


Procedural Dungeon Generation: A Survey

Breno M. F. Viana  [Universidade de São Paulo | bmfviana@gmail.com]

Selan R. dos Santos  [Universidade Federal do Rio Grande do Norte | selan.santos@ufrn.br]

Abstract Procedural content generation (PCG) is a method of content creation entirely or partially done by computers. PCG is popularly employed in game development to produce game content, such as maps and levels. Representative examples of games using PCG are *Rogue* (1998), which introduced the rogue-like genre, and *No Man's Sky* (2016), which generated whole worlds with fauna and flora. PCG may generate final contents, ready to be added to a game, or intermediate contents, which might be polished by human designers or work as an input level sketch to be interpreted by a level translator. In this paper, we survey the current state of procedural dungeon generation (PDG) research, a PCG subarea, applied in the context of games. For each work we selected in this survey, we examined and compared how they created game features, what type of level structure and representation they propose, which content generation strategy they applied, and, finally, we classify them according to the taxonomy of procedural content generation proposed by Togelius et al. (2016). The most relevant findings of our survey are: (1) PDG for 3D levels has been little explored; (2) few works supported levels with barriers, a game mechanic which temporarily blocks the player progression, and; (3) mixed-initiative approaches, i.e., software that helps human designers by making suggestions to the levels being created, are little explored.

Keywords: Survey, Procedural Content Generation, Dungeon, Game

1 Introduction

Togelius et al. (2016) defined Procedural Content Generation (PCG) as computer software capable of creating “*game content on its own, or together with one or many human players or designers.*” Over the years, PCG has become a valuable asset for the game development process because it may bring several benefits, such as reducing the high cost of production of game features by reducing the need of human designers to generate content; helping human designers to increase their creativity and productivity; controlling the game difficulty or help game balancing, or both, and; increasing the replay value of a game by providing unexpected content, for instance.

Dungeon level generation in games is a great example of how PCG can be very useful, particularly when it supports the creation of different dungeons every time the game is replayed. In this paper’s context, we are following the definition offered by van der Linden et al. (2014), who defined dungeons as labyrinthine environments that offer structured gameplay progressions through interrelated rewards challenges, and puzzles.

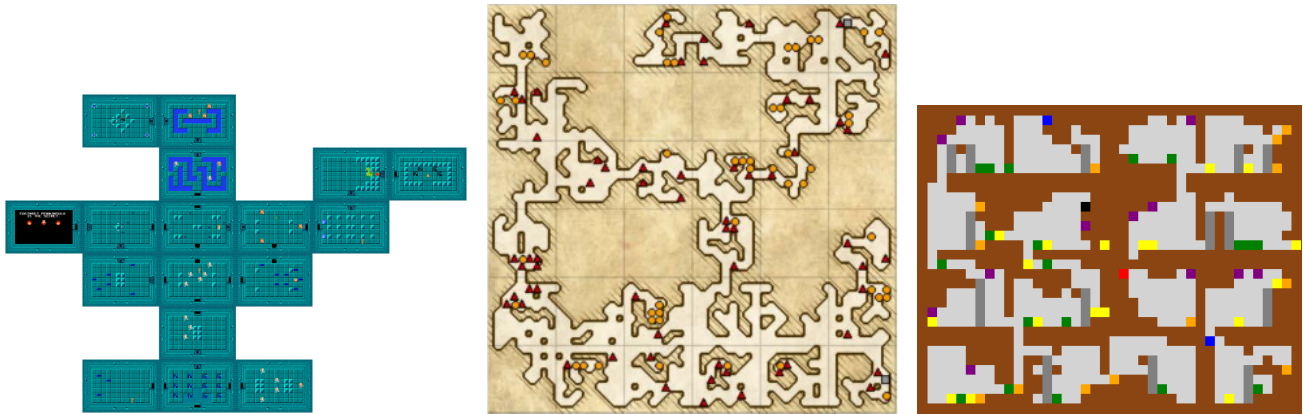
The game community usually classifies games that have computer-generated dungeons as rogue-like/rogue-lite games. Some examples are: *Rogue* (Toy and Wichman, 1980), the game which introduced the rogue-like genre, *Diablo* (Blizzard Entertainment, 1996), *The Binding of Isaac* (McMillen and Himsl, 2011), *Don't Starve* (Klei Entertainment, 2013), *Crypt of the NecroDancer* (Brace Yourself Games, 2015), *Moonlighter* (Digital Sun, 2018) and *Dead Cells* (Motion Twin, 2018).

In 2014 van der Linden et al. published a survey on Procedural Dungeon Generation (PDG). They compared different approaches to content generation and tried to understand how the control works in the surveyed methods. However, they did not attempt to classify the methods under a unified

taxonomy. This shortcoming motivated us to produce a review exclusively focused on PDG and target at classifying the research under the PCG taxonomy defined by Togelius et al. (2016). We believe this categorization might be useful since it helps us understand the approaches’ behavior better and identify shortcomings, strengths, and tendencies (or lack thereof) in the recent research body aimed at PDG.

In this paper, we extend our last survey on PDG – Viana and dos Santos (2019) – to improve the classification of some works and provide an updated review of state of the art by including recent papers in our review. We selected papers from Scopus, ACM Digital Library, and IEEE Xplore. We considered works related to the generation of dungeons, maze-like, cavern, and other types of levels similar to the dungeons as defined by van der Linden et al. (2014). Our review analyzed the selected papers under the following categories: (1) the *game genre*, such as RPG, rogue-like, action-adventure, 2D platform, etc.; (2) the *game space*, i.e., if the generated space is the level layout or just a level sketch; (3) the *game dimensionality* (two- or three-dimensional); (4) the *level representation*, i.e., which structures are used to represent the level; (5) the *solution strategies* adopted by the PDGs; (6) how well they fit the PCG *taxonomy* (Togelius et al., 2016), and; (7) the *level contents*, i.e., if the level is composed by rewards, challenges, etc.

The main contributions of this paper are the findings that resulted from our classification, of which we highlight the following: (1) few works presented solutions for 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 approaches when a human design steers the computer content generation. Other contributions are a revised version of Togelius’s PCG taxonomy by reorganizing some of the original categories and improving its descriptions, and the suggestion



(a) TDML dungeon. Source: Lavender and Thompson (2017).

(b) TDCL dungeon. Source: Liapis (2017).

(c) SS dungeon. Source: Baghdadi et al. (2015).

Figure 1. Types of 2D dungeon structures: top-down mansion-like (TDML), top-down cavern-like (TDCL), and side-scrolling (SS) dungeons.

of potential PDG related research problems to investigate.

This paper is structured as follows. Section 2 presents the definitions of dungeon levels in games and defines some game elements, features, and mechanics that we were interested in while evaluating the papers. We also briefly introduce the revised taxonomy of procedural content generation, and, finally, we overview some of the generation strategies we identified during the survey. In Section 3, we overview the current state of PDG under the parameters introduced in Section 2 by presenting the quantitative results of our survey. In Section 4, we discuss, based on the survey results, limitations, tendencies, and new possibilities of research to pursuit. Finally, in Section 5, we conclude the paper by providing some suggestions for open challenges in the field.

2 Background

Surveys are important mechanisms to understand an area better and help to map problems, failures, and successes, aiming at the progress of knowledge (Mulrow, 1994; Moher et al., 2009). They are relevant for researchers because they might help them to: (1) understand existing techniques within a common context or framework (e.g., taxonomy); (2) identify how to improve existing methods; (3) identify niches of little-explored methods; and, (4) discover new methods, perhaps combining elements or strategies that have been used successfully to date.

Therefore, this section presents the definitions of some game features and mechanics that we were interested in and the definition of dungeon levels. We also present the revised PCG taxonomy introduced by Togelius et al. (2016) and an overview of the solution strategies found in our survey.

2.1 Game Features

According to van der Linden et al. (2014), dungeons are labyrinthine environments mostly composed of rewards, challenges, and puzzles, distributed over the level to offer highly structured gameplay progressions. Several games of different genres use dungeons in some parts – e.g., *Pokémon* (Nintendo, 1996) – or the entire game – e.g., *The Binding of Isaac* (McMillen and Himsl, 2011).

Game genres are classifications of games based mainly on the gameplay and how players interact with the game, i.e., the game space and the game dimension. The game space is the environment where the gameplay happens through the user interaction, e.g., the game levels of a level-based game. The game dimension refers to the dimensionality of the game space, which usually are two- (2D), two-and-half- (2.5D), or three-dimensional (3D). The game space of dungeons usually tries to simulate real-world labyrinth-like environments, such as caves, intricate buildings, medieval dungeons, among others. The game’s genre, space, and dimension are essential features that directly affect the game level representation and the automatic generation method a PCG-based game may choose to follow.

In this survey, we have identified works that generate the *layout* of the level space, the *sketch* of the level space – i.e., meta-data that represents the layout of the level space that can be translated into an actual level space – or both, at different stages of the generative process. A layout of the level space is usually represented by grids or shapes, whereas a sketch of the level space is often represented by grids, graph nodes, or grammar strings.

