A Survey of Procedural Dungeon Generation

Breno M. F. Viana and Selan R. dos Santos
Departamento de Informática e Matemática Aplicada — DIMAp
Universidade Federal do Rio Grande do Norte — UFRN
Natal, RN Brazil
bmfviana@gmail.com, selan@dimap.ufrn.br

***Abstract*—Procedural content generation (PCG) is a method of content creation fully or semi-performed by computers. PCG is widely used in game development to generate game content, from Rogue (1998) to No Man's Sky (2016). PCG generates final contents, which are ready to be added to a game, or intermediate contents, which are might work as content sketch to be polished by human designers. In this paper we survey the current state of procedural dungeon generation (PDG) research, a subarea of PCG. We analyzed the works according to the game features they generate, the solution strategy employed and the taxonomy of procedural content generation. Some of the relevant findings of the survey are: (1) PDG for 3D levels has been little explored; (2) few works supported levels with barriers, a game mechanic which blocks, temporarily, the player progression, and; (3) and just a few solutions relied on mixed-initiative approach, where a human design content is combined with a computer generated level.**

***Keywords*-survey; procedural content generation; game content generation; procedural dungeon generation;**

I. Introduction

According to Togelius et al. [1], in games, Procedural Content Generation (PCG) "*refers to computer software can create game content on its own, or together with one or many human players or designers.*" In other words, PCG is a method of game content creation fully performed by computers or in association with human designers or gamers. PCG may be considered a valuable asset in game development mainly because (1) it may bring down the high cost of production of game features by reducing the need of a human designer to generate content; (2) it may help human designers to increase their creativity and productivity; (3) it may be applied to control the game difficulty, and/or help game balancing, and; (4) it may increase the replay value of a game by providing unexpected contents.

PCG is usually applied to generate dungeon levels in games. These games are often classified by the game community as rogue-like games. Some examples are: Rogue (Troy and Wichman, 1980), the game which introduced the rogue-like genre, Diablo (Blizzard Entertainment, 1998), The Binding of Isaac (Edmund McMillen, 2011), Don't Starve (Klei Entertainment, 2013), Crypt of the NecroDancer (Brace Yourself Games, 2015), Moonlighter (Digital Sun, 2018) and Dead Cells (Motion Twin, 2018).

Dungeon levels, according to van der Linden, Lopes and Bidarra [2], are labyrinthine environments which consists mostly of rewards, interrelated challenges and puzzles in order to offer highly structured gameplay progressions. The authors also published a survey on procedural dungeon generation (PDG), where they compared different approaches on content generation, and tried to understand how the surveyed methods were controlled. However, they did not attempt to classify the methods under an unified taxonomy. This shortcoming motivated us to produce a review exclusively focused on PDG and, most important, target at classifying the research under the taxonomy of PCG defined by Togelius et al. [1]. We believe that this categorization might be useful since it help us to better understand the approaches' behavior, to identify shortcomings and strengths, as well as tendencies (or lack thereof) in the recent body of research aimed at PDG.

In this paper, we survey the current state of PDG. We selected papers from ACM Digital Library, IEEE Xplore, and Scopus. We considered only works related to generation of dungeon or maze-like levels in the last 8 years. We focused our review on the following categories: *game dimensionality*, i.e., two- or three-dimensional; *the game genre*, i.e., RPG, rogue-like, action-adventure, 2D platform, etc.; *the level structure*, i.e., composed of caves, rooms, etc.; *the algorithm or solution strategies* adopted by the PDGs; and, how well they fit the *taxonomy* proposed by Togelius et al. [1]. Some of the main findings of our review were: (1) few works focused on PDG of 3D levels; (2) the generation of levels with barriers (a game mechanic which blocks, temporarily, the player progression), although interesting, has not been much explored, and; (3) surprisingly few approaches relied on mixed-initiative approach, when a human design steers the computer content generation.

This paper is structured as follow: the Section II presents the definitions of dungeon levels in games, defines some game features and mechanics that we were interested in while evaluating the papers, briefly introduces the taxonomy of procedural generation, and provides an overview of some of the automatic generation strategies we identified during the survey. The Section III presents the quantitative and qualitative results of the survey. We offer an overview of the current state of PDG under the parameters introduced in the Section II. We also highlight a few less explore avenues

of research associated with PDG. Finally, we conclude the paper providing some suggestions of open challenges in the field.

II. Background

In this section, we define dungeon levels, and some related game elements. We also present the taxonomy of PCG [1], and an overview of the automatic generation algorithms described in the surveyed papers.

A. Dungeon

A dungeon is a maze-like environment that is mostly composed by three main game features: rewards, challenges and puzzles [2]. This kind of environment is used by several game genres. In this review, we identified three types of level structures: (1) top-down mansion-like (Fig. 1a), e.g. The Legend of Zelda's dungeons; (2) top-down cavern-like (Fig. 1b), e.g. Pokémon's dungeons, and; (3) side-scrolling (Fig. 1c), e.g. Spelunky's dungeons. These types of level structures may be the same for 2D and 3D level's sketches. However, the scenario's structure in 3D are totally different from 2D games.

B. Game Features and Mechanics

In this section, we present the definitions of some game elements (features and mechanics). Some of the definitions presented here were taken from [2].

1) Room: is a large area a level connected with another rooms by doors or corridors. The Fig. 1a presents an example of this game feature.

2) Item: is a game element that enables the player to get some gameplay benefit. For example, a first aid kit that recovers the player's health.

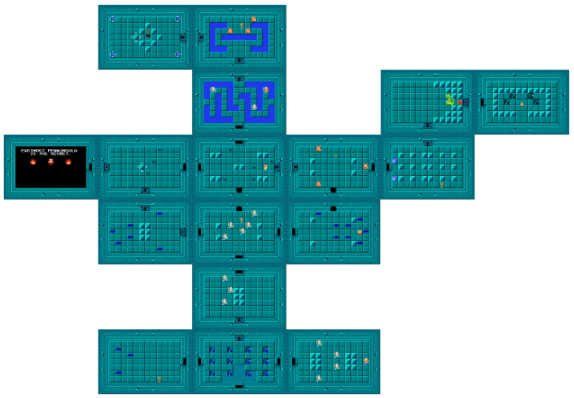
3) Skill: is a game element similar to an item; the difference between them is basically semantics (how they are used in a game).

