparent that some models have better predictions in specific games than others. Additionally, the mid-game phase model proved to have the best overall performance, achieving an F1-Score of 0.53 (0.78 accuracy) across all test games and game phases. It is also the strongest model in the individual mid and late phases, being bested in the early game phase only by the simple rule predictor implemented (which incorporates the human knowledge that gaining score leads to a win).

Regarding next steps, a hyper-heuristic agent will be built, able to switch between algorithms appropriately while playing the game, based on the predictions given by our system. The task can be split into two: identifying which features need improvement and which method leads to the desired behaviour. A prediction explanatory system could be responsible for the first part of this task and first steps towards this system are presented in Figure 2, which uses the LIME system[3]. The example provided is an explanation of the prediction of each model at game tick 300 in "Frogs" level 0, when played by 2-8-RHEA. There is an obvious difference between features and a clear signaling of which features currently indicate a loss. Therefore, a hyper-heuristic method could make use of this analysis to correct the loss indications.

Additionally, new methods could be introduced to the system in order to create stronger models, able to adapt to any style of play. The current system is robust enough to handle testing on new algorithms: Figure 3 shows predictions trained with data generated only by RHEA and RS variants, but tested live with an MCTS controller playing the game. If this is compared to Figure 4s, it can be seen that all models are able to maintain a similar shape and still accurately predict the outcome half way through the game.

Lastly, more features could be integrated to better describe player experience, such as empowerment [26], spatial entropy or characterization of agent surroundings [9].

Acknowledgment

This work was funded by the EPSRC Centre for Doctoral Training in Intelligent Games and Game Intelligence (IGGI) EP/L015846/1.

References

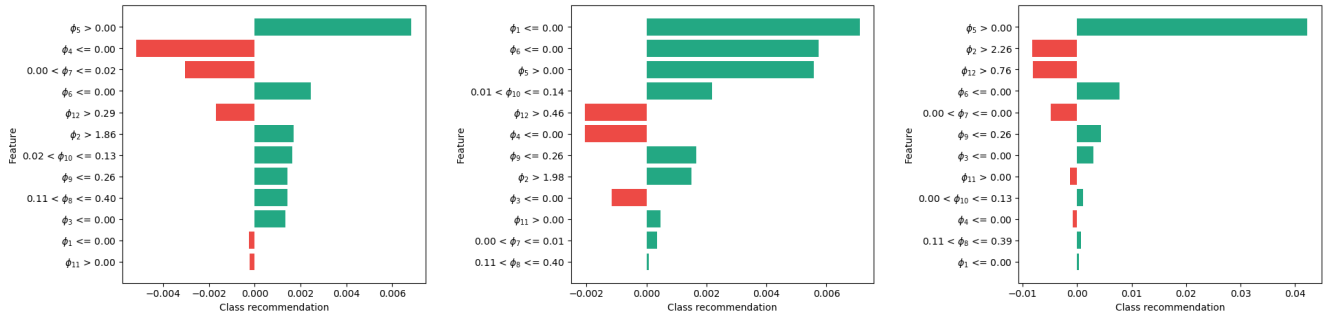
[1] D. Perez-Liebana, S. Samothrakis, J. Togelius, S. M. Lucas, and T. Schaul, "General Video Game AI: Competition, Challenges and Opportunities," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016, pp. 4335–4337.

[2] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents." *J. Artif. Intell. Res.(JAIR)*, vol. 47, pp. 253–279, 2013.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[4] T. Joppen, M. Moneke, N. Schroder, C. Wirth, and J. Furnkranz, "Informed Hybrid Game Tree Search for General Video Game Playing," *IEEE Transactions on Computational Intelligence and AI in Games*, 2017.

[3] https://github.com/marcotcr/lime



(a) Early model (b) Mid model (c) Late model

Fig. 2: Class predictions by features. Red signifies the model feature recommends a loss, green a win. The probability of class being selected based on individual feature recommendation is plotted on the X-axis.

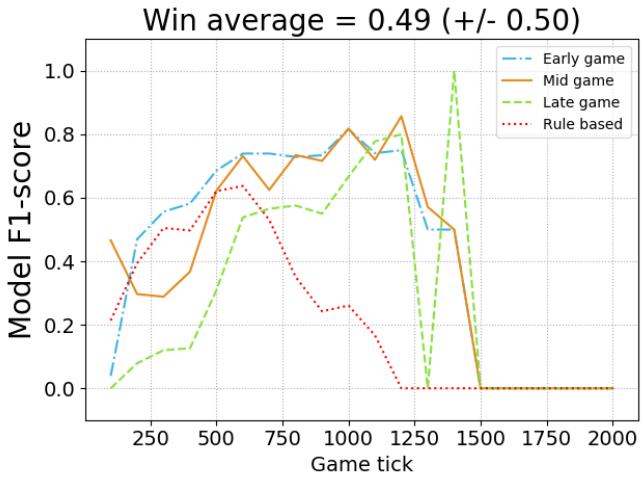


Fig. 3: Model F1-scores in the game `Ghost Buster`, trained on variations of RHEA and RS (80 training games) and tested on MCTS.

