The Unexpected Consequence of Incremental Design Changes

Nathan R. Sturtevant,[1] **Nicolas Decroocq,**[2] **Aaron Tripodi,**[3] **Matthew Guzdial**[1]
[1]Department of Computing Science, Alberta Machine Intelligence Institute (Amii), University of Alberta, Canada
[2]ESM Saint-Cyr, 1^{er} Bataillon de France, F-56380 GUER Cedex, France
[3]Department of Computing Science, University of Alberta, Canada
nathanst@ualberta.ca, nicolasdecroocq1@gmail.com, tripodi@ualberta.ca, guzdial@ualberta.ca

Abstract

Human designers may find it difficult to anticipate the impact of small changes to some games, particularly in puzzle games. However, it is not difficult for computers to simulate all mechanical impacts of such small changes. This suggests that computers might be able to aid humans designers as they build and analyze game levels. This paper takes one step towards this larger goal by studying how Exhaustive Procedural Content Generation (EPCG) can be used for analysis of incremental changes of existing game levels. Using an incremental EPCG approach, we analyze all of the levels in the popular puzzle game Snakebird, showing that incremental variations in the level designs can significantly increase the length of the shortest possible solution. A user study on a subset of these modified levels shows that the modified levels are both interesting and challenging for humans to play. Thus, through the analysis of Snakebird, we demonstrate the broader potential for incremental applications of EPCG.

1 Introduction

It is an expensive and time-consuming process for humans to generate and tune the large amount of novel content available in games. This is partly because it is difficult to understand the impact that small changes in design will have on gameplay (Korhonen 2010; Griesemer 2010).

One proposed solution to reduce this cost is procedural content generation (PCG). PCG is the practice of producing game content via algorithmic processes. There has been a great deal of research investigating potential applications of PCG (Shaker, Togelius, and Nelson 2016; Liapis 2020). However, the majority of this work has focused on a relatively small set of methods and games, with the vast majority of academic PCG work focused on search-based PCG (SBPCG) (Togelius et al. 2010) and platformer level generation, specifically *Super Mario Bros.* levels (Shaker et al. 2011; Liapis 2020). Comparatively, the vast majority of indie and industry PCG practitioners draw on constructive PCG (CPCG) approaches, such as constructive grammars (Short and Adams 2017).

It is still challenging for non-experts to create high-quality PCG systems from scratch. This is due to the authoring effort involved in creating a PCG algorithm. This has lead to increased interest in user-friendly PCG tools and so called mixed-initiative or co-creative systems. In these systems a human designer interacts with an existing PCG algorithm (Liapis, Smith, and Shaker 2016) to produce novel content. However, most academic mixed initiative systems also rely on SBPCG (Charity, Khalifa, and Togelius 2020).

SBPCG approaches such as genetic algorithms typically make large changes to the input population in each step because the core operators in a genetic algorithm (crossover and mutation) are stochastic. Thus, these approaches are better suited for applications where broad sampling is more important than methodical exploration of a problem space. Comparatively, CPCG approaches typically build a piece of content in one shot, which makes adding user control during the generation process nontrivial.

With a larger goal of broadening the number of techniques that can be used for assisting in game development, this paper studies techniques for the methodical exploration of puzzle game level designs. We focus on the impact of incremental design changes to an existing design, something that SBPCG and CPCG methods are not well-suited to model.

This project began with the hypothesis that small design changes could have a significant impact, particularly in puzzle games. Investigating this hypothesis, the paper makes the following contributions. First, it provides the first study of the game *Snakebird*, demonstrating how Exhaustive PCG (EPCG) (Sturtevant and Ota 2018) can be applied to the game. Second, it demonstrates that small design changes can significantly impact the solution length of puzzles in the game. Finally, a user study is then used to demonstrate that the suggested design changes result in levels that are more interesting to human players. Overall, this work provides the foundation for future applications of EPCG in mixed-initiative and co-creative research and broadens our understanding of the utility of this underexplored type of PCG.