4) Barrier: is a feature that blocks, temporarily, the player to reach some region of a level. The barriers usually need one or multiple items or some special skill to be unblocked.

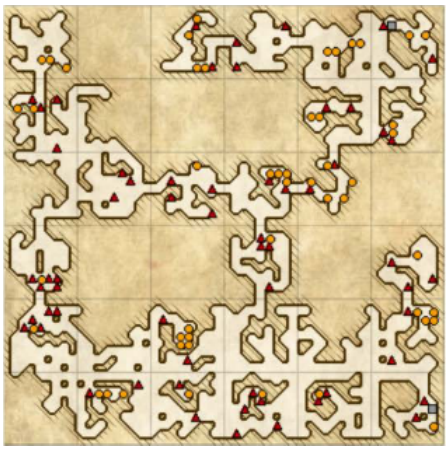
5) Challenge: is an obstacle that hinders the player's progression. In the dungeon context, challenges are usually treated as battle challenges.

6) Reward: is either an item, skill, or points the player may just pick up along the way or earn after accomplishing some difficult challenge.

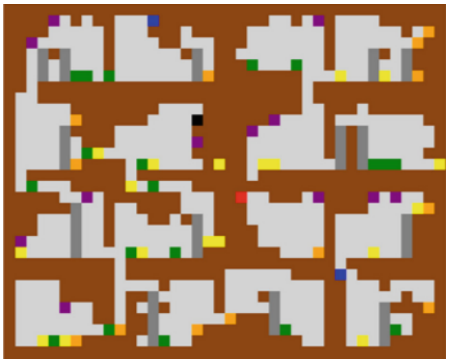
7) Puzzle: is a certain type of obstacle that requires from the player a certain level of reasoning to solve it. For example, certain puzzles involve sorting items in a specific order to unblock a barrier.



(a) Top-down mansion-like dungeon (Source: [3]).



(b) Top-down cavern-like dungeon (Source: [4]).



(c) Side-scrolling dungeon (Source: [5]).

Figure 1. Three main types of dungeon structures found in the survey.

C. Taxonomy of Procedural Content Generation

In this section, we present the taxonomy of PCG defined by Togelius et al. [1]. We renamed each title of the classifications to simplify the reading. The taxonomy defines 7 axes that are used to classify approaches of procedural generation:

1) Content Need: defines if a content generated by an approach is **necessary** for the game to work (for instance, the levels of a game) of it **optional** (for instance, to create several types of weapons in a first-person shooter).

2) Generation Time: defines when the content is generated. An approach is **offline** when it generates the content before the gameplay. Or it is **online** when it generates the content during the gameplay.

3) Generation Control: defines how the approaches can be parameterized and the dimensions of control. The control of an approach is made by **random seeds** that allows only one dimension of control. Or it is made by a **set of parameters** that allows more dimensions of control.

4) Generality: defines the audience that the content is being generated to. An approach is **generic** when the content is generated for several players. Or it is **adaptive** when the content is generated for a specific player.

5) Random Choice: defines the way that decisions are made while the generation algorithm runs. The **deterministic** approach generates the same content given the same set of parameters, whereas the **stochastic** approach generates different contents for the same set of parameters each time the algorithm runs.

6) Generation Method: defines how the generation can be performed. The **constructive** option generates a content in only one pass, while the **generate-and-test** alternates the generation and the consistency test of the content.

7) Content Authorship: defines the degree of the designer interference. In the **automatic generation** only the computer generates the content, whereas in the **mixed-initiative** the designer has a thinner control over content (not only by parameters) that is being generated.

D. Solution Strategies

We identified only two main solution strategies: constructive or search-based.

1) Constructive Algorithms: These algorithms are often quite different from each other, even if they are designed to generate the same content type. Most of them are based on random positioning and/or are based on algorithms created for other purposes, such as Cellular Automata and Generative Grammars. A Cellular Automaton (CA) is, according to Adams and Louis [6], *"a two-dimensional grid of cell and set of rules specifying cell-state transitions based on cells' neighbors."* In PCG, this approach is commonly employed to create the level's spatial structure. Generative Grammars are systems of grammatical rules that generates words [7]. Based on Generative Grammars (GG), other kind of grammars have been developed, e.g. Graph Grammars [8]. The graphs generated by Graph Grammars support the creation of levels organized in rooms which may contain rewards, and challenges [3], [9].

2) Search-based Algorithms: Solutions in this category are usually based on metaheuristics [10]. Unlike the constructive algorithms, the search-based ones always test the generated content to ensure consistency. Search-based algorithms works by generating and keeping track of a population of candidate contents, which are then evaluated according to some quality (fitness) function; the content with the highest score is the final result. Note that the fitness function value might work both to indicate whether a content is valid or not, but also to reflect the content overall quality. Most of search-based approaches we reviewed are based on Genetic Algorithms (GA) [6] [11] [12] [5] [13] [4] [14] [15] [16]. The GA is a metaheuristic algorithms based on neo-Darwinian theory of evolution. Some approaches use GA to evolve CA rules that create maze-like levels. Other approaches try to represent the content generation process in such a way that it is possible to apply Answer Set Programming (ASP) to generate content. ASP, according to [17], "*is a declarative logic programming approach aimed at modeling constrained combinatorial search problems*."

III. Survey

We selected the papers for this review from the ACM Digital Library, IEEE Xplore, and Scopus. The keywords of the search query were "dungeon", "labyrinth", "maze", "PCG", "game", and variations thereof. We chose these keywords because dungeons levels are often modeled as a maze-like structure. We selected only paper that stated clearly either in the title or the abstract that their focused on automatic dungeon level generation. For each selected paper we collected information regarding the game features presented in Section II-B, and the design strategy described in Section II-D.

