Game State Evaluation Heuristics in General Video Game Playing

Bruno S. Santos, Heder S. Bernardino
Departament of Computer Science
Universidade Federal de Juiz de Fora - UFJF
Juiz de Fora, MG, Brasil
Email: bruno.soares@ice.ufjf.br, hedersb@gmail.com

***Abstract*—In General Game Playing (GGP), artificial intelligence methods play a diverse set of games. The General Video Game AI Competition (GVGAI) is one of the most famous GGP competitions, where controllers measure their performance in games inspired by the Atari 2600 console. Here, the GVGAI framework is used. In games where the controller can perform simulations to develop its game plan, recognizing the chance of victory/defeat of the possible resulting states is an essential feature for decision making. In GVGAI, the creation of appropriate evaluation criteria is a challenge as the algorithm has no previous information regarding the game, such as win conditions and score rewards. We propose here the use of (i) avatar-related information provided by the game, (ii) spacial exploration encouraging and (iii) knowledge obtained during gameplay in order to enhance the evaluation of game states. Also, a penalization approach is adopted. A study is presented where these techniques are combined with two GVGAI algorithms, namely, Rolling Horizon Evolutionary Algorithm (RHEA) and Monte Carlo Tree Search (MCTS). Computational experiments are performed using 20 deterministic and stochastic games, and the results obtained by the proposed methods are compared to those found by their baseline techniques and other methods from the literature. We observed that the proposed techniques (i) presented a larger number of wins and F1-Scores than those found by their original versions and (ii) obtained competitive solutions when compared to those found by methods from the literature.**

***Keywords*-General Video Game Playing; Game State Evaluation; Monte Carlo Tree Search; Rolling Horizon Evolutionary Algorithm;**

I. Introduction

For a long time games have been used to test new artificial intelligence (AI). As the definition of intelligence varies, developing tests to measure AI techniques performances is considered a great challenge. Some characteristics associated with the intelligent behaviour are the capacity for logic reasoning, understanding, learning, planning and problem-solving. Depending on the game one or more of these characteristics are a necessary part of some players scope of abilities to perform well. Therefore, once games usually have a well-defined set of rules and objectives, developing game playing agents is considered to be an easy and valid way to test new AI techniques.

In past decades, game playing controllers have experienced significant improvement, even managing to surpass expert human players in some cases, such as in the board games GO (1) and chess (2). Despite the good results found by these methods, they are commonly designed to a specific game, having no capability of generalizing the skill used or learned to perform well in other games. This limitation brings the challenge of developing and studying controllers capable of such generalization. In order to provide an environment to study and develop AI methods with such characteristics, new forms of testing and creating controllers arose, where techniques can be tested in different games with only a little knowledge of the environment they are playing in. Three of these new environments are: (i) General Game Playing Competition (GGP) (3), that challenge the controllers with board games; (ii) Arcade Learning Environment (ALE) (4); and (iii) General Video Game Playing Competition (GVGAI) (5). The last two cases consider Atari 2600 based arcade games to challenge the AI with the difference that ALE presents the world to the controller as a screen capture. Here, the GVGAI competition framework and rules are used.

In game playing, the correct evaluation of the advantageous or disadvantageous features on a given game is a way of improving the playing skills, as one can plan to divide the winning condition in smaller and easier to conquer objectives. Using human knowledge to improve the performance of AI agents was first used in the pioneer self-learning checkers playing method developed by Samuel (6), where the technique chooses from a list of possible desirable characteristics the ones which better enhance the probability of winning the game. Since then, most of the high-level game playing algorithms perform some kind of state evaluation, being provided by humans (6) or automatically generated evaluations with machine learning (2).

In GVGAI, the controllers do not have much time to adapt to the games. So, performing the analysis necessary to find which conditions are favourable to that specific game is a challenge. Here, we explore the techniques of general state evaluation, analyzing three approaches and their impact when combined with two popular GVGAI algorithms, namely, Rolling Horizon Evolutionary Algorithm (RHEA) and Monte Carlo Tree Search (MCTS). The first technique considers avatar-related game features, its health and the number of resources in its possession. In the second approach, the exploration of the map is encouraged

through a penalization in the evaluation when the avatar remains in a same region of the level it is playing. The last technique proposed here involves different forms of using the knowledge obtained in the simulations. This is done by recording the outcome of the collision with the sprites during the game and increasing the evaluation score when the avatar is near beneficial sprites and decreasing when around harmful ones.

This paper is divided into 7 sections. A literature review of state evaluation techniques in both GVGAI and GGP is presented in Section II. Section III contains the background information about the framework and testing algorithms used. Our proposed techniques are detailed in Section IV. The computational experiments results and discussion is present in Section V. In Section VI we present our concluding remarks and future work.

II. Literature Review

This section presents algorithms found in the literature on state evaluation in general game playing.

Many works can be found in the literature regarding evaluating states in GGP. These techniques focus on gathering information from the game rules, being by feature selection (7)(8) or using neural networks (9). These methods achieved interesting results when compared to UCB (Section III-B) based techniques.

Some performance metrics to test GVGAI algorithms are proposed by Guerrero-Romero et al. (10) where the controllers have different objectives in the game, such as exploration maximization, knowledge discovery and estimation. A heuristic is proposed for each evaluation metric, which rewards its specific objective.

Perez et al. (11) in their work proposed the use of some penalty values in the evaluation; The opposite action penalty, that punishes using spacial redundant actions, which would not change the avatar position (e.g. moving left then right). Blocked movement penalty diminishes the evaluation of states obtained after applying movements that do not change the position os the avatar (e.g. moving against a wall). Repelling pheromones trails, a technique where the position visited by the avatar and its proximity is marked and its value is subtracted from the evaluation value.

