An Improved Rolling Horizon Evolution Algorithm with Shift Buffer for General 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

Eduardo Hauck
Departament of Computer Science
University of Tsukuba
Tsukuba, Ibaraki, Japan
Email: eduardohauck@gmail.com

***Abstract*—General Game Playing (GGP) is the design of artificial intelligence programs to play more than one game. Here, one of the most famous GGP frameworks, The General Video Game AI Competition (GVGAI) Framework, is used in order to design controllers for Atari 2600 inspired games. Recent advancements in the literature of GVGAI showed that the Rolling Horizon Evolution Algorithm (RHEA) is competitive when compared to other methods, encouraging the use and the research by improvements for this method. The use of a 1-Step-Look-ahead approach and a Redundant Action Avoidance policy during the creation of new individuals are proposed in this paper. The 1-step-look-ahead technique improves the action selection after the shift of the individual in RHEA with the shift buffer enhancement (RHEA-SB), and the redundant action avoidance policy decreases the chance of spatial redundant actions within the individual. Also, a parameter analysis of RHEA-SB is performed here, where different values of population size, depth of simulations, and number of individuals that remains in the population are evaluated. Results show that using 1-Step-Look-ahead and a redundant action avoidance policy improves the quality of the solutions found when compared to the original algorithm.**

***Keywords*-General Video Game Playing, Rolling Horizon Evolution Algorithm, Monte Carlo Tree Search**

I. Introduction

Since the beginning of the artificial intelligence (AI), games provide a solid environment for evaluating new techniques. The games are formed by a well-defined set of rules and objectives, which require the machine to explore the environment defined by these rules and to find a policy to reach the objectives in the most efficient possible way, mimicking a human intelligent behaviour.

With the developments of AI, many game playing controllers have been developed, such as GO (1) and chess (2). In some cases, the computational methods obtain a performance comparable to the best human players. Despite the good results found by these techniques, commonly they are designed to a specific game, having no capability of generalizing their skills to perform well in other similar games. This limitation brings the challenge of developing and studying controllers capable of such generalization. To facilitate and stimulate the research regarding these controllers, new competitions were created, where the programs are tested in many different game environments without previous knowledge of the objectives and challenges presented in the games.

General Game Playing (GGP) Competition (3) and the General Video Game Playing (GVGAI) (4) are the most relevant general controllers competitions in this area. While the first competition uses board games for evaluating the contenders and the second one considers video games to analyze the performance of the techniques. Another game environment created with the objective of testing general AI techniques is the Artificial Learning Environment (ALE) (5). Different from GVGAI, where controllers receive a more structured information on the game environment (as described in Section III-A), screen captures are presented to the controller in ALE. Here, the GVGAI competition framework and rules are used.

The Rolling Horizon Algorithm (RHEA) (6), in its vanilla version, did not achieve a good performance when compared to other algorithms. However, recent enhancements (7) (8) show that RHEA can be competitive with other methods from the literature, encouraging further studies. We consider here one of the most effective enhancement found so far, the Rolling Horizon Algorithm with Shift Buffer (RHEA-SB). This variant was chosen due to the results presented by Gaina (8), where it outperforms the vanilla RHEA in all configurations and is present in all the configurations with the best results.

Initially, a study of the behaviour of RHEA-SB was done variating different parameters, such as the population size, depth of simulations in each individual and number of individuals to be shifted after an action is chosen. Also, we propose two enhancements in the RHEA algorithm with shift buffers (8). The first one uses the 1-Step-Look-ahead algorithm in order to improve the choice of actions in the shifted individuals of the population. The second enhancement applies a redundancy avoidance policy to discourage choosing actions that cancel each others effects, allowing a better exploration of the states presented in the games.

Results obtained indicate that both modifications im-

proved the performance of the original algorithm, having more impact in the configurations with more individuals and larger simulation depths. The 1-Step-Look-ahead enhancement improved the winning rates when compared to the original algorithm and better performance than the other variants in some configurations. Redundancy avoidance policy presented a positive impact in the algorithm, mainly when using configurations with deeper simulations and larger populations. Also, using both 1-Step-Look-ahead and the redundancy avoidance policy improved the results in all metrics tested.

