

Evaluating the Performance of Agents for Deceptive Levels

Noor Us Sabah¹✉, Adeel Zafar¹

¹Riphah International University, Islamabad, Pakistan

noorsaba560@yahoo.com; adeel.zafar@riphah.edu.pk

Corresponding Author: noorsaba560@yahoo.com

Abstract

Deceptive levels are levels where agents are trapped in the game and the policy leads them away from the global optimum policy. In this paper, we examine the problem of evaluating deceptive levels. In our previous work, we generated deceptive levels and evaluated them using different agents. The interesting insight from that study was that there was no single agent being able to solve all types of deceptions. In order to further extend our work, we investigate the effect of different metrics on the evaluation of deceptive levels. In our current work, we use different metrics including reward penalization, discounted reward, noisy world, broken world, and broken forward model. Initially, we execute these agents by utilizing deceptive as well as non-deceptive levels to perform a comparison on the basis of total points and win-rate. After comparison, ranks are assigned to the agents. Contrary to our previous evaluation, these metrics give different results and rankings. The experiment implies that each agent has a different set of characteristics and weaknesses.

Article History

Manuscript Received
April 2, 2025

First Revision
May 30, 2025

Final Acceptance
June 11, 2025

Keywords: Procedural Content Generation; General Video Game Artificial Intelligence; General Video Game Level Generation; Video Game Description Language

1 Introduction

Procedural Content Generation (PCG) is an algorithmic approach to generate content for video games [1]. Different methods are used for content generation in games such as search-based [1], constructive [2] and machine learning-based methods [3].

The evaluation of general video game levels is considered a challenging task due to the diverse nature of games. Various evaluation approaches are already employed by academicians including shallow decision-making analysis [4], ranking approach [5], relative algorithm performance profiles (RAPP) [6]. In our previous work [7], we generated different deceptive levels and evaluated them using different agents. An interesting insight from that study was the limitation of each agent as no agent was able to solve all deceptions. In this paper, we extend our work by evaluating deceptive levels on different set of metrics. We use a rank-based approach, where we assign a rank to the game-playing agents. In addition, five metrics are also used for the ranking of agents including reward penalization, discounted reward, noisy world, broken world, and broken forward model.

$$\text{Formula 1 Scoring System} = \{25, 18, 15, 12, 10, 8, 6, 4, 2, 1\} \quad (1)$$

Equation (1) is used to assign rank to the game-playing agents. The formula assigns 25 points to the best controller and 1 point to the last. Our experiment results show that OLETS is the best-performing agent contrary to our previous experiment where Number27 was the best performing agent. In addition, this approach will help us to create better or optimal agent for deceptive levels.

The remainder of the paper is organized as follows. In section 2, we present related work and background knowledge. Section 3 presents the details of our proposed approach. Details of the experiment are discussed in section 4. Section 5 presents the discussion on the evaluation of agents. Finally, the paper is concluded in section 6.

2 Related Work and Background Knowledge

2.1 Level Evaluation

Level Evaluation can be performed using multiple approaches including computational metrics, user-based evaluation, and agent-based evaluation. In the next sub-sections, we cover the background knowledge on each one of them.

Computational Metrics

Computational metrics measure emergent properties of levels, rather than simply using the same parameters. In [8], authors demonstrated evaluation as a problem in PCG. A comparative evaluation technique was applied for the game of Super Mario Bros.

User Based Evaluation

User-based evaluation allows the user to classify the content [9]. Authors proposed a way to solve the problem of automatically generated game levels. Different techniques of modeling and evolutionary algorithms were also introduced for the construction of tracks. Similar user-based studies were also performed by authors in [18], [19] and [20].

Rank-Based Evaluation

Rank-based evaluation allows the user to rank the generated content. In [5], authors presented a research study to test the performance of various game-playing agents. Results indicate that the performances of some controllers were better in almost all conditions while the performances of other controllers were worst.

Agent-Based Evaluation

Agent-based evaluation is an evaluation-based approach of game-playing agents, where we test different controllers on game levels without human intervention. In [4], a method to measure the performance of agents was proposed. Results indicate that these methods give them a better understanding of the decision-making process of the agents. In [6], authors provided a technique to rank the content according to win percentage and mean score obtained from various agents.