We found a total of 26 articles that presented solutions for dungeon generation. The Table I and Table II summarize the data collected from these papers. Because two of the papers presented more than one PCG algorithms (for comparison purposes), we end up with a total of 31 approaches.

To simplify the view presented in Table I we abbreviated the solution strategy approaches and we assigned IDs to the taxonomy classes. The abbreviations of the solution strategies are: Evolutionary Algorithm (**EA**), Genetic Algorithm (**GA**), Genetic Programming (**GP**) Answer Set Programming (**ASP**), Generative Grammars (**GG**), Cellular Automata (**CA**) and Constructive Approach (**CO**). The IDs of the taxonomy are: Generation Time (**T1**); Generation Control (**T2**); Generality (**T3**); Random Choice (**T4**); Generation Method (**T5**); and Content Authorship (**T6**). T1: offline (Off) or online (On). T2: Random Seeds (RS) or Set of Parameters (P). T3: Generic (G) or Adaptive (A). T4: Deterministic (D) or Stochastic (S). T5: Constructive (C) or Generate-and-Test (GT). T6: Automatic Generation (AG) or Mixed-Initiative (MI).

TABLE I. COLLECTED DATA FROM ARTICLES I

Work	Dimension	Genre	Solution Strategy	Taxonomy					
				T1	T2	T3	T4	T5	T6
[18]	2D	-	GA	Off	P	G	S	GT	AG
[19]	2D	-	GA	Off	P	G	S	GT	AG
[20]	2D	-	GA	Off	P	G	S	GT	AG
[11]	2D	-	GA	Off	P	G	S	GT	AG
[21]	2D	Rogue-like	EA/ASP	Off	P	G	S	GT	AG
[9]	3D	Adventure RPG	GG	Off	P	G	S	C	AG
[22]	2D	-	GA	Off	P	G	S	GT	AG
[23]	2D/3D	-	CO	Off	P	G	D	C	AG
[24]	2D	Rogue-like	ASP	Off	P	G	S	GT	AG
[12]	2D	-	GA/CA	Off	P	G	S	GT	AG
[25]	2D	-	GA/CA	Off	P	G	S	GT	AG
[5]	2D	2D Plataform	GA	Off	P	G	S	GT	AG
[13]	2D	-	GA/CA	Off	P	G	S	GT	AG
[26]	-	Rogue-like	CO	Off	P	G	S	C	AG
[3]	2D	Action-Adventure	GG	Off	P	G	S	C	AG
[27]	-	-	CO	Off	P	G	S	C	AG
[4]	2D	-	GA	Off	P	G	S	GT	AG
[15]	2D	-	GA	Off	P	G	S	GT	MI
[14]	2D	-	GA	Off	P	G	S	GT	MI
[28] (5)	2D/3D	-	CO	Off	P	G	2D/3S	C	AG
[29] (2)	2D	-	CO	Off	P	G	S	C	AG
[30]	3D	Adventure RPG	GA	Off	P	G	S	C	AG
[17]	-	-	ASP	Off	RS	G	S	C	AG
[16]	2D	-	GA	Off	P	A	S	GT	MI
[31]	2D	-	CO	Off	RS	G	D	C	AG
[32]	2D	-	GP	Off	P	G	S	GT	AG

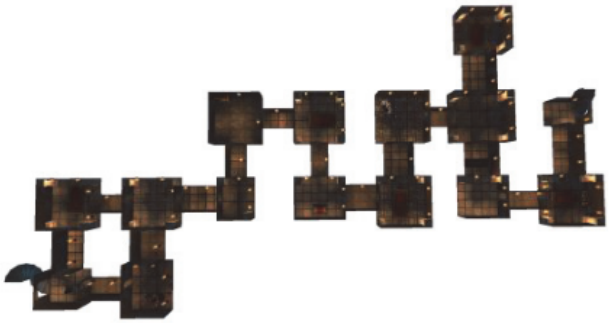
To simplify the view of the data of the Table II we provided aliases for each general type of dungeon spatial structure: Top-down mansion-like (**TDML**); Top-down cavern-like (**TDCL**); and Side-scrolling dungeon (**SS**).

A. *Quantitative Analysis*

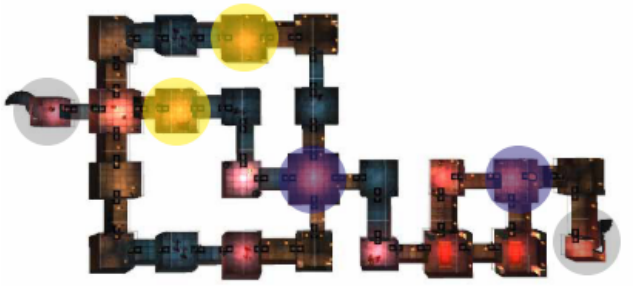
Regarding the dungeon level *dimensionality*, 19 (out of 26) presented solutions target at 2D dungeon level generation. The Fig. 1a, Fig. 1b and Fig. 1c present examples of 2D dungeon levels generated by Lavender and Thompson [3], Liapis [4] and Baghdadu et al. [5], respectively. Only the works from van der Linden, Lopes and Bidarra in [9], and Karavolos, Liapis and Yannakakis in [30] proposed solutions that generated 3D dungeon levels. Note, however, that these two works in fact did not generated a true 3D structure. Rather, they laid out 2D generic structures (called sketches) and then connected 3D chunks (rooms) previously created by a human designer based on these sketches. Both works generated levels for Dwarf Quest (Wild Card Games, 2013). Fig. 2 present examples from each paper.

The papers from Santamaria-Ibirika et al. [23], and Baron [28] described solutions that worked both for 2D and 3D. The Santamaria-Ibirika's work is based on Voronoi Diagram and Delaunay Triangulation which enables a less artificial level generation. The Fig. 3 presents an example of a level generated with their approach. The Baron's work, however, does not generate a true 3D level. Instead, it generates a 2D text-based sketch which undergoes a extrusion process to become a 3D level (see Fig. 4).