Concerning the works that generated 2D dungeons, we found three types of 2D dungeon structures: (1) top-down mansion-like dungeon (TDML) – example in Figure 1a –, e.g., *The Legend of Zelda*’s dungeons (Nintendo, 1986); (2) top-down cavern-like dungeon (TDCL) – example in Figure 1b –, e.g., *Pokémon*’s dungeons (Nintendo, 1996), and; (3) side-scrolling dungeon (SS) – example in Figure 1c –, e.g., *Spelunky*’s dungeons (Yu, 2008). In our survey, we also consider levels with both rooms and corridors as TDCL dungeons. It is worth mentioning that we could only find three papers that tackled 3D dungeons: Santamaria-Ibirika et al. (2014) dig terrain to create the dungeon; Baron (2017) only translates 2D level into a 3D environment; and, Antoniuk et al. (2018) that generated 3D dungeon rooms. Since only the first work truly generated connected 3D dungeons, we decided not to create an exclusive category just for 3D dungeon generation.

Next, we introduce some *gameplay elements* usually present in dungeon levels.

- **Room** is a structural part of a level that may be connected to other rooms by doors or corridors. Figure 1a

presents an example of a level structured into several rooms.

- **Item/Skill** is a collectible element or learned ability that grants to the player some gameplay benefit, such as a health kit that regenerates some of the player's health or when a player earns some limited capacity to fly over obstacles.
- **Barrier** is a game feature that temporarily blocks the player from reaching certain regions on a level. Overcoming a barrier usually requires the player to collect one or multiple items or to learn some special skills that unlock or remove the blockage.
- **Reward** is something the player is encouraged to locate and acquire through some effort. Items, skills, money, tools, for instance, are typical examples of rewards. Rewards may or may not be mandatory goals (originally defined by van der Linden et al. (2014)).
- **Challenge** is an obstacle that hinders the player's progression. In the dungeon context, challenges are usually treated as battle challenges or traps (originally defined by van der Linden et al. (2014)).
- **Puzzle** is an intellectual obstacle that requires the player some reasoning to solve it. Puzzles usually involve reorganizing items in a specific sequence to unblock a barrier or figuring out how to use game elements in an unconventional way, such as moving pieces of furniture to reach a tunnel initially out of reach. Sometimes, a puzzle is hidden in the game level, and the player has to find it first before attempting to solve it (originally defined by van der Linden et al. (2014)).

2.2 PCG Taxonomy

In this section, we present the PCG taxonomy defined by Togelius et al. (2016). We renamed each title of the classifications in an attempt to improve their understanding. The taxonomy defines seven axes to classify procedural generation approaches:

- **Content Requirement** defines if the generated content is *necessary* for the gameplay or if it is *optional*. For instance, a key may be necessary to open the door to the boss' room. In contrast, the level may have a blocked room that the player can ignore and yet finish the level.
- **Outcome Randomness** defines the way that choices are made during the generation process. An approach is *deterministic* when it generates the same content with the same set of parameters. Alternatively, it is *stochastic* when it generates different contents, even with the same set of parameters. Although this classification seems silly at first – since one of the PCG goals is to provide variety –, deterministic approaches may be useful, as discussed later in this paper.
- **Generation Time** defines when the content is generated. An approach is *offline* when it generates the content before the gameplay (i.e., when the player is playing the game), or it is *online* when it generates the content during the gameplay. For instance, the levels on *Moonlighter* (Digital Sun, 2018) are generated before the gameplay. On the other hand, the levels of any End-

less Runner are generated during the gameplay.

- **Generation Control** defines how to parameterize the approaches and dimensions of control. The control of an approach may be made by *random seeds* that allows only one dimension of control. Alternatively, the control may be made by a *set of parameters* that allows more dimensions of control. For instance, a set of parameters may be composed of width, height, and difficulty.
- **Generality** defines the audience that the content is being generated for. An approach is *generic* when the content is generated for several players. Alternatively, it is *adaptive* when the content is generated for a specific player. For instance, *Left4Dead 2* (Valve Corporation, 2009) send zombie hordes of different sizes according to the player's gameplay rhythm.
- **Generation Method** defines how the generation can be performed. An approach is *constructive* when it generates content in only one pass without validation of the generated content. Alternatively, an approach is *generate-and-test* when it alternates the generation and the consistency or evaluation test or both of the content. Constructive approaches are usually faster than generate-and-test techniques. However, they are more susceptible to content inconsistency. Within the generate-and-test, a nested classification called search-based PCG defines the PCG approaches in which the test function grades the generated content. For instance, in Evolutionary Algorithms (EA), the grading function is the fitness function. Another nested class of generate-and-test is the solver-based PCG (Summerville et al., 2018). For instance, we classify as solver-based PCG the approaches built with Answer Set Programming (ASP). Summerville et al. (2018) described a new PCG paradigm based on Machine Learning (ML) where the content is generated directly from ML models. However, Reinforcement Learning (RL) cannot be classified as ML-based. The contents generated by this approach are not created by a model but by agents instead. Charity et al. (2020) classify these approaches as RL-based.
- **Content Authorship** defines the degree of human interference in the generative process. An approach is *automatic generation* when only the computer generates the content. Alternatively, it is *mixed-initiative* when it allows the designer (or the gamer) to have finer control over content (not only by parameters) that is being generated. Usually, mixed-initiative programs are dedicated to increasing the productivity of game designers.

2.3 Solution Strategies

In this section, we present an overview of the solution strategies from the papers selected for this survey. We begin with the Cellular Automata (CA). CAs were created as a model of biological self-reproduction (Wolfram, 1983). A CA-based algorithm follows a set of rules to define cell-state transitions based on the cells' neighbors in a grid (Adams and Louis, 2017). In PCG, CAs have been adapted to generate game levels (Johnson et al., 2010); in particular, they work quite well to create cavern-like levels.

Generative Grammars (GG) are systems of grammatical

rules that generate words (Chomsky, 2006). Based on GGs, other kinds of grammars have been developed, e.g., Graph and Shape Grammars. Graph Grammars define rules for node and edge transformations in order to generate new graphs (Rozenberg, 1997). The graphs generated by these systems can be used to generate several types of game content, e.g., the works of Lavender and Thompson (2017) and van der Linden et al. (2013). On the other hand, Shape Grammars define rules that transform geometric two- or three-dimensional shapes (Halatsch et al., 2008). This kind of GG is suitable for the generation of level layouts.

Voronoi Diagram (or Voronoi Tessellation) is a method of dividing space into regions based on a set of points (Voronoi, 1908). This method is applicable in several fields. In PCG, however, it is mostly used for the generation of terrains (Olsen, 2004). Delaunay Triangulation is a technique that triangulates the space based on a set of points (Delaunay et al., 1934). This technique was also used to generate terrains (De Kok et al., 2007).

Since PCG can be interpreted as Constraint Satisfaction Problems (CSPs), we can use search algorithms to generate content – these are the search-based PCG approaches. Simulated Annealing is a metaheuristic based on annealing in metallurgy. This technique involves heating and controlled cooling of a material (Van Laarhoven and Aarts, 1987). Evolutionary Computation is a family of algorithms based on theories of evolution. They are widely used for PCG purposes. The most common is the Genetic Algorithms (GA), a metaheuristic that is based on the neo-Darwinian theory of evolution (Gendreau et al., 2010). GAs evolves lists that are usually represented as strings. Similar to the latter approach, there is Genetic Programming (GP) that evolves individuals represented as syntax trees (Gendreau et al., 2010). There is also an approach that evolves two populations of feasible and infeasible individuals, FI2Pop GA (Kimbrough et al., 2005).

Still in evolutionary computation, a new subclass has emerged, the Quality Diversity (QD) algorithms. This new paradigm prioritizes the search for a set of diverse individuals first and then maximizes their quality (Pugh et al., 2016). QD approaches are especially interesting for PCG due to the content variety provided by them. No wonder that some researchers started to explore QD algorithms on their PCG solutions, as reviewed by Gravina et al. (2019). Some QD algorithms are: Deluged Novelty Search Local Competition (DNLSC); MAP-Elites, and; Constrained MAP-Elites (a combination of MAP-Elites and FI2Pop GA).

PCG approaches also have been developed using solver-based solutions. Answer Set Programming (ASP) is a kind of declarative logic programming created to solve search problems (Smith et al., 2018). Since, as we said, PCG problems can be mapped as search problems, ASP may be a useful tool for generating content. Some content constraints – e.g., level layout constraints – can also be interpreted as integer linear constraints. Therefore, the generation of these kinds of contents can be solved by Satisfiability Modulo Theories (SMT) solvers. Whitehead (2020) describes this approach better and its use of STM for PCG purposes.

