Promoting Human Collaboration in Procedural Content Generation Tools for Game Development

Benjamin P. Burgess
burge337@morris.umn.edu
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

Abstract

The question being addressed in this research is whether mixed-initiative (human-computer collaboration) implementations of procedural content generation in games can aid in their creation. This paper argues that a mixed-initiative approach can potentially alleviate some strain from the generation of content in iterative processes as well as offer novel approaches to design. This proposal is supported by three papers [1, 3, 4] investigating the opinion of people who had collaborated with an algorithm on how their design process was affected by their collaboration. It also includes an analysis of how mixed-initiative approaches can be implemented via the Evolutionary Dungeon Designer's Feasible-Infeasible Two Population Genetic Algorithm.

Keywords: mixed-initiative, procedural content generation, genetic algorithms

1 Introduction

Creating video games is a difficult and time-consuming process. It often involves creating and testing numerous prototypes for game features, level designs, music, and more. In an effort to combat these difficulties, there have been attempts to automate the development process. As with many other creative endeavors however, the development process is often based on heuristics rather than well defined rules or guidelines. As such, automating these tasks has proven to difficult. The evaluation of game quality is a particularly challenging effort due to the nebulous guidelines used to determine said quality. It is so challenging in fact, that for an algorithm to accurately mimic the human evaluation process for games would require an AI-complete solution[7] (an algorithm that is as smart as a human) which we don't yet have.

This paper explores the use of mixed-initiative implementations in procedural content generation tools to combat this very issue. We will first go over two examples of Procedural Content Generation (PCG) approaches and examine them through the lens of Liapis' six game facets. We will then provide a brief overview of genetic algorithms to provide the required background knowledge for a series of studies done by Baldwin et al. on their mixed-initiative PCG (MIPCG) tool: The Evolutionary Dungeon Designer. We will then go over their findings from user studies on the perceived usefulness of such a system and discuss the implications of these findings. Then there will be discussion of Guzdial et al.'s MIPCG tool and a review of their findings from their user studies.

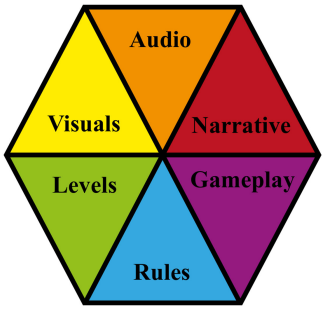


Figure 1. The six facets of game design [7]

2 Background

2.1 Overview of Procedural Content Generation in Games

Procedural content generation in games refers to the automated generation of game content such as levels, visuals, and music via an algorithm. While PCG has been implemented in games as long ago as 1981's Rogue, and with greater mainstream popularity of the Rogue-like genre occurring in the early 2000's, its prominence in the academic community is fairly recent [6]. The target of PCG in games varies in scope, from more ambitious attempts to automate the generation of entire games, to automating singular facets of games such as level designs or music, or even designing algorithms that generate content alongside people in a mixed-initiative effort.

2.2 Six Facets of Game Design

In part of one such ambitious effort to automate the generation of entire games, Liapis defines six facets of game design: audio, narrative, gameplay, rules, levels, and visuals (see Figure 1). These six facets help clarify the target of PCG in games and highlight the interactions between the algorithm(s) responsible for each facet. Liapis proposes two different approaches to automating game design: a top-down

approach where some algorithm is responsible for orchestrating the function of the other generative algorithms, and a bottom-up approach where each generative algorithm builds and iterates upon its created content using some form of shared memory with all the other generative algorithms. Both of these approaches share a significant weakness: evaluating the coherence and quality of generated content, even in single facets of design, is nearly impossible to do as well as people can, without an AI-complete solution [7].

2.3 Analysis of Example PCG Games Through the Six Facets

The automated generation of games being a problem of AI-complete difficulty suggests a need for human intervention during their creation and/or evaluation, and in fact many attempts at incorporating PCG in games have incorporated just that. Angelina —a piece of software that generates simple platformer games (games where you navigate through various obstacles, often by running and jumping)— incorporates this human intervention element asynchronously by scraping information from online sources to supply image backgrounds and sound bytes based on the contents of internet articles (see Figure 2). While this doesn't solve the issue of coherence evaluation for the game as a whole, individual features of each facet that were created by humans should be coherent. The in-game screenshot from Figure 2 highlights this disparity, as an image of a child's face is covering up a large portion of the level.