2 Related Work

Procedural content generation represents a class of approaches for the automated creation of game content (To-

gelius et al. 2011). The majority of existing academic PCG work focuses on platformer games (Shaker et al. 2011; Liapis 2020), however some work exists in applying PCG to puzzle games. Khalifa and Fayek (2015) compared constructive and search-based PCG for Sokoban level generation, finding that both approaches performed roughly equivalently. Bento, Pereira, and Lelis (2019) showed how to create Sokoban levels that were both hard and guaranteed to be solvable. Smith, Butler, and Popovic (2013) employed answer set programming, an example of constraint-based PCG, to generate *Refraction* puzzles that were guaranteed to fit specific design constraints. One common domain for PCG in puzzle games is *Angry Birds*, which has had a wide variety of PCG approaches applied to it (Ferreira and Toledo 2014; Stephenson and Renz 2016; Jiang, Harada, and Thawonmas 2017). Because the game is a physics-based game, small changes to the level design are less likely to greatly impact the puzzle solution.

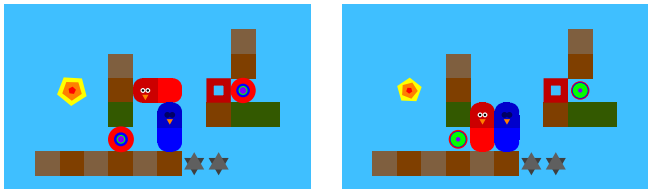
Outside of full level generation, there are mixed-initiative approaches in which a human works with an existing PCG approach to produce content (Karavolos, Bouwer, and Bidarra 2015). There has been very little research into the ways puzzle games might benefit from mixed-initiative PCG. However, *Angry Birds* has seen an example of SBPCG mixed-initiative level design (Campos et al. 2017). Recently, Charity et al. (Charity, Khalifa, and Togelius 2020) employed a SBPCG approach for mixed-initiative design of *Baba is You* levels. Other mixed-initiative approaches for non puzzle games exist, which produce variations on a designer's current level (Liapis, Smith, and Shaker 2016; Powley et al. 2016; Campos et al. 2017). This work parallels our own with Snakebird, focused on modifying levels rather than producing entirely new content or content based on user input. However, as we demonstrate below, EPCG allows us to make specific, minimum changes that would not be possible with other SBPCG approaches.

This paper builds on Exhaustive Procedural Content Generation (EPCG) (Sturtevant and Ota 2018), an approach that uses a generator to exhaustively build content from which an evaluator selects the best content, but one that has also been used in other fields (Gil-Gala et al. 2020). In this work we use EPCG incrementally to create modifications of full levels in Snakebird, laying the framework for future work building mixed-initiative and co-creative systems using EPCG.

3 Snakebird Domain

Snakebird is a 2015 game by Noumenon Games, with a 2019 follow-up called Snakebird Primer. The goal of each level is to have one or more snakebirds eat all of the fruit in the level and then for all snakebirds to leave via the exit.

An example level before and after an action is taken is shown in Figure 1. This level requires the use of many of the mechanics in the game, including multiple snakebirds (red and blue), spikes (6-sided gray stars), the exit (the yellow and orange hexagon), movable blocks (the hollow red square), and portals (the colored circles), although it does not contain fruit. At a surface level the game is similar to many other snake games, where eating fruit (seen at the top of Figure 3) makes a snake longer. However, in Snakebird



(a) (b)

Figure 1: Example Snakebird Level

there is also gravity, which causes snakebirds to fall back to the ground if they are not supported by any objects. In a player action a single snakebird can move a single step in any unblocked direction, after which gravity is applied. Figure 1(b) shows what happens when the red snakebird in Figure 1(a) moves up. After moving up one step, gravity is applied, and the red bird falls back down onto the platform.

Snakebirds can push objects or other snakebirds, assuming there is no solid object blocking the way. Falling onto a spike will kill a snakebird. Both snakebirds and objects can teleport between portals, as long as they will not be inside another object after teleporting. The level in Figure 1 is difficult because when the snakebirds try to directly reach the goal, they are teleported to the upper platform by the portal. One possible solution to the level requires placing the block on the upper portal, as this will prevent the snakebirds from being teleported to the upper platform, after which they can move past the portal to the goal.