Finally, the last 3 papers only generated a high level representation for a dungeon level that could neither be classified as 2D nor as 3D.



(a) Level generated by van der Linden, Lopes and Bidarra [9].



(b) Level generated by Karavolos, Liapis and Yannakakis [30].

Figure 2. Generated levels for the Dwarf Quest.

In terms of *game genre* only 7 (out of 26) papers mentioned the game genre their solution were designed for [3]

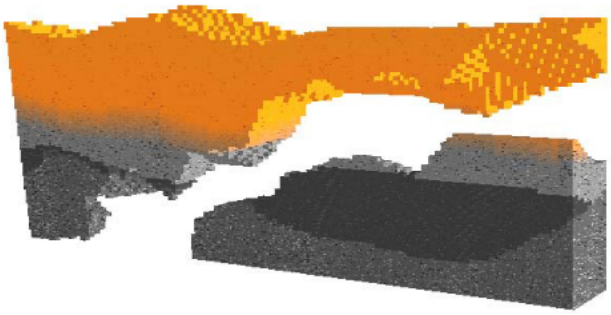
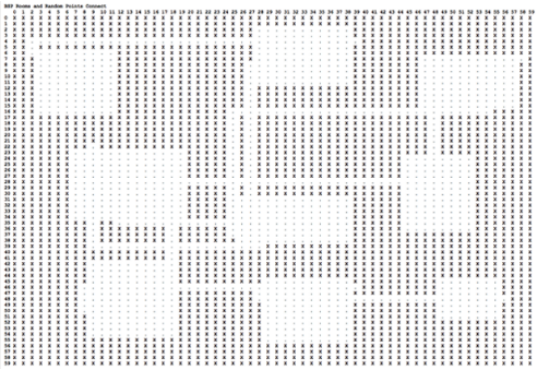
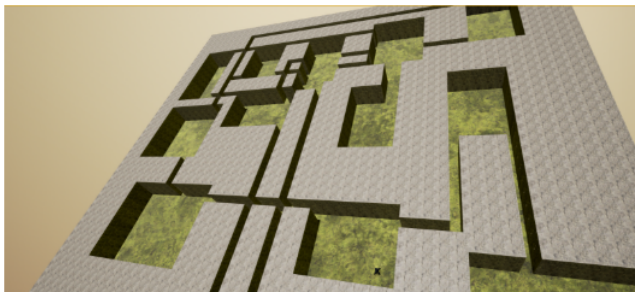


Figure 3. Slice of a cave generated by Santamaria-Ibirika et al. [23].



(a) 2D Level generated by Baron's work [28].



(b) 3D Level generated by Baron's work [28]

Figure 4. 2D and 3D Levels generated by Baron's work [28].

[5] [9] [21] [24] [26] [30], and 5 of them specified the game the solutions were developed for. This means that most works prefer to provide solutions that are game genre agnostic.

In terms of *solution strategy* 18 papers described an evolutionary based approach. From these, 14 papers relied on genetic algorithms or derivations, 2 applied ASP algorithms to generate dungeon levels ([24] and [17]); 1 used genetic programming [32], and; the last one involved an variation of evolutionary algorithms combined with ASP [21].

The remaining 8 papers described constructive approaches, derived from different strategies. Lavender and Thompson [3] and van der Linden, Lopes and Bidarra [9] developed solutions based on generative grammar approach that creates a missions' graph which aims to build the level. Santamaria-Ibirika et al. [23], as we already said, is based on Voronoi Diagram and Delaunay Triangulation. Forsyth [26] developed a algorithm that generates level sketches by simply placing rooms randomly. Hell, Clay and ElAarag [27] developed a algorithm that generates level sketches from the expansion of a tree which represents the level sketch. Baron [28] presented five different algorithms which are combinations of Room-Generation and Corridor-Generating algorithms, they are: Random Room Placement and Random Point Connect; Random Room Placement and Drunkard's Walk; BSP Room Placement and Random Point Connect; BSP Room Placement and Drunkard's Walk; and SP Room Placement and BSP Corridors. Hilliard, Salis and ElAarag [29] introduced two algorithms: Span* that generates a set of of random points which becomes rooms and connect them with Prim's algorithm [33]; and Growth that generates a level by a set of points which becomes rooms and new random rooms are connected with them. Sampaio et al. [31] presented a solution to level generation based only on Random Room Placement.

Only 4 papers employed hybrid approaches to generate dungeon levels. Ashlock [25] and Pech et al. [12] [13] used GA to evolve CA rules. More specifically, CA rules generates the maze-like dungeons levels, while the GA evolved the CA rules to satisfy some level's constraints. Togelius, Justinussen and Hartzen [21] presented a combination of EA-ASP, with each of them having different responsibilities. While the ASP is responsible to ensure the level viability (well-formedness, playability and winnability), the EA evolves the level in order to optimize the challenge and skill differentiation.

The division of responsibility is a common feature in constructive approaches. However, due to the high number of contents or the representation of the level, one search-based algorithm rely on this feature to facilitate its the development and to improve its results. Liapis [4] introduced two similar search-based algorithms: the first one generates the dungeon sketch and the second one generates the rest of the dungeon.

Next, we analyze the papers in terms of the Togelius's taxonomy. We begin with the *Generation Time* (T1). All papers presented offline generation. This result was also found by van der Linden, Lopes and Bidarra [2]. Clearly we need to begin to explore online solutions, since it has the advantage of saving memory and the content is generated while the player is progressing.

In terms of *Generation Control* (T2), only 2 papers presented solutions with random seeds as generation control [17] [31]. The solutions of the remaining papers were controlled by a set of parameters. The overwhelming choice for set of parameters makes sense since they enable the human designer to have a finer control over the results.

Considering the *Generality* (T3), all but one presented solutions for generic level generation. Only the paper from Alvarez et al. [16] were adaptive in the sense that an human designer is responsible for selecting levels to be "evolved" by the automatic process. Ultimately, the human designer decides when the generation process stops. However, the existence of only one paper on adaptive approach does not mean that the generic approach generates perfect solutions. There are still some game features that influence the level structure and, therefore, might be generated in conjunction with the dungeon level.

