Procedural content generation in education

Orchestration of content using PCG

Bora Tolgay Mete[1]
Supervisor(s): Rafael Bidarra[1]
[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Bora Tolgay Mete
Final project course: CSE3000 Research Project
Thesis committee: Rafael Bidarra, Sicco Verwer

Abstract

Procedural Content Generation (PCG) is a powerful content generation technique that can be used to automatically generate content (an example would be exercises for a quiz or a game). As it stands, PCG is able to create content with zero human interaction which makes it a technique worth exploring for educational purposes. In the context of PCG, this research focuses on how PCG is used to orchestrate the simultaneous generation of content of various types and identify what could be done more, possibly towards educational use. The method chosen for this research is literature review. As such, this paper provides a summary of "orchestrative" PCG approaches found in recent literature and evaluates said approaches through the lens of "PCG for education". Findings are presented in an exhaustive manner with each piece of literature occupying a subsection. A discussion section that explores how these approaches could benefit education and/or what could be done to facilitate that is also present.

1 Introduction

Procedural content generation (PCG) is an automated content generation technique in which a "content" (a common example would be exercises for a math quiz/game [1]) is created by a computer *automatically*, as opposed to the content coming together through manual contributions. PCG is able to produce content with limited-to-none human interaction which makes it an interesting and useful technique for educational use. However, the details of the technique itself and how to actually use it to make education better is mostly unexplored territory. This area of research shows great importance because if PCG is able to improve the quality of educational content, that would positively impact millions of people around the world. Considering "PCG for education" as an umbrella term, this research focuses on **how** PCG is used to "orchestrate" the simultaneous generation of content of various types and **what** has been achieved so far. Here, the word orchestrating refers to the process of combining two or more procedural content generators in some way to generate content (up to a complete game) [2].

The main method of this research is literature review and the paper aims to provide a detailed summary answering (the research question) "What has been achieved so far orchestrating the simultaneous generation of content of various types?" and the following sub-questions:

- Which methods are used in the industry when implementing PCG that orchestrate the simultaneous generation of content?
- How or why is PCG used to orchestrate the generation of multiple types of content? Is it preferred over other alternatives?
- What are the similarities/differences between applications of PCG that orchestrate the generation of a game? Based on these, are there implementations of PCG that are better/worse?

The paper follows a conventional structure, starting with this introduction. The next section elaborates on the methodology of the research as well as necessary background information. Sections 3 to 6 each contain a different set of papers and they talk about the relevant literature through the lens of education in an attempt to organize the papers based on their similarities & differences. Section 7 provides a recap of the most important literature included in the previous sections and talks about what has been done so far in

terms of orchestrating the generation of educational content & what could be done more (possibly with inspirations from other domains). Section 8 talks about the ethical aspects of the research. The next section is the last one before the references and it summarizes the conclusions of the research.

2 Background Information & Methodology

The usage of PCG for educational purposes has been mostly focused on automatically generating exercises or questions for educational games (such as games aimed at teaching mathematics [1], programming [3] and English [4] etc.). Note that the mentioned papers do not apply PCG in a way to orchestrate multiple types of content. Most examples of orchestration using PCG is outside the educational domain (simultaneously generating levels and their soundtracks based on a "tension progression" [5], automatically creating missions and spaces for them [6]).

The aim of this research is to investigate the current state of PCG in educational applications through a comprehensive literature review. By reviewing existing literature and examining industry practices/applications, this study also attempts to identify the methods & processes used to orchestrate the simultaneous generation of content of various types and how can those be used to improve education.

2.1 Literature Review Methodology

The literature review process of this research is outlined below:

- Identifying sources (databases): For searching existing literature and relevant papers, databases such as Google Scholar, IEEE Xplore and ACM were used. These are some examples (not an exhaustive list) of search keywords used on said databases: "PCG in education", "Procedural content generation in education", "PCG AND orchestration", "Procedural content generation AND full games".
- Inclusion/Exclusion criteria: Sources/papers which discussed PCG in the domain of education and/or demonstrated an application of orchestration of content using PCG were considered relevant. Papers which mention PCG but do not provide details on how it was implemented or papers that were not peer-reviewed were excluded.
- Processing: For filtering, tagging and grouping the papers found, Zotero was utilized. The relevancy of each source was re-evaluated.
- Analysis: The content of each paper that remained after the previous step was studied closely and relevant information was extracted. The data was organized in order to identify patterns, similarities and differences between different implementations of PCG. A table was created which shows the occurrences of various PCG methods in different sources.

3 Orchestration of educational content

This section and the following few sections (4, 5 and 6) present the literature that was considered "relevant" as described in the previous chapter. This section contains the literature that applies PCG orchestration in an educational context. The next section talks

about why examining PCG orchestration approaches for games could be useful and follows a paper which provides key information regarding the orchestration process in games as well as some case studies. The following 2 sections (5 and 6) each talk about a different set of papers that are clustered together based on the method of PCG used to orchestrate the generation of game facets. Note that each "orchestrative" implementation of PCG will be analyzed through the lens of education. Finally, a table is provided at the end which shows the similarities & differences between all of the literature presented throughout this chapter.