A knowledge-based evaluation (KB) approach is one of the proposed techniques by Soemers, Dennis et al. (12) to enhance MCTS performance. This technique keeps a record of the avatar collisions occurred through the game and its outcomes. Thus it rewards or penalizes the proximity with sprites that are to be considered beneficial or prejudicial to the avatar, respectively.

Similarly to KB evaluations, Park, Hyunsoo and Kim, Kyung-Joong (13) proposed a heuristic where the information gathered in the simulations is used to define the goodness of sprites, but instead of using the distance to the components of the game this approach builds an influence map using this information. This influence map is used then, to bias the evaluation based on the position of the avatar.

III. Background

A background is presented in this section, containing a description of GVGAI framework (with the rules of the competition), the main GVGAI controllers (RHEA and MCTS), and the algorithms used in the comparative analysis of the computational experiments.

A. GVGAI Framework

GVGAI framework is a Java-coded environment that allows the creation of controllers to play a set of single and multiplayer 2D Atari inspired games. These games can be classified into different categories and difficulties (14; 15), challenging the controllers to discover and complete different types of objectives.

To describe the different games a Video Game Description Language (VGDL) (16) was created, that allowed another venue of research to arise, the automatic generated games and levels. The competition rules are defined in a way that all games must have a win condition, a time limit for that condition to be reached and a game score. In the competition, results are defined using a Formula-1 scoring system (F1-Score). Once all controllers have played one game, a rank is made by sorting first by the number of victories, followed by the average game score and the average time they spent to finish a given game. According to this rank position, the agents receive 25, 18, 15, 12, 10, 8, 6, 4, 2 and 1 points, from the first to the tenth ranked player, with the rest receiving 0 points. When compared across different games, the controller that achieves the highest F1-Score is considered the winner.

Before starting, the player has no previous information about which game it is going to play. However, in order to better plan its strategy, it is provided with a series of data about the current state of the game. A list of observations is given with the position of each sprite, the category of that sprites type, whether it is a wall, non-player character (NPC), portal, resource, movable or immovable. A set of avatar-related information is also provided such as its current position, available actions, type, health points, resources, score and game time. This information is used together with a simulation system called Forward Model (FM) where, given a previous current or simulated state, allows for the controller to obtain one possible resulting state (as games may be stochastic) after performing an action. The controller must inform a valid action every game tick (defined as 40ms in the competition). Hence, a good strategy with a low computational cost is important.

B. Monte Carlo Tree Search (MCTS)

Since the success obtained by a Monte Carlo Tree Search (17) combined with a neural network in playing

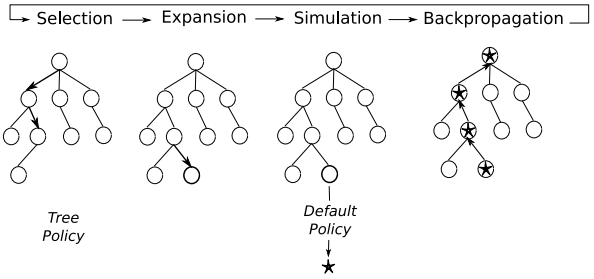


Figure 1. Monte Carlo Tree Search scheme

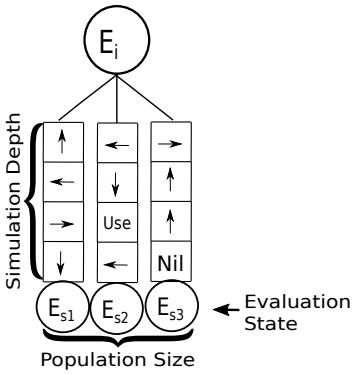


Figure 2. Illustration of RHEA population with the initial (E_i) and simulated (E_s) states.

the game GO (1) this technique has been explored in the game playing field of study obtaining significant results. Throughout GVGAI competition, MCTS-based algorithms have dominated other techniques. A vanilla version of the algorithm provided by the framework is used here in the computational experiments.

In MCTS algorithm a root node is created at each game step. Then, as shown in Figure 1 the algorithm repeats four steps during the given time budget. First, a non-terminal node with unvisited children is selected by descending the root node using a tree policy. Then this node is expanded by adding a new child to it. From this node, a default policy (applying random actions) is used to simulate using the FM until a predefined depth is reached. Finally, the state reached after the simulation step is evaluated using a heuristic and its value is used to update all the nodes that have been visited during this iteration. The algorithm returns the child of the root node that is considered the best action (e.g. that with the highest evaluation value or the most visited one).

The tree policy is obtained by using a *Upper Confidence Bounds* (UCB) derived function called *Upper Confidence Bound for Trees* (UCT) (Equation 1) to balance the exploration-exploitation and obtain the maximum reward. This policy consists of, for every node visited, a child node j is chosen to maximize UCT function

$$UCT = \overline{X}_j + 2 * C_p \sqrt{\frac{2 \ln n}{n_j}} \tag{1}$$

where $\overline{X}_j$ is the average reward from arm j, n is the number of times the current node has been visited, n_j the number of times child j has been visited and $C_p > 0$ is a constant.

C. Rolling Horizon Evolutionary Algorithm (RHEA)

Rolling Horizon Evolutionary Algorithm is an algorithm first proposed by Perez et al. (18) where a set of actions are evolved to play real-time single player games. As shown in Figure 2, in GVGAI each individual is represented by the horizontal lines containing each a sequence of actions that are performed from the current state E_i using FM. After the simulation, the final simulated E_s state evaluation value becomes the fitness of that individual. Once all individuals are evaluated, the algorithm evolves to a new population, this process is repeated until the time budget is reached. When the time budget is reached, the individual with the best evaluation value first action is applied to the game and the controller starts evolving a new population in the next game tick.