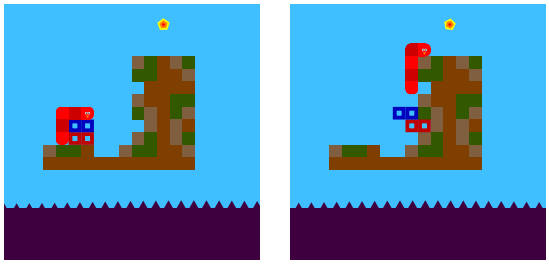
Snakebird has 46 regular levels, 6 difficult (star) levels, and 1 very difficult (black hole) level. Snakebird Primer has 69 regular levels, 6 star levels and 1 black hole level. All together there are 129 levels, of which 108 can be solved easily (see Section 5). Of these, the Snakebird levels have an average optimal solution length of 37, while the Snakebird Primer levels have an average optimal length of 24.

4 EPCG For Incremental Design

EPCG describes approaches for generating procedural content where "all possible content is methodically generated and evaluated" (Sturtevant and Ota 2018). The basic approach requires a generator G and an evaluator E. The evaluator is applied to all generated content, and the content with the highest evaluation is selected.

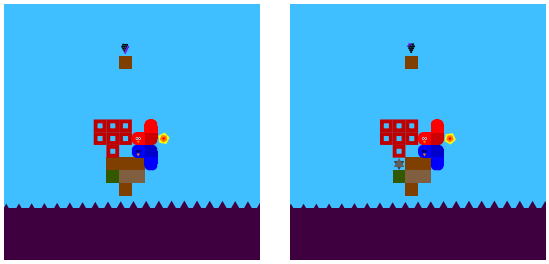
EPCG was originally evaluated with generators that built complete levels from scratch. But, for a game like Snakebird there are too many possible levels to generate all of them. Even a 20×20 level with three terrain types (sky, ground, and spikes) results in $3^{400} = 7 \times 10^{190}$ possible levels. But, this significantly underestimates the total number of possible levels when other gameplay elements are considered. In the game The Witness, there are similarly large numbers of levels if all puzzle sizes and types of constraints from the game are taken into account. In that context, previous work applied EPCG to exhaustively generate subsets of content for the game (Sturtevant 2019), generating all levels of a particular size that use a particular mix of constraints.

Because our ultimate goal beyond this paper is to build



(a) (b)

Figure 2: Solver insights



(a) (b)

Figure 3: Largest incremental change

a mixed-initiative co-creative system that can assist human designers, our approach is to exhaustively consider small changes to an existing level. This is both because the computational costs are smaller and because small changes are more easily understood. In the context of this paper we analyze levels from the original games, but these could very easily be changes to a level being co-authored with a human user in a mixed-initiative co-creative system.

We believe that it is difficult to understand the implications of complex dynamics, especially early in the design stages of a game. Thus, it is possible that level designers may not be aware of the implications that small changes to their designs might have. Because computers are well-suited for rote exploration tasks, our hypothesis at the beginning of this project was that only small changes would be required to significantly change the solution to a puzzle, and that such changes would be both non-obvious and useful to designers. If our hypothesis is true, we should be able to use EPCG to find interesting changes to existing levels.

Formally, our generator G, works as follows. It goes through every tile in the level that is not occupied by the snakebirds, the exit, a portal, fruit, or a movable object. These tiles are either sky, ground, or spikes. It then generates every 1-step change to a level, where a single tile is changed to one of the two differing tile types. Thus, the generator in an $m \times n$ level will make exactly $2(m \times n - |S| - |P| - |F| - |O| - 1)$ calls to the evaluator, where S is the set of cells occupied by snakebirds, P is the set of cells occupied by portals, F is the set of cells occupied by fruit, and O is the set of cells occupied by other objects. The exit occupies a single cell.

Our evaluator E uses a breadth-first search (BFS) to explore a level and return the optimal solution length.[1] We use this as a proxy for how interesting a player might find a modified level. Other research has developed better metrics for Sokoban (Jarušek and Pelánek 2010; Bento, Pereira, and Lelis 2019), but our results will confirm that players found levels with longer solutions more interesting in Snakebird.

In the final step, the EPCG procedure can either select the generated content which minimizes or maximizes the solution length, which in our case makes the puzzle respectively easier or harder.

[1]An A* search could be more efficient, but the BFS was sufficient for most levels in the game.