[5] D. Perez-Liebana, J. Liu, A. Khalifa, R. D. Gaina, J. Togelius, and S. M. Lucas, "General video game ai: a multi-track framework for evaluating agents, games and content generation algorithms," *arXiv preprint arXiv:1802.10363*, 2018.

[6] D. Ashlock, D. Pérez-Liébana, and A. Saunders, "General Video Game Playing Escapes the No Free Lunch Theorem," in *2017 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2017.

[7] P. Bontrager, A. Khalifa, A. Mendes, and J. Togelius, "Matching games and algorithms for general video game playing," in *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2016, pp. 122–128.

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

[9] V. Volz, D. Ashlock, S. Colton, S. Dahlskog, J. Liu, S. M. Lucas, D. P. Liebana, and T. Thompson, "Gameplay Evaluation Measures," in *Articial and Computational Intelligence in Games: AI-Driven Game Design (Dagstuhl Seminar 17471)*, E. André, M. Cook, M. Preuß, and P. Spronck, Eds. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, pp. 36–39.

[10] A. Liapis, C. Holmgård, G. N. Yannakakis, and J. Togelius, "Procedural personas as critics for dungeon generation," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2015, pp. 331–343.

[11] N. Shaker, S. Asteriadis, G. N. Yannakakis, and K. Karpouzis, "Fusing visual and behavioral cues for modeling user experience in games," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1519–1531, Dec 2013.

[12] W. Sombat, P. Rohlfshagen, and S. M. Lucas, "Evaluating the enjoyability of the ghosts in ms pac-man," in *2012 IEEE Conference on Computational Intelligence and Games (CIG)*, Sept 2012, pp. 379–387.

[13] A. Isaksen, C. Holmgård, J. Togelius, and A. Nealen, "Characterising score distributions in dice games," *Game and Puzzle Design*, vol. 2, no. 1, 2016.

[14] V. Volz, G. Rudolph, and B. Naujoks, "Demonstrating the feasibility of automatic game balancing," *CoRR*, vol. abs/1603.03795, 2016. [Online]. Available: http://arxiv.org/abs/1603.03795

[15] C. Browne and F. Maire, "Evolutionary game design," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 1, pp. 1–16, 2010.

[16] A. Khalifa, A. Isaksen, J. Togelius, and A. Nealen, "Modifying mcts for human-like general video game playing." in *IJCAI*, 2016, pp. 2514–2520.

[17] H. Horn, V. Volz, D. Prez-Libana, and M. Preuss, "MCTS/EA hybrid GVGAI players and game difficulty estimation," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, Sept 2016, pp. 1–8.

[18] D. Perez-Liebana, S. Samothrakis, J. Togelius, T. Schaul, S. Lucas, A. Couetoux, J. Lee, C.-U. Lim, and T. Thompson, "The 2014 General Video Game Playing Competition," in *IEEE Transactions on Computational Intelligence and AI in Games*, vol. PP, no. 99, 2015, p. 1.

[19] R. D. Gaina, S. M. Lucas, and D. P. Liébana, "Rolling Horizon Evolution Enhancements in General Video Game Playing," in *Proceedings of IEEE Conference on Computational Intelligence and Games*, 2017.

[20] R. D. Gaina, J. Liu, S. M. Lucas, and D. P. Liébana, "Analysis of Vanilla Rolling Horizon Evolution Parameters in General Video Game Playing," in *Springer Lecture Notes in Computer Science, Applications of Evolutionary Computation, EvoApplications*, no. 10199, 2017, pp. 418–434.

[21] R. D. Gaina, S. M. Lucas, and D. P. Liébana, "Population Seeding Techniques for Rolling Horizon Evolution in General Video Game Playing," in *Proceedings of the Congress on Evolutionary Computation*, 2017.

[22] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A Survey of Monte Carlo Tree Search Methods," in *IEEE Trans. on Computational Intelligence and AI in Games*, vol. 4, no. 1, 2014, pp. 1–43.