Though not many studies where made regarding RHEA, recent results obtained from modified versions (19)(20) suggests it can achieve competitive results to MCST algorithm.

D. Baseline Algorithms

The algorithms described in this section are used in the experiments to provide a baseline towards the efficiency of our proposed heuristics. These algorithms were chosen as they propose different evaluation methods.

1) Win Score: The first approach consists of the same testing algorithm using the simple state heuristic provided in by the framework. It prioritizes the two main objectives considered in the competition. As shown in Equation 2, this is done by returning a huge positive or negative score for winning or losing states respectively, and returning the current score P when none of these states is reached.

$$E(S) = \begin{cases} \infty & \text{if Win State} \\ -\infty & \text{if Lose State} \\ P & \text{Otherwise} \end{cases} \tag{2}$$

This heuristic is also used as a baseline for our proposed methods.

2) Influence Map: This is the algorithm proposed by Park, Hyunsoo and Kim, Kyung-Joong (13)[1] the proposed approach consists of determining the goodness of each sprite to create an influence map based on that information, and use the value on where the avatar is positioned to bias MCTS UCT equation.

[1] dropbox.com/s/ex0iriqwqfg40hx/GVGAI.zip?dl=0

3) MaastCTS2: This algorithm created by Soemers, Dennis et al. (12)[2] and was the champion of 2016 single player and runner-up on the Two-player competition track. It consists of an MCTS-based algorithm with several enhancements. The evaluation technique used was named *Knowledge-Based Evaluations*, and consists of setting a weight for each sprite and a dynamic update to its values through every game tick, recording the collision outcome with sprites and slightly increasing all weights where no information is obtained. It uses A* algorithm (21) to calculate the distance between them and the closest of each sprite and penalizes or increases the evaluation score for being near bad or good sprites respectively.

Other enhancements are also implemented, among them: *Tree Reuse*, considers the entire subtree rooted in the node corresponding to the action taken in the game. *Progressive History and N-Gram Selection Technique*, introduce a bias in the respective steps towards playing actions, or sequences of actions, that performed well in earlier simulations. *Breadth-First Tree Initialization*, generates the direct successors of the root node before starting the search. *Safety Prepruning*, that count the number of immediate game losses and only keep the actions leading to nodes with the minimum observed number of losses. *Loss Avoidance* where the algorithm ignores losses by immediately searching for a better alternative whenever a loss is encountered the first time a node is visited. *Novelty-Based Pruning* uses Iterated Width (IW) (22) algorithm to prune redundant lines of play during the selection step of MCTS. *Deterministic Game Detection*, detects the type of game it is playing and treats deterministic and stochastic games differently.

4) TeamTopbug_NI: This refers in the result section to the RHEA variation proposed by Perez et al. (11)[3], that achieved the best results when compared to the tree-based techniques with the same changes proposed in that paper and was the 4^{th} best technique of the GVGAI 2014 Competition.

The evaluation technique proposed is based on the Win Score heuristic with some penalties applied. The penalties are the use of opposite actions such as moving left after moving right and using moving actions with no change in the avatar position, e.g. moving against a wall. Also, a repelling pheromone trail is secreted by the avatar and the amount of pheromone in the position being evaluated is subtracted from that state reward. This stimulates the controller to explore new regions of the game level. These pheromones are updated every game tick with the avatar location according to a diffusion equation and a decay factor is applied.

IV. Proposed Game Heuristics

The baseline of the proposed approach is the heuristic provided by the competition framework (Section III-D1), as it prioritizes the main game objectives: lose avoidance, winning and score. It is hard to observe modifications in $E(S)$ in some GVGAI games, given the restricted computational budget. Thus, we propose strategies for differentiating states with the same score. In this way, Equation 2 is replaced by

$$E(S) = \begin{cases} \infty, & \text{if win state} \\ -\infty, & \text{if lose state} \\ P - \sum_{c=1}^{C} \varepsilon_c \times N_c, & \text{otherwise} \end{cases} \quad (3)$$

where N_c represents the penalty value when the characteristic c is observed, ε_c is the penalty coefficient and indicates the relevance of the c-th characteristic, and C is the number of characteristics. Similarly to the baseline algorithm, a very huge positive (or negative) value is returned in case of a win (or lose) state. Otherwise, $E(S)$ is the score P penalized using a static penalty method. One can notice that larger values of $E(S)$ are preferred. The proposed N_c values are defined in the following sections.

A. Avatar Status

In games, the characteristics of the avatar are good indicators of its probability of winning the game. An avatar status penalty (ST) value is created and it is obtained combining (i) the avatar's current health points (HP) and (ii) the number of resources gathered (RG). As HP and RG vary during the game, we decided to normalize them between 0 and 1.

The difference between the current HP (Cur_{HP}) and its maximum value (Max_{HP}) in the evaluated state is normalized as

$$N_{HP} = \frac{Max_{HP} - Cur_{HP}}{Max_{HP}} \quad (4)$$

As there may be more than one resource in the game, a summation is necessary. For each resource i, its current quantity (Cur_{RG_i}) normalized by the maximum amount gathered until that moment in the game (Max_{RG_i}) is considered to calculate N_{RG}. Thus, $-N_{RG}$ is the mean of the normalized values of the resources and is calculated as

$$N_{RG} = -\frac{\sum \frac{Cur_{RG_i}}{Max_{RG_i}}}{|RG|}, \quad (5)$$

where $|RG|$ the number of resources in the game. Due to the improvement in the controller performance when gathering more resources, N_{RG} is a negative value.