The PCG orchestration applications shown in this section orchestrate the generation of content directly for educational purposes. For clarification, a research needs to satisfy 2 properties to be included here: It should orchestrate the generation of content (combining multiple generators in some form) and the application should be within the educational domain.

3.1 *Procedural generation of problems for elementary math education*

Xu et al. (2021) presented an approach that is able to procedurally generate elementary math problems. Their question generator takes a Knowledge Component (KC) as input (example KCs and their respective abstract forms are shown in Figure 1) that is used to generate an abstract math problem as the first step [7]. Then, the generated abstract problem is processed by 3 other generators which are described below:

- Distractor Generation: This generator generates "distractors" (wrong answers) for the generated question. To achieve this, **Xu et al.** (2021) analyzed the common mistakes students would do on fill-in-blank arithmetic problems and used this information to create distractor generation rules. As an example: If the common mistake is identified as "correct operation but wrong calculation" the distractor generator rule would be "generate distractors similar to the correct answer" [7].

- Textual Content Generation: The text generation process consists of 4 steps. The first one is to create a "logic schema" using the abstract math problem which basically defines the entities (like a person or X amount of apples etc.) that are going to appear in the question text and the relationships between them. The next step is *lexicalization* which is responsible for assigning actual words (choosing from the candidate words) to the linguistic variables. The third step is to fully generate the sentence to be shown as the question text and this is achieved by matching the logic schema with a predetermined syntactic sentence template based on the schema entities and relations. Said syntactic template generates the sentence using a context-free grammar [7]. The final step is post-processing which aims to improve the quality of the text and possibly fix grammar mistakes. An example sentence generation is shown in Figure 2.

- Visual Content Generation: This generator is the simplest of the 3 modules, it just retrieves images that are thematically-related to the generated question text from a large database.

The PCG orchestration approach presented here is highly valuable from an educational standpoint. It is able to generate a large number of math problems with a suitable text and a visual as well as meaningful distractors while requiring close-to-none human interaction. Moreover, this research is one of the few researches which aim to automate the generation of exercises/questions similar to actual exercises prepared by humans. In that sense, it is

Knowledge Component	Math problem category	Abstract form
Integer addition or subtraction within 100	Arithmetic	Equation (Addition/Subtraction)
Equal sharing problems within 12	Arithmetic	Equation (Division)
Compare integers within 20, and order them on a number line	Comparison	Number Sequence (Integers)
Compare the amount of money	Comparison	Number Sequence (Decimal numbers)
Change length units from m to dm and cm, vice-versa	Mathematical Relationship	Ratio Table
Read the percentages from a pie chart	Mathematical Relationship	Percentage Chart
Compare surfaces of flat objects	Geometry	Grid Paper (with Flat Figures)
Calculate the perimeter of a square or rectangle	Geometry	Grid Paper (with Polygon)

Figure 1: Knowledge Component examples and their abstract forms [7]

Input Logic schema	BUY(Anna, 6, Brownie, Cafe)
Matched Syntactic Template	SUBJECT VERB NUM OBJ PREP DET NOUN
Output Sentence	Anna buy six brownie at the cafe

Figure 2: Example sentence generation [7]

inspirational for the future researches given that it lays out a detailed approach on how to actually orchestrate the generation of different facets of a question.

3.2 *The Effects of Mathematics Game-based Learning with Random Maze Generation*

Lifindra et al. (2023) conducted an experiment to observe the effects of *game-based mathematics learning* using a game consisting of randomly generated mazes. The approach involved procedurally generating mazes using a recursive backtracking algorithm and randomly putting "locks" between the rooms of the maze [8] which would ask the player an algebra question. The goal of the game was to find a ball which was randomly placed in the maze. The questions were generated using simple algebraic question templates and substituting random numbers to the variables in the said templates. Figure 3 shows the questions implemented within the game.

As illustrated, the orchestration approach presented here is simple and unsophisticated (both in terms of the generated mazes and questions). Nevertheless, according to the results of the experiment, using this maze game in teaching maths could provide a better learning experience than relying solely on traditional teaching methods [8]. This research goes to show that even highly straightforward and unsophisticated PCG approaches like this one could prove to be useful for education.

Template	Objective	Answer
$x + a = b$	Answer x	b - a
$\frac{x^a y^b z^c}{x^i y^j z^k} = x^p y^q z^r$	Answer p, q, and r	p = a – i q = b – j r = c - k
$x + y = a$ $x - y = b$ $x^2 - y^2 = \cdots ?$	Answer x^2 - y^2	a × b
$x + y = a$ $x^2 + y^2 = b$ $2xy = \cdots ?$	Answer 2xy	a^2 - b

Figure 3: Question examples [8]