Regarding the PCG ML paradigm, several approaches models can be used *directly* to generate game content (Summerville et al., 2018). However, in this survey, we are in-

terested only in Generative Adversarial Networks (GAN). GANs are generative models that are trained to produce content based on the learned content (Gutierrez and Schrum, 2020). As for RL-based algorithms, we found a work that used Reinforcement Learning (RL) to create agents that are used to generate content (Sutton and Barto, 2018).

It is worth mentioning that some works based on constructive approaches have exclusive steps that depend only on the content type. These exclusive algorithms are usually based on the random or constrained random placement of game spaces. For instance, the Drunkard’s Walk (or Random Walk Simulation) in PCG may work as a digger to create levels.

Finally, all of the described strategies can be combined to take benefit of their advantages. Hybrid approaches act in a way to generate different contents or the same content in different steps of the generative process. For instance, a hybrid approach can generate level sketches using Graph Grammars and generate level rooms using CA rules using previously generated sketches (Gellel and Sweetser, 2020).

3 Survey

We carried out this survey based on the guidelines of the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) model (Moher et al., 2009; Liberati et al., 2009). Such methodology is widely applied in several research fields for gathering comprehensive literature reviews (Harper et al., 2021; Moray et al., 2021; Assadi et al., 2020; Masotta et al., 2020). The PRISMA states the following steps to perform a systematic literature review: (1) literature search; (2) selection of eligible papers; and (3) data extraction and summarizing.

We collected data from Scopus¹, ACM Digital Library², and IEEE Xplore³. Table 1 presents search terms divided into three groups we defined to compose the search string: PCG, level, and game.

Table 1. Search terms. The terms were divided into three groups we defined to compose the search string: PCG, level, and game

| PCG Group | Level Group | Game Group |
|-------------------------------|-------------------------------|------------|
| PCG | Dungeon | Game |
| Procedural Generation | Labyrinth | |
| Procedural Content Generation | Maze | |
| Procedural Generated Content | Cave | |
| Level Generation | Cavern | |
| Procedural Level Generation | Rogue-like | |
| Automatic Generation | | |
| | Procedural Dungeon Generation | |
| | Procedural Playable Cave | |
| | Procedural Playable Cavern | |

We build the search string by connecting each word from each term group with the boolean operator *OR*. Then, we combined the exclusive term of PCG and Level group with the operator *AND* and, then, the shared terms with *OR*. Finally, we combined the resulting string with the Game group with *AND*. Below, we present the final search string.

¹Scopus: <https://www.scopus.com/>.

²ACM Digital Library: <https://dl.acm.org/>.

³IEEE Xplore: <https://ieeexplore.ieee.org/>.

((((PCG OR “Procedural Generation” OR “Procedural Content Generation” OR “Procedural Generated Content” OR “Level Generation” OR “Procedural Level Generation” OR “Automatic Generation”) AND (Dungeon OR Labyrinth OR Maze OR Cave OR Cavern OR Rogue-like)) OR (“Procedural Dungeon Generation” OR “Procedural Playable Cave” OR “Procedural Playable Cavern”)) AND Game

To validate the search string, we verified whether the top 3 most relevant papers of the PDG scenario were returned as a valid result. These target papers are: Togelius et al. (2011) because this paper introduced the PCG taxonomy, and it is widely cited by the PCG papers; van der Linden et al. (2014) because it was the first survey on PDG; and, Johnson et al. (2010) because it is one of the most cited papers on the PDG scenario. Despite this, we are aware that the use of a search string has its own limitations. It is not possible to ensure that all the possible papers related to the subject we are interested in will be captured with a search string.

We selected conference and journal papers published in the last ten years (2010 to October 2020) in which their title and abstract specifically mention the generation of dungeon, cave, or maze-like levels. We also selected works that generated levels similar to the dungeons as defined by van der Linden et al. (2014) and used the GVG-AI framework⁴ to generate dungeon systems of Zelda-like games. Although one may argue that mazes are a type of labyrinths, we discarded any work that strictly generated mazes because they have a slightly different type of level structure. Mazes are puzzles that mainly create a labyrinthine path connection between a starting point and an end point, while dungeons are more complex environments that present rooms, doors, floors, enemies, among others (van der Linden et al., 2014). From now on, we call all of them just dungeon levels.

The search and the selection process resulted in a total 41 papers, 3 of which described and compared more than one dungeon generation approach. Therefore, this pool yielded a total of 52 different approaches for dungeon generation. The data we collected from these 41 papers were: the game genre; the game space, type of level that is generated (i.e., either the entire level layout or just a sketch); the level dimensionality; the level representation; the type of solution strategy to generate dungeons; the categories of the PCG taxonomy; and, the game features. The description for each of these data items was provided in Section 2. To classify the approaches’ game genre, dimensionality, representation, and game features of the levels generated by the works as follows: (1) based on what the authors themselves wrote (self-classification); (2) based on the methodology used by the works; and, (3) based on the images provided in the works. The Tables 2, 3 and 4 summarizes the collected data. In the following subsections, we present these data in detail.

3.1 Solution Strategy Classification

Game genre From the 41 papers, 18 of them described solutions target at specific *game genre*, whereas the rest were

game genre-agnostic. Only six of them specified the game the solutions were developed for (van der Linden et al., 2013; Baghdadi et al., 2015; Lavender and Thompson, 2017; Karavolos et al., 2016; Gutierrez and Schrum, 2020; Charity et al., 2020), and only the works from Pereira et al. (2018), Sheffield and Shah (2018), Goandy et al. (2020) and Gutierrez and Schrum (2020) developed game prototypes to collect player’s feedback about the level quality.

Level space A total of 25 papers generated directly the level layout. Valtchanov and Brown (2012), van der Linden et al. (2013), Karavolos et al. (2016), Forsyth (2016), Hell et al. (2017), Smith et al. (2018), Pereira et al. (2018), and Sheffield and Shah (2018) generated only the level sketches. However, only Forsyth (2016), Hell et al. (2017), and Smith et al. (2018) did not present a level translator for the level sketches they generated or translations to an existing game. The remaining papers presented solutions that generated both, first a sketch and then the level layout (Dormans and Bakkes, 2011; Smith and Bryson, 2014; Baghdadi et al., 2015; Lavender and Thompson, 2017; Liapis et al., 2015; Nepožitek and Gemrot, 2018; Gutierrez and Schrum, 2020; Gellel and Sweetser, 2020).

Level dimensionality A total of 30 papers presented solutions exclusively target at 2D dungeon level generation. For instance, Figure 1a, Figure 1b and Figure 1c presented examples of 2D dungeon levels generated by Lavender and Thompson (2017), Liapis (2017) and Baghdadi et al. (2015), respectively. Just Antoniuk et al. (2018) proposed the use of Shape Grammars, which is capable of generating 3D dungeon levels. Santamaria-Ibirika et al. (2014) and Baron (2017) described solutions that worked both 2D and 3D levels. Santamaria-Ibirika’s work is based on Voronoi Diagram and Delaunay Triangulation, enabling a less artificial level generation. Figure 2b presents an example of a level generated with their approach. The Baron’s work, however, does not generate true 3D levels. Instead, it generates a 2D text-based sketch which undergoes an extrusion process to become a 3D level (see Figure 3). Finally, the works from Valtchanov and Brown (2012), van der Linden et al. (2013), Karavolos et al. (2016), Pereira et al. (2018), and Sheffield and Shah (2018) generated level sketches and translated them into levels. van der Linden et al. (2013) and Karavolos et al. (2016) translated the sketches into 3D levels. Note, however, that these two works, in fact, did not generate true 3D structures. 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). Figure 4 presents examples from each paper. Finally, Forsyth (2016), Hell et al. (2017), and Smith et al. (2018) generated a high level representation (sketches) for a dungeon level that could neither be classified as 2D nor as 3D.

Level representation We identified 23 papers that represented their levels as grids. Liapis (2017), for instance, represented both the dungeon level and its rooms as grids. van der Linden et al. (2013), Forsyth (2016), Hell et al. (2017) and Smith et al. (2018) expressed the levels they generate as graphs, all of them created levels sketches and only van der Linden et al. (2013) translated them into real levels. Santamaria-Ibirika et al. (2014), Antoniuk et al. (2018) and

⁴GVG-AI (General Video Game AI) is a framework for AI competitions (<http://www.gvgai.net/>).

Whitehead (2020) represented their levels directly as shapes. Pereira et al. (2018) and Valtchanov and Brown (2012) described their levels as trees. Some works presented combinations of different representations. Ashlock and McGuinness (2014), Lavender and Thompson (2017), Karavolos et al. (2016), Alvarez et al. (2018), and Gellel and Sweetser (2020) use grids and graphs to represent the level. However, Ashlock et al. (2011a) generate the level as a grid and then extract the room adjacency graph to populate the level. The other authors generated missions as sketches to the levels. Dormans and Bakkes (2011) and Nepožitek and Gemrot (2018) represented level sketches as graphs and then generate the level shapes based on them. Smith and Bryson (2014) represented their levels as Answer Sets, working as sketches, and grids (the result of the solved Answer Sets). Gutierrez and Schrum (2020) generated both levels and their rooms, the level sketches are represented by graphs and converted into grids, and the rooms are directly represented as grids.