2.2 VGDL, GVG-AI, and GVG-LG Framework

Video Game Description Language (VGDL) [10] is a simple, expressive, and human-readable language. Authors highlighted that this language is needed to understand the game logic and it also requires

computational agents. The general video artificial intelligence (GVG-AI) framework [11] is proposed to test different artificial intelligent agents. In [12], authors proposed a GVG-LG framework as a benchmark for general video game level generation. It allows competitors to create generators that can generate levels for different games.

2.3 Deceptive Level Generator

The concept for deceptive games has been introduced by [13]. In [7], authors demonstrate that these games were designed in a way where controllers get rewards that can lead them away from the global optimum policy. Zafar et al. [7] proposed a deceptive level generator. The level generator generates three types of traps including greedy traps, smoothness traps, and generality traps. The results of this study showed that some agents perform poorly on deceptive levels and no single algorithm was able to solve all the deceptions. Figure 1 shows the deceptive levels for the game of Zelda. Figure 2 shows the deceptive levels for the game of Solar fox.



Figure 1: Deceptive levels for game Zelda (The authors have taken copyright permission from (Zafar et al. 2018) to use the figure)



Figure 2: Deceptive levels for game Solarfox (The authors have taken copyright permission from (Zafar et al. 2018) to use the figure)

3 Method

This section presents the proposed methodology used to evaluate the deceptive levels using a variety of agents. Our proposed approach is a rank-based approach, where we assign ranks to the game playing agents on the basis of total points and win-rate. The flow chart of our proposed evaluation methodology is illustrated in Figure 3. In total, there are 1250 deceptive and non-deceptive levels executed by all the agents. The deceptive levels used are generated by [7] in the GVG-LG framework and non-deceptive levels are selected from the GVG-AI framework. Five metrics are also used including reward penalization, discounted reward, noisy world, broken world, and broken forward model. By looking at the nature of the algorithm used by each agent, we have embedded these metrics in the GVG-AI framework. After the execution of the agents, we computed results such as total points, win rate and mean score. Then finally, Equation (1), is applied that depicts the performance evaluation criteria. The controllers are sorted based on their rankings. To decide the best controller all points of five games are added. After adding up the points, the agent with the highest sum among all other agents is considered as the winner. When the points in the game are considered as a draw, the first position ties the ranking and further proceeds to the second and third positions. Afterwords on the basis of total points and win rate, we decide the best-game playing and worst-game playing agents.

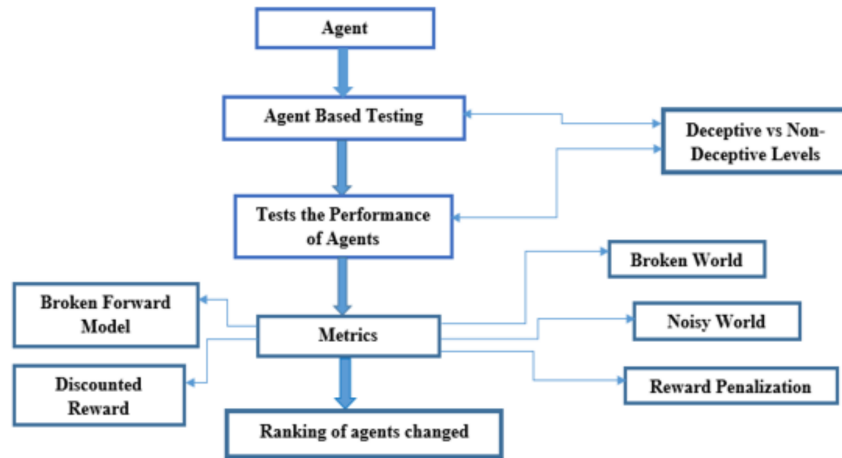


Figure 3: Flow chart of our proposed evaluation methodology

3.1 Selection of Games

A brief description of the games [7] used in our research are as follows:

- **Zelda:** The goal of the player in this game is to find the key and escape. The player is also equipped with the sword to kill enemies.
- **Catapults:** The goal of this game is to reach the exit door. The player can not jump on the water and can use catapults to reach it.
- **Cakybaky:** The primary goal for such a game is to bake the cake. In order to bake the cake, the player has to collect some ingredients.
- **Digdug:** The purpose of this game is to gather all gold coins and gems in the cave, exploring the way through it.
- **Solar fox:** The player's objective in this game is to collect diamonds and avoid the attacks of enemies.