Given the N_{HP} and N_{RG} values calculated, respectively, in Equations 4 and 5, N_{ST} can be calculated as

$$N_{ST} = N_{HP} + N_{RG}. \quad (6)$$

N_{HP} and N_{RG} values are only considered in games with hit points measure or resources to be gathered. As $N_{HP} \in [0, 1]$ and $N_{RG} \in [-1, 0]$, N_{ST} assumes values between -1 and 1.

[2] github.com/DennisSoemers/MaastCTS2

[3] github.com/xaedes/open-loop-search-for-general-video-game-playing

B. Spacial Exploration Maps

A spacial influence map is a heuristic developed in order to stimulate the spatial exploration. This is similar to the pheromones of $TeamTopbug\_NI$ (Section II). However, the matrix here is updated with the position of the avatar only (no diffusion is used), the penalization value is updated considering also the area explored through simulation, higher penalization values (> 1) are used, and there is no decay over time. An example of the techniques to stimulate exploration mentioned here are given in Figure 3. With this modification, we expect (i) to reduce the processing time spent by removing the matrix diffusion and decay update, (ii) to enhance the exploration by penalizing the area explored during the simulations and (iii) to reduce the chance of penalizing a potentially good location as only the places where the avatar explored are penalized (the neighbour is not penalized).

In the proposed approach, a two-dimensional matrix is created with the same size of the maze, i.e., all positions of the game. Two approaches are tested here: Evaluation Map (EM), that keeps score of how many times an evaluation was taken in that position; and Position Map (PM), that records how many times during the simulations the avatar was in that position. The value of N_{EM} and N_{PM} used in Equation 3 (corresponding to a given N_c) is the positive integer value of the matrix in the same position being evaluated.

C. Knowledge-Based Evaluations

As shown in Section III-D, some studies were already made using information acquired during simulations to improve the state evaluations in GVGAI. The weighted sum of the distances to each sprite, as

$$N_{KB} = \sum w_i \times d(i), \tag{7}$$

where w_i is the weight of the i-th sprite and $d(i)$ is the distance to the i-th sprite.

In computational experiments, the sprites receive an initial weight $w_+ = 1$, which corresponds to a curiosity value, as the strategy requires the agent exploring unknown sprites to acquire information about the game. This value is updated during the game with $w_{null} = 0$ (when the collision does not provide a change in score) $w_- = -1$ (when the score decreases), and $w_{loss} = -10$ or $w_{win} = 10$ (when a lose or win state is observed, respectively).

The distance can be calculated using several measures. Also, walls, portals and traps can affect the real distance to the sprites. Here we propose the use of the Euclidean and Manhattan distances. In the second case, the walls present in the field are considered: the distance is calculated and stored for every pair of positions at the beginning of the run, the pairs that do not present free paths receive a maximum value established as $maze\_width+maze\_height+1$, where $maze\_width$ and $maze\_height$ are the width and height of the field, respectively.

In addition, two variants of this approach are tested. In the first, only the closest sprite of each type is considered in the distance calculation. In the latter case, the distance is calculated using all sprites of the same type.

TABLE I. GAMES USED IN THE EXPERIMENT, WITH ITS ID IN THE FRAMEWORK AND ITS CLASSIFICATION AS DETERMINISTIC (D) OR STOCHASTIC (S)

Id	Name	Type		Id	Name	Type
0	Aliens	S		4	Bait	D
13	Butterflies	S		15	Camel Race	D
18	Chase	D		20	Chopper	S
25	Crossfire	S		29	Dig Dug	S
36	Escape	D		46	Hungry Birds	D
49	Infection	S		50	Intersection	S
58	Lemmings	D		60	Missile Command	D
61	Modality	D		66	Plaque Attack	D
75	Roguelike	S		77	Sea Quest	S
84	Survive Zombies	S		91	Wait for Breakfast	D

V. COMPUTATIONAL EXPERIMENTS

The results of computational experiments are presented in this section. Due to a large number of games present in the framework a subset was selected for the experiments. This subset is presented in Section V-A along with the parameters and tests used for evaluating the controllers. Also, the source-code of the proposal is available[4].

A. Setup

A subset of the available games is used to test the heuristic variations. It contains the 20 games presented in Table I, that are equally divided in stochastic and deterministic games. These games are used as a testing set in many (10)(19)(20) studies. Each controller plays 20 times in each of the 5 different levels available in the framework resulting in 2000 independent runs for each test.