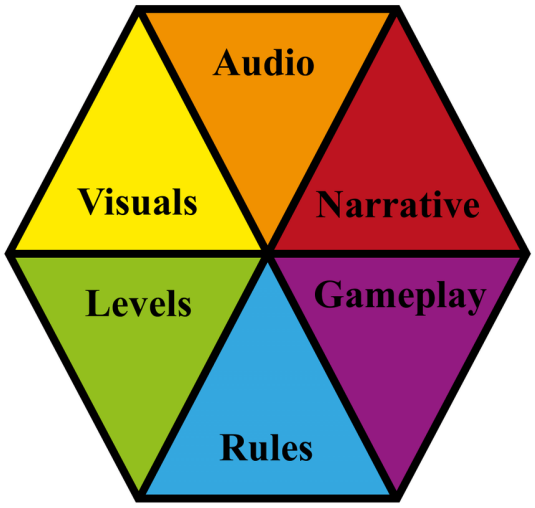

Figure 4: Creative facets of games [2]

4 Orchestration of game facets

Given that the applications of PCG orchestration in an educational context are limited, it makes sense to examine the orchestration approaches outside the educational domain and possibly determine if said approaches could be useful for education in some form. To that end, it seems appropriate to look at PCG orchestration techniques developed for **games** for the following reasons:

- a lot of research has been done about how PCG can be used for games
- same techniques could also be useful for *educational* games

As such, the literature presented in this section and the following 2 sections was picked such that it has PCG implementations acting on multiple creative facets of games and attempts to orchestrate the generation of content. Figure 4 shows the different facets of games which are: Audio, Visuals, Levels, Rules, Gameplay and Narrative.

Figure 4 is taken from the paper *Orchestrating Game Generation* by **Liapis et al.** (2019) [2] which talks about the orchestration process (for games) in-depth: the facets of games to

be orchestrated, different approaches of how orchestration can be performed, the inputs & outputs of the orchestration process as well as various case studies (games that have PCG implementations acting on multiple facets). The approaches which could be considered useful for education among those case studies are discussed below.

4.1 *Game-O-Matic*

Game-O-Matic is a game generator that takes a small-scale narrative as input and tries to generate an arcade game based on it [9]. The narrative is used to generate rules and the (single) level of the game. According to **Liapis et al.** (2019), *Game-O-Matic* is mainly intended for journalists so that they can automatically and quickly create news-games [2] which can provide more context to the public about some news in an entertaining way. This PCG method could prove itself useful for education: Given that it is able to transform a short narrative into a playable simulation, it could be used (for example) to turn elementary mathematics or physics problems for children into simulations which would help teach simple concepts in an entertaining manner.

4.2 *Data Adventures*

Data Adventures aims to create adventure games using open access data to generate the game components. It utilizes Wikipedia articles to create a story (narrative) that's based on "real" people and searches for images on the internet based on the article titles. Levels of the game are generated with the help of OpenStreetMap and they are real locations all around the world [10]. In terms of education, *Data Adventures* could be useful for teaching basic geography or history in a fun way. As an example, with small changes to the implementation, every level could be an exercise that asks about a known figure in history while automatically showing visuals of other people or pictures of places that's related as hints.

4.3 *AudioInSpace*

AudioInSpace is a unique space shooter which utilizes 2 compositional pattern producing networks (CPPNs) to facilitate the bi-directional connection between the audio and the rules of the game. The first CPPN takes the audio's current pitch information and the position of the bullets as input which allows the audio facet to indirectly control the trajectory & color of the players' bullets (rules facet). The other CPPN inputs are the position where the bullet was fired, the time passed after firing, whether it hit an enemy and its color [11]. As such, the players are able to influence the audio with their firing behavior (gameplay), forming a loop in which the player influences the audio and the audio affects the rules & changes the visuals of the game. Admittedly, it is unclear how the method of PCG utilized in *AudioInSpace* could be useful for education. Nevertheless, it is presented here because of the unique pattern of orchestration (a loop where the player can influence the audio & visuals via their gameplay and vice versa).

4.4 *Mechanic Miner*

Mechanic Miner makes small tweaks to an existing source code in order to change the rules of the game and then procedurally generates new levels with the updated ruleset. The playability of the generated levels are tested with the help of an agent performing random

actions [12]. From an education perspective, this pattern of orchestration could be used to tweak values in pre-written questions/exercises to automatically generate similar ones. However, testing the validity (whether it is solvable or not) of the new exercises could prove to be a challenge given that trying random actions won't result in a valid solution.

5 Orchestration using generative grammars

The PCG approaches presented in this section utilize one or more generative grammar(s) in some form to orchestrate the generation of content. A generative grammar typically has a set of rules and an alphabet (set of symbols like "a" or "B" which are used to create the rules of the grammar). The rules dictate *how* words can be derived and in general they are shown in the following form: S $\rightarrow$ ab which means the symbol "S" can be replaced with the string "ab". Generative grammars could be used to describe games if symbols in the alphabet are used to represent game-specific concepts (for example: obstacle, enemy, collectable) and the rules of the grammar are used to define appropriate ways of combining said concepts [6].