3.2 Selection of Controllers

A brief description of the controllers used in our research are as follows:

- **Number27:** It is an agent with such a population size which involves individuals with a set sequence of actions [14].
- **Thorbjrn:** This agent attempts 49 actions and optimizes various game-playing algorithms and output differences. This agent uses evolutionary and MCTS algorithms [15].
- **OLETS:** OLETS is the version of MCTS, where agent's states are never correlated with the node of the network [5].
- **SampleRS:** The sampleRS controller generates the individuals randomly instead of evolving them. It has length of 10 and it can produce as many sequences in the given time duration [16].
- **SampleOLMCTS:** It is a very common tree search technique which iteratively creates an asymmetric tree in memory to estimate the value of the various actions available from a given state [5].
- **NovTea:** NovTea implements tree-search algorithms and returns the actions [15].
- **CatLinux:** It is an agent that implements Rolling Horizon Evolutionary Algorithms (RHEA). The RHEA algorithms evolves a population of sequences of actions, and the length of each sequence is fixed which is equal to the fixed depth of simulation [6].
- **SampleMCTS:** The sampleMCTS is an MCTS implementation with frequently random roll-outs up to a maximum depth of 10. Once the agent is initialized, the tree policy should be defined, while the heuristic used to determine game states is a combination of the score plus an eventual bonus [5].
- **SampleOneStep:** It rolls the forward model from available actions and selects the best value action. It also determines the score maximization function that minimizes the NPCS distance [4].

- **DoNothing:** It is the basic controller and it always returns *ACTION_NIL*, so it does not performs any action during the game [17].

4 Results

This section presents the details of the experiment carried out to evaluate the performance of different game-playing agents. As discussed earlier, five metrics are used to rank the agents. Later, we will discuss the results of the experimentation of each metric.

4.1 Evaluation of Deceptive vs Non-Deceptive levels

We executed top ten agents including Number27, OLETS, thorbjrn, OLMCTS, sampleRS, NovTea, CatLinux, sampleMCTS, SampleOneStep and doNothing. These agents are selected from the GVG-AI competitions. The performance of the agent is dependent on win-rate and total points. In Table 1, G-0 represents the zelda game, G-1 represents the catapults game, G-2 represents the cakybaky game, G-3 represents the dig dug game and G-4 represents the solarfox game (the same abbreviation is used for all other tables). In the case of deceptive levels, Number27 is the best-performing agent with respect to total points and win-rate, while doNothing is the least-performing agent with respect to total points and win-rate. All agents successfully played the levels of Zelda game, whereas there is no agent which is capable of playing the levels of Solarfox game. Table 1 illustrates the ranking of agents by using deceptive levels. In the case of non-deceptive levels, Number27 is the agent with better performance with respect to total points and win-rate, while sampleOneStep is the agent with worst performance with respect to total points and win-rate. In this scenario, almost all agents are capable of playing the Zelda game levels, while there is no agent that is capable of playing the Solarfox game levels. Table 2 illustrates the ranking of agents by using non-deceptive levels.