Figure 2. Facet orchestration diagram with in-game screenshot of Angelina with narrative, audio, and visuals being pulled from online sources. Arrows between facets indicate the influence of one facet's content on the generation of another's. Levels are procedurally generated, represented by the computer pointing to that facet. [7]

Game-O-Matic —an AI based game generator that creates games where "simulation meets political cartoons"—[7] takes another approach to facet orchestration. In this setup, human-authored relationships between different entities are used to generate the rules, visuals, and level layout of the game (See Figure 3). The relationships are referred to as microrhetorics or concept maps, and are simply a directed graph connecting the entities in the relationship through verbs.

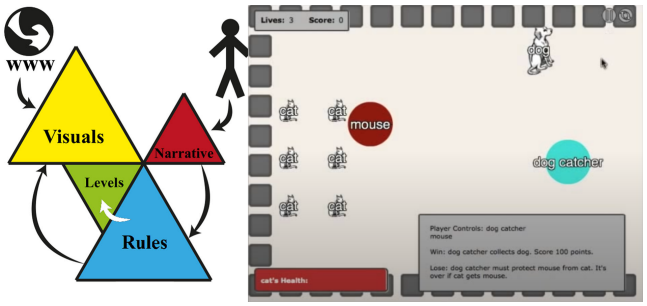


Figure 3. Facet orchestration diagram and in-game screenshot for Game-O-Matic with human generated narrative informing generation of rules and levels, and visuals being scraped from the internet in accordance with game rules. [7]

These examples highlight Liapis' observation about coherence evaluation. While individual, human-created pieces of content within these games are consistent, their overall aesthetic is either simplistic or incoherent. This suggests that PCG in its current form may be best utilized in mixed-initiative capacities as a means of providing quality control until significant advancements are made to computer-based coherence evaluation algorithms. An example of such a program based on genetic algorithms will be discussed later in the paper.

2.4 Genetic Algorithm Basics

Genetic Algorithms (GAs) seek progressively better solutions for a given objective without guarantee that optimal solutions will be found or even recognized [5]. The methodology behind this process emulates natural selection. Candidate solutions (often referred to as individuals) within a population are evolved toward better solutions by modifying one or more of their properties (often referred to as chromosomes or genotypes) either directly to an individual's property(s) via mutation or indirectly when two parent solutions are bred and pass down their chromosomes via crossover. These parents are typically individuals selected for the quality of their solution in the hopes that those desirable qualities pass down to their children. The overall quality of these individuals is referred to as their fitness, which is determined by the value of the objective function, while the fitness value of a specific property of an individual is measured by a fitness function. [8]

2.5 A Typical GA Run

A typical run of a genetic algorithm consists of several steps looped over and over until a desired solution is obtained:

- Initialization: Some number of possible solutions are generated (often randomly).
- Selection: A portion of the population (typically the most fit individuals) are selected to breed a new generation.

- Genetic Operators: The selected population are paired up and undergo some combination of mutation and crossover to produce children, some of which will be more fit than their parents.

2.6 Feasible-Infeasible Two-Population Genetic Algorithm Overview

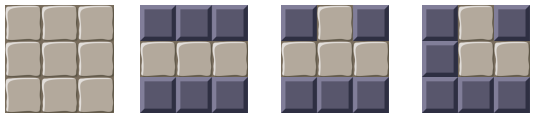
The FI-2PopGA works in essentially the same way, save for the use of two distinct populations that are maintained throughout generations of the algorithm. These populations are divided into categories based on how well they satisfy the problem constraints. Measures of an individual's solution quality are taken via fitness function to determine which category they belong in. The first population consists of candidate solutions that satisfy the problem constraints and are known as feasible individuals. The second population consists of infeasible individuals that fail to do so. These populations are then evaluated under different constraints. Individuals within a feasible population are evaluated according to objective function values that measure how well they follow the given constraints for the problem. This measure is also referred to as an individual's fitness score. Infeasible individuals instead have their measure of fitness evaluated with respect to a function of their constraint violations. They are penalized according to the severity of their constraint violations in order to remove particularly low fitness individuals from the population. In this way feasible individuals are selected to increase payoff while disregarding potential constraint violations, and infeasible individuals are selected with the goal of repair while disregarding potential payoffs.