In terms of *Random Choice* (T4), only 3 papers were deterministic [23] [28] [31]. The remaining papers were have stochastic approaches. Again, this result is somehow expected since it is more interesting to generate different contents to increase the replayability of the game.

Regarding the *Generation Method* (T5), we have 16 search-based solutions, which fall under the category generate-and-test (GT). This result indicates that ensuring the consistency of levels is a hard task for PCG since it is necessary to perform consistency tests to validate the generated level. That is why this approach is often much more computationally expensive than constructive approaches. The remaining papers are Constructive based (C).

In terms of *Content Authorship* (T6), automatic generation dominates over mixed-initiative. Only the two algorithms introduced in [14] [15] and one presented in [16] are mixed-initiative. These 3 papers are all from the same research group. Their work enables the human designer to (1) edit a level which was automatically generated, (2) observe this level evolve into a set of possible levels, (3) choose the best one, and, if necessary, (4) repeat the evolutionary cycle until he or she is satisfied. The Fig. 5 presents the tool (Eddy - Evolutionary Dungeon Designer) developed by their research group.

Next we list some quantitative results that accounts for the presence of the game elements described earlier in Section II-B.

Recall that a dungeon is ideally composed by rewards, challenges and puzzles. However, the results show us that most works do not present solutions that generate levels with all these game features simultaneously. Only 2 works generated dungeons with rewards [3] [27]. Only 3 papers generated dungeons with challenges [9] [17] [21]. Only 6 works presented levels with rewards and challenges [5] [4] [14] [15] [16] [31]. Finally, only [30] presented levels with all three elements, i.e. rewards, challenges and puzzles.

Only 8 works presented solutions that consider the level difficulty as an input information while generating a level [5] [9] [14] [15] [16] [26] [27] [31]. All these works allow the control over the level's difficulty by setting a difficulty value. As the player progresses through the levels, the difficult value also increases, which makes the generator create more

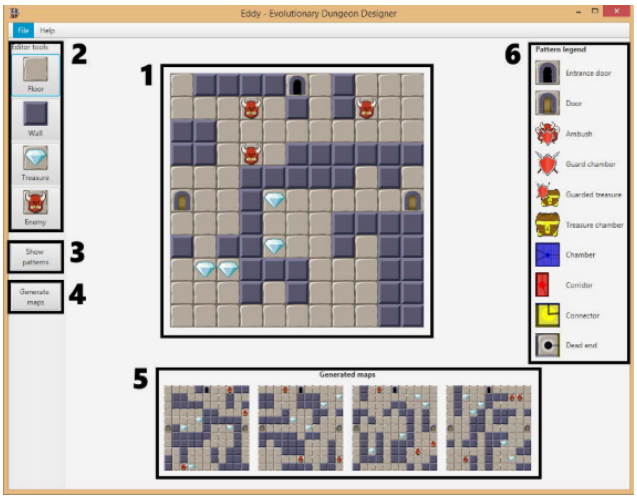


Figure 5. The mixed-initiative Baldwin et al. [15].

complex (hard) levels.

Few papers presented a level generator that supports the mechanic of barriers. There were two types of barriers: (1) a barrier that needs only one key to be unlocked; and (2) several barriers that are opened by a single key. The first kind of barrier was presented as a door (a barrier) that needs only one key to be unlocked in [3] [9] [32]. Lavender and Thompson [3] and van der Linden, Lopes and Bidarra [9] did not focused on mechanic of barriers per si. Instead, they treated it together with other kind of contents such as missions. Thus, some interesting characteristics of this mechanic could not be proper explored. Pereira, Prado and Toledo [32], however, present a solution that generates dungeon levels with barriers that are opened with corresponding keys, as depicted in Fig. 6 by the color association between doors and keys.

In contrast, Smith, Padget and Vidler [17] generated both the first and the second kind of barrier. For the second one, the player needs to take a bow (a weapon that works as a key) to defeat enemies (they work as barriers).

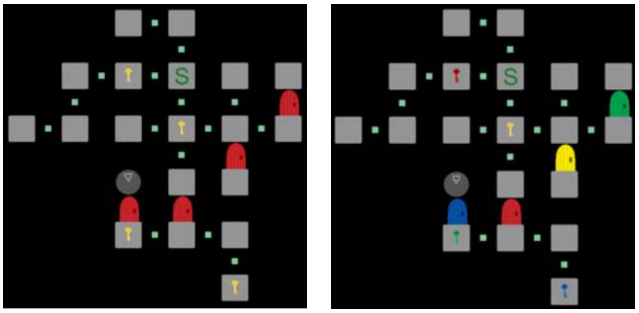
Barriers are clearly a promising research topic because they play an important role in engaging the player: overcoming barriers requires skill (to acquire the keys) and planning (to figure it out the sequence of keys that open up the barriers).

Most works presented solutions for generation of TDCL dungeons (16 out of 26). Only 6 works presented solutions that generate TDML dungeons [3] [9] [11] [24] [30] [32]. Only one work presented a solution for generation of SS dungeons. This means that there are still much to explore in the TDML dungeons and SS dungeons level generation researches.