Table 1: Ranking of agents by using deceptive levels

Agents Ranking									
Rank no.	Agents	Total Points	Mean Score	Win Rate	G-0	G-1	G-2	G-3	G-4
1	Number27	125	0.006	52%	25	25	25	25	25
2	OLETS	118	0.63	40%	25	25	25	25	18
3	sampleRS	118	0.63	28%	25	25	18	18	25
4	Thorbjrn	111	0.71	40%	25	25	25	18	18
5	OLMCTS	108	0.01	24%	25	25	25	18	15
6	CatLinux	108	0.01	20%	25	25	18	25	15
7	sampleMCTS	103	0.45	16%	25	25	18	10	25
8	NovTea	103	0.45	20%	25	25	25	10	15
9	sampleOneStep	93	0.13	8%	25	25	18	10	15
10	doNothing	86	0.02	4%	25	25	25	10	1

Table 2: Ranking of agents by using non deceptive levels

Agents Ranking									
<i>Rank no.</i>	<i>Agents</i>	<i>Total Points</i>	<i>Mean Score</i>	<i>Win Rate</i>	<i>G-0</i>	<i>G-1</i>	<i>G-2</i>	<i>G-3</i>	<i>G-4</i>
1	Number27	125	0.006	36%	25	25	25	25	25
2	OLETS	118	0.63	30%	25	25	18	25	25
3	sampleRS	108	0.01	29%	25	25	18	15	25
4	Thorbjrn	108	0.01	33%	25	25	25	15	18
5	OLMCTS	103	0.45	16%	25	25	18	10	25
6	CatLinux	100	0.01	30%	25	25	25	15	10
7	sampleMCTS	93	0.13	20%	25	18	25	15	10
8	NovTea	93	0.13	23%	25	25	18	15	10
9	doNothing	90	0.12	8%	25	25	18	10	12
10	sampleOneStep	86	0.02	6%	25	25	10	1	25

4.2 Reward Penalization

In this metric, we change the settings of the GVG-AI framework in such a way to reduce the score s in the game by one point when the action is performed by the agent [5]. So, that better-performing controllers would get reward in the rank. The formula for the calculation is: $s = s - \text{actions}$. We compared the rankings of agents with deceptive as well as non-deceptive levels. Table 3 depicts the ranking for this metric. In this setting, the ranks are assigned to the agents according to total points but the ranking is also dependent on the win-rate. All agents performed well in the reward penalization settings except Number27 and sampleRS. Some agents achieved an increase in the win-rate such as CatLinux. The Table 3 show that in this setting, the OLMCTS is the first-ranked agent, OLETS is the second-ranked agent and Number27 is the third-ranked agent.

Table 3: Ranking of agents in the Reward Penalization setting by using deceptive levels

Reward Penalization Settings									
<i>Rank no.</i>	<i>Agents</i>	<i>Total Points</i>	<i>Mean Score</i>	<i>Win Rate</i>	<i>G-0</i>	<i>G-1</i>	<i>G-2</i>	<i>G-3</i>	<i>G-4</i>
1	OLMCTS	125	0.04	25%	25	25	25	25	25
2	OLETS	118	0.06	35%	25	25	25	18	25
3	Number27	108	0.03	23%	25	25	25	18	15
4	thorbjrn	95	0.05	30%	25	25	18	15	12
5	CatLinux	93	0.96	36%	25	18	25	15	10
6	NovTea	93	0.02	36%	25	18	15	25	10
7	sampleMCTS	90	0.04	17%	25	25	18	12	10
8	sampleRS	88	0.05	10%	25	25	18	10	10
9	sampleOneStep	88	0.64	4%	25	18	25	10	10
10	doNothing	79	0.02	2%	25	18	25	1	10

4.3 Discounted Reward

In this metric, we changed the settings of the GVG-AI framework in such a way, that it depends on the Discounted Factor D when the forward model returns the score s [5]. The value of the discounted factor D is set to 0.9. The formula for the calculation is: $s = s * D$. We compared the rankings of agents with deceptive as well as non-deceptive levels. Table 4 shows that in this setting, the ranks are assigned to the agents according to total points but the ranking is also dependent on the win-rate. In this setting, the sampleRS agent suffers a drop in win-rate. Thus, the performance of the controllers has the highest impact on the game Zelda. In the previous setting, the first-ranked controller is the OLMCTS, whereas in this setting the first-ranked controller is OLETS, the second-ranked controller is thorbjrn and the third-ranked controller is OLMCTS.