This paper is structured in 6 sections. Section II presents the more relevant works from the literature regarding different variants of RHEA and other algorithms with similar features as those ones proposed here. Section III details the framework and background algorithms. Section IV discusses the modifications proposed. Sections V and VI present the computational experiments and the results obtained, respectively. Finally, Section VII concludes the paper and presents some future works.

II. Related Work

In this section, studies relevant to this research are considered. A study of evolutionary computing controllers applied to single games is shown. We then summarize RHEA and its enhancements previously proposed in the GVGAI literature. Also, some techniques similar to those proposed here and applied to GGP are discussed.

A rolling evolution inspired algorithm, the online evolution, is proposed by Justesen et al. (9). It is designed to play *Hero Academy* (Robot Entertainment 2012), a game where the technique controls a number of units and evolves a set of 5 different actions of every turn. The results presented showed that the online evolution was able to perform better than Monte Carlo Tree Search (MCTS) in 98% of the times. MCTS has been dominant in game playing since it's success in GO (1) and overcame other greedy techniques. Wang et al. (10) modified the idea to select script portfolio to play *StarCraft* (Blizzard Entertainment 1998), outperforming Upper Confidence Bound for Trees (UCT) based algorithms (in Section 2 MCTS and UCT are described with more detail).

Perez et al. (11) were the first to propose the Rolling Horizon Evolutionary Algorithm, which uses simple macro actions to solve the Physical Travelling Salesman Problem (PTSP), a real time game where an agent must find a path through a maze while maximizing the number of way points reached. The algorithm presented competitive results with Monte Carlo Tree Search algorithm. A rolling horizon algorithm was first submitted in the 2014 GVGAI competition as part of the sample controllers. Perez et al. (12) provides a brief description and the ranking of the used algorithm (12th). The 1-Step-Look-ahead algorithm (ranked as the 16th best technique) is also into that sample of algorithms and it is used as a possible improvement proposed o RHEA.

Recent studies presented by Gaina et al. analyze different configurations of the vanilla RHEA (6). Gaina et al. also proposed several enhancements to the vanilla version. For instance, they proposed the use of other methods, such as MCTS, for initial population seeding of RHEA (7). That proposal showed a positive impact on the overall results. A bandit-based mutation, Statistical-tree, Rollouts and a Shift buffer enhancements were also explored (8), having the last one the best improvement in the efficiency of the algorithm. As the best version so far, RHEA with Shift buffer was chosen as the baseline of this study.

Redundancy avoidance policy has been previously tackled as an enhancement for the MCTS algorithm in many different forms. The first redundant action avoidance policy was proposed by Soemers, Dennis (13) and used an approach based on Iterated Width algorithm (IW) (14) to prune expanded redundant states in MCTS. Perez, et al. (15) proposed a penalty method to the reward function after executing opposing moves. A probability penalty was proposed by (16) in the selection phase of the actions in MCTS. All modifications listed above presented performance improvement when compared to the original algorithm.

III. Background

A. General Video Game AI Framework (GVGAI)

The GVGAI framework comprises a set of different 2D games, containing more than 100 single and 50 multiplayer games. These games are based on Atari 2600 console, with no more than 6 action choices, namely: left, right, up, down, use (press button) and nil (do nothing). GVGAI games can be divided into different categories (e.g. puzzle, shooting, and survival) providing many challenges to the controllers.

In the competition rules, the controller has no information about which game it is playing and in order to take decisions, it is provided a minimum information about each state of the game, like the position of the avatar and game elements (e.g. walls, non player characters (NPCs) and resources), available actions and score. The framework also provides a Forward Model (FM), which allows for controllers to perform simulations considering different actions. As the games in the competition may be stochastic, one FM call gives one possible resulting state for an action when the game is not deterministic. The controller must also inform an action every game tick (40ms in the competition). This is a small time for the controller to compute and to return an action and, hence, a good strategy with a low computational cost is important.

