IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 22, Issue 6, Ser. II (Nov. – Dec. 2020), PP 08-15
www.iosrjournals.org

Mario AI Competition

Rabeya Sultana
Prime University, Dhaka, Bangladesh

Ferdousi Barira
Prime University, Dhaka, Bangladesh

Abstract
The Level Generation Competition, Mario AI Championship, was to the knowledge of the world's first procedural content generation competition. Competitors participated by submitting level generators — software that generates new levels for a version of Super Mario Bros tailored to individual players' playing style. This paper presents the principles of the competition, the scoring procedure, the submitted level generators, and therefore the results of the competition. Here also discuss what are often learned from this competition, both about organizing procedural content generation competitions and about automatically generating levels for platform games.

Keywords: *Mario, game AI, reinforcement learning, competitions.*

Date of Submission: 12-12-2020 Date of Acceptance: 27-12-2020

I. Introduction

In the last few years, a number of game AI competitions have been run in association with major international conferences, several of them sponsored by the IEEE Computational Intelligence Society. These competitions are based either on classical board games (such as Othello and Go) or video games (such as Pac-Man, Super Mario Bros, and Unreal Tournament). In most of these competitions, competitors submit controllers that interface to the game through an API built by the organizers of the competition. The competition is won by the person or team that submitted the controller that played the game best, either on its own (for single-player games such as Pac-Man) or against others (in adversarial games such as Go). One interesting variation on this formula is the 2k BotPrize, where the submitted entries are not supposed to play the game as well as possible, but in a human-like manner as possible [1]. Several of these competitions have spurred valuable research contributions as reported in [2], [3].

Here the **'Mario AI Competition'** which ran from 2009 to 2012 is focused: providing an introduction to the problem area and the challenges it presented. Here I explore one of the most interesting areas to have arisen in Artificial Intelligence for games in recent years: procedural content generation. The challenges faced in developing AI that creates content that is assessed primarily by the fun it generates for the player. This is the focus of a particular track of the competition.

II. Infinite Mario Bros

Infinite Mario Bros (Markus Persson 2008) is a public domain clone of Nintendo's classic platform game Super Mario Bros (1985). The original Infinite Mario Bros is playable on the web, where Java source code is also available. The gameplay in Super Mario Bros consists of moving the player-controlled character, Mario, through two-dimensional levels, which are viewed sideways. Mario can walk and run to the right and left, jump, and (depending on which state he is in) shoot fireballs. Gravity acts on Mario, making it necessary to jump over holes to get past them. Mario can be in one of three states: Small, Big (can crush some objects by jumping into them from below), and Fire (can shoot fireballs). The main goal of each level is to get to the end of the level, which means traversing it from left to right. Auxiliary goals include collecting as many as possible of the coins that are scattered around the level, finishing the level as fast as possible, and collecting the highest score, which in part depends on the number of collected coins and killed enemies. Complicating matters is the presence of holes and moving enemies. If Mario falls down a hole, he loses a life. If he touches an enemy, he gets hurt; this means losing a life if he is currently in the Small state. Otherwise, his state degrades from Fire to Big or from Big to Small. However, if he jumps and lands on an enemy, different things could happen. Most enemies (e.g. goombas, cannonballs) die from this treatment; others (e.g. piranha plants) are not vulnerable to this and proceed to hurt Mario; finally, turtles withdraw into their shells if jumped on, and these shells can then be picked up by Mario and thrown at other enemies to kill them. Certain items are scattered around the levels, either out in the

open or hidden inside blocks of brick, and only appearing when Mario jumps at these blocks from below so that he smashes his head into them. Available items include coins, mushrooms which make Mario grow Big, and flowers which make Mario turn into the Fire state if he is already Big. No textual description can fully convey the gameplay of a particular game. Only some of the main rules and elements of Super Mario Bros are explained above; the original game is one of the world's best selling games, and still very playable more than two decades after its release in the mid-eighties. Its game design has been enormously influential and inspired countless other games. The original Super Mario Bros game does not introduce any new game mechanics after the first level, and only a few new level elements (enemies and other obstacles). It is also very little in the way of the story. Instead, the player's interest is kept through rearranging the same well-known elements throughout several dozens of levels, which nevertheless differ significantly in character and difficulty. This testifies to the great importance of level design in this game (and many others in the same genre), and to the richness of the standard Super Mario Bros vocabulary for level design.