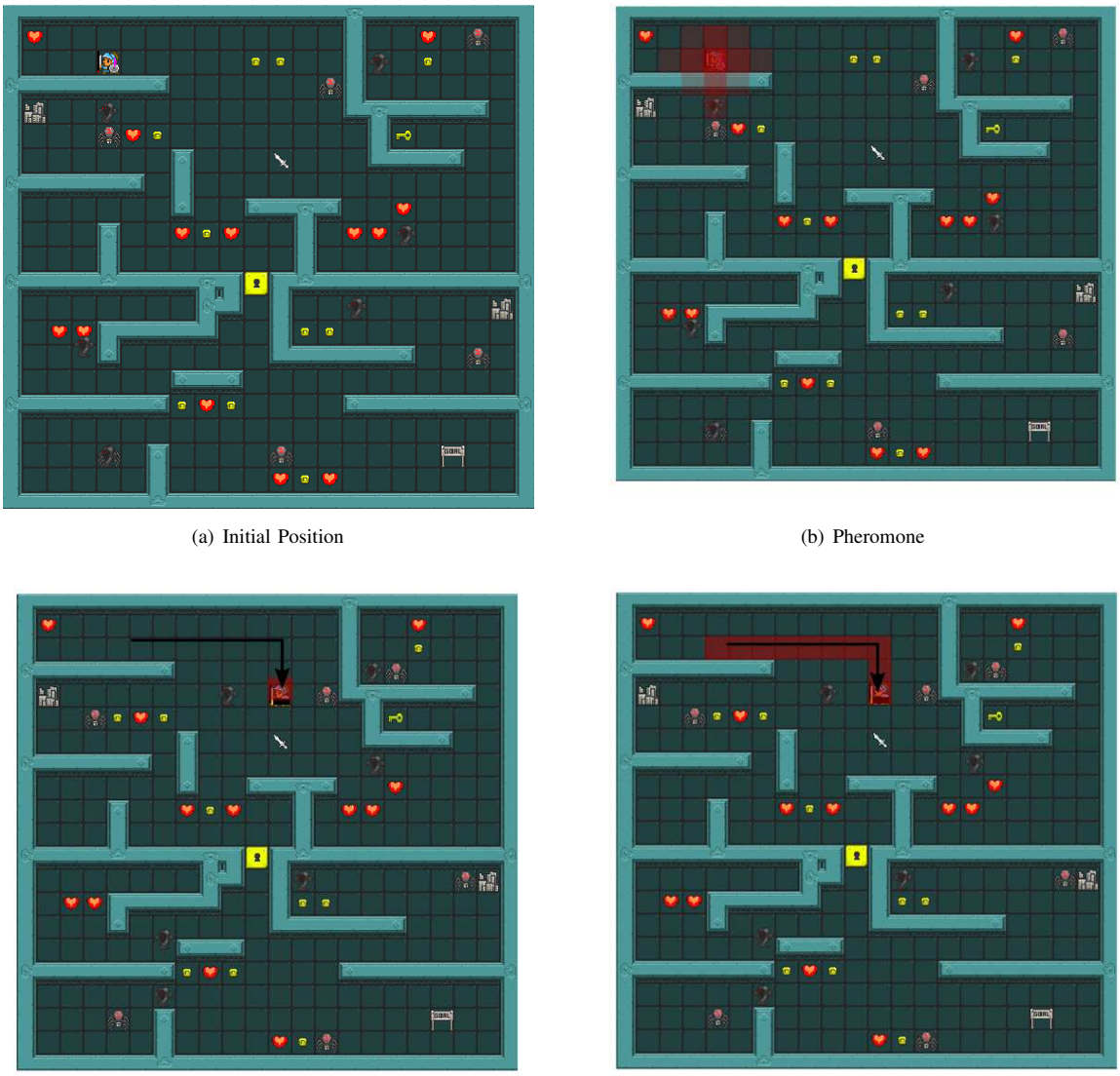
The chosen RHEA parameters for the experiments are both population size and simulation depth equal 10.

To analyze the performance of the controllers three metrics are used here: (i) The official competition metric, Formula-1 Score presented in Section III-A, (ii) the total number of wins in all games and (iii) a statistical analysis using Kruskal-Wallis H test followed by a Mann-Whitney non-parametric U test (p-value < 0.05) applied to the scores obtained in each game individually.

B. Parameter Setting

To define the ϵ parameters for each algorithm proposed the tests presented in Figure 4 were performed, this value determines the feature priority relative to the score in the state evaluations. As the executions are very time costly, in our experiments the values subset $\{0.001, 0.01, 0.1, 1\}$ is

[4]https://github.com/BSant/GVGAI_evaluation.



(a) Initial Position

(b) Pheromone

(c) Evaluation Map

(d) Simulation Map

Figure 3. Examples of exploring stimulation in the game *Roguelike* starting at (a). TeamTopbug_NI pheromones technique (b) penalizes the space where the avatar is and its surroundings. In (c) and (d) is shown a simulation done with the area to be penalized in EM and PM techniques, respectively.

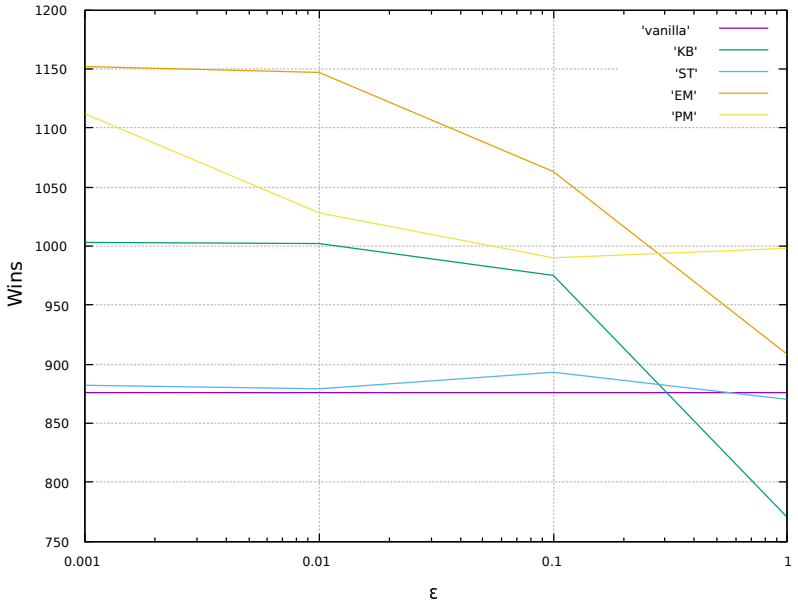
used for all proposed techniques being using the Euclidean distance and only the nearest sprite of each type for the KB evaluations tests.