5.1 *Adventures in level design: generating missions and spaces for action adventure games*

One of the earlier examples of using generative grammars to orchestrate game facets was demonstrated by **Dormans** (2010) [6]. This paper uses graph grammars as the basis of its approach to PCG. A graph grammar is a special form of a generative grammar in which the alphabet of the grammar consists of nodes and edges connecting the nodes (essentially, graph components) instead of plain characters. Similarly, graph grammars produce graphs (one or more nodes connected with zero or more edges) instead of words. **Dormans** (2010) claims that graph grammars are well suited to generate missions ("the series of tasks the player needs to complete in order to reach the end of the level" as defined in the same paper) because of their non-linear nature [6].

This paper performs orchestration with the help of another type of generative grammar: shape grammars. Shape grammars have shapes instead of characters or graph components as their alphabet and define appropriate ways of replacing/combining said shapes with the help of their rules. The unique part of this PCG approach is that **Dormans** (2010) uses shape grammars to generate space (the geometrical lay-out of the level) that is suitable for a *given* mission. In other words, this paper aims to automate the generation of levels by generating random missions via graph grammars and then automatically generating level lay-outs (spaces) that are suitable to the generated missions. It can be observed that in this orchestration pipeline, the narrative facet of the game (missions) influence the level facet (spaces).

Even though this approach to PCG orchestration is not relevant from an educational standpoint, it was presented here because it is one of the first examples of using generative grammars to orchestrate the generation of content. Moreover, it also demonstrates using a graph grammar in combination with some other technique (in this case shape grammars) to perform orchestration and that is a popular approach which shows itself in nearly all papers included in this subsection.

5.2 *Mixed-Initiative Design of Game Levels: Integrating Mission and Space into Level Generation*

Karavolos, Bouwer and Bidarra (2015) present an approach utilizing the "Ludoscope" tool, which is a mixed-initiative design tool that aims to help the generation by making a model of the design process [13]. Ludoscope tries to break down the generation process into smaller executable modules and each module either receives input from/gives output to one another. The level generation process starts by the designer providing an initial graph as input to Ludoscope. The tool provides an example graph and the designer has the ability to add/remove nodes and make/break connections (edges) on this graph. This initial graph is then processed through multiple modules (note that the order of the modules/operations are also defined by the designer) to generate an action graph which can be fine-tuned. The nodes in the resulting graph are all terminal nodes and loosely correspond to rooms in the level [13]. Finally, the action graph is transformed into a level by a layout solver.

Similar to the one presented in the previous subsection, this approach also aims to use the narrative facet in the level generation process. The difference between them is that for this research, the missions (narrative) are generated based on both a generative grammar (human input) and Ludoscope whereas the previous research generated random missions just using graph grammars. In short, Ludoscope aims to grant more control to the designers and makes it easier for them to guide the orchestration process since everything is separated into executable modules. From an educational standpoint, Ludoscope could be useful if the aim of the orchestration approach is education. As an example, assume that the initial input to the tool was a generative grammar that generates some sort of puzzle/exercise/question. The designer would be able to easily look at different variations of the exercise (between/after operations) and compare them with the help of Ludoscope's modular nature.

5.3 *Graph grammar-based controllable generation of puzzles for a learning game about parallel programming*

This paper is slightly different than the rest of the literature presented in this section since it applies orchestration of content directly for an educational purpose. I just chose to present it here instead of Section 3 because of clarity seasons, given that it uses graph grammars to generate content. The aim of this approach is designing a PCG system that is capable of generating a variety of solvable programming puzzles that satisfy given set of input parameters [14]. The implementation utilizes 4 different graph grammars consecutively, which are outlined below:

- Puzzle Outline: Generates the overall structure of the puzzle.
- Challenges: Converts the "challenge" nodes in the graph to actual challenges. It also dictates which programming concepts are going to be featured in the puzzle.
- Refinements: Refines the structure of the graph & ensures that the tracks (lines of execution) are properly connected.
- Solution Removal: Removes a set of nodes from the graph to create a puzzle (otherwise it's a *solved* puzzle). This grammar also sets the difficulty of the puzzle.

The resulting graph is embedded into a 2D grid that will be used to display the puzzle in-game.

As illustrated, this approach performs the orchestration of programming puzzles for an educational game using 4 different generative graph grammars. Each graph grammar is responsible for a different set of operations (like which programming concepts are going to show up in the puzzle or the overall difficulty & complexity) which makes it easier to control the desired aspects of the puzzle. In terms of education, the research is a highly valuable contribution to literature: It shows that it is possible to generate a variety of solvable exercises using generative graph grammars.

5.4 *Graph-based generation of action-adventure dungeon levels using answer set programming*

Smith, Padget and Vidler (2018) used Answer Set Programming (ASP) with a graph grammar to generate graph models for action-adventure dungeon levels (similar to Zelda [15]). ASP is a declarative logic programming approach that aims to help at modelling constrained combinatorial search problems, as defined in the same paper. **Smith, Padget and Vidler** (2018) also mention that they used the approach laid out by **Smith and Mateas** (2011) [16] to model the design space. Said approach is able to model content generation problems as ASP logic programs which result in answer sets that each represent a single instance of valid content [15]. In short, there are 3 types of rules that guide the generation of nodes and the overall graph structure:

- *choice* rules that generate a selection of available nodes which form the design space
- *deduction* rules that deduce additional necessary nodes/edges
- *integrity* constraints that forbid undesirable outcomes to constrain the design space

Similarly to the literature explored before, the graphs generated by ASP are transformed into playable levels by a layout solver.