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. The score is defined using a Formula-1 scoring system (F1-Score). Once all controllers have played one game, a rank is made by sorting first by number of victories, followed by the average score and average time they spent to finish a given game. According to

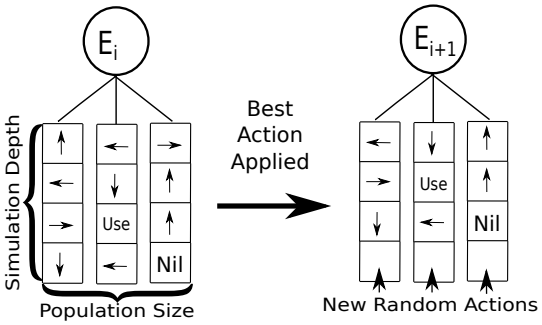


Figure 1. Shift buffer scheme.

their 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 winner is determined by summing the points obtained in all the games.

B. Rolling Horizon Evolutionary Algorithm with Shift Buffer (RHEA-SB)

The Rolling Horizon Evolutionary Algorithm is a bio-inspired populational technique proposed by Perez et al. (11) where an agent evolves a set of actions in an imaginary model, performs the first action of its plan, and repeats these steps until the game is over. When applied to GVGAI games, one individual is composed by a sequence of available actions in the game and its fitness is given by evaluating the last simulated state after using the FM to simulate its sequence. At the beginning of each game tick, a population is created and evolved with a time limitation. In the sequence, the first action of the fittest individual is run.

Vanilla RHEA, as presented in GVGAI framework, creates a new population every game tick, losing the information acquired in the last simulations. In order to use that information, Gaina et al. (8) proposed the shift buffer enhancement, where the information in the previous game tick is used in the current game state. As shown in Figure 1, the first action is removed from all individuals of the population, the remaining actions are shifted in order to place each action in its next position, and other randomly generated actions are included as the last ones in each individual. Thus, the population maintains some information from the previous game tick.

C. Monte Carlo Tree Search (MCTS)

The dominant techniques in GVGAI are mainly based on Monte Carlo Tree Search (MCTS) (17). The GVGAI framework provides an MCTS variant where most competition entries are based. This vanilla version is used here as a baseline in the computational experiments.

At each game step, the algorithm begins by creating a root node to its search tree. Then each iteration consists of four

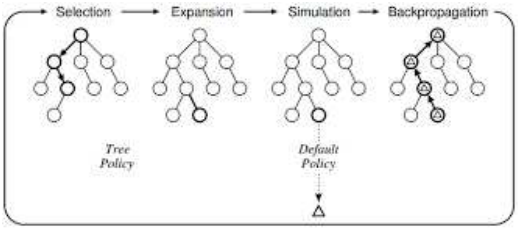


Figure 2. MCTS scheme.

steps (Figure 2): selection, expansion, simulation and back-propagation. In selection phase, an expandable node (with a non-terminal state and with unvisited children) is selected using the tree policy. This node is then expanded by adding a new child (by choosing an action to perform and to lead to a new game state). From the added node, actions are randomly selected using FM to simulate the game using the sequence of actions defined by the tree. 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. These iterations are repeated until a time limit is reached. 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 used in the first step consists of descending in the existing tree using the *Upper Confidence Bound for Trees* (UCT) (Equation 1) as a policy. This policy consists of, for every node visited, a child node j is chosen to maximize UCT function

$$UCT = \overline{X}_j + 2C_p\sqrt{\frac{2\ln(n)}{n_j}} \quad (1)$$

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.

In Section VI-C is presented a comparison between the best configuration of RHEA proposed here and the vanilla version of MCTS available in the GVGAI framework.

IV. Enhancements proposed

The combination of RHEA-SB with 1-Step-Look-ahead and Redundant Action Avoidance is proposed here[1]. 1-Step-Look-ahead is used in order to improve the choice of the action to complete the shifted individual. Thus, the best sequence of actions previously obtained is augmented by including a good (hopefully the best) next action. A Redundant Action Avoidance prevents the agent of spending time simulating action sequences which are unlikely to provide good results.

[1]Source code available at https://github.com/BSant/GVGAI_RHEA

A. 1-Step-Look-ahead (1Step)

The 1-Step-Look-ahead algorithm is a simple method and it consists of using the forward model to test every available action from the current state. In this paper, 1-step-look-ahead is used after the shift buffer phase of the RHEA-SB algorithm to choose an action to complete the shifted individual with the best action in the sequence.