Algorithm strategy A group of 16 papers described evolutionary-based approaches, as follows: 5 papers relied on GA (Ashlock et al., 2011a,b; McGuinness and Ashlock, 2011; Ashlock and McGuinness, 2014; Baghdadi et al., 2015), 5 on FI2Pop GA (Liapis et al., 2015; Liapis, 2017; Baldwin et al., 2017a,b; Alvarez et al., 2018) 2 on GP (Valtchanov and Brown, 2012; Pereira et al., 2018), 2 on Constrained MAP-Elites (Alvarez et al., 2018; Charity et al., 2020), 1 on DNSLC (Melotti and de Moraes, 2018), and 1 on a general EA (Karavolos et al., 2016).

Another group of 15 presented distinct solutions, as follows. Johnson et al. (2010) suggested CA-based algorithms capable of generating infinite cave-like dungeon levels in real-time. van der Linden et al. (2013) created a Graph Grammar approach (Gameplay Grammar) that generates a missions' graph, which aims to build the level. Forsyth (2016) developed an algorithm that generates level sketches by simply placing rooms randomly. Hell et al. (2017) proposed an algorithm that generates level sketches from the expansion of a graph, representing the level sketch. Hilliard et al. (2017) introduced two algorithms: Span* that generates a set of random points which becomes rooms and connect them with Prim's algorithm (Prim, 1957); and Growth that generates a level by a set of points which becomes rooms, and new random rooms are connected with them. Baron (2017) presented five different algorithms, which are combinations of Room-Generation and Corridor-Generating algorithms: Random Room Placement and Random Point Connect; Random Room Placement and Drunkard's Walk; Binary Space Partitioning (BSP) Room Placement and Random Point Connect; BSP Room Placement and Drunkard's Walk; and SP Room Placement and BSP Corridors. Like Baron, Sampaio et al. (2017) presented a solution to level generation derived from Random Placement of rooms and corridors. Smith et al. (2018) implemented an ASP program capable of generating sketches as graphs for dungeon levels. Nepožitek and Gemrot (2018) applied Simulated Annealing search to find the best fit of room shapes for dungeon levels from graphs as sketches. Sheffield and Shah (2018) created RL agents to perform digging and distribute game features in a grid. Like the latter, Goandy et al. (2020) used a digger to generate dungeon levels, but they used the Drunkard's Walk algorithm instead.

Antoniuk et al. (2018) introduced a L-system (Shape Grammar) solution for the creation of 3D dungeon levels. Green et al. (2019) combined three algorithms for level creation (creators) and three for object placement (furnishers). Each set of algorithms is constraint-based, CA-based, and agent-based. Finally, Whitehead (2020) created SMT formulas and applied SMT solvers to distribute rooms of different sizes.

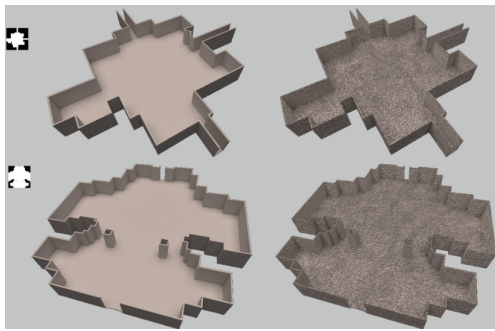
The last group of 11 papers employed hybrid approaches to generate dungeon levels. Togelius et al. (2012) presented a combination of ASP programs that generate the dungeon levels, with an EA that evolves values that define the level (e.g., level height and width). While the ASP is responsible for ensuring level viability (well-formedness, playability, and winnability), the EA evolves level values to optimize the challenge and skill differentiation. Dormans and Bakkes (2011) and Lavender and Thompson (2017) generated missions from Graph Grammars and then generated the dungeon levels with Shape Grammars. Similarly, Smith and Bryson (2014) generate ASP programs and run them in a Python solver to generate dungeon levels. Santamaria-Ibirika et al. (2014), as we already said, use Voronoi Diagram and Delaunay Triangulation algorithms to dig volumes and place game elements. Ashlock (2015), Pech et al. (2015), Pech et al. (2016) and Kreitzer et al. (2019) used GA to evolve CA rules which generate the dungeon levels. More specifically, CA rules generate the maze-like dungeon levels, while the GA evolved the CA rules to satisfy some level's constraints. Gutierrez and Schrum (2020) generated missions using Graph Grammars and use them as sketches to generate the dungeon levels. Moreover, they also applied GANs to generate the layout of each room of the levels. Finally, Gellel and Sweetser (2020) also generated missions by using Graph Grammars. To generate the levels, however, they introduced a new CA-inspired solution and applied a heuristic to maximize the quality of the levels.

The division of responsibility in generative processes is a common feature in constructive approaches. However, due to the high number of contents or the level representation, one search-based algorithm relies on this feature to facilitate its development and improve its results. Liapis (2017) introduced two similar search-based algorithms: the first one generates the dungeon sketch, and the second generates the rest of the dungeon (the dungeon rooms).

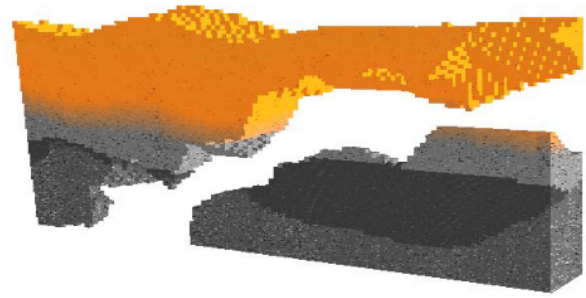
3.2 Taxonomy Classification

In this section, we analyze the papers in terms of Togelius' PCG taxonomy. We decided not to consider the *Content Requirement* category in this survey because this is a difficult feature to identify just by reading the paper and not actually playing the game.

Outcome Randomness Only three papers were clearly deterministic (Sampaio et al., 2017; Smith et al., 2018; Goandy et al., 2020). The remaining papers had stochastic approaches. This result is somehow expected since it is more interesting to generate different contents – dungeon levels in this case – to increase the replayability of the game. However, deterministic approaches may present useful applications in the game context. For instance, mixed-initiatives software could implement undo functionality without saving all the

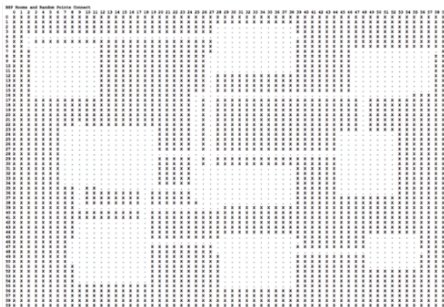


(a) Rooms of a dungeon generated by Antoniuk et al. (2018).

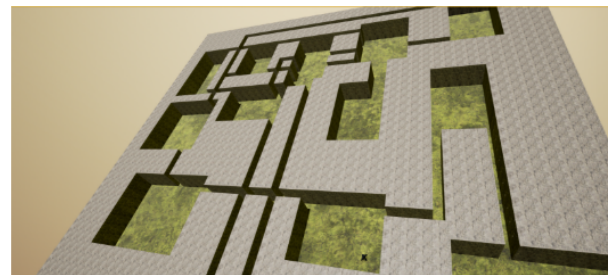


(b) Slice of a cave generated by Santamaria-Ibirika et al. (2014).

Figure 2. 3D dungeons.

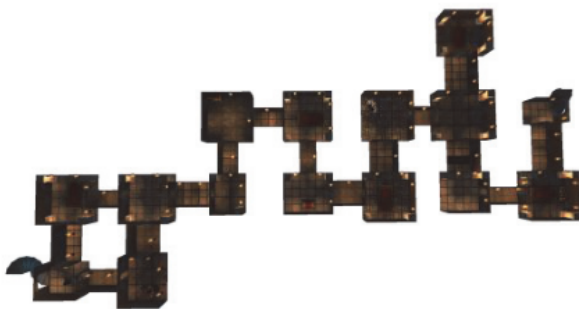


(a) 2D dungeon.

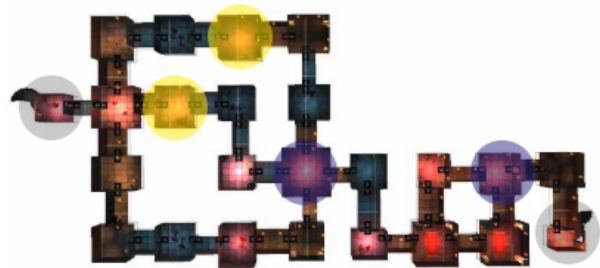


(b) 3D dungeon.