III. The Mario Ai Championship

The Mario AI Championship was set up as a series of linked competitions based on Infinite Mario Bros. In 2009, the first iteration of the Championship (then called the Mario AI Competition) was run as a competition focusing on AI for playing Infinite Mario Bros as well as possible. A write-up of the organization and the results of this competition can be found in [3]. The 2010 Mario AI Championship was a direct successor of this competition, but with a wider scope. It consisted of three competition tracks (the Gameplay Track, the Learning Track, and the Level Generation Track) that were run in association with three international conferences (EvoStar, IEEE Congress on Evolutionary Computation, and IEEE Conference on Computational Intelligence and Games). While the championship was open to participants from all over the world, the cash prizes (sponsored by the IEEE Computational Intelligence Society) could only be awarded to competitors that were physically present at the relevant competition event.

IV. A Little Bit of History

As noted earlier, the competition has run since 2009. However, the area of interest, the procedural generation track, did not run until 2010. The original emphasis of the competition, much like Ms. Pac-Man before it, was to create characters that could learn to play the game. The classic *Super Mario* game is one that is almost ingrained within modern culture. It provided an alternative to Ms. Pac-Man given that it a completely different style of game, given that it is a two-dimensional platformer.

Outside of the gameplay mechanics, it also acted in much the same way the Ms. Pac-Man had done previously. *Super Mario Bros.* is an equally, if not more so, recognizable game than Ms. Pac-Man. So it acted on two fronts: a legitimate problem to explore and an iconic game to rally interest. The team pushed the competition heavily through social media in order to get the message out there and encourage submissions not only from researchers but hobbyists too.

The Mario AI competition provides a legitimate AI problem to explore in an iconic game to rally interest.

The competition itself is reliant upon the Infinite Mario Bros. Clone that was developed by Markus 'Notch' Persson, who is more famously known for that little project he developed called *Minecraft*.

4.1 Game Play Track

The gameplay track was the inaugural track of the competition in 2009 and was aimed at researchers adopting any technique they wanted to play Mario levels. This resulted in a range of implementations (15 to be precise) that varied from hand-scripted to the use of genetic algorithms, neural networks, and reinforcement learning.

4.2 The Learning Track

As discussed in (Togelius et al., 2010) there was a gulf in the performance of the A* and handcrafted submissions versus those that were reliant upon machine learning algorithms. As a result, a separate track was introduced in 2010. However, it didn't seem to be as popular and only ran that year. The A* implementation made people think that the problem was 'solved' and did not merit further consideration.

4.3 The Turing Track

The behavior is not remotely human. As a result, the team also introduced a Turing track, whereby AI players must behave as human-like as possible. In many ways inspired by a similar competition running in Unreal Tournament 2004: the 2K BotPrize (Hingston, 2010). The Turing track has only run in the 2012 iteration of the competition and as noted in (Togelius, 2013a), bots were not capable of fooling the judges.

Interestingly the moments they appeared more human were when they were standing still or second-guessing their actions.

V. The Level Generation Track

The level generation track in the Mario competition is the first of its kind. Unlike the other tracks that were focused on producing software agents that could act in the players' stead as Mario, the focus was to create a level generator that would create levels to play. As mentioned in (Togelius et al., 2013a), when the first PCG track ran back in 2010, there were no real benchmarks that could be used to argue the validity of a generated piece of terrain.

Of course, in order to evaluate the generated content, it needs to be played! So the traditional method of simply allowing the AI to run against a problem doesn't work. So for judging to take place, human players would play against the generated levels. However, as mentioned in (Shaker et al., 2011), each judge was asked to play a test level that acted solely to accrue metrics from the players' performance. This information was passed on to the level generators to allow them to run their procedural generation. In addition, there are a number of constraints passed to each generator that dictate some expectations of the generated level. This included things such as the number of pits, the number of coin/power-up blocks.