3 The Evolutionary Dungeon Designer

The Evolutionary Dungeon Designer (EDD) is a pattern-based MIPCG tool where users design dungeon levels similar to those from the original Legend of Zelda for fantasy role playing games [2]. It extends the work of the Evolutionary World Designer, which generated world maps and dungeons for adventure games without giving enemies, treasures, and other such elements exact positions within them. EDD uses a feasible-infeasible two population genetic algorithm (FI-2PopGA) to include the placement of these elements within the dungeons it creates.

3.1 EDD Implementation

EDD creates dungeons that are an $M \times N$ grid of tiles that consist of six different types: enemy, floor, wall, door, treasure, and entrance, where the player can move through all tile types except walls. EDD generates suggestions for individual room layouts using the FI-2PopGA where two populations—consisting of feasible and infeasible rooms—are evolved in parallel via separate genetic algorithms. Feasible rooms are rooms that satisfy the playability constraint. They contain at least one treasure and enemy and allow access between the entrance and all other doors, treasures, and enemies, while infeasible rooms violate this constraint in some capacity. These populations are interbred in order to promote solution diversity within the overall solution space that contains all possible solutions for the FI-2PopGA. This space can be further subdivided into the feasible and infeasible solution spaces, which contain their respective solution type. By including the infeasible population we'll occasionally have infeasible solutions crossover into the feasible solution space and thus explore its outer border. This is particularly important for constrained optimization problems such as this that will have the optimal solution somewhere between the feasible and infeasible regions.



(a) Chamber **(b)** Corridor **(c)** Joint **(d)** Turn

Figure 4. Examples of micro-patterns detected by the generator [2]

Individuals from these populations are evaluated based on fitness scores derived from the quality of various micro and meso patterns [3]. An example of a function behind these fitness scores is seen in equation 1, where the quality (Q_{corridor}) of a given corridor (c) is a simple min function that returns 1 if the length of the corridor (Area(c)) is greater than or equal to the user desired corridor length, and $\frac{Area(c)}{T_{\text{corridorlength}}}$ if the length is less than desired, thus rewarding corridors that are at least as long as the user desires.

$$Q_{\text{corridor}}(c) = min(1.0, \frac{Area(c)}{T_{\text{corridorlength}}}) \tag{1}$$

Micro patterns are further subdivided into inventorial and spatial patterns, where the former are base tile types consisting of enemies, doors, and treasure tiles, and the latter are a combination of wall and floor tiles consisting of joints, chambers, turns, and corridors (see Figure 4). Meso patterns include ambushes, guard chambers, treasure chambers, dead ends, and guarded treasure, which are combinations of micro and meso patterns. Quality scores for these patterns are based on user-selected parameters for desired quantities and types of these patterns, which are then used in fitness functions to determine an individual's feasible or infeasible fitness score.

In addition to allowing the user to guide content generation through micro and meso pattern parameters, users are also able to directly edit rooms EDD generates via the room editing view (see Figure 5). Once a user has finished editing the room, they can then generate four new room suggestions using the FI2PopGA. Two of these suggestions are designed to mimic only the types of micro and meso patterns contained within the user-edited map, while the other two are

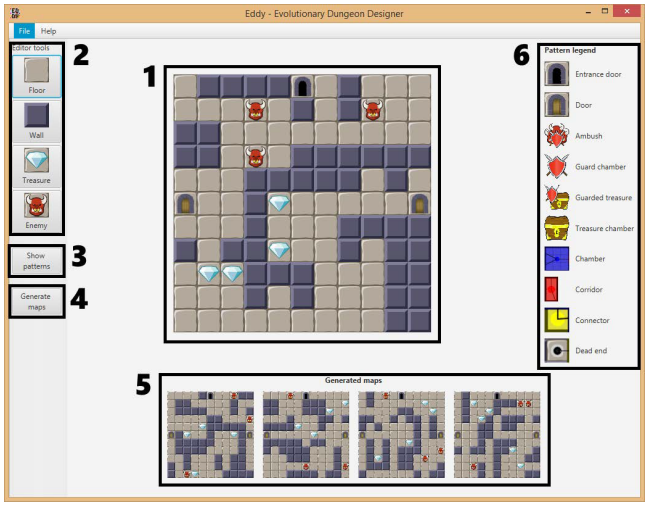


Figure 5. The Room Editing View [3]

intended to mimic user patterns as well as the approximate layout of the user map. The former two suggestions only include the user-edited map in the first generation of the genetic algorithm. The latter also does this, but also includes a mutation of the user-edited map within each generation. These mutations keep the individuals of these suggestion types more closely approximate to the current map. In this way, the system is allowed some freedom for novel generation of content while also closely preserving the user's intent should they want more control.