B. Redundant Action Avoidance (RedAvoid)

As the controller has no information on the game, there is no guarantee of witch actions will lead to more promising states. Also, the exploration of states with different characteristics (e.g. different positions for the avatar) leads to a better exploration of the possibilities in the game, increasing the chances of exploring winning states.

Redundant action avoidance techniques were explored before in other algorithms with promising results (13)(15)(16), as it requires low computational resources to identify simple redundant actions. Thus, a strategy to avoid redundant movement actions is also included in the current proposal, where a new action is randomly selected whenever one of the following sequences is found in the sequence of actions of an individual of the population:

- right - left
- left - right
- up - down
- down - up

This verification is performed for every pair of actions of each newly created individual (by the application of shift buffer, or by means of crossover and mutation). Once a redundant action is found, it is replaced by another one randomly generated. This makes those redundant sequences of actions less likely to appear in individuals, but it does not remove them completely as the new randomly generated action may be the same as the previous one. Thus, the redundant actions are avoided but they are not completely removed from the possible solutions. The redundant action may be useful; they can be adopted for dodging an enemy or to acquire some sort of bonus or resource, for example.

V. Experimental Setup

Experiments were run using three main parts in this study. The first part was proposed to analyze the impact of different parameter settings on the efficiency of the proposed algorithm. The parameters considered here are: population (pop), depth of simulations (depth), and number of individuals shifted and copied to the next game tick (shift). The second part involves the evaluation of the proposals. The last part compares the best proposed method with Vanilla MCTS. The experiments were performed according to (8); the same games and restrictions were used.

All the parameter settings and proposed RHEA-SB variants were evaluated using a set of 20 games, and playing 20 times on all 5 levels of each game. Thus, 100 independent runs were performed for each game and algorithm. The budget given to each algorithm was 900 FM calls, so as to eliminate bias from variations in the machine used to run the experiments. The maximum configuration tested was pop10-depth14 due to the fact that if it were larger, by adding rollouts, the limited budget would not allow for even one full population to be evaluated in one game tick.

TABLE I. Games used in the experiments, with its identifier (Id) in the framework and 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

The first part of the experiment was conducted by varying one parameter while keeping the others static. The fixed parameters were chosen based on those from the literature: pop= 5, depth= 10, and shift= 1. The possible parameter values are: pop={1,2,5,10} (Table IV), depth={6,8,10,14} (Table II), and shift={1,2,3,4,5} (Table V). The case proposed in the literature where shift=pop is also tested here, and the results are presented in Table III.

In the second part of the experiment, 4 different core parameter configurations (Population-Length = {1-6, 2-8, 5-10, 10-14}) were used for all algorithms. These configurations are from the literature (8). As it will be shown in Section VI-A, shifting only one individual between game tics is the best policy. All tests with shift buffer were made shifting only the best individual. A Vanilla RHEA provided by the framework was also included in the computational experiments in order to provide a baseline.

VI. Results and Discussion

This section presents the results and discussions on the computational experiments. Two performance metrics are used in the comparative analysis performed here: (i) Formula-1 Score (Section III-A) and (ii) the number of wins. Also, a Kruskal-Wallis H test followed by a Mann-Whitney non-parametric U test is applied in order to identify when the results are statistically different and best (p-value < 0.05).

A. Parameters Testing

Table II shows an increase in win rate and a better F1-Score with deeper simulations in each individual. With this results we can conclude that, for that parameter range, it is possible to take deeper simulations without affecting the evaluation afterwards. This suggests that RHEA-SB provides

TABLE II. DEPTH TESTS RESULTS

Pop	Depth	Shift	Win(%)	F1-Score
5	6	5	40.90	309
5	8	5	43.75	336
5	10	5	43.80	368
5	14	5	**44.40**	**387**

TABLE III. RESULTS WHEN VARYING THE PARAMETER POP.

Pop	Depth	Shift	Win(%)	F1-Score
1	10	1	**47.04**	**413**
2	10	2	42.80	312
5	10	5	43.80	334
10	10	10	43.85	341

the best results when evolving deeper sequences of actions.