Only 6 works presented solutions that perform area or room distinction to control what kind of content (e.g. enemies, rewards, etc.) will appear in the respective area or

TABLE II. COLLECTED DATA FROM ARTICLES II (LEVEL STRUCTURES: TOP-DOWN MANSION-LIKE OR TDML; TOP-DOWN CAVERN-LIKE OR TDCL; AND SIDE-SCROLLING DUNGEON OR SS)

Work	Reward	Challenge	Puzzle	Barrier	Difficulty	Area/Room distinction	Level Structure
[18]	-	-	-	-	-	-	TDCL
[19]	-	-	-	-	-	-	TDCL
[20]	-	-	-	-	-	-	TDCL
[11]	-	-	-	-	-	×	TDML
[21]	-	×	-	-	-	-	TDCL
[9]	-	×	-	×	×	-	TDML
[22]	-	-	-	-	-	×	TDCL
[23]	-	-	-	-	-	-	TDCL
[24]	-	-	-	-	-	-	TDML
[12]	-	-	-	-	-	-	TDCL
[25]	-	-	-	-	-	×	TDCL
[5]	×	×	-	-	×	×	SS
[13]	-	-	-	-	-	-	TDCL
[26]	-	-	-	-	×	-	-
[3]	×	-	-	×	-	-	TDML
[27]	×	-	-	-	×	-	-
[4]	×	×	-	-	-	×	TDCL
[15]	×	×	-	-	×	-	TDCL
[14]	×	×	-	-	×	-	TDCL
[28]	-	-	-	-	-	-	TDCL
[29]	-	-	-	-	-	-	TDCL
[30]	×	×	×	-	-	-	TDML
[17]	-	×	-	×	-	×	-
[16]	×	×	-	-	×	-	TDCL
[31]	×	×	-	-	×	-	TDCL
[32]	-	-	-	×	-	-	TDML



(a) Level without door distinction. (b) Level with door distinction

Figure 6. Examples of sketches generated by Pereira, Prado and Toledo [32].

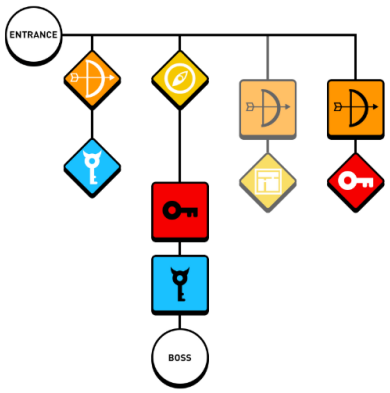


Figure 7. Examples of sketch generated by Smith, Padget and Vidler [17].

room. The works presented different ways to do the area or room distinction. Valtchanov and Brown [11] (Fig. 8) presented a solution that is designed to identify the entrance and event rooms. Smith, Padget and Vidler [17] (Fig. 7) present a solution that only differentiates the entrance and the boss' room from the other (they only generates the dungeon level sketch). Ashlock and McGuinness [22] (Fig. 9) presented a solution that assigns areas to the level's components such as the entrance, armory, and enemies location (goblins and magical beings). Ashlock [25] (Fig. 10) presented a solution that generates levels with different types of terrains, e.g. a level with ground, water, and lava. Baghdadi et al. [5] (Fig. 11) presented a solution that detects the type of room and uses this information to evolve the levels. The rooms in their work can become an ambush area (only enemies in a dead-end), a guard chamber (only enemies), a treasure chamber (only treasure), or a heavily guarded treasure chamber (a room full with enemies and treasure). And, finally, Liapis [4] presented a solution that may create 8 types of rooms (which he called segments): wall segment (blocked for the player); empty segment (just an empty room, with no monsters or treasures); simple segment (special room with few monsters and treasures); exit segment (a room with the level's exit); sparse challenge segment (contains more monsters than rewards); sparse reward segment (contains more rewards than monsters); high challenge segment (ops, only monsters here!); and high reward segment (bingo! just rewards). Liapis also distinguishes rooms based on the type of connection they support (see Fig. 12).

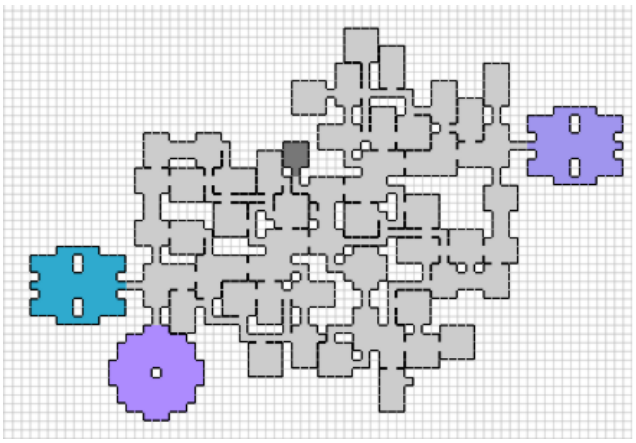


Figure 8. Level generated by Valtchanov and Brown [11].

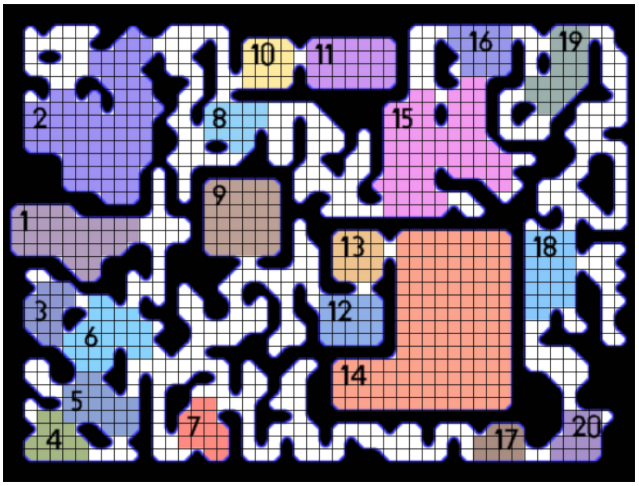


Figure 9. Level generated by Ashlock and McGuinness [22].

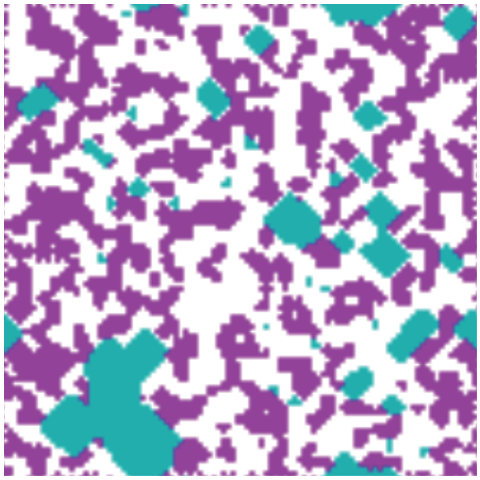


Figure 10. Level generated by Ashlock and McGuinness [25].