This set of constraints were introduced for a reason: it would be naive to assume that these level generators will create something algorithmically without any constraints being imposed upon them. This might seem cynical, but there's a good chance that an unchallenged level generator is randomly sampling from collections of hand-crafted, human-designed solutions. That isn't really the focus of the competition. In fact, it would be cheating! This allows the competition to focus solely on content crafted procedurally.

NOTE: *One of the key things to remember about this competition is that the levels are not being designed to mimic ACTUAL Mario levels. Such a problem is – arguably – even more challenging, given that what we constitute as a Mario level adheres to a rule book that has never been released to the public domain and is locked in the minds of Nintendo's finest designers. While some generators hope to invoke some of these principles, they are often inferred from the experience of playing Mario games. The rules on what makes a level a "Mario level" may not actually exist.*

There are a number of different implementations that were developed for the 2010 competition. Here a handful and show the variety of approaches employed by different teams is shown. Some are driven by AI algorithms, whereas others have a more prescribed nature.

5.1 A 'Flow'-Driven Generator

The untitled submission by Tomonori Hashiyama (University of Electro-Communications, Tokyo) and Tomoyuki Shimizu (formerly Uni. Electro-Com, Tokyo), developed a system designed to make experiences 'flow' between one another. This is achieved by using what is known as an Interactive Evolutionary Computational algorithm. In short, an evolutionary algorithm is used to build the solutions, but the fitness criteria are first determined by the player's performance at the test level.

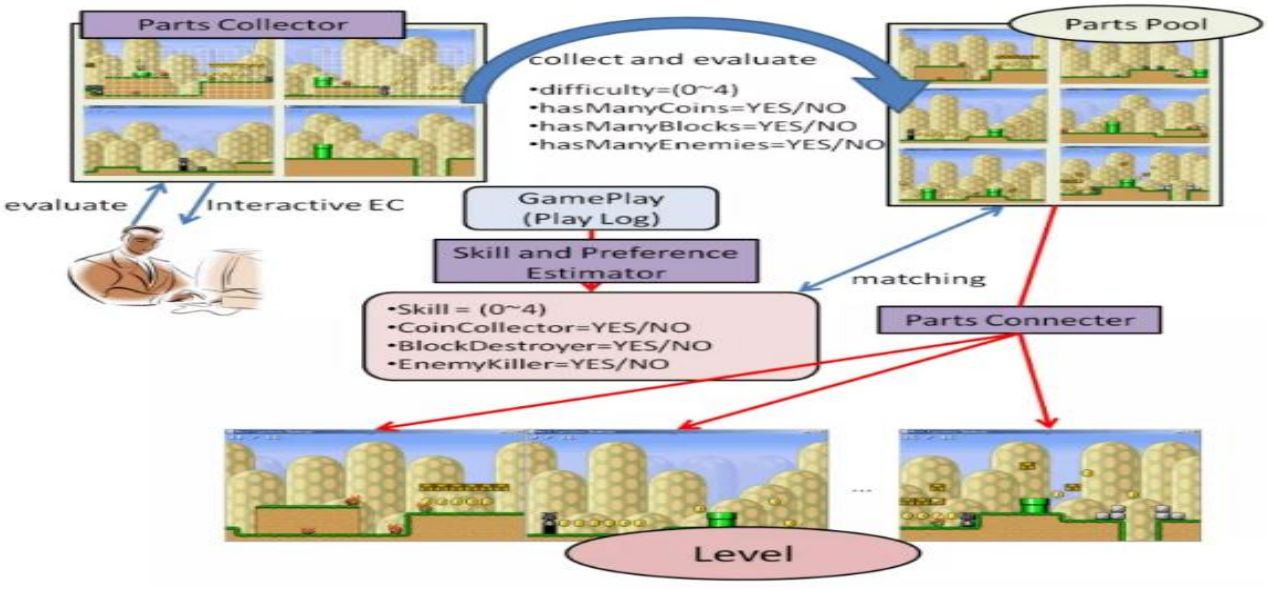

Figure 1: An overview of the architecture of the Shimizu and Hashiyama generator. (Shaker et al., 2011)

A skill and preference estimator derives key information from the player's data from the test level. Identifying your preference for collecting coins, killing enemies, and smashing blocks. The system is then reliant upon a collection of randomly crafted level 'chunks' which are assessed with respect to the metrics you have defined. The system then creates levels by matching chunks to the skill level, followed by the preference of the player.