It is possible to see in Tables III and III that the variation which obtained best individual win rate was that one with only one individual, where the algorithm uses only the mutation operator. The number of wins decreases for 2 individuals and then starts increasing when more than 2 individuals are used. Thus, we can conclude that the algorithm behaves better while evolving only one individual with mutation as moving operator. The increase on the win rate shown while increasing the population in the tests where $Pop \geq 2$, however, suggests that larger populations with crossover operator can obtain good results, but it would require more population diversity.

RHEA-SB, as shown in Table III, had better performance when only the best individual is shifted and copied to the next game tick. This is expected, as only the first action of the best individual is performed at the end of each game tick. As a result, the current state of the game is similar to that of FM when the action is performed, and the remaining actions in the best individual are suitable for this case.

Table V presents the results obtained when the proposal uses 5 individuals and depth equals to 10. In this table, the number of individuals which are copied and shifted in the next game tick is varied, and one can observe that shifting more than 1 individual has no positive impact in the performance of the algorithm.

In (8), RHEA-SB with the same parameter setting (population size equals to 5 and depth of the simulation equals to 10) achieved 40.05% with respect to win rate, and the best hybrid technique (EA-shift-roll with pop10-depth14) obtained 42.35%. The results presented in Table V vary from 43.8% to 45.75% with respect to win rate, surpassing both

TABLE IV. RESULTS WHEN VARYING THE PARAMETER POP AND SHIFT= 1.

Pop	Depth	Shift	Win(%)	F1-Score
1	10	1	**47.04**	**402**
2	10	1	44.40	302
5	10	1	45.75	320
10	10	1	45.05	376

TABLE V. RESULTS WHEN VARYING THE PARAMETER SHIFT.

Pop	Depth	Shift	Win(%)	F1-Score
5	10	1	**45.75**	**372**
5	10	2	45.10	362
5	10	3	44.70	318
5	10	4	44.50	264
5	10	5	43.80	284

TABLE VI. WIN RATE FOR EACH "POP-DEPTH" PARAMETER VALUES.

	win%			
Variant	1-6	2-8	5-10	10-14
RHEA	44.10	38.95	44.45	46.85
RHEA-SB	43.75	42.45	45.75	47.55
RHEA-SB-1Step	**44.40**	43.65	44.90	49.30
RHEA-SB-RedAvoid	43.60	**44.70**	45.30	**49.95**
RHEA-SB-RedAvoid-1Step	43.85	44.45	**46.90**	49.50

of the previous mentioned approaches.

B. Comparing the Proposed Approaches

Table VI shown the win rate in all configurations tested. Table VII presents F1-Score calculated separately for each parameter setting (each column).

As expected, vanilla RHEA got the lowest F1-Scores in all cases. Also, it is possible to observe that the gap in the results is smaller in the parameter settings with 1 individual, getting even a higher number of wins than some of its enhanced version. This shows that the modifications proposed have a larger impact with larger population sizes and deeper simulations are adopted.

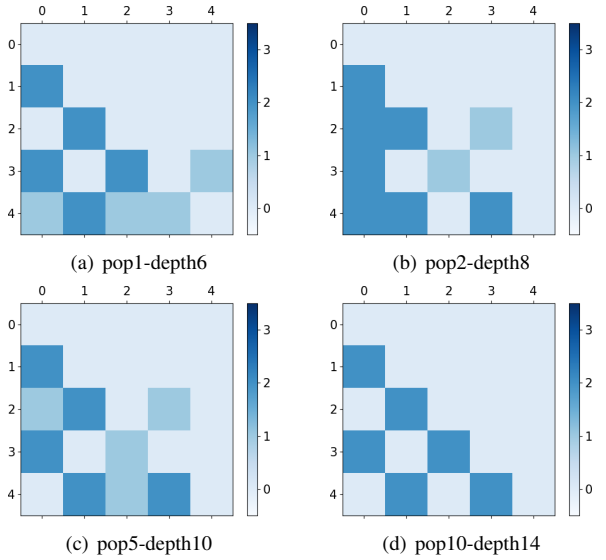
Although the technique with 1-Step-Look-ahead presented similar F1-Score than the original algorithm, it overcomes RHEA-SB in 3 of the 4 tests when the win rate is considered, and even achieves the best overall variations in pop1-depth6 configuration (44.4% wins).