One can notice that in both graphs that the quality of the algorithm is dependent on the evaluation of game states. One can notice that in the heuristics tests where the penalty value is not normalized, with exception of PM with MCTS, prioritizing the game score over the other metrics is the way to go. Also, ST presented the most stable performance through the different parameters, with a small gain over its vanilla forms.

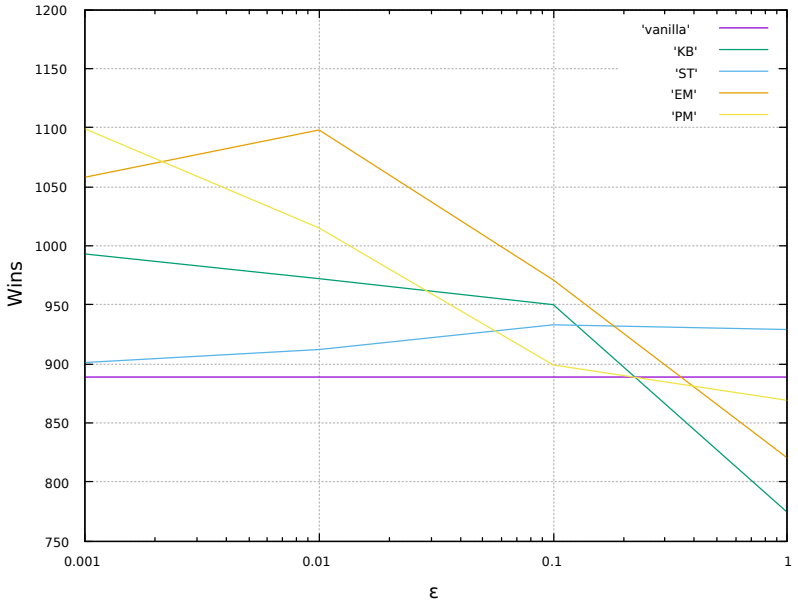
C. Techniques Comparison

In Table II it is shown the KB results for the permutation of each variation proposed. The parameter $\epsilon = 0.001$ for all variants.

All variations performance were really alike in this tests. Using Manhattan distance increased the results while the number of sprites did not show much difference in MCTS algorithm. In RHEA, calculating the value for all sprites presented an increase in wins and lowers $F1\_Score$, while using euclidean calculating distance metric got the best out of Manhattan with only the closest sprite of each type and



(a) MCTS parameter test single wins



(b) RHEA parameter testing single wins

Figure 4. Number of single wins in all games in all parameter tests performed in relation to the vanilla algorithm.

TABLE II. RESULTS ON KNOWLEDGE-BASED VARIATIONS.

Algorithm		MCTS		RHEA	
Sprites	Path	Wins(%)	F1_Score	Wins(%)	F1_Score
Closest	Euclidean	50.15	364	49.65	**374**
Closest	Manhattan	**51.95**	343	49.45	356
All	Euclidean	50.10	320	50.10	334
All	Manhattan	51.85	**373**	**50.80**	336

loses when calculating with all sprites.

Due to this dubious results, no definitive conclusion of which configuration performed better. But as the configuration with all sprites and Manhattan distance low gap (0.1) form the configuration with most wins and best $F1\_Score$ in MCTS, best win rate in RHEA algorithm, this variation will be considered the best for the upcoming tests.

The results presented in Table III considers the parameter ϵ that achieve better single wins: 0.01 for RHEA+EM and 0.001 for the other variants. $TeamTopbug\_NI$ controller is also compared here due to its similarity to out proposed

penalty maps techniques.

TABLE III. SINGLE WINS AND $F1\_Score$ ON TECHNIQUES CITED IN THIS ARTICLE.

Algorithms	MCTS		RHEA	
	Wins(%)	F1_Score	wins(%)	F1_Score
PM	55.60	356	**54.95**	**417**
EM	**57.60**	**421**	52.90	374
TeamTopbug_NI	54.70	383	54.70	369

From the table, it is possible to see that our proposed heuristics overall results outperformed the $TeamTopbug\_NI$ map exploring stimulation techniques.

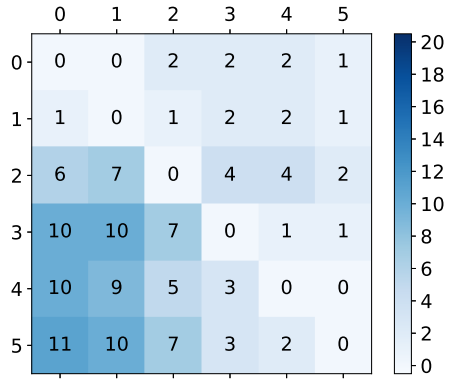
Our evaluation techniques presented different behaviours with each algorithm. As shown in the table, when combined with MCTS algorithm EM presented the best performance while PM synergizes better with RHEA. Though there are many differences in both algorithms in relation to the number of evaluations, like the use of it (individual fitness or in UCT equation) we believe that PM gives more information on RHEA individuals sequence of action to the algorithm that is provided by MCTS tree structure.