5.2 The 'Hopper' Level Generator

This generator used a rule-driven approach to place tiles across the map from left to right, utilizing probabilities that were hand-crafted for things such as widths of gaps and enemy frequency. Depending on the incoming user-stats, different probabilities would be applied.

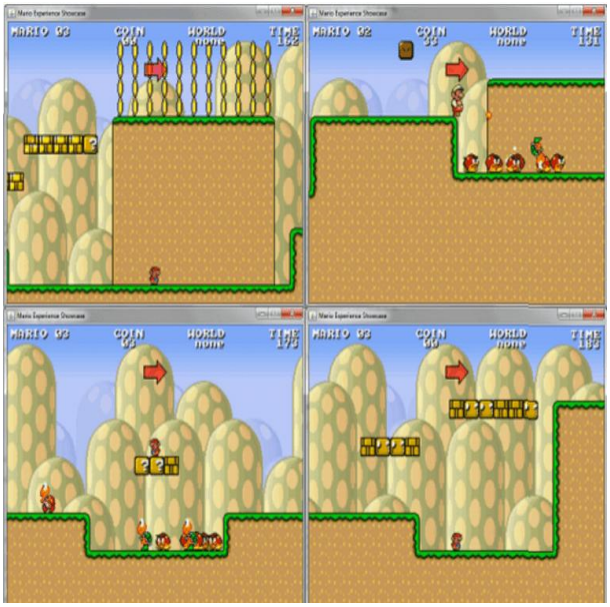

Figure 2: Screenshots showcasing the special 'zones' that the 'Hopper' generator by Glen Takahashi and Gillian Smith would implement. Clockwise from top-left: hidden coin zone, fire zone, super jump zone, and shell zone. (Shaker et al., 2011)

5.3 ProMP

Meanwhile, Ben Weber, a PhD student at the University of California, devised the Probabilistic Multi-Pass Generator (ProMP) that adds content to the level in stages. For each level generated, the system would make multiple passes in a specific order: main terrain, hills, pipes, enemies, blocks then coins. This is shown in an example below.

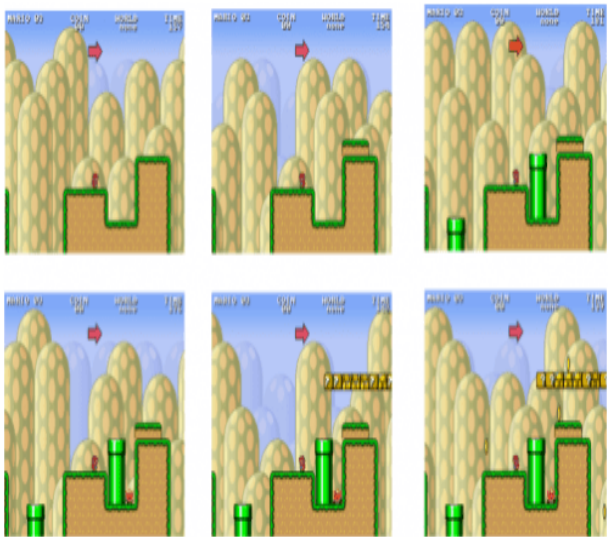

Figure 3: An overview of the level generation process in Ben Weber's implementation.

This process is also parameterized, allowing the levels to be tailored to suit the player's skill levels. These levels are pretty intense and would be insurmountable for many human players. Interestingly, it also identifies some limitations of the A* player.