The use of a redundancy avoidance policy produces the best win rates in 3 of the 4 tested cases and the best F1-Scores. Also, the variant RHEA-SB-RedAvoid with pop= 10 and depth= 16 achieved the best win rate, and RHEA-SB-RedAvoid-1Step with pop= 2 and depth= 8 obtained the largest F1-Scores.

The Kruskal-Walis H test was performed in order to determine when results (with respect to the score obtained in each game) obtained by the five techniques for each game are statistically different. The Mann-Whitney U test is used to identify the best variants when the null hypotheses (results are statistically equal) is rejected (p-value < 0.05). These

TABLE VII. F1-SCORE FOR EACH "POP-DEPTH" PARAMETER VALUES.

	F1-Score			
Variant	1-6	2-8	5-10	10-14
RHEA	301	234	270	259
RHEA-SB	331	299	309	304
RHEA-SB-1Step	319	317	295	310
RHEA-SB-RedAvoid	310	365	**373**	**372**
RHEA-SB-RedAvoid-1Step	**339**	**385**	353	355



(a) pop1-depth6 (b) pop2-depth8

(c) pop5-depth10 (d) pop10-depth14

Figure 3. Heat Maps with the number of games in which the variant on the row found results significantly better than those on the column, where 0–RHEA, 1–RHEA-SB, 2–RHEA-SB-1Step, 3–RHEA-SB-RedAvoid, and 4–RHEA-SB-RedAvoid-1Step.

two tests are non-parametric. The heat maps in Figure 3 present the number of games in which the variant on the row found results significantly better than those on the column.

As expected, vanilla RHEA did not achieve the best scores in any case (first row), and its results are worse than other variants in some games (first column). Also, one can notice that RHEA-SB-RedAvoid-1Step is not worse than the other variants in all the cases (fourth column), except to the variant RHEA-SB-RedAvoid when pop= 1 and depth= 6 in only one game.

C. Comparison with MCTS

Table VIII presents the percentage of wins, the mean score and standard deviation obtained. Vanilla MCTS is used and RHEA-SB-RedAvoid pop10-depth14- the RHEA variant with the best number of wins, as shown in Table VI.

RHEA-SB-RedAvoid variant presented a better game win rate 999 (49.95%), against vanilla MCTS 876 (43.8%). However, when analyzing F1-Score, we notice that both algorithms with 50% of the best results in games. One can observe in Table VIII that MCTS achieved the best score in 14 games, against 11 of RHEA. Among the 5 games where the algorithms achieved the same win rate ({0,29,50,58,75}), the winning rate is either a 100% or 0%. Also, in these games, the RHEA algorithm achieved the best game score in 4 out of this 5 games, losing only in game 0 (Aliens). Considering only the games scores, RHEA performed better than MCTS in 11 games. As a result, RHEA algorithm is more likely to get better scores in most of the games, but still presented more difficulties than MTCS to win the games.

TABLE VIII. WINS AND SCORE WITH DEVIATION FOR EACH GAME FOR THE VANILLA MCTS AND RHEA-SB-REDAVOID (POPULATION 10, SIMULATION DEPTH 14 AND ONE INDIVIDUAL SHIFTED)

Game	MCTS		RHEA-SB-RedAvoid	
#	Wins	Score	Wins	Score
0	**1**	**68.73±5.73**	**1**	66.35±1.35
4	**0.12**	2.51±1.49	0.1	**5.66±0.66**
13	**0.98**	29.36±15.36	0.94	**32.6±18.06**
15	0.04	-0.76±0.24	**0.05**	**-0.75±0.25**
18	**0.06**	**3.38±0.62**	0.03	2.7±2.3
20	0.97	15.94±3.94	**0.99**	**16.52±2.48**
25	0.01	**0.03±0.03**	**0.05**	0±0
29	**0**	13.64±13.64	**0**	**16.05±16.95**
36	0	0±0	**0.43**	**0.43±0.43**
46	**0.38**	**38±38**	0.27	27.4±27.4
49	0.95	**16.47±5.47**	**0.99**	15.7±9.7
50	**1**	1±0	**1**	**3.87±2.87**
58	**0**	-3.06±3.06	**0**	**-0.14±0.14**
60	**0.72**	**5.28±2.72**	0.6	4.95±0.05
61	**0.24**	**0.24±0.76**	0.22	0.22±0.22
66	**0.9**	**46.93±23.07**	0.89	43.8±37.2
75	**0**	5.11±4.89	**0**	**6.45±5.55**
77	**0.89**	**2362.02±1677.98**	0.41	1413.82±620.18
84	**0.43**	2.87±2.87	0.41	**3.08±3.08**
91	0.07	0.07±0.07	**0.68**	**0.68±0.32**