A game score comparison among our developed evaluation heuristics is shown in Figure 5 heatmap, where the number in each cell represents the number of games the algorithm represented by the number in the row is significantly better than the one in the column. The algorithm combining all techniques (ALL) in both cases used ST with $\epsilon_{ST} = 0.1$, EM and KB (ALL sprites and Manhattan distance) with $\epsilon_{EM} = \epsilon_{KB} = 0.001$. The use of EM is due to when together with the other techniques, empirical tests presented a better performance, even if when applied alone, PM shows better results combined RHEA.

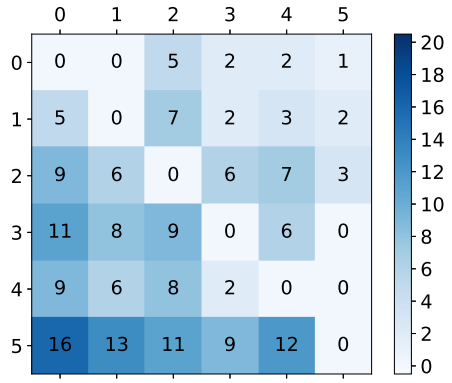
When with MCTS, the expected ST performance is observed in 5.(a) first and second rows, a little enhance from the $Vanilla$ form and losing to the other techniques. Surprisingly, when combined with RHEA it is statistically better than KB in more games, even with a lower win rate.

Considering the individual techniques, in both algorithms, the exploring stimulation presented the most improvement when compared to its respective $Vanilla$ variations. This heatmaps also support the results presented in the Table III having the heuristic using EM being statistically better more times than PM for MCTS (EM 3 x 1 PM) and the contrary for RHEA (EM 2 x 6 PM).

As expected, ALL variation presents the best performance when combined with both algorithms. With MCTS, this performance boost can be seen in the direct comparison between the techniques shown in the lasts row and column. Besides that direct comparison enhancement, with RHEA it achieves the best values out of each column showing the synergy between techniques when used to evaluate the single individuals.



(a) MCTS



(b) RHEA

Figure 5. Heatmap representing the number of games the row algorithm achieved significant better scores than column. ϵ value and configuration that achieved the most wins used for each of the following configuration: 0 – $Vanilla$; 1 – ST; 2 – KB; 3 – PM; 4 – EM; 5 – ALL.

D. Literature Comparison

Here, the proposed evaluation heuristics with best performances are compared with approaches from the literature. Table IV shows the score values obtained through the F1_Score system and the win percentages rates of the considered algorithms. Figure 6 presents the statistical results.

TABLE IV. WIN RATE AND F1_SCORE CALCULATED WITH VANILLA, LITERATURE'S AND OUR BEST ALGORITHMS.

Algorithm	wins(%)	F1_Score
RHEA	44.45	203
MCTS	43.80	230
RHEA(KB+EM+ST)	62.15	310
MCTS(KB+EM+ST)	58.30	337
$InfluenceMap$	34.15	165
$TeamTopbug\_NI$	54.70	240
$MaastCTS2$	**72.35**	**395**

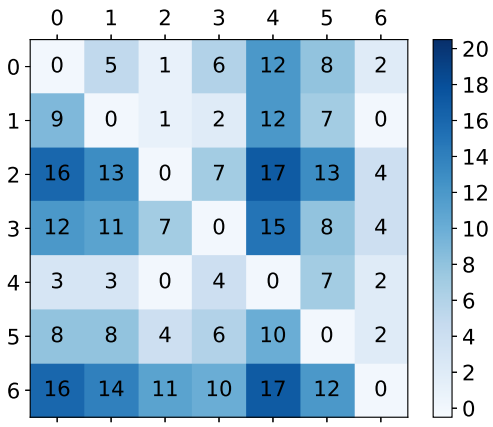


Figure 6. Heatmap representing the number of games the row algorithm achieved significant better scores than column one. 0 – Vanilla RHEA; 1 – $VanillaMCTS$; 2 – RHEA(KB+EM+ST); 3 – MCTS(KB+EM+ST); 4 – $InfluenceMap$; 5 – $TeamTopbug\_NI$; 6 – $MaastCTS2$

One can notice in Table IV that $InfluenceMap$ obtained a low win rate. This is due to its high computational cost on building the evaluation matrices every step, leaving a small time for the simulations. On the other hand, the approach with $MaastCTS2$, that contains many enhancements to MCTS along with the evaluation enhancements, obtained the best results in all metrics used. Therefore, analyzing $InfluenceMap$ and $MaastCTS2$ results it is possible to conclude that though enhance evaluation heuristic have a major impact on performance, having a good simulations strategy and distribute the time budget is essential.

According to Figure 6, the proposed variants of RHEA and MCTS (third and fourth rows) obtained results better than those found by its vanillas forms, $InfluenceMap$, and $TeamTopbug\_NI$. Also, when compared to $MaastCTS2$, one of the best GVGAI algorithms from the literature, the proposed approaches achieved statistic better results in 4 games. The same can be observed with respect to the win rate and F1_Score showed in Table IV. Though that in the direct comparison with $MaastCTS2$ our proposals showed lower results, when looking to the rows, the similarity on the results when comparing with other algorithms is remarkable, especially RHEA that achieved a slightly better result comparing to when comparing to $TeamTopbug\_NI$ (13 x 12). Achieving this results without any modifications on the simulation decisions mechanics shows the importance of state evaluation in general game playing.

Comparing RHEA and MCTS approaches one can notice notice that MCTS overcomes RHEA when using the competition F1_Score metric and the evolutionary algorithm achieves better win rates. Additionally, the gap between the the win rates increase when applying the proposed modifications. Also, the statistical tests show that the number of games where each algorithm achieves a significantly better game score when the vanilla versions are adopted is larger using MCTS (RHEA 5 x 9 MCTS). On the other hand, a tie is observed when the proposed approaches are considered (RHEA 7 x 7 MCTS). Thus, one can argue that a good state evaluation heuristic is more beneficial to RHEA than MCTS. This result also suggests that RHEA can achieve competitive solutions when compared to those found by MCTS based techniques.