There are two possible ways this approach could fail. First, we could analyze levels from the game and discover that there are no modifications that significantly increase or decrease the lengths of the solutions to the levels. Second, we could find possible modifications, but they might not be interesting to human players. The success and failure of the incremental approach used in this paper will be a feature of both the game evaluated and the incremental approach to design. Our results will show that both of these failures are avoided in Snakebird. Experimental results will show that the incremental design changes proposed to Snakebird levels have a significant impact on the solution length and are interesting to humans.

After using EPCG to generate incremental changes to levels, we then looked at the resulting levels. While there are many interesting results, we highlight two levels here.

Solution Analysis We begin by looking at level 39 from Snakebird in Figure 2(a). The design of this level, and our own solution, suggests that the two blocks should be put in the two notches in the structure on the right-hand-side of the level so the snakebird can support itself on them and reach the exit. Looking at a walkthrough provided on YouTube,[2] this is the suggested solution to the level. However, the EPCG analysis suggested putting another block at the top of this structure. Directly running the solver on the level revealed to our surprise, as shown in Figure 2(b), that it is possible to exit the level without placing the second block into the second notch.

We have not spoken to the designers of the game, so we can only speculate about this level. But, evidence suggests that most people do not notice that the top of the level can be reached after placing just the first block. This suggests that it is easy to overlook design consequences, especially because adding another row of blocks on the top of the structure would easily remove this shorter solution.

Maximal Change In Figure 3 we show the level that had the largest absolute increase in solution length. The original level is shown in Figure 3(a), with the modified level in Figure 3(b). By changing the ground under the object to be a spike instead of ground, the minimal solution length is increased from 27 to 58, an increase of 31 moves. In the original level the challenge is to get the first snakebird to lift the

[2]https://www.youtube.com/watch?v=PITJZnLW2ug

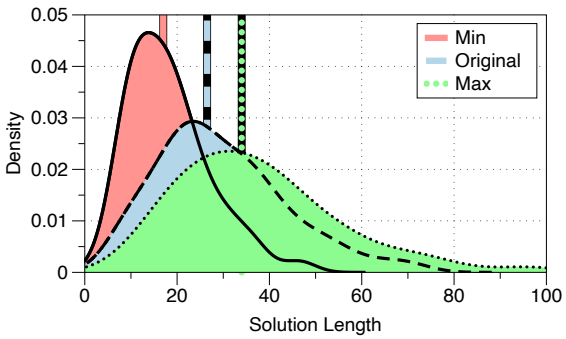


Figure 4: Comparison of the three distributions of solution lengths.

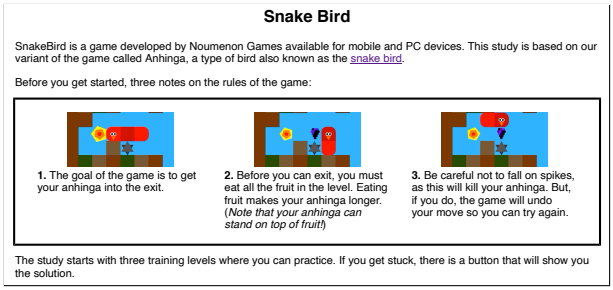


Figure 5: Study landing page.

object from underneath while the second snakebird is raised on top to reach the fruit.

The EPCG change to the level preserves the original challenge of the puzzle while adding an additional challenge. Now, the object must be moved from the left to the right side of the platform before the original solution can be used. The spike makes this particularly difficult. The enhanced level could be either used as is, or could be broken into two separate levels, each with a different challenge.

This level, along with several other interesting levels, are playable online from http://movingai.com/snakebird.html, and a demo of the EPCG system is also available (Sturtevant et al. 2020). The full source code for the entire project is available as part of the HOG2 code environment (https://github.com/nathansttt/hog2/).

5 Quantitative Analysis

To begin, we evaluate our hypothesis that there exist small design changes that significantly impact Snakebird levels, measured according to the minimum solution length. In this section, we present a quantitative analysis of the EPCG level modifications to evaluate this hypothesis. We produced two modifications for each Snakebird level, employing EPCG to find the one change that most increased solution length and the one change that most decreased the solution length. We ran this process on all 129 levels of the game, but terminated the evaluator when the BFS required more than 1 million state expansions, giving us results for 108 levels.[3] The final result is three sets of levels: the originals, the variations of each level by maximizing the solution length, and the variations of by minimizing the solution length. We refer to these sets of levels as originals, max, and min respectively.