VII. CONCLUSION AND FUTURE WORK

In this paper we presented a study of the state of the art in rolling horizon evolutionary algorithms (RHEA) in GVGAI and proposes two new approaches. The first proposed variant uses the 1-Step-Look-ahead algorithm after the shift buffer phase of RHEA-SB in order to improve the individual which is shifted and copied to the next game tick. The second proposal applies a spatial redundant action avoidance policy that replaces actions that would undo the previous movement. Also, the combination of both approaches with RHEA-SB is proposed (labelled as RHEA-SB-RedAvoid-1Step).

The computational experiments were divided in three main parts. In the first experiment, some parameter settings were evaluated and the results show that using less shifted individuals and deeper simulations improve win rates and F1-Scores. Also, the quality of the solutions is decreased when a middle population size is adopted.

The second experiment analyzed the two proposed approaches when combined with RHEA. One can conclude that 1-step-look-ahead had a positive impact in the overall performance of RHEA, achieving better win rates. Also, the use of redundancy avoidance policy presented the best results in most cases and reached the best F1-Scores. The best win rates are obtained when the redundancy avoidance policy is used, population size is equal to 10 and the depth of the simulation is 14. The proposed RHEA-SB-RedAvoid-1Step obtained the best F1-Score when population size is equal to 2 and the depth of the simulation is 8.

The proposed RHEA variation have shown competitive results when compared to vanilla MCTS. Besides that, when compared to MCTS, RHEA obtained a higher overall game

score in most of the games considered here. However, when applying the GVGAI competition ranking method both algorithms reached similar F1-Scores.

The results obtained indicate that RHEA when combined with 1-Step-Look-ahead and a redundancy avoidance policy is promising and this achievement encourages further research. For instance, the adoption of other methods to assist RHEA, such as seeding techniques and rollouts.

There are studies where enhancements are proposed to the MCTS algorithm and some of them can also be considered to RHEA. Thus, the analysis of those enhancements when applied to both RHEA and MCTS provides an interesting future work. Finally, the application of other search techniques, such as GRASP, tabu search and path relink, can be adopted to create new rolling horizon controllers.

Acknowledgment

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

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] J. Levine, C. Bates Congdon, M. Ebner, G. Kendall, S. M. Lucas, R. Miikkulainen, T. Schaul, and T. Thompson, "General video game playing," *Dagstuhl Follow-up*, 2013.

[5] 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.

[6] R. D. Gaina, J. Liu, S. M. Lucas, and D. Perez-Liebana, "Analysis of vanilla rolling horizon evolution parameters in general video game playing," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2017, pp. 418–434.

[7] 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.

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

[9] N. Justesen, T. Mahlmann, and J. Togelius, "Online evolution for multi-action adversarial games," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2016, pp. 590–603.

[10] C. Wang, P. Chen, Y. Li, C. Holmgård, and J. Togelius, "Portfolio online evolution in starcraft," in *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2016, pp. 114–120.

[11] D. Perez-Liebana, 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.

[12] D. Perez-Liebana, S. Samothrakis, J. Togelius, T. Schaul, S. M. Lucas, A. Couëtoux, J. Lee, C.-U. Lim, and T. Thompson, "The 2014 general video game playing competition," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 8, no. 3, pp. 229–243, 2016.

[13] 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.

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

[15] 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.

[16] E. H. dos Santos and H. S. Bernardino, "Redundant action avoidance and non-defeat policy in the monte carlo tree search algorithm for general video game playing," *Proceedings do XVI Simpsio Brasileiro de Jogos e Entretenimento Digital*, 2017.

[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.