Table 4: Ranking of agents in the Discounted Reward setting by using deceptive levels

Discounted Reward Settings									
Rank no.	Agents	Total Points	Mean Score	Win Rate	G-0	G-1	G-2	G-3	G-4
1	OLETS	118	0.07	35%	18	25	25	25	25
2	thorbjrn	108	0.04	30%	25	25	18	25	15
3	OLMCTS	108	0.04	15%	25	18	25	15	25
4	Number27	100	0.01	30%	25	15	25	10	25
5	CatLinux	100	0.01	38%	25	25	15	10	25
6	sampleMCTS	95	0.02	21%	25	18	15	25	12
7	sampleRS	91	0.05	10%	25	18	15	25	8
8	NovTea	86	0.01	38%	25	18	10	25	8
9	sampleOneStep	69	0.04	13%	25	10	8	25	1
10	doNothing	63	0.01	2%	25	12	15	10	1

4.4 Noisy World

In this metric, when action is performed by an agent, noise is added to the actions [5]. The noise is added in the actions when $p = 0.25$. Table 5 illustrates the rankings of agents obtained with noisy world settings. In this setting, ranks are assigned to the agents according to total points but the ranking is also dependent on the win-rate. In this setting, the OLETS agent is still in the first position as in the discounted reward setting, second-ranked agent is Number27. OLETS is the best controller tested with the highest points. And doNothing is the least-performing controller. The loss of scores in Zelda, Catapults, Cakybaky, and Digdug is smaller as compared to the Solarfox. The higher scores are achieve-able only in the Zelda game where the key is awarded to the agent in the game.

Table 5: Ranking of agents in the Noisy World setting by using deceptive levels

Noisy World Settings									
Rank no.	Agents	Total Points	Mean Score	Win Rate	G-0	G-1	G-2	G-3	G-4
1	OLETS	100	0.07	20%	25	25	15	10	25
2	Number27	92	0.01	30%	25	15	12	25	15
3	thorbjrn	87	0.05	20%	25	10	12	15	25
4	sampleRS	87	0.07	15%	25	25	15	12	10
5	OLMCTS	87	0.04	30%	25	15	12	25	10
6	CatLinux	85	0.01	20%	25	15	12	8	25
7	sampleMCTS	83	0.02	16%	25	15	8	25	10
8	NovTea	85	0.01	20%	25	15	10	25	10
9	sampleOneStep	80	0.4	19%	25	18	15	12	10
10	doNothing	79	0.02	4%	25	18	10	25	1

4.5 Broken World

In this metric, the real game introduces the noise in the actions, while the forward model is always accurate [5]. But the noise is added in the actions when $p = 0.25$. In this setting, the agent has no idea how to evaluate states so some states or actions can be missed by the agent. Table 6 shows that in this setting, the ranks are assigned to the agents according to total points but the ranking is also dependent on the win-rate. The results obtained in this setting are completely different as compared to the noisy world settings. Number27 is the highest-ranked agent that achieves the best results in the five games and doNothing is the least-rank agent that achieves poor results in the five games. The agent with the highest increase in the win-rate is OLETS. It is important to note here that the errors are now introduced in the rankings of the agents.

Table 6: Ranking of agents in the Broken World setting by using deceptive levels

Broken World Settings									
<i>Rank no.</i>	<i>Agents</i>	<i>Total Points</i>	<i>Mean Score</i>	<i>Win Rate</i>	<i>G-0</i>	<i>G-1</i>	<i>G-2</i>	<i>G-3</i>	<i>G-4</i>
1	Number27	108	0.03	23%	25	18	25	25	15
2	sampleRS	98	0.06	17%	25	25	18	15	15
3	thorbjrn	93	0.03	18%	25	15	18	25	10
4	OLETS	91	0.04	18%	25	18	15	8	25
5	OLMCTS	88	0.02	31%	25	18	10	10	25
6	NovTea	86	0.04	38%	25	18	10	25	8
7	sampleOneStep	84	36%	0.05	25	18	8	8	25
8	sampleMCTS	83	0.04	15%	25	10	15	25	8
9	CatLinux	80	0.01	20%	25	18	15	12	10
10	doNothing	69	0.02	4%	25	18	10	8	8