Figure 3. Dungeon levels generated by Baron (2017).



(a) Dungeon generated by van der Linden et al. (2013).



(b) Dungeon generated by Karavolos et al. (2016).

Figure 4. Dungeon levels generated for Dwarf Quest.

previous suggestions. These suggestions could be recreated by using the same set of parameters.

Generation Time All papers presented offline generation. This result was also found by van der Linden et al. (2014). The only work that might be considered online was Johnson et al. (2010). They applied CA to generate cavernous dungeons in real-time. However, there is no information about the level being generated during gameplay in their paper. Clearly, we need to begin to explore processing efficient online solutions since it has the advantage of saving memory as the content is generated while the player is progressing.

Generation Control Only two papers presented solutions with random seeds as the only dimension of the approach’s generation control (Smith et al., 2018; Goandy et al., 2020). The solution of Sampaio et al. (2017) has only two parameters: the random seed, and the level difficulty value. The solutions of the remaining papers were controlled by a set of parameters with more than two parameters. The overwhelming choice for sets of parameters makes sense since they enable

the human designer to have finer control over the results.

Generality All but two presented solutions for generic level generation. Only the papers from Alvarez et al. (2018) and Alvarez et al. (2019) were adaptive in the sense that a human designer is responsible for selecting levels to be “evolved” by the automatic process, edit and evolve to maintain the designer’s level aesthetic. Both works are from the same research group. To maintain the human designer aesthetics, their software – the Evolutionary Dungeon Designer (EDD), see Figure 5 – present the option of “freeze” cells of the grid, so they do not be able to evolve. Ultimately, the human designer decides when the generation process stops. Nonetheless, the existence of only two papers on the 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.

Generation Method We have identified 29 generate-and-test solutions. Ensuring dungeon levels’ consistency is a chal-

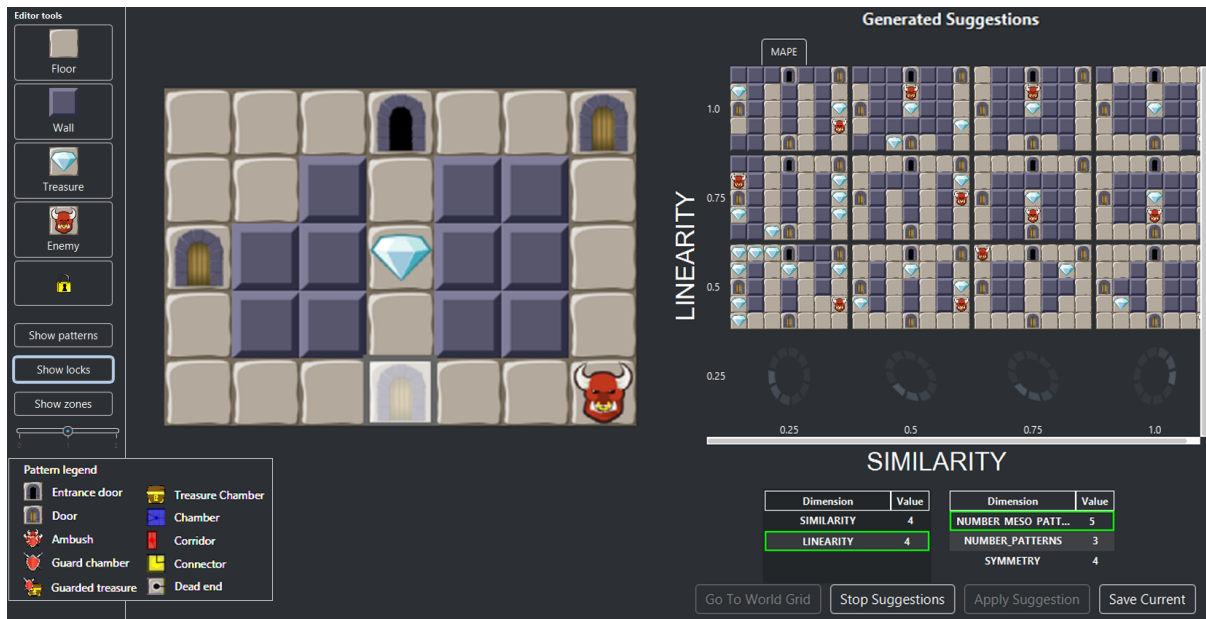


Figure 5. Evolutionary Dungeon Designer by Alvarez et al. (2019).

lenging task for PCG since it is necessary to perform consistency tests to ensure the generated level is playable. That is why this approach is often much more computationally expensive than constructive approaches. Within these 29 papers, 17 papers are solely search-based generative processes – e.g., Pereira et al. (2018). Nine of the generate-and-test works are hybrid, with emphasis on the works of Pech et al. (2015), Ashlock (2015), Pech et al. (2016) and Kreitzer et al. (2019) that hybridize a search-based approach (GA) with a constructive approach (CA). One of the hybrid works presented a ML-based solution for dungeon generation together with a CA-inspired approach. The other hybrid works present a simple test stage in their approach. The remaining papers are constructive solutions, and most of them are based on exclusive algorithms.

Content Authorship The automatic generation dominates over mixed-initiative. Only four papers presented solutions for mixed-initiative PCG software (Baldwin et al., 2017a,b; Alvarez et al., 2018, 2019). These four papers are all from the same research group. All of them present the same tool, the EDD. Their tool enables the human designer to (1) edit a level that 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. Figure 5 presents the EDD tool developed by their research group.

3.3 Game Features Classification

In this section, we list the quantitative results that account for the presence of the game elements described earlier in Section 2.1.

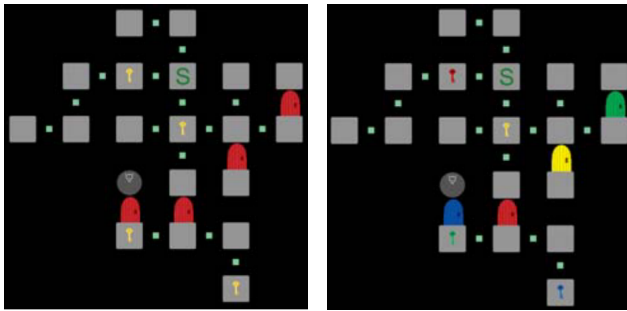
Recall that a dungeon is ideally composed of *rewards*, *challenges*, and *puzzles*. However, the results show us that most works, 31 out of 41, do not present solutions that simultaneously generate levels with all these game features⁵.

⁵Note that, in this context, we are considering the barrier mechanics as a type of puzzle.

When we examined which works were capable of creating dungeons with puzzles, rewards, and challenges, we identified these possibilities:

- Solutions that *only generate puzzles*: a total of 9 papers presented solutions for generating puzzles (Dormans and Bakkes, 2011; van der Linden et al., 2013; Lavender and Thompson, 2017; Karavolos et al., 2016; Smith et al., 2018; Pereira et al., 2018; Sheffield and Shah, 2018; Gutierrez and Schrum, 2020; Gellal and Sweetser, 2020), and only 3 papers, namely Karavolos et al. (2016), Smith et al. (2018), and Gutierrez and Schrum (2020), are capable of creating other puzzles besides the barrier-style puzzle.
- Solutions that *only offer rewards and challenges*: the work of Hell et al. (2017) only offers rewards, whereas Togelius et al. (2012) and Melotti and de Moraes (2018) created dungeons that only offer challenges. A total of 12 works are capable of creating levels with both rewards and challenges. They are: Ashlock and McGuinness (2014); Smith and Bryson (2014); Santamaria-Ibirika et al. (2014); Liapis et al. (2015); Baghdadi et al. (2015); Liapis (2017); Baldwin et al. (2017b,a); Sampaio et al. (2017); Alvarez et al. (2018, 2019); Green et al. (2019).
- Solutions that *offer all three elements*, puzzles, rewards, challenges, simultaneously: Dormans and Bakkes (2011), van der Linden et al. (2013), Lavender and Thompson (2017), Karavolos et al. (2016), Smith et al. (2018) and Sheffield and Shah (2018) presented levels with all three elements, if we consider barriers as puzzles. Although Charity et al. (2020) also generated levels with rewards, challenges, and barrier puzzles, they actually generated rooms, not entire levels as such. It is worth mentioning that only a few of these works provided finer control over the number of rewards or challenges.

Another relevant aspect we were interested in was to un-



(a) Level without door distinction. (b) Level with door distinction.
Figure 6. Examples of dungeon sketches generated by Pereira et al. (2018).

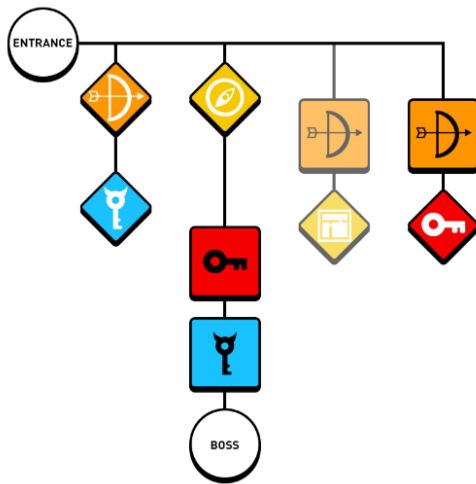


Figure 7. Example of a sketch generated by Smith et al. (2018).