From an orchestration perspective, this approach is not that interesting given that it still (similar to most approaches seen before in this section) aims to generate levels using a narrative generated by a graph grammar. From an educational perspective, it might prove to be useful in the future since **Smith, Padget and Vidler** claim that using ASP enabled them to easily target the desired areas of the design space while being able to satisfy hard gameplay/level-design constraints [15]. For example, if the underlying graph grammar was designed to produce exercises then with the help of ASP it would be easier to adapt the difficulty (targeting desired areas of the design space) and generate exercises that are both similar to each other & valid (satisfying hard constraints).

6 Orchestration using machine learning

The PCG approaches presented in this section utilize some kind of machine learning/deep learning technique to help orchestrate the generation of game facets. Naturally, machine learning techniques often require large amounts of data for training. For the first piece of literature that's included in this section, said training data is gathered through manual testing in combination with a questionnaire. The remaining 2 papers generated their datasets randomly with the help of artificial agents.

6.1 *An experiment on game facet combination*

Prager et al. (2019) experimented with a maze runner game on 4 different audiovisual setting combinations to observe their perceived effects on players. The settings consist of 2 homogeneous and 2 heterogeneous audiovisual combinations: (happy, happy), (horror, horror) and (happy, horror), (horror, happy). **Prager et al.** (2019) experimented with 20 people for this research and asked them to compare different setting combinations as *more*, *less* or *same* based on 2 metrics: fun and difficulty [17]. According to the results of the questionnaire, the players perceived homogeneous audiovisual settings as more fun and less difficult to complete compared to heterogeneous combinations. The authors also observed that the players self-reported having higher *arousal* (strong emotions like joy, laughter or stress, fear [17]) when playing with homogeneous audiovisual settings.

Prager et al. (2019) also applied pairwise preference learning using Support Vector Machines (SVM) to see if the arousal effect of different facet combinations on a player can be modeled. They found out that the homogeneous audiovisual settings can be modeled with a higher accuracy compared to heterogeneous combinations. Another observation they made was that the audio facet seemed to be more powerful in terms of predictive modeling than the visual facet[17].

This research explores the effects of game facet orchestration (specifically audio and visual) on players, assuming a machine learning approach. The insights gathered by **Prager et al.** (2019) are highly valuable for the improvement of PCG because said insights are needed in order to orchestrate the generation of game facets in a way that is perceived as "fun" by the players. Same insights could also be used to compare different orchestration approaches for an educational game to observe which approach is more enjoyable for the players.

6.2 *A multi-faceted surrogate model for search-based procedural content generation*

Karavolos, Liapis and Yannakakis (2019) developed an approach that uses deep learning to create a surrogate model which is able to assess the fitness of a level and 2 characters for a one-versus-one First Person Shooter (FPS) game. The dataset used to train the model consists of randomly generated levels and 2 character classes (one for each player). The class-level-class pairings were simulated with the help of artificial agents [18] and 2 types of gameplay outcomes were recorded: kill ratio (the ratio of the first player's kills to total kills) and the duration of the match. They then trained a Convolutional Neural Network (CNN) that takes the level and 2 character classes as its input and tries to predict the mentioned gameplay outcomes. Note that the purpose of this surrogate model is to solely evaluate possible class-level-class combinations. For actually generating the facets of the game (the level and 2 character classes) an evolutionary algorithm is used. The initial population of the evolutionary algorithm is based on human designs [18].

As a recap, this research proposes a surrogate model that is able to map a certain level (level facet) and 2 character class (rules facet) combination to its gameplay outcome (gameplay facet). The model is used to evaluate candidate combinations generated by an evolutionary algorithm. **Karavolos, Liapis and Yannakakis** (2019) claim that their model could be used to adjust human designs towards desired gameplay outcomes.

Even though this research does not have *direct* educational applications, it was included here thanks to its uniqueness. The research shows that it is possible to "model" a certain game facet orchestration approach and evaluate the fitness of said approach. Moreover, the

orchestration pipeline presented in this paper (random generation of facets → simulation → training the CNN to predict gameplay outcome) is inspiring for education because it can be translated to generate educational content apart from the simulation step. Using artificial agents which perform random actions to simulate an FPS game is perfectly fine, however, it would not be ideal (nor sufficient) for simulating most educational material like exercises, questions or visuals. From an education perspective, it makes the most sense to use human input for simulating (evaluating) randomly generated educational material but doing that would defeat the purpose of using PCG in the first place.