3.2 EDD User Study

A user study was conducted in order to determine the relevance of the mixed-initiative component as well as discover useful features for future iterations of the software [3]. The study had five people from the game development industry test the software in order to determine the effectiveness of the MIPCG component. They each designed three increasingly difficult 11×11 rooms that would be a part of the same dungeon level. Their levels were saved for analysis upon completion and participants took part in a structured interview.

Users generally believed EDD to be an interesting and useful tool for dungeon design, with only one of the five saying otherwise. Four out of five of them also considered EDD's pattern detection important, with the dead end detection capabilities being deemed especially so. There was friction however, where one participant attempted to design their level without adhering to the established design patterns and consequently the tool failed to recognize it. A similar complaint was voiced by two of the participants who wished to be able to modify existing design patterns, or even create entirely new ones. Four of the participants wished for a graph-like view of all the designed rooms in a dungeon to more easily parse how rooms are connected as a whole.

These findings suggest that while users find value in EDD's ability to save time by generating levels as an editing base, they desire more control over the design process even if it potentially hinders the generation of novel content.

3.3 EDD Improvements

A new iteration of EDD [1] was created in response to the findings from the aforementioned user study with the following improvements:

- Users can now create and edit a grid-based dungeon of varying dimensions with interconnected rooms where they previously could only edit a single room.
- Users receive more information about consequences of their alterations to individual rooms as well as the differences between the current room and the suggestions proposed by the algorithm.
- Navigation tools were added within and between views to provide an overview of the dungeon and provide better context of the edited room.
- The algorithm was updated to better allow the preservation of a designer's aesthetic by implementing locked sections of rooms and by extending the evaluation function to include measurements of symmetry and similarity in provided suggestions.

3.4 Follow-up User Study

A follow-up user study with five participants from the games industry was conducted to assess how the improvements affected the user experience while collaborating with the tool. This study had users view a brief demonstration of EDD to showcase its workflow during the creation of a 3×3 dungeon. The researchers then gave them ten minutes to create their own 3×3 dungeon. Their work was saved for later analysis and users went through a structured interview upon completion of the task. A questionnaire about their background in game design, dungeon-based games, and previous experience with mixed-initiative tools was also administered.

3.5 Study Findings

The researchers found their main goal of establishing a mixed-initiative interaction with flexible human and computer design roles was only partially achieved. They found that the room suggestions and whole dungeon navigation definitely supported user decision making during the design process, and while the users didn't feel restricted by the system, they still expressed a desire for more control over the EDD when designing their levels. Other user concerns included unclear design pattern visualization, a lack of usefulness for the world view, and doubts about the accuracy and sufficiency of information provided about generated room suggestions. In spite of these concerns however all five participants in the study found the new version of EDD to be

overall good and intuitive, and believed the generated map suggestions for each room to be the most helpful to their design process.

4 Morai Maker

Morai maker is another mixed-initiative level creation tool wherein a person works collaboratively alongside an AI agent to build Super Mario Bros.-like platformer levels [4]. This tool was used in a pair of studies to first investigate the development needs of the tool, and then later explore the ways that the tool impacted the behavior of practicing game designers.

4.1 Morai Maker Implementation

Morai Maker works with a user to create two dimensional platformer levels using three different AI partners: Markov Chain, Bayes Net, and a Long Term Short Term Memory Recurrent Network (LSTM), whose lower-level function details are outside the scope of this paper. Differences on their function within the context of Morai Maker are relatively simple however. The Markov Chain only looked at a 2x2 grid of level content, the Bayes Net examined a width of 16 grid points, and LSTM considered almost the whole level [4].

4.2 User Workflow

User interaction with the software takes place within the level editor interface (see Figure 6). Users can select blocks and place them within the level. They can place as many blocks as they like, and when they are satisfied with their changes they can press the 'End Turn' button to prompt AI level additions. The AI were not allowed to do anything other than add components to the level. These additions are added piece by piece to the current level and the camera scrolls to where the changes are taking place. All changes to the level are logged and tagged as either being created by the human or AI partner. The user is then able to resume editing the level, and the process can be repeated for as long as needed. Users can also test their levels by pushing the 'Run' button at any time.