Understand how solutions control the *level of difficulty* of their created levels. We found out that 10 works presented solutions that consider the level difficulty as input information while generating a level (Dormans and Bakkes, 2011; van der Linden et al., 2013; Baghdadi et al., 2015; Forsyth, 2016; Hell et al., 2017; Baldwin et al., 2017b,a; Sampaio et al., 2017; Alvarez et al., 2018, 2019). All these works support control over the level’s difficulty by setting a difficulty value.

Providing control over the level of difficulty of dungeons automatically generated by an algorithm may be considered a valuable asset for game designers. They would naturally appreciate having some influence over the outcome of these algorithms since the level difficulty is a feature that often directly impacts the player’s overall experience while playing a game.

The next aspect we wanted to consider is the *insertion of barriers* in a level as a gameplay mechanics. We learned that just 9 papers presented a level generator that supports barrier mechanics. 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 Dormans and Bakkes (2011); van der Linden et al. (2013); Lavender and Thompson (2017); Karavolos et al. (2016); Pereira et al. (2018); Sheffield and Shah (2018); Gutierrez and Schrum (2020); Gellel and Sweetser (2020). These works present solutions that generate dungeon levels with barriers that are opened with corresponding keys, as depicted in Figure 6 by the color association between doors and keys (Pereira et al., 2018).

All other papers did not focus on the barrier mechanics per se. Instead, they treated it together with other kinds of content, such as missions. Thus, some interesting characteristics of this mechanic could not be properly explored. In contrast, Smith et al. (2018) generated both the first and the second kinds of barriers (Figure 8a). 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). Since Charity et al. (2020) generated only rooms and not entire levels, we could not classify the generated doors and keys of their 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 out the sequence of keys that open up the barriers).

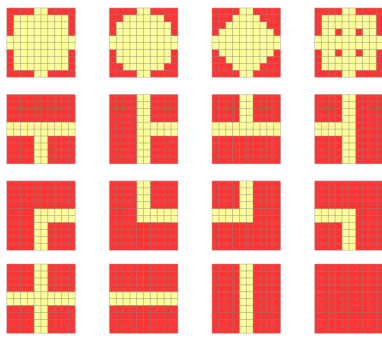
One more aspect we were concerned with was the *2D dungeon structures*. Here is what we discovered:

- A total of 27 works presented solutions for the generation of top-down cavern-like dungeons.
- Just 8 works generated top-down mansion-like dungeon (TDML); these solutions were introduced by Valtchanov and Brown (2012), van der Linden et al. (2013), Lavender and Thompson (2017), Karavolos et al. (2016) Pereira et al. (2018), Sheffield and Shah (2018), Gutierrez and Schrum (2020) and Gellel and Sweetser (2020).
- Just Baghdadi et al. (2015) created side-scrolling (SS) dungeons (*Spelunky’s* dungeons).

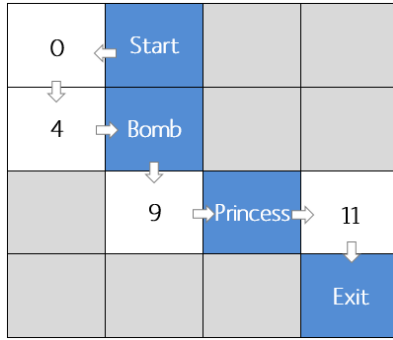
This result means that there is still much to explore in level generation research of both TDML and SS dungeons.

The next aspect we analyze is how the papers addressed the *presentation of level spaces*. Some approaches need to distinguish the spaces of the levels they are generating so that they can determine which game element will appear in these spaces. A total of 13 papers fall in this category. Most of them use this feature to control what kind of content (e.g., enemies, rewards, among others) will appear in the respective area/room. However, the works presented different ways to do this. Let us provide more details on that.

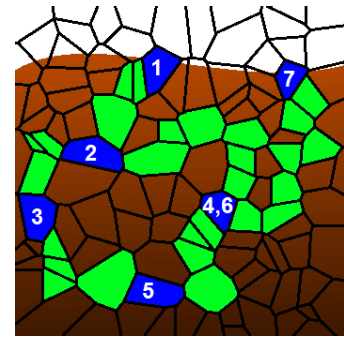
- Smith et al. (2018) introduced a solution that only differentiates the nodes in the level sketch, e.g., the entrance node, the boss’ node, enemy node, key node, among others (Figure 7).
- Smith and Bryson (2014) presented rooms and corridors with different shapes and connection directions (Figure 8a). Rooms may or may not contain rewards; however, this distinction is not presented visually in their work.
- Baghdadi et al. (2015) distinguish the chambers of their levels by defining: the entrance, the exit, where the bomb is, and where the princess is (Figure 8b).
- Santamaria-Ibirika et al. (2014), after creating the chambers, distinguish the rooms to perform later the game elements’ placement, e.g., rewards and enemies (Figure 8c).
- Ashlock (2015) and Kreitzer et al. (2019) (Figure 8d) generate levels with different types of terrains, e.g., a level with ground, water, and lava.



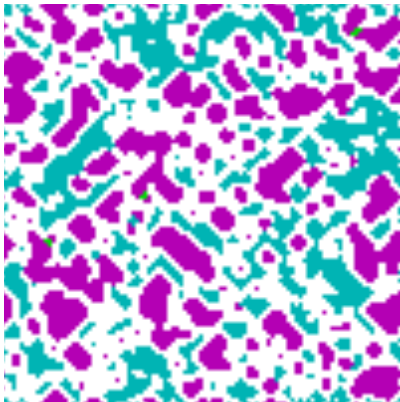
(a) Rooms of levels by Smith and Bryson (2014).



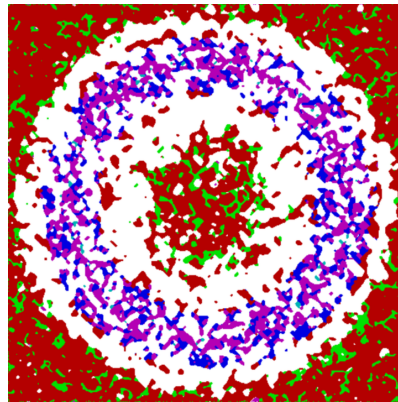
(b) Sample of a level sketch by Baghdadi et al. (2015).



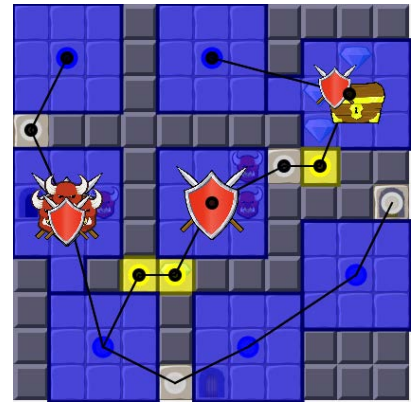
(c) Dungeon by Santamaria-Ibirika et al. (2014).



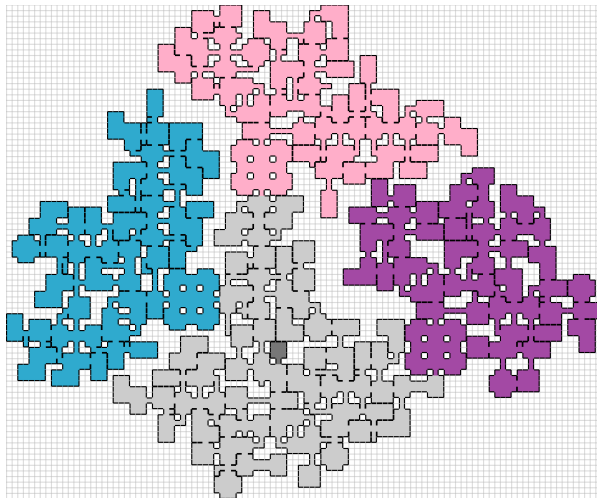
(d) Level by Ashlock (2015).



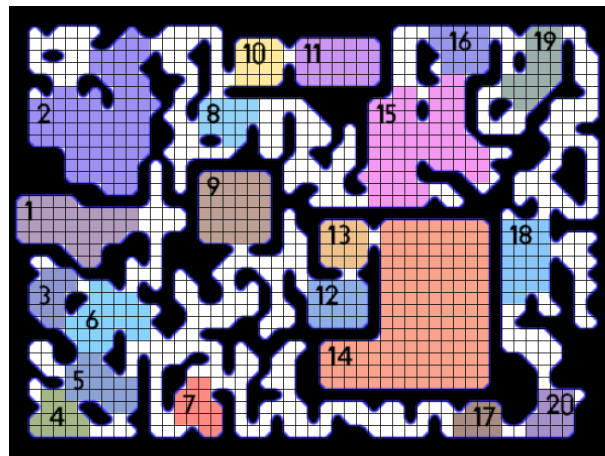
(e) Level by Kreitzer et al. (2019).