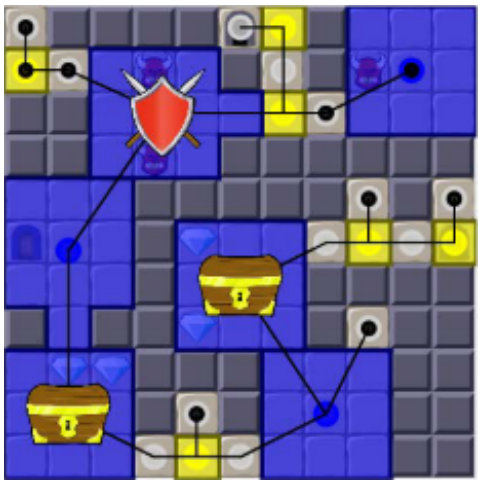


Figure 11. Level patterns of a level generated Baldwin et al. [15].

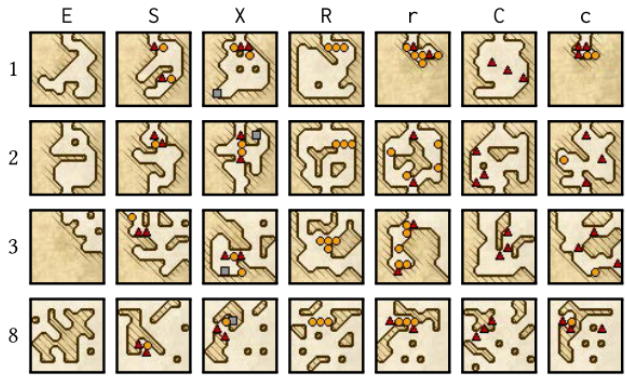


Figure 12. Some results by room type with four different kind of connections of Liapis' work (Source: [4]).

B. Unexplored Research Topics

When we tried to match the data from both tables we identified a few interesting points, highlighted in this section. The work purposed by Santamaria-Ibirika et al. [23], for instance, is the only one that generates the level sketch and the 3D dungeon environment. Their work generate levels with enemies and treasures. However, they do not support other game features, such as puzzles or barriers.

Alvarez et al. [16] were the only authors that presented a solution for adaptive dungeon generation. Their work used area/room distinction and geometric patterns to help with the adaptive generation. However, there are some game features they do not generate in their approach that are interesting to explore. Some of the game features could be used are several types of puzzles or mechanic of barriers. Besides, the Alvarez et al. work only generates TDCL dungeons. Therefore, we might explore alternatives to generate TDML and SS adaptive dungeons.

Alvarez et al. [16] work also presented a solution for mixed-initiative approach. Therefore, another unexplored

topic is the association of mixed-initiative approach with the puzzles and mechanic of barriers features and the TDML and SS dungeons.

Surprisingly, only 4 papers (from the total of 26 papers reviewed) presented dungeons with barriers [3] [9] [17] [32], exclusively focused on TDML dungeon generation. Again, new solutions that support barriers for TDCL or SS dungeon levels might be promising.

C. Analysis Summary

The follow list summarizes the analysis of our survey:

- Interesting characteristics:
  - Most dungeon level generators favors solutions that are game genre agnostic;
  - The search-based approach is predominant;
  - Few works presented combination of approaches;
  - It may be useful to separate the generation process in steps, even for the search-based algorithms;
  - It might be helpful to differentiate rooms during the generation process, so that the method might assign different purpose and/or content to rooms.
- The found major problems:
  - Few works presented solutions for PDG of truly 3D levels;
  - Just one work presented a solution for Platform 2D dungeons;
  - Not a single paper supported the balancing of the level's difficulty;
  - Few works considered barriers;
  - Few works distinguished areas and/or room;
  - Only one work presented a solution for dungeon level generation as defined by van der Linden, Lopes and Bidarra [2];
  - Not a single paper supported online generation;
  - Only one work supported adaptive generation;
  - Few works presented were mixed-initiative approaches.

IV. Conclusion

In this paper, we surveyed peer-reviewed research papers on procedural dungeon generation. Our review pointed out that there is a clear preference for search-based algorithms. This behavior probably happens due to the high number of constraints that are involved in dungeon level generation. Most works disassociate the level generation from the game genre. However, we believe that it might be important to define the game genre a priori because it, in turn, might narrow down the contents and constraints involved in the level generation process.

From the overview analysis, we identified nine open problems and, some unexplored research topics related to PDG. In terms of open problems we highlight: (1) few works presented solutions for PDG of 3D levels; (2) few works presented solutions for levels with the mechanic of barriers; and (3) few works presented solutions for mixed-initiative approach. In terms of research topics we underline the lack of more PDG solutions that support barriers and all the complexities that comes with it, specially when applied to top-down mansion-like, and side-scrolling dungeons. Also, we need to further explore adaptive and mixed-initiatives approaches, because they present the potential to generate complex and challenging levels.

We conclude that the procedural generation of dungeons still remains a complex problem. We have suggested a few research problems to investigate. Some works were difficult to evaluate due to the lack of information. To avoid these problems in the future we suggest that published papers should: (1) define the game genre or the specific game which the level will be generated for; (2) define the level structure (TDML, TDCL, SS, or any other); (3) classify the solution based on the Togelius's PCG taxonomy, and; (4) define the generation problem mathematically, if it is possible. These suggestions will hopefully improve the overall understanding of the field, as well as help us keep track of open problems and map important avenues of research.

References

[1] J. Togelius, N. Shaker, and M. J. Nelson, "Introduction," in *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*, N. Shaker, J. Togelius, and M. J. Nelson, Eds. Springer, 2016, pp. 1–15.

[2] R. van der Linden, R. Lopes, and R. Bidarra, "Procedural generation of dungeons," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6, no. 1, pp. 78–89, 2014.