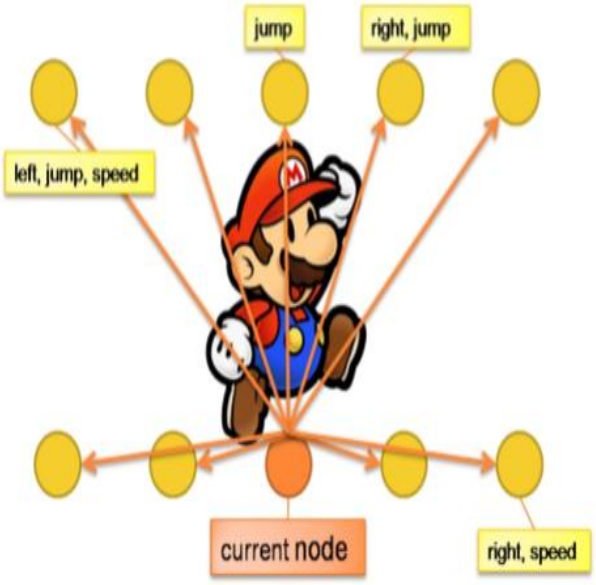

Figure 4: Illustration of node generation in A* search.

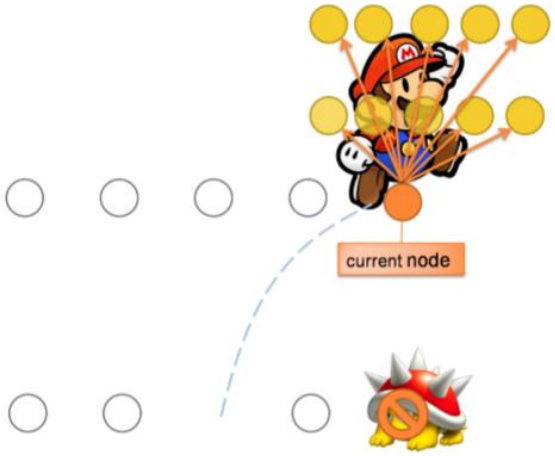

Figure 5: Illustration of node placement in A* search in Infinite Mario. The best node in the previous simulation step (right, speed) is inaccessible because it contains an obstacle. Thus the next best node is chosen, which results in a jump over the obstacle. The search process continues by generating all possible neighbors. The speed of Mario at the current node distorts the positions of all the following nodes.

5.4 Scoring procedure

The rationale behind the scoring was that the level generator which generates levels that were preferred by most players should win. As mentioned earlier, the primary aim of the competition was the generation of personalized Super Mario levels for particular players. For this purpose, we used human judges as Mario players to assess the quality of each submitted competitor; everyone who was present at the competition event was encouraged to participate in the judging. Each human judge was given a test level to play, and his or her performance on that level was recorded and passed on to the level generators. The judge then played two generated levels from two competing generators and ranked them according to how much fun they were to play. A two-alternative forced-choice questionnaire was used according to which each judge expressed a pairwise preference of fun after completing the two levels (i.e. "which game of the two was more fun to play?"). (The concept of "fun" was deliberately not defined further, so as not to bias judges more than what is unavoidable.) The adoption of this experimental procedure was inspired by earlier attempts to capture player experience via pairwise preference self-reports which were introduced by the competition organizers (see [4], [5], [6] among

others). For all competition entries to be treated fairly all generators had to be played an equal number of times by the judges and compared against all other generators submitted. On that basis, the required minimum number of judges was 15 given that there were six competitors (i.e. all possible combinations of 2 games out of 6 competitors). To control for order of play effects, each pair was played by the same judge in both orders.
To make sure that each pair of competitors were judged at least once in both orders we set up an online SQL database that initially contained all possible pairs marked as "unplayed". Whenever a game session started, the software connected to the database and asked for an unplayed pair to load. Once the two-level generators in the pair had been chosen from the database, the levels were generated according to the judge's gameplay behavioral statistics and the judge was set to play the generated two levels in both orders. The level generators had access to player metrics such as numbers of player jumps and coins collected. When the two games and the questionnaire were completed, the judge's preferences and gameplay statistics were stored in the database and the pair was marked as "played". The experiment is reset if there are no more pairs available in the database to play (all pairs are marked as "played").