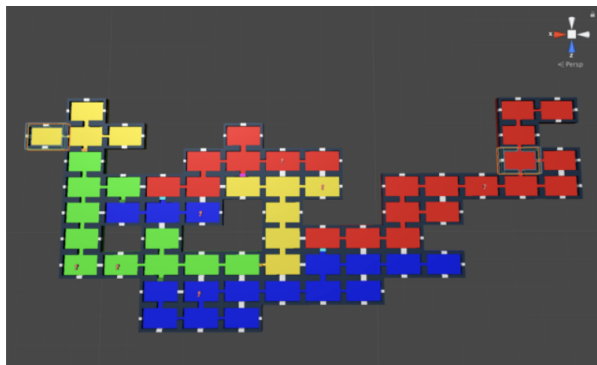
(f) Level patterns by Baldwin et al. (2017a).



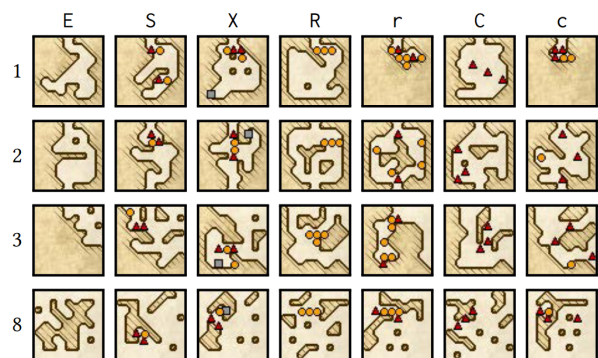
(g) Level by Valchanov and Brown (2012).



(h) Level by Ashlock and McGuinness (2014).



(i) Level by Gellel and Sweetser (2020).



(j) Level generated by Liapis (2017).

Figure 8. Some works that presented area or room distinction in their levels.

- Baldwin et al. (2017b), Alvarez et al. (2018) and Alvarez et al. (2019) detect the type of room and uses this information in their heuristics to evolve the levels (Figure 8f). 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 of enemies and treasure).
- Valtchanov and Brown (2012) is designed to identify the entrance and event rooms, in single rooms or in regions (Figure 8g).
- Ashlock and McGuinness (2014) assign areas to the level's components such as the entrance, armory, and location of enemies, e.g., goblins and magical beings (Figure 8h).
- Gellel and Sweetser (2020) distinguish their levels by what they call subsections. Since they generate levels with barriers, the keys must be placed to be collected by the player and reach all the level's rooms. Therefore, the definition of subsections is a feature that assists their approach to ensure the reachability of keys.
- Finally, Liapis (2017) creates eight different 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 (a 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 Figure 8j).

4 Discussions

The data collected in this survey showed us that the search-based approaches are popular in PDG research (Figure 9). In particular, evolutionary algorithms are the most popular approaches in dungeon generators, corresponding to 16 out of 17 occurrences of the search-based approaches. This result is somewhat expected since search-based approaches usually try to ensure the feasibility of levels and optimize the quality of them by, for instance, ensuring degrees of level linearity (Pereira et al., 2018).

Within EA, Quality Diversity approaches are slowly beginning to be explored in this area. This class of algorithms focuses on diversity, increasing the variety of the generated levels (Gravina et al., 2019). Constrained MAP-Elites is particularly interesting because they combine the QD approach of MAP-Elites with the evolutionary structure of FI2Pop, another strategy to increase diversity. Although ML-based approaches are also promising since there are very successful uses of learning models in other areas, we found out just a single paper that implements such a solution for PCG (Gutierrez and Schrum, 2020). Since one of the PCG goals is to increase replayability, most works should explore these techniques to ensure a good variety of dungeons.

Just a handful of solutions presented solver-based meth-

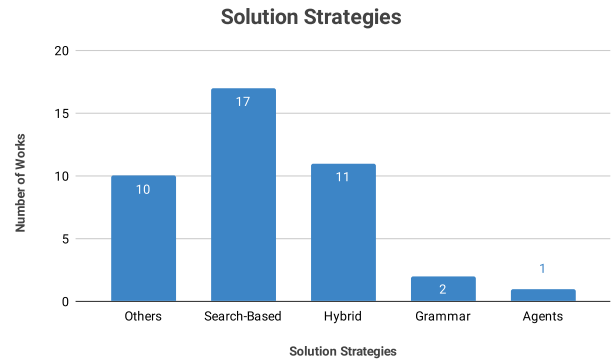


Figure 9. Number of works by solution strategy.

ods in some stages of their generative process. This result might be explained by the fact that these solutions are often developed with logic-based programming languages, which are not as popular as, say, imperative or object-oriented languages. Nevertheless, exclusive constructive approaches are still being explored. Clearly, this has to continue since these approaches are faster than the others since they do not perform tests to evaluate the content they generate.

As Whitehead (2020) described, the dungeon generative process is composed of: (1) mission generation; (2) layout generation, and; (3) interior decoration and layout of rooms. The missions work as sketches to generate the level layout and, sometimes, to determine the content that will be present in the interior of rooms. However, as we observed in our survey, these steps may appear combined in some approaches. For instance, Charity et al. (2020) generated the room layout and placed challenges and rewards.

An aspect interesting explored by the solution introduced by Liapis (2017) was the division of responsibility in its generative process. He combined two search-based approaches to perform the generation in different stages of the generative process in his solution. He could easily add another stage that could perform the generation of another kind of content. For instance, he could add the generation of barriers. However, since Liapis' approach represents its levels as grids, it could be hard to ensure his levels' feasibility. We believe that a solution similar to proposed by Gellel and Sweetser (2020) – called subsections – could help with this problem.

In terms of the *game genres*, most works disassociate the level generation from the game genre (Figure 10). We observed that the works that defined the game genre *a priori* narrowed down the contents and constraints involved in the level generation process. For instance, Platform Rogue-like games, e.g., *Dead Cells* (Motion Twin, 2018), are structurally different from Top-Down Rogue-like games, e.g., *Moonlighter* (Digital Sun, 2018). The former needs ladders, which are intrinsically related to the level structure, to allow the player to access other areas, while the latter does not require such a feature. Moreover, these games also present different strategies for the placement of challenges, rewards, and puzzles in games. Therefore, different strategies could prioritize the placement of these features.

On the other hand, general approaches are interesting since they can deal with the content generation of multiple game genres. However, there are also drawbacks in using general approaches such as performance loss, and too general con-

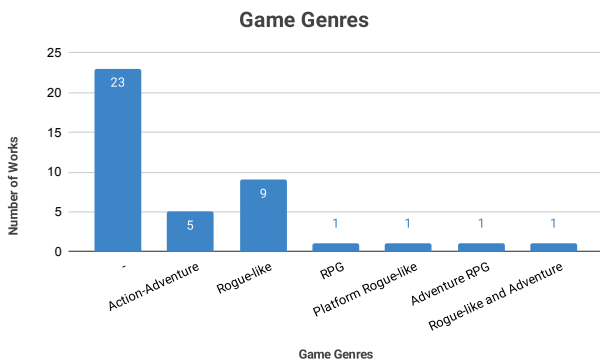


Figure 10. Number of works by game genre.

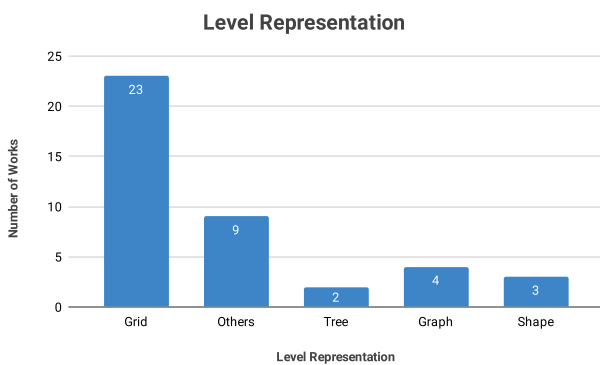


Figure 11. Number of works by level representation.