[3] T. Thompson and B. Lavender, "A generative grammar approach for action-adventure map generation in the legend of zelda," in *Artificial Intelligence Simulation and Behaviour*. https://arro.anglia.ac.uk/700077/, 2016.

[4] A. Liapis, "Multi-segment evolution of dungeon game levels," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2017, pp. 203–210.

[5] W. Baghdadi, F. S. Eddin, R. Al-Omari, Z. Alhalawani, M. Shaker, and N. Shaker, "A procedural method for automatic generation of spelunky levels," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2015, pp. 305–317.

[6] C. Adams and S. Louis, "Procedural maze level generation with evolutionary cellular automata," in *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on*. IEEE, 2017, pp. 1–8.

[7] N. Chomsky, *Language and mind*. Cambridge University Press, 2006.

[8] G. Rozenberg, *Handbook of Graph Grammars and Comp.* World scientific, 1997, vol. 1.

[9] R. Van der Linden, R. Lopes, and R. Bidarra, "Designing procedurally generated levels," in *Proceedings of the the second workshop on Artificial Intelligence in the Game Design Process*, 2013.

[10] F. W. Glover and G. A. Kochenberger, *Handbook of metaheuristics*. Springer Science & Business Media, 2006, vol. 57.

[11] V. Valtchanov and J. A. Brown, "Evolving dungeon crawler levels with relative placement," in *Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering*. ACM, 2012, pp. 27–35.

[12] A. Pech, P. Hingston, M. Masek, and C. P. Lam, "Evolving cellular automata for maze generation," in *Australasian Conference on Artificial Life and Computational Intelligence*. Springer, 2015, pp. 112–124.

[13] A. Pech, M. Masek, C.-P. Lam, and P. Hingston, "Game level layout generation using evolved cellular automata," *Connection Science*, vol. 28, no. 1, pp. 63–82, 2016.

[14] A. Baldwin, S. Dahlskog, J. M. Font, and J. Holmberg, "Towards pattern-based mixed-initiative dungeon generation," in *Proceedings of the 12th International Conference on the Foundations of Digital Games*. ACM, 2017, p. 74.

[15] ——, "Mixed-initiative procedural generation of dungeons using game design patterns," in *Computational Intelligence and Games (CIG), 2017 IEEE Conference on*. IEEE, 2017, pp. 25–32.

[16] A. Alvarez, S. Dahlskog, J. Font, J. Holmberg, and S. Johansson, "Assessing aesthetic criteria in the evolutionary dungeon designer," in *Proceedings of the 13th International Conference on the Foundations of Digital Games*. ACM, 2018, p. 44.

[17] T. Smith, J. Padget, and A. Vidler, "Graph-based generation of action-adventure dungeon levels using answer set programming," in *Proceedings of the 13th International Conference on the Foundations of Digital Games*. ACM, 2018, p. 52.

[18] D. Ashlock, C. Lee, and C. McGuinness, "Search-based procedural generation of maze-like levels," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 260–273, 2011.

[19] C. McGuinness and D. Ashlock, "Decomposing the level generation problem with tiles," in *Evolutionary Computation (CEC), 2011 IEEE Congress on*. IEEE, 2011, pp. 849–856.

[20] D. Ashlock, C. Lee, and C. McGuinness, "Simultaneous dual level creation for games," *IEEE Computational Intelligence Magazine*, vol. 6, no. 2, pp. 26–37, 2011.

[21] J. Togelius, T. Justinussen, and A. Hartzen, "Compositional procedural content generation," in *Proceedings of the The third workshop on Procedural Content Generation in Games*. ACM, 2012, p. 16.

[22] D. Ashlock and C. McGuinness, "Automatic generation of fantasy role-playing modules," in *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*. IEEE, 2014, pp. 1–8.

[23] A. Santamaria-Ibirika, X. Cantero, S. Huerta, I. Santos, and P. G. Bringas, "Procedural playable cave systems based on voronoi diagram and delaunay triangulation," in *Cyberworlds (CW), 2014 International Conference on*. IEEE, 2014, pp. 15–22.

[24] A. J. Smith and J. J. Bryson, "A logical approach to building dungeons: Answer set programming for hierarchical procedural content generation in roguelike games," in *Proceedings of the 50th Anniversary Convention of the AISB*, 2014.

[25] D. Ashlock, "Evolvable fashion-based cellular automata for generating cavern systems," in *Computational Intelligence and Games (CIG), 2015 IEEE Conference on*. IEEE, 2015, pp. 306–313.

[26] W. Forsyth, "Globalized random procedural content for dungeon generation," *Journal of Computing Sciences in Colleges*, vol. 32, no. 2, pp. 192–201, 2016.

[27] J. Hell, M. Clay, and H. ELAarag, "Hierarchical dungeon procedural generation and optimal path finding based on user input," *Journal of Computing Sciences in Colleges*, vol. 33, no. 1, pp. 175–183, 2017.

[28] J. R. Baron, "Procedural dungeon generation analysis and adaptation," in *Proceedings of the SouthEast Conference*. ACM, 2017, pp. 168–171.

[29] N. Hilliard, J. Salis, and H. ELAarag, "Algorithms for procedural dungeon generation," *Journal of Computing Sciences in Colleges*, vol. 33, no. 1, pp. 166–174, 2017.

[30] D. Karavolos, A. Liapis, and G. N. Yannakakis, "Evolving missions for dwarf quest dungeons," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, Sept 2016, pp. 1–2.

[31] P. Sampaio, A. Baffa, B. Feijó, and M. Lana, "A fast approach for automatic generation of populated maps with seed and difficulty control," in *2017 16th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*. IEEE, 2017, pp. 10–18.

[32] L. T. Pereira, P. V. Prado, and C. Toledo, "Evolving dungeon maps with locked door missions," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–8.

[33] R. C. Prim, "Shortest connection networks and some generalizations," *The Bell System Technical Journal*, vol. 36, no. 6, pp. 1389–1401, 1957.