4.3 Study 1: Investigating Design Considerations

Guzdial et al's initial study was run to derive design lessons about the interface as well as the AI system. They were interested in what impact the choice of AI algorithm would be on the user experience as well as getting feedback on their UI's usability. The study included 91 participants that took part, but only 84 followed the study guidelines correctly and interacted with their AI partner. Their age range was skewed younger, with 64 participants being in the 18-22 age range, 19 in the 23-33 range, and only one in the 34-55 range. While they were not game developers as in the previous studies, the vast majority of them had played Super Mario games before, with only seven of them having never been exposed to them before. Participants were given a short tutorial on the level editor and its function, and were allowed to view two examples of two level types from the original Super Mario Bros. Then they worked with two randomly assigned AI partners back to back to design two different levels. Participants were tasked with designing either an above ground or below ground level for each design session, where each session was capped at fifteen minutes. They were required to interact with their AI partner by pressing the 'End Turn' button at least once for each design session. They were then given a survey with questions about which of the two agents were preferred in terms of most fun, frustrating, challenging, helpful, which lead to the most surprising and valuable ideas, and which they would most want to use again.

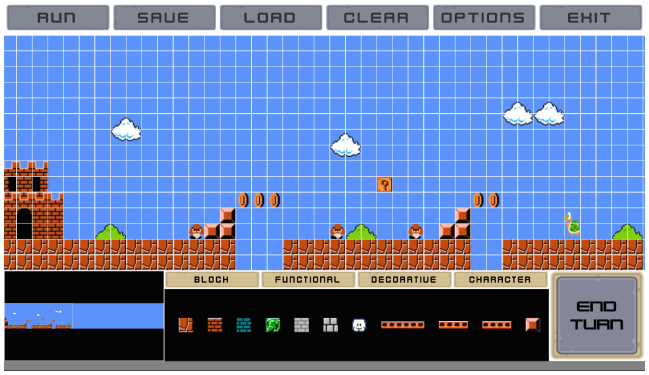


Figure 6. Screenshot of Morai Maker level editor [4]

4.4 Results

The results were split into three conditions based on the pair of AI partners interacted with. Each pair of partners was then ranked based on experiential features such as most fun, creative, etc. Ranking of these partners by experiential features was inconsistent, suggesting that none of the individual AI agents are significantly better creative partners and that it was simply a matter of individual preference (see [4] for more detailed information). Their rankings for most surprising and valuable partner highlight these inconsistencies as Bayes was ranked higher than LSTM, Markov higher than Bayes, but comparing Markov to LSTM found no significant relationship. The researcher's key takeaways are as follows:

- No single partner could meet all of the users' different expectations.
- Participants consistently created levels that were significantly different from traditional Super Mario Bros level structures.
- Participants didn't understand how their AI partners worked, but were willing to invent explanations for their behaviors.

4.5 Study 2: Morai Maker's Impact on User Process

A second study was run using an updated version of Morai maker that took into account the user feedback from the first study. This study targeted practicing, published game designers and sought to answer three different research questions:

- Does leveraging active learning to adapt the AI partner to a user allow a tool to better serve level designer needs?
- Can Explainable AI allow users a better understanding of the AI, and thus allow greater utilization of the tool?
- Will the overall changes lead to beneficial experiences for the designers?

The updated version of Morai Maker had a few core changes made. In response to the lack of user satisfaction with any of their prior algorithms, Guzdial et al. swapped to a more co-creative model of their system. In this version, they used a semi-Markov Decision Process with a three layer Convolutional Neural Network (CNN) as their AI agent that was trained on interactions of the 91 participants using the "Reuse" ranking (1 or -1) for the final reward. While the finer details of its function are beyond the scope of this paper, a CNN was chosen because it tends to focus on small, local features that are helpful in making global decisions, which seemed most promising based off of user feedback from the prior study. They also included a small negative reward if a human deleted an AI addition and a small positive reward if a human kept the changes in order to better adjust the agent to a user's preferences. These rewards were local to an addition's placement, so deletion of a particular tile type didn't mean that tiles of that type would be deleted outside of that part of the level. These changes effectively created a tool capable of adapting to user preferences during use by picking up on a user's local level structures.

This new study gave 14 game designers a full run-down of the tool and provided them time to ask any questions they had to study personnel. This included explanations of the AI. Participants again designed two levels using the tool, but this time they interacted with a single agent that was not reset between levels. They were tasked with creating either an above or below ground level each time. Participants were encouraged to voice their thoughts during the design process and asked the following questions at the session's conclusion:

- Did you prefer the AI behavior in the first or second session?
- Would you prefer this tool with or without the AI partner?
- Did you feel the agent was collaborating with you?
- Did you feel the agent was adapting to you?
- If you asked for explanations, did you find that they improved your experience?