A visual comparison of the optimal solution length for the three distributions of levels is found in Figure 4. The x-axis is the solution length and the y-axis is what percentage of the distribution has that particular solution length, while the vertical lines indicate the medians of each distribution. Visually, we can already identify a clear divide between the three distributions.

However, a visual analysis does not offer sufficient support for our hypothesis. To do this, we ran a statistical analysis to see if these distributions differed significantly from one another. First, we confirmed that the distributions of solution lengths differed significantly from the normal distribution via the Shapiro test ($p < 0.01$). Given this, we then used a two-sample Wilcoxon test to show that the distribution of original and the min, and the original and the max levels differed significantly ($p < 0.001$). This indicates that the modified levels are substantially different than the original levels.

As a point of reference, suppose that the max approach was only able to increase the solution length of each level by 1, perhaps by placing a trivial obstacle in front of the snakebird. Our analysis shows that the resulting distribution would not pass the Wilcoxon test ($p = 0.5$). Thus, at least quantitatively, incremental changes can lead to significant differences in the length of the optimal solution for a level.

6 User Study

In Section 5 we quantitatively validated that incremental EPCG changes can lead to significantly different solution lengths. But, this does not guarantee that players will find these levels to differ significantly when playing them. To investigate this further, we ran a user study in which participants play pairs of original and modified levels. This was used to gather evidence as to whether the incremental changes impact a user's experience of a level.

6.1 Study Methodology

We advertised the study on social media, including Twitter, Slack, Facebook, and Discord. After clicking through a consent form participants were directed to a landing page for the study that had instructions and animated .gif images showing the dynamics of the game, as shown in Figure 5. We called our version of the game *Anhinga*, after a bird also known as the *snakebird*. Participants were required to play three training levels, and then were assigned two pairs of test levels, with the modified and original levels given to the user in random order. If an individual participant could not solve a particular level they could use an animated solver to see a solution. After playing both pairs of levels, participants were asked to complete a brief survey that asked the participant to rank the pairs of levels in terms of which was

[3] A higher limit, a better heuristic, and/or symmetry detection should solve 19 of the remaining levels, while two levels have more objects than our engine supports.

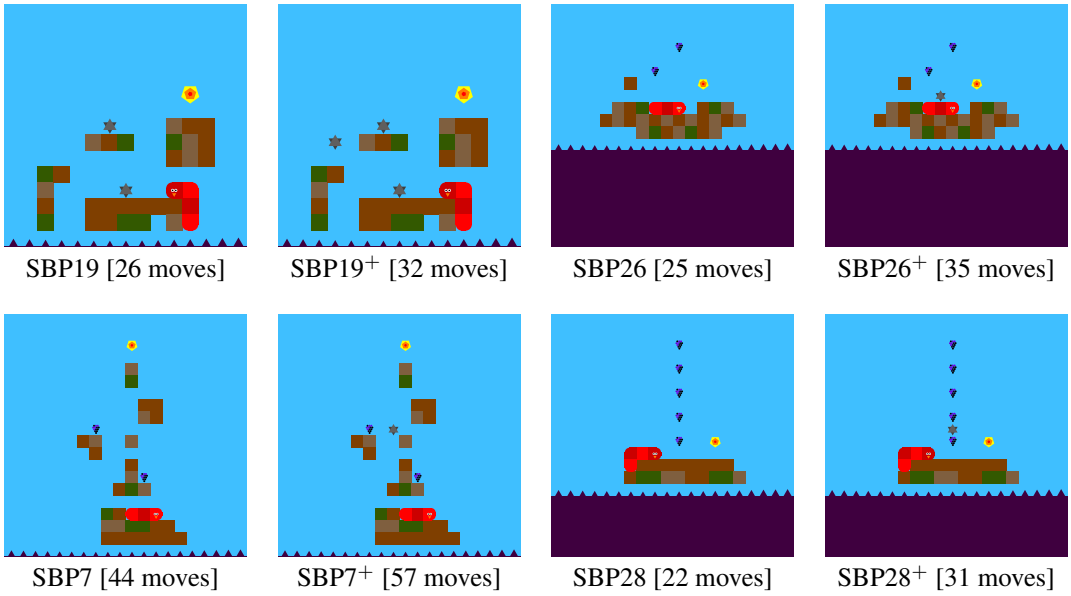


Figure 6: Levels from Snakebird Primer used in our user study. The $^+$ indicates a level from incremental EPCG analysis.

more fun, frustrating, surprising, interesting, enjoyable, and challenging. We chose to employ rankings instead of ratings as we were specifically interested in comparing the original and modified levels. We also asked each participant to pick the "best" level overall. Finally, we asked for demographic information in terms of gender, age, how often the participants played and designed games, and whether or not the participants had played Snakebird before the study.