4.6 Broken Forward Model

In this metric, the forward model introduces the noise with $p = 0.25$, while the actions provided to the game are not changed [5]. Table 7 shows that in this setting, the ranks are assigned to the agents according to total points but the ranking is also dependent on the win-rate. In the settings of the broken forward model, OLETS is the only controller that achieves the highest position in the rankings. The Number27 agent has the same points as the OLETS agent but it is a second-ranked agent. As a result, when the noisy actions are executed, it is only present in the game but not in the forward model. The results show that the behavior of the agents in these metrics is changed due to increase or decrease in win-rate and total points. It is important to note here that OLETS is the best-performing agent. It works better in the metrics including discounted reward, noisy world and broken forward model because in these metrics it is the top-ranked agent as it attains the highest points, whereas in these metrics the points of other agents are comparatively low. OLETS uses Hierarchical Open-Loop Optimistic Planning (HOLOP) for playing different games and it turns out that its performance on a broader scale is better. OLETS's performance is worst in reward penalization metric because in this metric the OLETS agent achieves second position as it attains lowest points. But in this metric, OLMCTS agent performs better because it attains highest points, as it is a variant of Monte-Carlo Tree Search (MCTS) and is designed to work better in stochastic environments. It uses the forward model to reevaluate the actions instead of keeping the states of the game in the nodes of the tree. The performance of OLETS agent is also worst in broken world metric because in this metric it achieves fourth position as it attains lowest points. But in this metric, Number27 is the best-performing agent in the broken world metric as it attains highest points. The Number27 agent consists of a genetic algorithm. It uses multiple action sequences instead of using a single one. As each agent uses different algorithms, so it is also important to look at the the nature of the algorithm to decide best-performing agent. We can conclude here that the use of Hierarchical Open-Loop Optimistic Planning (HOLOP) is better for deception problems because when the agent plays any level of a game, the value of next actions are also assessed. It not only includes the average value of the node but also the maximum value among its children.

Table 7: Ranking of agents in the Broken Forward Model setting by using deceptive levels

Broken Forward Model Settings									
Rank no.	Agents	Total Points	Mean Score	Win Rate	G-0	G-1	G-2	G-3	G-4
1	OLETS	108	0.03	23%	25	25	25	18	15
2	Number27	108	0.03	18%	25	25	15	25	18
3	Thorbjørn	100	0.06	17%	25	25	25	15	10
4	OLMCTS	85	0.02	31%	25	15	25	10	10
5	sampleRS	100	0.06	17%	25	25	25	15	10
6	CatLinux	86	0.01	44%	25	25	18	10	8
7	sampleMCTS	85	0.04	15%	25	25	15	10	10
8	NovTea	80	0.04	38%	25	25	12	10	8
9	sampleOneStep	69	0.05	20%	25	18	10	8	8
10	doNothing	63	0.02	4%	25	15	10	12	1

5 Conclusion

The focus of this research is to assign ranks to the game-playing agents. Initially, we execute the agents on deceptive as well as non-deceptive levels to perform a comparison on the basis of total points and win-rate. There are 1250 deceptive levels played by each controller. Five metrics are also used including reward penalization, discounted reward, noisy world, broken world and broken forward model. These metrics analyze how the rankings of the game-playing agents are changed. The proposed approach successfully assigns a rank to the agents based on the characteristics of the agents including total points and win-rate. The experiments indicate better performance of OLETS agent in the majority of metrics. Few agents are not able to perform well in noisy conditions including sampleOneStep and doNothing. By analyzing the performance of the agents in detail, we have a better understanding of the ranking of the agents.