6.3 *SuSketch: Surrogate Models of Gameplay as a Design Assistant*

Migkotzidis and Liapis (2021) applied the framework presented in the previous section to *SuSketch* which is a mixed-initiative design tool. The bulk of the surrogate model implementation is the same:

- the model is trained using a randomly generated class-level-class combination dataset for the same FPS game
- levels are simulated with the help of artificial agents
- a CNN is trained to predict the gameplay outcomes using the level and character classes as its input

The part that is unique to this approach is the gameplay outcomes. Instead of using the kill ratio and the match duration to express the gameplay outcomes, **Migkotzidis and Liapis** (2021) introduced 3 new metrics: Death Heatmap (DH), Dramatic Arc (DA) and Combat Pacing (CP) [19]. DH (as its name would suggest) shows a heat map of player deaths by dividing the level into 4x4 cells. DA is similar to kill ratio, however, it's computed more than once for specific points in time rather than calculating it once in the end. Finally, CP counts the number of kills scored in a specific time interval to capture high-action moments.

In terms of educational applicability, this research is slightly better compared to the previous one. The main difference between the two implementations is that the previous research used the surrogate model to evaluate the combinations generated by the evolutionary algorithm. Here, it is used to help *SuSketch* generate suggestions for the designer in the form of alternative class pairings and power-up placements [19]. From an educational standpoint, this approach is more suitable since it aims to use the model to help the human designer whereas the previous approach used the model to (essentially) improve the evolutionary algorithm.

Table 1 summarizes all of the PCG orchestration approaches explored in this section as well as their educational applicability (which is the last column denoted with "E"). The rest of the columns represent the type of content generated by the approach which are the same as the game facets discussed in Section 4 (Visuals, Audio, Narrative, Gameplay, Rules, Levels). A "P" pertains to partial generation of that type or that the approach is partially useful for/applicable to education.

Author	V	A	N	G	R	L	E
Xu et al.	X	-	X	-	-	-	X
Lifindra et al.	-	-	-	-	P	X	X
Game-O-Matic, Treanor et al. (2012)	X	-	P	-	X	P	P
Data Adventures, Green et al. (2018)	X	-	X	-	-	P	P
AudioInSpace, Hoover et al. (2015)	X	X	-	-	P	-	-
Mechanic Miner, Cook et al. (2013)	-	-	-	P	X	X	P
Dormans (2010)	-	-	X	-	-	X	-
Karavolos, Bouwer and Bidarra (2015)	-	-	X	-	-	X	P
Valls-Vargas et al. (2017)	P	-	-	-	-	X	X
Smith, Padget and Vidler (2018)	-	-	X	-	-	X	P
Prager et al. (2019)	X	X	-	-	-	-	P
Karavolos, Liapis and Yannakakis (2019)	-	-	-	X	X	X	-
Migkotzidis and Liapis (2021)	-	-	-	X	X	X	-

Table 1: Orchestration approaches

7 Discussion

The previous sections presented various PCG orchestration approaches with differing levels of educational applicability. Based on the approaches reviewed, this section discusses the types of educational content which can be generated together and determine if doing so would be advantageous from an educational perspective.

As illustrated in Section 3, **Xu et al.** (2021) developed an approach that is able to generate a large number of elementary math problems with essentially no human input. The approach consisted of generating abstract math questions using templates and then utilizing 3 different generators (which take the generated abstract question as input) to generate distractors for the question, a suitable question text and a related visual. This approach shows that it is possible to orchestrate the generation of both *textual* and *visual* content together with an educational exercise/question.

Lifindra et al. (2023) proposed a very simple PCG orchestration approach in the form of an educational game which has randomly generated mazes as levels and asks the player template-based algebra questions between the rooms of the maze. According to the results of the research, using this game as a teaching tool could slightly improve the educational experience compared to relying only on traditional methods [8]. In terms of orchestrating the generation of educational content, this research shows that generating a simple maze-based or puzzle-based game and embedding template-based algebra questions within the levels could be useful for education.

Section 4 followed a research which talked about orchestration of game facets as well as some case studies. One of the case studies were *Game-O-Matic*, which is capable of generating short arcade games based on a small-scale narrative it takes as input. Taking inspiration from the work of **Xu et al.** (2021), this approach shows potential in terms of generating educational content: For example it could be used to orchestrate the generation of playable simulations of elementary math or science problems (similar to the visual content generator in **Xu et al.** (2021)'s work).

Section 5 presented the literature that utilized generative grammars to orchestrate the generation of game facets. Starting with the only paper in that section which directly tries

to create educational content: **Valls-Vargas et al.** (2017) developed a modular PCG orchestration approach which uses 4 generative graph grammars to automate the generation of puzzles for an educational programming game. This approach is highly valuable from an educational perspective because it shows that other orchestration techniques that use graph grammars could be useful for education as well. As an example, **Karavolos, Bouwer and Bidarra** (2015) combined graph grammars with a mixed-initiative design tool called "Ludoscope" to orchestrate the generation of levels. Another approach (by **Smith, Padget and Vidler** (2018)) used graph grammars in combination with ASP to generate action-adventure dungeon levels. Both of these techniques were concerned with orchestrating the generation of the narrative and level facets of games. However, taking inspiration from the approach by **Valls-Vargas et al.** (2017), these techniques (using graph grammars together with a mixed-initiative tool or ASP) *could* be useful & should be explored from an educational perspective.