[23] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[24] J. Zhu, S. Rosset, H. Zou, and T. Hastie, "Multi-class AdaBoost," *Ann Arbor*, vol. 1001, no. 48109, p. 1612, 2006.

[25] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 2013, pp. 108–122.

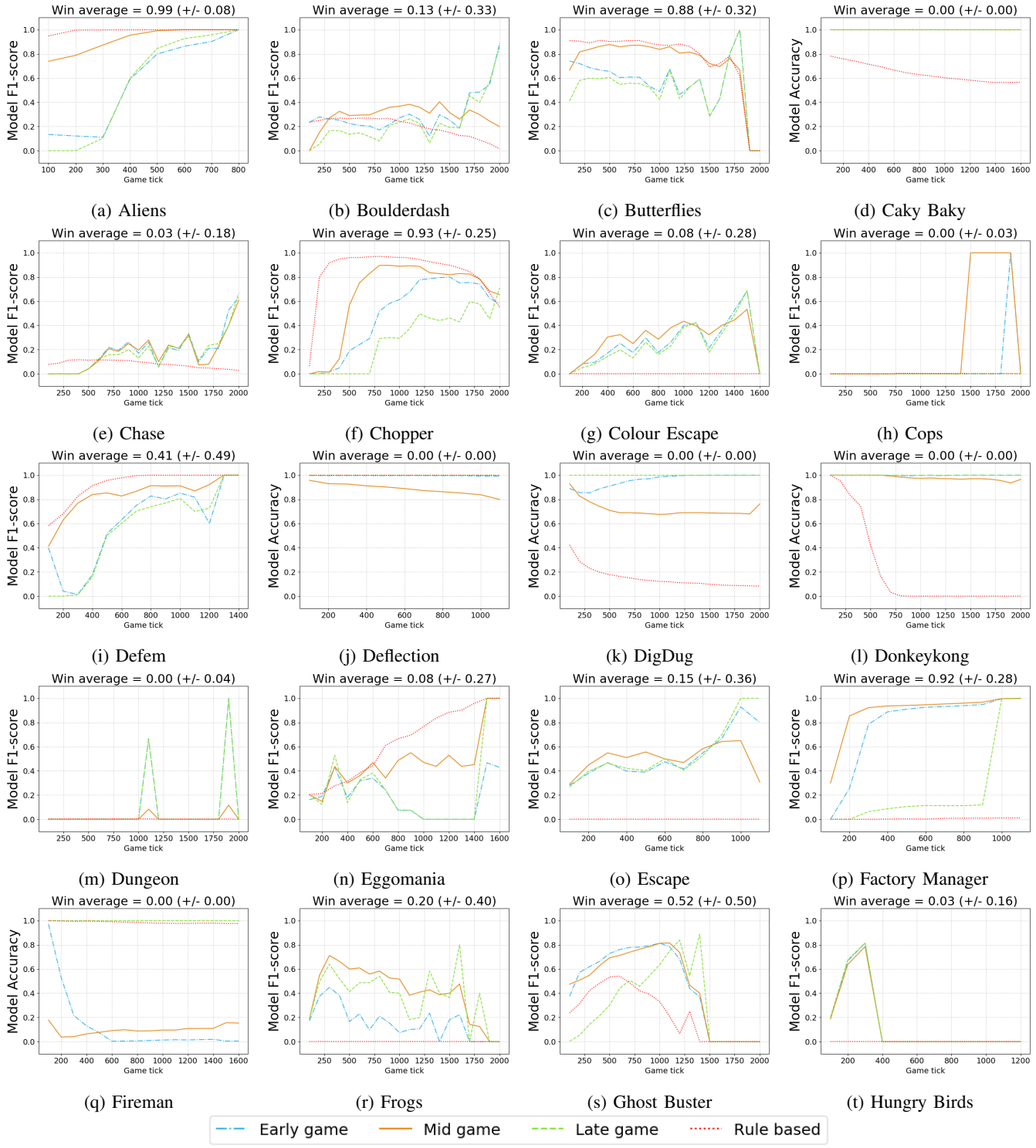


Fig. 4: Model F1-scores for each game in the test set, averaged over up to 1400 runs, 14 agents, 100 runs per game. Game ticks are displayed on the X axis, maximum 2000 game ticks. 3 different predictor models trained on early, mid and late game data features, as well as the baseline rule-based predictor. If F1-scores were 0 for all models, accuracy is plotted instead. Additionally, win average is reported for each game.

[26] C. Guckelsberger, C. Salge, J. Gow, and P. Cairns, "Predicting player experience without the player.: An exploratory study," in *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, ser. CHI PLAY '17. New York, NY, USA: ACM, 2017, pp. 305–315.