VI. Conclusions and Future work

This paper presented a study on state evaluation and three techniques are proposed using (i) penalties to enhance the map exploration, (ii) different approaches on the collision information gathered by the avatar during the gameplay, and (iii) features of the avatar. A combined version of these ideas is also considered here. The proposed approaches are combined with two popular GVGAI algorithms, namely, MCTS and RHEA.

Preliminary tests were performed in order to determine the importance of each technique when compared to the main game objectives (winning and gathering points). It is noticeable that among our techniques the one that encourages the map exploration performed better, followed by the use of knowledge-based information, with both methods presenting better results when applied with a lower priority regarding the direct game objectives. As the avatar-related characteristics values are normalized in our approach, the technique presented a constant small improvement, independent of its prioritization.

In the computational experiments performed, the proposals improved the performance of the baseline methods. This result indicates that the integration of state evaluation approaches to MCTS and RHEA is a good venue to develop new general controllers.

Compared to MCTS, RHEA presented more sensitivity to change in the proposed evaluation techniques given the direct impact of the state evaluation on the fitness of the individuals and its importance to the evolutionary process. On the other hand, the proposed evaluation approaches did not overcome *MaastCTS2*, an improved version of MCTS. This occurred due to the large number of other components used by *MaastCTS2*. Thus, studying the combination of the techniques proposed here with enhanced versions of the algorithms from the literature is a good direction to create better general player controllers.

The development of state evaluating algorithms is encouraged by the results found here. One idea is to adapt the values ϵ of the penalty method. Also, the investigation of methods which can quickly select and use more knowledge-based information is an interesting research venue.

ACKNOWLEDGMENT

The authors thank the financial support provided by UFJF, PPGCC, Capes, CNPq and FAPEMIG.

The authors would like to thank Denis Soemers, Martin Hünermund and Hyunsoo Park, for their availability in providing the source code for our experiments.

REFERENCES

[1] 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.

[2] M. Campbell, A. J. Hoane Jr, and F.-h. Hsu, "Deep blue," *Artificial intelligence*, vol. 134, no. 1-2, pp. 57–83, 2002.

[3] M. Genesereth, N. Love, and B. Pell, "General game playing: Overview of the aaai competition," *AI magazine*, vol. 26, no. 2, p. 62, 2005.

[4] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, 2013.

[5] J. Levine, C. B. Congdon, M. Ebner, G. Kendall, S. M. Lucas, R. Miikkulainen, T. Schaul, and T. Thompson, "General video game playing," in *Artificial and Computational Intelligence in Games*, 2013, pp. 77–83.

[6] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of research and development*, vol. 3, no. 3, pp. 210–229, 1959.

[7] J. Clune, "Heuristic evaluation functions for general game playing," in *AAAI*, vol. 7, 2007, pp. 1134–1139.

[8] K. Walędzik and J. Mańdziuk, "An automatically generated evaluation function in general game playing," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6, no. 3, pp. 258–270, 2014.

[9] D. Michulke and M. Thielscher, "Neural networks for state evaluation in general game playing," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2009, pp. 95–110.

[10] C. Guerrero-Romero, A. Louis, and D. Perez-Liebana, "Beyond playing to win: Diversifying heuristics for gvgai," in *Computational Intelligence and Games (CIG), 2017 IEEE Conference on*. IEEE, 2017, pp. 118–125.

[11] D. Perez Liebana, J. Dieskau, M. Hunermund, S. Mostaghim, and S. Lucas, "Open loop search for general video game playing," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2015, pp. 337–344.

[12] D. J. Soemers, C. F. Sironi, T. Schuster, and M. H. Winands, "Enhancements for real-time monte-carlo tree search in general video game playing," in *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*. IEEE, 2016, pp. 1–8.

[13] H. Park and K.-J. Kim, "Mcts with influence map for general video game playing," in *Computational Intelligence and Games (CIG), 2015 IEEE Conference on*. IEEE, 2015, pp. 534–535.

[14] 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.

[15] H. Horn, V. Volz, D. Pérez-Liébana, and M. Preuss, "Mcts/ea hybrid gvgai players and game difficulty estimation," in *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*. IEEE, 2016, pp. 1–8.

[16] T. S. Nielsen, G. A. Barros, J. Togelius, and M. J. Nelson, "Towards generating arcade game rules with vgdl," in *Computational Intelligence and Games (CIG), 2015 IEEE Conference on*. IEEE, 2015, pp. 185–192.

[17] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[18] D. Perez, S. Samothrakis, S. Lucas, and P. Rohlfshagen, "Rolling horizon evolution versus tree search for navigation in single-player real-time games," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*. ACM, 2013, pp. 351–358.

[19] R. D. Gaina, S. M. Lucas, and D. Perez-Liebana, "Population seeding techniques for rolling horizon evolution in general video game playing," in *Evolutionary Computation (CEC), 2017 IEEE Congress on*. IEEE, 2017, pp. 1956–1963.

[20] ——, "Rolling horizon evolution enhancements in general video game playing," in *Computational Intelligence and Games (CIG), 2017 IEEE Conference on*. IEEE, 2017, pp. 88–95.

[21] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[22] T. Geffner and H. Geffner, "Width-based planning for general video-game playing," *Proc. AIIDE*, pp. 23–29, 2015.