VI. Lessons learned

Four years of running the Mario AI Championship has taught us a few things about what to do and what not to do when running a game-based AI competition. Let's start with what did right. Basing the competition on a version of a famous game and keeping an active presence in social media helped to get attention to the competition. There also went to great lengths to ensure that the API is very simple to use – the target was for anyone familiar with Java to be able to have a controller up and running within five minutes – and that the framework is computationally lightweight and lacks external dependencies. These factors together seem to be responsible for the impressive adoption of the software for both research and teaching (the software is used for assignments in dozens of AI classes worldwide). However, most of the controllers and level generators developed for research purposes have not been submitted to any track of the competition, and after 2010 the Gameplay and Learning tracks have eff ectively stagnated. We believe the main reason for this is our failure to keep a central, coordinated source of information for the competition. There has been confusion regarding which version of the software and which settings have been used for particular competition events, and it has not always been easy to find updated information about results or the submitted controllers themselves. This has diminished the value of the software as a benchmark. A better example to follow is the Ms. Pac-Man competition, which evaluates controllers automatically via a web interface and keeps an updated league table at all times.

VII. The Future of the Competition

The Mario AI Championship is launched under the name "The Platformer AI Competition", with reworked graphics and sounds from the open-source game SuperTux 4. The name and graphics change are meant to avoid problems associated with Nintendo's ownership of the name "Mario" and the Mario graphics, but also to signify a reinvigoration of the competition taking into account several of the lessons learned while running the Mario AI Championship. In particular, the new competition will adopt a better approach to making canonical benchmark code available for each competition event, and making the code of competition submissions available. Initially there concentrate on the Turing Test and Level Generation tracks, given that these contain the currently most fertile research challenges and seem to draw the most interest from the academic community.

VIII. Discussion

This section discusses what can learn from this round of the Level Generation Track (which was also the first academic PCG competition and the first competition about adaptive or controllable PCG), both about organizing a PCG competition and about generating levels for platform games.

8.1 Organizing a PCG competition

Compared to other game AI competitions the PCG competition attracted a reasonably large set of competitors, representing a considerable diversity both geographically and in particular in terms of algorithmic approaches to the particular content generation problem. All of the entries submitted contain novel elements, most of the approaches are sophisticated and some of them are connected to the competitors' ongoing research programs. The number and quality of submissions indicate a fairly strong interest in the field of procedural content generation forming a sub-community devoted to PCG that lies within the broader game AI and computational intelligence and games communities. Therefore, it seems very plausible that given a simple enough interface and an interesting enough content generation problem, future PCG competitions will attract good attention. In organizing this competition, the organizers drew on the experience of organizing several previous game AI-related competitions, as well as a set of "best practices" that have been accumulated within

the computational intelligence and games community over the past few years. One core principle is that the competition should be as open as possible in every sense, both in terms of source code, rules, procedures, and participation. Another key principle is that the software interface should be so simple that a prospective competitor is able to download the software and hack together a simple entry in five minutes. Limitations in terms of operating systems and programming languages should be avoided wherever possible. It has also become customary to provide a cash prize in the range of a few hundred dollars, along with a certificate, to the winner. We believe that these principles have served us well.