6.2 Level Selection

We selected two sets of levels for the study. Training levels were selected to bring players up to speed with the game and teach the core mechanics. Testing levels were selected for study purposes. For brevity we refer to levels from Snakebird as SBx, and levels from Snakebird Primer as SBPx.

We used introductory levels from the game as training levels. These included SB0 and SPB4. We then modified SBP5 to illustrate to players that they could stand on fruit and how spikes worked.

To select levels for the study, we looked at the results of the analysis in Section 5. After looking at the changes and considering the mechanics of the games, we chose four pairs of levels for the study: SBP7, SBP19, SBP26, and SBP28, as shown in Figure 6, along with their EPCG modified counterparts. We chose these levels because they all had simple mechanics: a single snakebird and no portals or objects. Additionally, the single change suggested for each of these levels was to add a spike. Overall, we expected the levels would be simple enough for users who had not played the game before to solve.

6.3 Results

In our study, 127 individuals loaded the main study page, with 89 participants completing at least one level, and 67 participants completing the final survey. However, due to variations within the playtests, such as recording multiple plays of a single level, we only collected usable results from 58 participants. 44 of these subjects self-reported as male, 12 as female, and the remaining listed their gender as other or declined to report. Our participants trended toward the younger end of the spectrum, with twenty-five of our participants falling within 18-25, twelve within 25-30, and fifteen within 31-40. In terms of level design experience, 27 of our participants reported having never designed a level, 14 reported having done so once, 16 reported having done so "many" times, and one self-reported as being a level design expert. Since over half of our participants had level design experience, we can expect that this population should have been well-suited to evaluating levels.

Perhaps unsurprisingly, our participants had a major bias toward regularly playing games, with 37 reporting that they played games daily and 11 reporting playing at least once a week. However, our participants had less experience with puzzle games, with only seven reporting playing puzzle games daily, 20 reporting at least weekly, and 15 reporting at least monthly. Similarly, only seven participants reported having played Snakebird before this study.

The first step for our results was to determine the impact of the various conditions and demographic information. Using a multi-way ANOVA or MANOVA (Huberty and Olejnik 2006), we determined there was no significant impact on any of the ranking results from any of our demographic information. Further, there was no significant impact on the results based on the order that the participants encountered the pairs of levels (original and modification). However, there was a significant impact ($p \ll 0.05$), on the ranking results based on the order in which users encountered the parts of each pair. In other words, whether a user would rank either the original or modification higher according to the different subjective features (fun, frustrating, etc.)

	Fun	Frustrating	Surprising	Interesting	Enjoyable	Challenging
EPCG Variation %	48.3%†	61.7%	48.3%†	61.7%	55.0%†	63.3%

Table 1: The percentage of time the incremental EPCG variation was ranked first in terms of the given features. †Not significant.

depended on what order they encountered the levels. This is unsurprising given the recency bias (Summerfield and Tsetsos 2015). Whether or not the level being ranked was an EPCG modification or original also had a significant impact ($p \ll 0.05$). This indicates that our changes had a perceptual impact on user experience, which we delve into in more detail below. However, we found that particular individual levels had no significant impact on the ranking results. In other words, each level was individually roughly as likely to be ranked first or second for each feature. This indicates that it is more fruitful to consider the levels in the aggregated groups of originals and modifications instead of examining the specific changes made to particular levels.

We next needed to determine the impact of whether a level was an original or modification on the different subjective experience features: which level was more fun, frustrating, surprising, interesting, enjoyable, and challenging. This would tell us what general impacts, if any, came from the EPCG modifications. Given that the ordering within pairs of original and modified levels had a significant impact on these rankings, we randomly sampled a subset of our results where there was an equal number of instances of both kinds of levels occurring first and second. This new dataset had 60 pairs of original and variation levels, for 120 total comparisons. We note that this removed a bias in the original data where more users saw the modified level before the original level. One possible reason for this bias was participants quitting when they saw the easier level (original) before the more difficult variant. The percentage of time that the variation was ranked first, according to each experiential feature, is found in Table 1.