8 Responsible Research

While conducting the literature review as part of this research, several responsible research principles were adhered to such as avoiding plagiarism, credibility of sources, respecting authorship and reproducibility of the overall research. All sources of inspiration and/or information are properly cited and referenced according to the BibTex scheme which can be found at the end. The literature included in this review was selected from credible sources such as peer-reviewed journals or conference proceedings. To respect authorship, it was ensured that appropriate credit has been given to authors of the said literature. Finally, to ensure reproducibility, the literature review process has been conducted in a systematic and transparent manner. The details of the methodology such as the databases used to search the literature or the inclusion/exclusion criteria can be found in Section 2.1.

9 Conclusion

Procedural Content Generation is a content generation technique that is capable of generating content with close to none human input. That makes PCG an interesting technique from an educational standpoint. However, most of the educational applications of PCG are one dimensional and generate only one type of content (so no orchestration of content). This research focuses on this knowledge gap and conducts an extensive review on recent literature related to PCG in an attempt to answer the question "What has been achieved so far (from an educational standpoint) orchestrating the simultaneous generation of content of various types?". To that end, a variety of PCG orchestration approaches both within and outside the educational domain have been analyzed.

The literature reviewed have been divided into 4 categories: Orchestration of educational content, Orchestration of game facets, Orchestration using generative grammars and Orchestration using machine learning. Starting with orchestration of educational content, **Xu et al.** (2021) showed that it was possible to orchestrate the generation of a large number of elementary math problems together with suitable question texts and thematically-related images. Similarly, **Lifindra et al.** (2023) also developed an approach which aims to use PCG orchestration for teaching maths in the form of a simple maze game that asks the player template-based algebra questions between the rooms of the maze. Another approach that was inspiring for elementary mathematics or science education was *Game-O-Matic*

from the domain of games. *Game-O-Matic* is able to generate a short arcade game based on a small-scale narrative [9], which could be used to orchestrate the generation of playable simulations/visuals for basic math or physics problems. From an educational standpoint, these approaches show that PCG is capable of generating a variety of exercises/questions together with textual and/or visual content supporting the generated question. The reason why maths are explored (for PCG orchestration) before other subjects under education could be the fact that it is relatively easy to generate math exercises (using templates).

9.1 Future Work

In terms of orchestrating the generation of educational content, it can be seen that the number of approaches developed for this purpose are limited. On the contrary, a lot of research has been done about how PCG could be used to orchestrate the generation of game facets. Some of the techniques used in that area of research could be inspiring or useful from an educational perspective as well. For example, generative graph grammars are widely used for orchestrating game facets, mainly for generating better (more suitable) levels [6] [13] [15]. However, as demonstrated by **Valls-Vargas et al.** (2017), graph grammars are also capable of generating puzzles for an educational programming game. This shows that generative grammars (especially graph grammars) are powerful tools and makes it a technique worth exploring for orchestrating the generation of educational content. Another PCG method used for orchestrating game facets is machine learning (as described in Section 6) which mainly focuses on "modeling" the facet generation process as a form of evaluating different orchestration options. Directly using ML techniques for generating educational content without human input could be dangerous, however, a mixed-initative approach like *SuSketch* by **Migkotzidis and Liapis** (2021) could be promising for education.

References

[1] Luiz Rodrigues, Robson Bonidia, and Jacques Brancher. *A Math Educational Computer Game Using Procedural Content Generation*. Oct. 2017. DOI: 10.5753/cbie.sbie.2017.756.

[2] Antonios Liapis et al. "Orchestrating Game Generation". In: *IEEE Transactions on Games* 11.1 (2019). https://doi.org/10.1109/TG.2018.2870876, pp. 48–68. URL: http://graphics.tudelft.nl/Publications-new/2019/LYNPB19.

[3] T.J. Tiam-Lee and K. Sumi. "Procedural Generation of Programming Exercises with Guides Based on the Student's Emotion". English. In: 2018, pp. 1465–1470. ISBN: 978-1-5386-6650-0. DOI: 10.1109/SMC.2018.00255.

[4] D. Hooshyar, M. Yousefi, and H. Lim. "A Procedural Content Generation-Based Framework for Educational Games: Toward a Tailored Data-Driven Game for Developing Early English Reading Skills". English. In: *Journal of Educational Computing Research* 56.2 (2018), pp. 293–310. ISSN: 0735-6331. DOI: 10.1177/0735633117706909.

[5] Phil Lopes, Antonios Liapis, and Georgios N Yannakakis. "Framing Tension for Game Generation". en. In: (2016).

[6] Joris Dormans. "Adventures in level design: generating missions and spaces for action adventure games". en. In: *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. Monterey California: ACM, June 2010, pp. 1–8. ISBN: 978-1-4503-0023-0. DOI: 10.1145/1814256.1814257. URL: https://dl.acm.org/doi/10.1145/1814256.1814257 (visited on 05/15/2024).

[7] Yi Xu, Roger Smeets, and Rafael Bidarra. "Procedural generation of problems for elementary math education". en. In: *International Journal of Serious Games* 8.2 (June 2021), pp. 49–66. ISSN: 2384-8766. DOI: 10.17083/ijsg.v8i2.396. URL: http://journal.seriousgamessociety.org/index.php/IJSG/article/view/396 (visited on 03/26/2024).