This not to say the current competition has been without its fair share of problems, actual as well as potential. It was until the last moment unknown how many members of the audience would be willing and able to participate in the judging, and it would, in general, be desirable to have a larger number of votes cast in order to increase the statistical validity of the scores. One of the key limitations of the existing survey protocol is that all entries need to be played against each other; ideally multiple times from different judges. That generates a large number of judges — which is combinatorial with respect to the number of entries — required to sufficiently assess the entries. This problem can be solved, in part, with a fair sampling of the pairs and an adaptive protocol which is adjusted according to the number of judges existent in the competition room. It is also questionable how representative of the general game-playing population an audience of game AI researchers is. As already mentioned, an Internet-based survey is currently running, where the software is included on a public web page and judges are solicited through mailing lists and social networking sites; this approach would undoubtedly come with its own set of limitations, such as preventing the competitors from gaming the system by voting multiple times themselves. Additional minor problems include the short time period given for the presentation of the competition; the competitors agree that it would have been very useful to have on spot presentations of their submissions as well. Moreover, one of the entries included a trivial but severe bug that was only discovered during the scoring, and which was arguably responsible for the very low score of that entry. The competition software repeatedly locked up on several of the judges' laptops during level generation for as yet unknown reasons. A potential problem is that someone could submit a "level generator" that essentially outputs the same human-designed level each time and, if that level is good enough, it could win the competition. As we have abandoned the idea of forcing additional constraints on the level generators for fear of restricting them too much, such a case would probably have to be decided by the organizers of the competition based on some fairly fuzzy guidelines. The deeper problem is that the distinction between a level and a level generator and is not clear. It should rather be thought of as a continuum with intermediate forms possible, e.g. a fixed level design that varies the number and distribution of enemies according to the player's skill level. (Bear in mind that several of the submitted level generators included complete human-designed level chunks of different sizes.) A possible solution to the above problem would be to let the judge play not one but several levels generated by the same level generator with the same player profile as parameters. In such a setting, a generator that always outputs the same level would probably come across as boring. This solution would also ensure that the judges rate that the actual design capacity of the generator rather than just the novelty value of a single generated level. If this is done, the player metrics might be updated as the player plays, allowing the generators to continuously adapt to a player's changing playing style. It would require that each judge spends more time on judging, which might lead to a shortage of willing judges, but given the considerable advantages, it seems like a good idea that the next level generation competition lets judges play several levels from each generator.

There are certain aspects of the questionnaire protocol used that could be improved on the next iteration of the competition. A 4-alternative forced-choice questionnaire scheme [7] could be adopted to improve the quality of self-reported preferences. Such a questionnaire scheme would include two more options for equal preferences (i.e. "Both levels were equally fun" and "Neither level was fun") and thereby eliminate experimental data noise caused by judges who do not have a clear preference for one of the two levels. In the future, we might consider including hand-authored levels (e.g. original Super Mario Bros levels) among the generated levels; a litmus test for whether the (personalized or other) level generators are really successful would be whether they have generally preferred over professionally hand-authored levels. We would also like to try to answer not only the "which" question about fun levels but also the "why" question; asking judges why they prefer a particular level over another would be interesting but would require significant human effort in interpreting the data. Another method would be to ask not only which level was more fun, but also which was more challenging, interesting, etc., similar to the questionnaires used in [8]. Another takeaway from previous CIG competitions is that competitions usually benefit from repetition. When basically the same competition is run a second or third time, competitors get a chance to perfect their entries and learn from each other, meaning that much better entries are submitted. Refining individual entries also mean that techniques that are more appropriate for the problem stand out from initially interesting ideas that fail to deliver on their promise. In other words, the scientific value of competition in general increases with the number of times it is run.

8.2 Generating levels for platform games

The main point to note about the competition results is that the simplest solution won. Ben Weber's ProMP level generator does not search and backtrack while constructing the level, does not include any human-designed level chunks, and does not in any way adapt to the judge's playing style. Above all, it does not attempt any form of large-scale level structure, pacing, or anything similar, but simply places individual level elements in a context-free manner.

It would be premature to conclude that the above-mentioned features (adaptation, human-designed chunks, search in level space, and macrostructure), which were attempted by the other generators, cannot in principle add to the quality of generated levels. Rather, we believe that imperfect implementation and a lack of fine-tuning was responsible for the relative failure of the more complex level generators. It is clear that the entrants need more time to perfect their entries, and possibly recombine ideas from different approaches. In addition, player behavioral information could assist the generation of more personalized, and thereby preferred, levels (as in [9]). While level generation studies in Super Mario indicate features that are responsible for a level's high aesthetical value [8] we are still far from identifying the complete set of features — which could be represented computationally — that would yield a highly engaging platform game. Earlier findings suggest that this feature set needs to be individualized for each player's behavioral type [8]. In other words, the competition needs to run again to give the competitors further opportunities to improve their level generators.