We ran the paired Wilcoxon signed-rank test, which is appropriate given that our data is in pairs and is ordinal but non-numeric (ranking information). We found no significant difference in the results for the features of fun, surprising, or most enjoyable. In other words, when the effect of ordering was neutralized, there was no significant difference in how players ranked the original and modifications in terms of these features. However, we found that users were significantly more likely to rank the variation as being more frustrating, more interesting, and more challenging than the original ($p < 0.05$). We also found that users were much more likely to rate a variation level as the best level overall, as 65% of the best levels in the final dataset were variations ($p \ll 0.05$). This gives support to the supposition that even outside of human guidance that EPCG can be successfully applied to Snakebird levels to produce variations that are more challenging and interesting to human players. However, we anticipate that human expertise is needed to help mitigate frustration with difficult puzzles, whether by level sequencing or by other design clues. Many Snakebird Primer levels, for instance, have the solution visually encoded in the level.

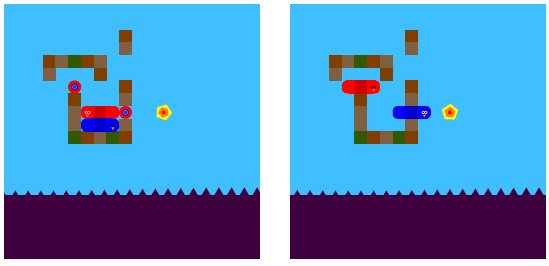
7 Discussion and Limitations

It is important to qualify that we are not claiming that any of the levels used in our study are *better* than the original levels when placed in the context of the full game. Game designers carefully design progressions of levels to teach users about mechanics and to challenge them in new ways. Thus, any single level in a game typically serves a design purpose within the larger progression of the game. The changes suggested by the incremental EPCG may help a designer explore new ideas during design, that will then be placed in the larger framework of the game.

Similarly, our evaluator works to maximize solution length, but solution length may not always be the most important thing to maximize. In Snakebird level 37, shown in Figure 7(a) we typically use a straightforward solution which requires 19 moves. The key state in this solution is in Figure 7(b). Looking at an online walkthrough for the game on YouTube, the same solution is provided. There is, however, a less straightforward solution, involving non-intuitive use of the portals, which only requires 17 moves. There is a trivial modification to the original level which will disallow the 19 move solution and only permit the 17 move solution, making the level more challenging. This change is not selected by the EPCG evaluator because it does not increase the minimum solution length. Thus, EPCG could still be missing interesting level changes based on human difficulty. That being said, our own understand in this example is a direct consequence of using the EPCG analysis.

8 Conclusions

This paper began with the hypothesis that there exists some games where incremental design changes can have significant impacts on the experience of playing a level, and that



(a) (b)

Figure 7: Solution length is not always indicative of solution complexity.

EPCG can be applied to find these changes. We demonstrated strong support for this hypothesis in the game of Snakebird through both a quantitative analysis of these changes and a user study. Thus, we anticipate that these results can be generalized to puzzle games outside of Snakebird. There is clear application to puzzle games where EPCG has been previously applied, such as in The Witness. But, we expect that incremental analysis may be insightful in many other grid-based puzzlers such as Cosmic Express, Causality, or Dissembler.

More broadly, this work has laid the framework for using EPCG with mixed-initiative co-creative design, in which a user can work creatively the EPCG system to design levels. Future work will further develop this possibility.

Acknowledgements

This work was funded by the Canada CIFAR AI Chairs Program. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Bento, D. S.; Pereira, A. G.; and Lelis, L. H. 2019. Procedural generation of initial states of sokoban. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 4651–4657. AAAI Press.

Campos, C.; de Oliveira Sá, W.; Teixeira, J. M. G.; and Lelis, L. 2017. Mixed-initiative tool to speed up content creation in physics-based games. *SBGames* 2017:590–593.