	First	Second
Most Fun	5	9
Most Frustrating	8	6
Most Aided	5	9
Most Creative	5	9
Preference	6	8
	Yes	No
Collaborating	7	7
Adapting	9	5

Figure 7. User responses to survey questions on AI collaboration experience [4]

4.6 Results

Participant responses to the survey can be seen in Figure 7. While people tended to rate working with the more adapted AI more highly in terms of experiential ratings, it seems as though collaboration and adaptation rankings suffered. Qualitative answers to the aforementioned research questions were also gleaned from user remarks during their sessions. They found that the AI consistently adapted to the participants, and that people generally found the tool valuable in the design experience by providing inspiration for design ideas either unintentionally or not. Those who didn't find consistent value in the tool often complained about the seeming randomness of the AI decisions. The researchers did not find a meaningful answer to their question regarding the usefulness of an Explainable AI.

5 Conclusions and Future Work

With video games becoming increasingly popular and often more complex, the demand for tools to ease the development process will likely only increase going forward. In this paper we review a number of different approaches to these sorts of tools and find that an MIPCG approach has been viewed positively by a handful of game designers, and that MIPCG has the potential to alleviate some of the game development burden by generating content automatically.

In general, it appears as though MICPG has been well-received. Most users enjoyed mixed-initiative design efforts and found particular value in MIPCG tools as sources of inspiration. This is evidenced by positive user responses to EDD's generated map suggestions and user comments on Morai Maker as a useful source of inspiration for when they ran out of ideas. While these findings do highlight potential benefits of MIPCG, studies investigating user opinion on their value are quite rare and tend to have small sample sizes. Further research should consider running larger user studies to better understand how users would most benefit from collaboration with PCG tools.

Acknowledgments

I'd like to thank Kristin Lamberty for her advisement throughout the project and Elena Machkasova and Brian Mitchell for their feedback on project drafts.

References

[1] Alberto Alvarez, Steve Dahlskog, Jose Font, Johan Holmberg, Chelsi Nolasco, and Axel Österman. 2018. Fostering creativity in the mixed-initiative evolutionary dungeon designer. *Proceedings of the 13th International Conference on the Foundations of Digital Games*, 1–8. https://doi.org/10.1145/3235765.3235815

[2] Alexander Baldwin, Steve Dahlskog, Jose M. Font, and Johan Holmberg. 2017. Mixed-initiative procedural generation of dungeons using game design patterns. *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, 25–32. https://doi.org/10.1109/CIG.2017.8080411

[3] Alexander Baldwin, Steve Dahlskog, Jose M. Font, and Johan Holmberg. 2017. Towards pattern-based mixed-initiative dungeon generation. *Proceedings of the 12th International Conference on the Foundations of Digital Games*, 1–10. https://doi.org/10.1145/3102071.3110572

[4] Matthew Guzdial, Nicholas Liao, Jonathan Chen, Shao Yu Chen, Shukan Shah, Vishwa Shah, Joshua Reno, Gillian Smith, and Mark O. Riedl. 2019. Friend, collaborator, student, manager: How design of an AI-driven game level editor affects creators. *Conference on Human Factors in Computing Systems - Proceedings*. https://doi.org/10.1145/3290605.3300854

[5] Steven Orla Kimbrough, Gary J. Koehler, Ming Lu, and David Harlan Wood. 2008. On a Feasible-Infeasible Two-Population (FI-2Pop) genetic algorithm for constrained optimization: Distance tracing and no free lunch. *European Journal of Operational Research* 190 (10 2008), 310–327. Issue 2. https://doi.org/10.1016/j.ejor.2007.06.028

[6] Antonios Liapis. 2020. 10 Years of the PCG workshop: Past and Future Trends. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3402942.3409598

[7] Antonios Liapis, Georgios N. Yannakakis, Mark J. Nelson, Mike Preuss, and Rafael Bidarra. 2019. Orchestrating Game Generation. *IEEE Transactions on Games* 11 (3 2019), 48–68. Issue 1. https://doi.org/10.1109/TG.2018.2870876

[8] Wikipedia contributors. 2022. Genetic algorithm — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Genetic_algorithm&oldid=1075487198 [Online; accessed 25-March-2022].