While Ben Weber's level generator did not generate any macrostructure, it can be argued that it generates more micro-structure than several of the other level generators. Individual images of levels generated by Ben Weber's generator tend to be densely filled with items, creatures, and landscape features and frequently give the false appearance of macrostructure, such as there being multiple paths through the level. This suggests that the current evaluation mechanism incentivizes judges to make judgments on level quality early or based only on local features. On a positive note, all the entries produced levels that were, at least once, judged to be more entertaining than some level generated by another entry. Also, the score difference between the winner and the runner-up was very small, despite the level generators being very dissimilar. This suggests that widely differing approaches can successfully be used to generate fun levels for Super Mario Bros. This particular content generation problem is still very much an open problem. Here also attempted to see how much of the preference for certain levels of others, and therefore the quality of level generators, can be explained by simple extracted features using linear correlations. The analysis showed that there are particular key level attributes, such as the number of coins and rocks as well as the average gap width and the even placement of enemies, that affect the fun preference of judges. These features are all negatively correlated; more items and more irregularly distributed items are associated with less fun. The most succinct summary of the statistical analysis would be that the less clutter, the more fun level. At the same time, the correlations are far from strong enough to explain all of the expressed preferences, suggesting that the relationship between level features and quality is too complex to be captured by linear correlations. We also know from previous research that level preferences are highly subjective. It is likely that an analysis of more extracted features, including playing style metrics, from a larger set of levels played by a larger set of judges could help us understand the complex interplay of the different aspects of level design better.

References

[1]. P. Hingston, "A new design for a turing test for bots," in Proceedings of the IEEE Conference on Computational Intelligence and Games, 2010.

[2]. D. Loiacono, P. L. Lanzi, J. Togelius, E. Onieva, D. A. Pelta, M. V. Butz, T. D. L¨onneker, L. Cardamone, D. Perez, Y. Saez, M. Preuss, and J. Quadflieg, "The 2009 simulated car racing championship," IEEE Transactions on Computational Intelligence and AI in Games, 2010.

[3]. J. Togelius, S. Karakovskiy, and R. Baumgarten, "The 2009 mario ai competition," in Proceedings of the IEEE Congress on Evolutionary Computation, 2010.

[4]. G. N. Yannakakis and J. Hallam, "Towards Optimizing Entertainment in Computer Games," Applied Artificial Intelligence, vol. 21, pp. 933– 971, 2007.

[5]. G. N. Yannakakis, H. P. Mart´ınez, and A. Jhala, "Towards Affective Camera Control in Games," User Modeling and User-Adapted Interaction, vol. 20, no. 4, pp. 313–340, 2010.

[6]. G. N. Yannakakis and J. Hallam, "Real-time Game Adaptation for Optimizing Player Satisfaction," IEEE Transactions on Computational Intelligence and AI in Games, vol. 1, no. 2, pp. 121–133, June 2009.

[7]. G. N. Yannakakis, "How to Model and Augment Player Satisfaction: A Review," in Proceedings of the 1st Workshop on Child, Computer and Interaction. Chania, Crete: ACM Press, October 2008.

[8]. C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling Player Experience for Content Creation," IEEE Transactions on Computational Intelligence and AI in Games, vol. 2, no. 1, pp. 54–67, 2010.

[9]. C. Pedersen, J. Togelius, and G. N. Yannakakis, "Modeling Player Experience in Super Mario Bros," in Proceedings of the IEEE Symposium on Computational Intelligence and Games. Milan, Italy: IEEE, September 2009, pp. 132–139.

Rabeya Sultana and Ferdousi Barira. "Mario AI Competition." *IOSR Journal of Computer Engineering (IOSR-JCE)*, 22(6), 2020, pp. 08-15.