Charity, M.; Khalifa, A.; and Togelius, J. 2020. Baba is y'all: Collaborative mixed-initiative level design. *arXiv preprint arXiv:2003.14294*.

Ferreira, L., and Toledo, C. 2014. Generating levels for physics-based puzzle games with estimation of distribution algorithms. In *Advances in Computer Entertainment Technology*, 1–6.

Gil-Gala, F. J.; Mencía, C.; Sierra, M. R.; and Varela, R. 2020. Exhaustive search of priority rules for on-line scheduling. In *European Conference on Artificial Intelligence (ECAI)*.

Griesemer, J. 2010. Design in detail: Changing the time between shots for the sniper rifle from 0.5 to 0.7 seconds for halo 3. Game Developer Conference.

Huberty, C. J., and Olejnik, S. 2006. *Applied MANOVA and discriminant analysis*, volume 498. John Wiley & Sons.

Jarušek, P., and Pelánek, R. 2010. Difficulty rating of sokoban puzzle. In *Starting AI Researchers' Symposium (STAIRS 2010)*, 140–150.

Jiang, Y.; Harada, T.; and Thawonmas, R. 2017. Procedural generation of angry birds fun levels using pattern-struct and preset-model. In *Computational Intelligence and Games (CIG)*, 154–161. IEEE.

Karavolos, D.; Bouwer, A.; and Bidarra, R. 2015. Mixed-initiative design of game levels: Integrating mission and space into level generation. In *Foundations of Digital Games (FDG)*.

Khalifa, A., and Fayek, M. 2015. Automatic puzzle level generation: A general approach using a description language. In *Computational Creativity and Games Workshop*.

Korhonen, H. 2010. Comparison of playtesting and expert review methods in mobile game evaluation. In *International Conference on Fun and Games*, 18–27.

Liapis, A.; Smith, G.; and Shaker, N. 2016. Mixed-initiative content creation. In *Procedural content generation in games*. Springer. 195–214.

Liapis, A. 2020. 10 years of the pcg workshop: Past and future trends. In *FDG Workshop on Procedural Content Generation*.

Powley, E.; Gaudl, S.; Colton, S.; Nelson, M.; Saunders, R.; and Cook, M. 2016. Automated tweaking of levels for casual creation of mobile games. In *Computational Creativity and Games Workshop*.

Shaker, N.; Togelius, J.; Yannakakis, G. N.; Weber, B.; Shimizu, T.; Hashiyama, T.; Sorenson, N.; Pasquier, P.; Mawhorter, P.; Takahashi, G.; et al. 2011. The 2010 mario ai championship: Level generation track. *Transactions on Computational Intelligence and AI in Games* 3(4):332–347.

Shaker, N.; Togelius, J.; and Nelson, M. J. 2016. *Procedural content generation in games*. Springer.

Short, T., and Adams, T. 2017. *Procedural generation in game design*. CRC Press.

Smith, A. M.; Butler, E.; and Popovic, Z. 2013. Quantifying over play: Constraining undesirable solutions in puzzle design. In *Foundations of Digital Games (FDG)*, 221–228.

Stephenson, M., and Renz, J. 2016. Procedural generation of complex stable structures for angry birds levels. In *Computational Intelligence and Games (CIG)*, 1–8. IEEE.

Sturtevant, N. R., and Ota, M. J. 2018. Exhaustive and semi-exhaustive procedural content generation. In *Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 109–115.

Sturtevant, N. R.; Decroocq, N.; Tripodi, A.; Yang, C.; and Guzdial, M. 2020. A demonstration of anhinga: A mixed-initiative epcg tool for snakebird. In *Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*.

Sturtevant, N. R. 2019. Exploring epcg in the witness. In *Knowledge Extraction from Games (AAAI workshop)*, 58–63.

Summerfield, C., and Tsetsos, K. 2015. Do humans make good decisions? *Trends in cognitive sciences* 19(1):27–34.

Togelius, J.; Yannakakis, G. N.; Stanley, K. O.; and Browne, C. 2010. Search-based procedural content generation. In *European Conference on the Applications of Evolutionary Computation*, 141–150. Springer.

Togelius, J.; Yannakakis, G. N.; Stanley, K. O.; and Browne, C. 2011. Search-based procedural content generation: A taxonomy and survey. *Transactions on Computational Intelligence and AI in Games* 3(3):172–186.