[8] Benny Hansen Lifindra, Darlis Herumurti, and Imam Kuswardayan. "The Effects of Mathematics Game-based Learning with Random Maze Generation". In: *2023 International Seminar on Application for Technology of Information and Communication (iSemantic)*. Sept. 2023, pp. 69–74. DOI: 10.1109/iSemantic59612.2023.10295276. URL: https://ieeexplore.ieee.org/document/10295276 (visited on 04/25/2024).

[9] Mike Treanor et al. "The micro-rhetorics of Game-o-Matic". In: *Proceedings of the International Conference on the Foundations of Digital Games*. FDG '12. New York, NY, USA: Association for Computing Machinery, May 2012, pp. 18–25. ISBN: 978-1-4503-1333-9. DOI: 10.1145/2282338.2282347. URL: https://doi.org/10.1145/2282338.2282347 (visited on 06/11/2024).

[10] Michael Cerny Green et al. "DATA Agent". In: *Proceedings of the 13th International Conference on the Foundations of Digital Games*. arXiv:1810.02251 [cs]. Aug. 2018, pp. 1–10. DOI: 10.1145/3235765.3235792. URL: http://arxiv.org/abs/1810.02251 (visited on 06/11/2024).

[11] Amy K. Hoover et al. "AudioInSpace: Exploring the Creative Fusion of Generative Audio, Visuals and Gameplay". en. In: ed. by Colin Johnson, Adrian Carballal, and JoÃ£o Correia. Vol. 9027. Book Title: Evolutionary and Biologically Inspired Music, Sound, Art and Design Series Title: Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 101–112. ISBN: 978-3-319-16497-7 978-3-319-16498-4. DOI: 10.1007/978-3-319-16498-4_10. URL: https://link.springer.com/10.1007/978-3-319-16498-4_10 (visited on 06/11/2024).

[12] Michael Cook et al. "Mechanic Miner: Reflection-Driven Game Mechanic Discovery and Level Design". In: Apr. 2013, pp. 284–293. ISBN: 978-3-642-37191-2. DOI: 10.1007/978-3-642-37192-9_29.

[13] DaniÃ«l Karavolos, Anders Bouwer, and Rafael Bidarra. "Mixed-Initiative Design of Game Levels: Integrating Mission and Space into Level Generation." In: *FDG* 2015 (2015), p. 8. URL: https://www.academia.edu/download/79239689/fdg2015_paper_25.pdf (visited on 05/11/2024).

[14] Josep Valls-Vargas, Jichen Zhu, and Santiago OntaÃ±^{3}n. "Graph grammar-based controllable generation of puzzles for a learning game about parallel programming". In: *Proceedings of the 12th International Conference on the Foundations of Digital Games*. FDG '17. New York, NY, USA: Association for Computing Machinery, Aug. 2017, pp. 1–10. ISBN: 978-1-4503-5319-9. DOI: 10.1145/3102071.3102079. URL: https://dl.acm.org/doi/10.1145/3102071.3102079 (visited on 05/04/2024).

[15] Thomas Smith, Julian Padget, and Andrew Vidler. "Graph-based generation of action-adventure dungeon levels using answer set programming". en. In: *Proceedings of the 13th International Conference on the Foundations of Digital Games*. MalmÃ¶ Sweden: ACM, Aug. 2018, pp. 1–10. ISBN: 978-1-4503-6571-0. DOI: 10.1145/3235765.3235817. URL: https://dl.acm.org/doi/10.1145/3235765.3235817 (visited on 06/11/2024).

[16] Adam M. Smith and Michael Mateas. "Answer Set Programming for Procedural Content Generation: A Design Space Approach". en. In: *IEEE Transactions on Computational Intelligence and AI in Games* 3.3 (Sept. 2011), pp. 187–200. ISSN: 1943-068X, 1943-0698. DOI: 10.1109/TCIAIG.2011.2158545. URL: http://ieeexplore.ieee.org/document/5783900/ (visited on 05/02/2024).

[17] Raphael Prager et al. "An Experiment on Game Facet Combination\". In: Aug. 2019, pp. 1–8. DOI: 10.1109/CIG.2019.8848073.

[18] Daniel Karavolos, Antonios Liapis, and Georgios Yannakakis. "A Multifaceted Surrogate Model for Search-Based Procedural Content Generation". In: *IEEE Transactions on Games* 13.1 (Mar. 2021). Conference Name: IEEE Transactions on Games, pp. 11–22. ISSN: 2475-1510. DOI: 10.1109/TG.2019.2931044. URL: https://ieeexplore.ieee.org/abstract/document/8778792 (visited on 05/11/2024).

[19] Panagiotis Migkotzidis and Antonios Liapis. "SuSketch: Surrogate Models of Gameplay as a Design Assistant". In: *IEEE Transactions on Games* 14.2 (June 2022). arXiv:2103.11726 [cs], pp. 273–283. ISSN: 2475-1502, 2475-1510. DOI: 10.1109/TG.2021.3068360. URL: http://arxiv.org/abs/2103.11726 (visited on 06/11/2024).