References

- [1] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, “Search-based procedural content generation,” in *Applications of Evolutionary Computation (EvoApplications 2010)*, ser. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2010, pp. 141–150.
- [2] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural Content Generation in Games*, ser. Computational Synthesis and Creative Systems. Cham, Switzerland: Springer International Publishing, 2016.
- [3] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgård, A. K. Hoover, A. Isaksen, A. Nealen, and J. Togelius, “Procedural content generation via machine learning (PCGML),” *IEEE Transactions on Games*, vol. 10, no. 3, pp. 257–270, 2018.
- [4] I. Bravi, D. Perez-Liebana, S. M. Lucas, and J. Liu, “Shallow decision-making analysis in general video game playing,” in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2018, pp. 1–8.
- [5] D. Pérez-Liebana, S. Samothrakis, J. Togelius, T. Schaul, and S. M. Lucas, “Analyzing the robustness of general video game playing agents,” in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2016, pp. 1–8.
- [6] T. S. Nielsen, G. A. B. Barros, J. Togelius, and M. J. Nelson, “General video game evaluation using relative algorithm performance profiles,” in *Applications of Evolutionary Computation (EvoApplications 2015)*, ser. Lecture Notes in Computer Science. Cham: Springer, 2015, pp. 369–380.

- [7] A. Zafar, H. Mujtaba, M. O. Beg, and S. Ali, “Deceptive level generator,” in *Joint Proceedings of the AIIDE 2018 Workshops*, ser. CEUR Workshop Proceedings, vol. 2282, Edmonton, Canada, 2018. [Online]. Available: https://ceur-ws.org/Vol-2282/EXAG_105.pdf
- [8] G. Smith and J. Whitehead, “Analyzing the expressive range of a level generator,” in *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, 2010, pp. 1–7.
- [9] W. M. P. Reis, L. H. S. Lelis, and Y. Gal, “Human computation for procedural content generation in platform games,” in *2015 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2015, pp. 99–106.
- [10] M. Ebner, J. Levine, S. M. Lucas, T. Schaul, T. Thompson, and J. Togelius, “Towards a video game description language,” in *Artificial and Computational Intelligence in Games*, ser. Dagstuhl Follow-Ups, S. M. Lucas, M. Mateas, M. Preuss, P. Spronck, and J. Togelius, Eds. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2013, vol. 6, pp. 85–100.
- [11] 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.
- [12] A. Khalifa, D. Perez-Liebana, S. M. Lucas, and J. Togelius, “General video game level generation,” in *Proceedings of the Genetic and Evolutionary Computation Conference 2016 (GECCO)*, 2016, pp. 253–259.
- [13] D. Anderson, M. Stephenson, J. Togelius, C. Salge, J. Levine, and J. Renz, “Deceptive games,” in *Applications of Evolutionary Computation (EvoApplications 2018)*, ser. Lecture Notes in Computer Science. Cham: Springer, 2018, pp. 376–391.
- [14] D. Perez-Liebana, J. Liu, A. Khalifa, R. D. Gaina, J. Togelius, and S. M. Lucas, “General video game AI: A multitask framework for evaluating agents, games, and content generation algorithms,” *IEEE Transactions on Games*, vol. 11, no. 3, pp. 195–214, 2019.
- [15] 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.
- [16] D. Anderson, P. Rodgers, J. Levine, C. Guerrero-Romero, and D. Perez-Liebana, “Ensemble decision systems for general video game playing,” in *2019 IEEE Conference on Games (CoG)*. IEEE, 2019, pp. 1–8.
- [17] R. D. Gaina, D. Pérez-Liebana, and S. M. Lucas, “General video game for 2 players: Framework and competition,” in *2016 8th Computer Science and Electronic Engineering (CEECE)*. IEEE, 2016, pp. 186–191.
- [18] C. Pedersen, J. Togelius, and G. N. Yannakakis, “Modeling player experience in Super Mario Bros,” in *2009 IEEE Symposium on Computational Intelligence and Games*. IEEE, 2009, pp. 132–139.
- [19] H. Desurvire, M. Caplan, and J. A. Toth, “Using heuristics to evaluate the playability of games,” in *CHI’04 Extended Abstracts on Human Factors in Computing Systems*, 2004, pp. 1509–1512.
- [20] P. Sweetser and P. Wyeth, “GameFlow: A model for evaluating player enjoyment in games,” *Computers in Entertainment (CIE)*, vol. 3, no. 3, p. 3, 2005.