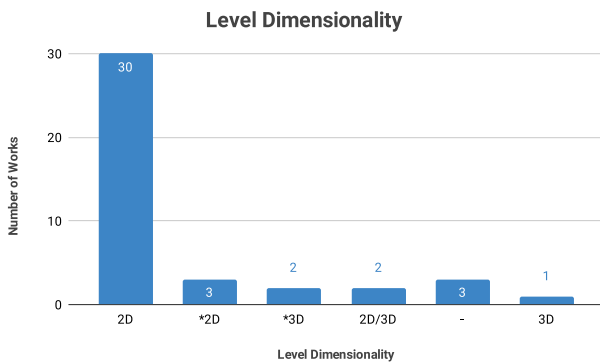


Figure 12. Number of works by level dimensionality. The ‘*’ occurrences mean that the work only performed a translation from a generated sketch to a level layout.

straints may be hard to develop in a single approach. For instance, Pereira et al. (2018) evolved levels represented as trees to ensure the barrier mechanics’ feasibility. Ideally, the levels should be represented as graphs to allow the presence of cycles, but this can lead to problems of level feasibility. This problem remains unsolved because Smith et al. (2018) generated acyclic dungeon levels although they applied graph representation.

With respect to the *level representation* of dungeons, the most common is the grid (Figure 11). Most of the works directly generated the layout of 2D levels (Figure 12). Other papers first generate sketches before creating the level layout. All works produced some kind of planar sketches, i.e., 2D sketches. Here we observe that the dungeon sketch generation may be extended by producing 3D structures to create dungeons of similar levels as “*The Legend of Zelda*” dun-

geons (Nintendo, 1986) or *Pokémon* dungeons (Nintendo, 1996) – at least the games from GBA. These dungeons contain multiple floors, which increases the labyrinthine degree. The works proposed by Santamaria-Ibirika et al. (2014) and Antoniuk et al. (2018) are the only two that yield truly 3D dungeons. Both works represented their dungeons as shapes. Only Santamaria-Ibirika et al. (2014) generated levels with enemies and treasures. However, both works do not support other game features, such as puzzles or barriers. Furthermore, dimensionality is not always a matter of visual representation only. There are mechanics that are intrinsically related to how the player interacts with the level that was not explored by these works. For instance, vertical paths, 3D vertical exploration (jumping between floors or climbing walls), 3D puzzles, barriers, among others.

As for the *dungeon features*, as defined by van der Linden et al. (2014), only six works generated rewards, challenges, and puzzles (Dormans and Bakkes, 2011; van der Linden et al., 2013; Lavender and Thompson, 2017; Karavolos et al., 2016; Smith et al., 2018; Sheffield and Shah, 2018). The most uncommon feature of all works was the puzzles. Some authors probably decided to focus on other dungeon features, but the puzzle generation is also a challenge in PCG, as surveyed by De Kegel and Haahr (2019). For instance, De Kegel and Haahr (2019) present the generation of Sokoban puzzles as a tough challenge. Furthermore, Sokoban-like puzzles also appear in dungeons of games, e.g., *The Legend of Zelda* dungeons (Nintendo, 1986) or *Pokémon* dungeons (Nintendo, 1996). Therefore, generate dungeons is not trivial, especially when it involves puzzles.

In relation to the *outcome randomness* in PDG, only Sampaio et al. (2017), Smith et al. (2018) and Goandy et al. (2020) proposed deterministic solutions for PDG. Even though variety is a goal in PCG, deterministic may become very useful for mixed-initiative software. For instance, a deterministic generator could allow the software to have a undo functionality that saves RAM memory since it is not necessary for saving all the previous suggestions.

Online PDG remains a mostly unexplored problem, as earlier observed by van der Linden et al. (2014). According to them, it is the nature of the dungeons’ environment since they usually have a *fixed* final goal to reach, e.g., a boss or the dungeon exit. Yet, this does not mean that we should not be exploring new forms of dungeons in games, such as an “Endless Dungeon Crawling”. In short: exploring online solutions in dungeon generation may payoff somehow.

Concerning the *Content Authorship*, only Baldwin et al. (2017a), Baldwin et al. (2017b), Alvarez et al. (2018) and Alvarez et al. (2018) (who belong to the same research group), presented the progress of their solution (EDD) for mixed-initiative approach. Considering the definition of dungeons we presented earlier, we suggest adding puzzles and barrier mechanics features or providing support for SS dungeons as an extension for this work. Alvarez et al. (2018) and Alvarez et al. (2019) also were the only authors that presented solutions for adaptive dungeon generation. Their work used area/room distinction and geometric patterns to help with the adaptive generation. Besides, they provide a feature to “freeze” cells in a grid (a dungeon’ room), these cells cannot evolve, and the system uses this information to learn the de-

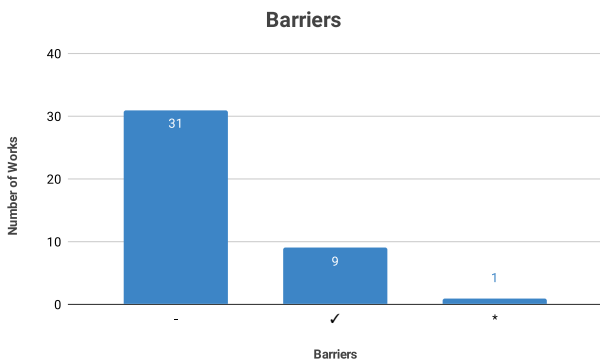


Figure 13. Number of works that generated levels with barriers. The ‘*’ occurrences mean that the paper was not clear in the respective classification.

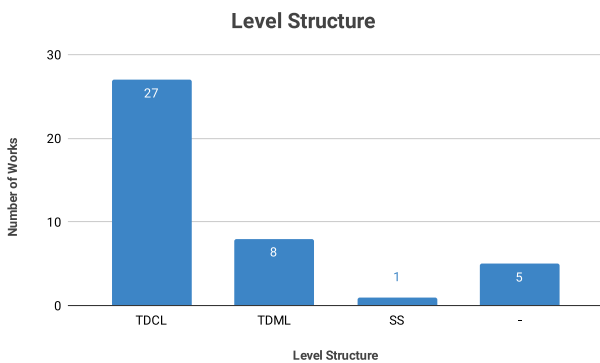


Figure 14. Number of works by level structure.

signer aesthetics. To extend this functionality, they could allow the designer to select cells from the suggestions of rooms which s/he wants to appear in other new. Another appealing feature they can add to their software is using a ML-based approach to learn the final version of the levels and improve the suggestions. The latter approach would be more interesting if the model comprehends the differences in the levels’ difficulty.

The control over the levels’ difficulty is an essential feature of a dungeon generator, even though we notice that few works supported this feature. To employ generators in real games, game developers and designers must produce subsequent levels with increasing difficulty. The same can be considered with rewards; they must be placed in strategic locations to encourage players to explore the levels. However, too many rewards make the game seem too easy or even make the player devalue the rewards. As we observed, area/room distinction is a promising way to control game features like difficulty and rewards. For instance, this concept was greatly applied by Liapis (2017). He generated dungeons’ rooms with different connections and content accordingly with the types of rooms (Figure 8j), which presented different degrees of rewards and enemies. Besides, this feature in generators also may support the production of levels with different terrains.

Surprisingly, just 9 papers (from a total of 41 articles reviewed, see Figure 13) promote dungeons with barriers (Dormans and Bakkes, 2011; van der Linden et al., 2013; Laverder and Thompson, 2017; Karavolos et al., 2016; Smith et al., 2018; Pereira et al., 2018; Sheffield and Shah, 2018; Gutierrez and Schrum, 2020; Gellel and Sweetser, 2020), and only Smith et al. (2018) proposed two different mechanics that

we can interpret as barriers: (1) a single barrier that requires a single key to be unblocked, and; (2) several barriers that require a single key to be unblocked. Here we can observe two natural extensions of this mechanics: (1) a single barrier that requires several keys to be unblocked, and; (2) several barriers that require several keys to be unblocked. This extension is particularly interesting for generative grammars. It is especially challenging if we consider that there might be barriers in the path towards a key to other barriers. Surprisingly, not a single work generated levels with these mechanics in a 3D level structure. Most of these works focused exclusively on TDML dungeon generation (Figure 14), and, again, new solutions that support barriers for TDCL or SS dungeon levels might be promising. Finally, other kinds of game features that are intrinsically related to the players’ interaction with levels and affect gameplay progression should also be explored. For instance, portals (Green et al., 2019), spatial puzzles (De Kegel and Haahr, 2019), and climbing (e.g., God of War (Sony, 2005)).

5 Conclusion

In this paper, we surveyed research papers on procedural dungeon generation. Our review indicates that there is a clear preference for evolutionary algorithms for generating dungeon levels. This behavior probably happens due to the high number of constraints that are involved in this process. We identified that the quality diversity approaches just started to be explored for dungeon generation. In terms of dungeon content, few papers presented solutions for dungeons as defined by van der Linden et al. (2014). However, the main problem is that the puzzle generation itself is a very complicated challenge (De Kegel and Haahr, 2019).

Here is a brief list of our main findings, which may indicate potential research topics to pursuit in PDG: (1) the procedural generation of 3D dungeons needs more investigation because it has the potential to support new gameplay experiences; (2) a small number of works introduced solutions for levels with the barrier mechanics, and; (3) a handful of papers presented mixed-initiative solutions. In terms of research topics, we underline the lack of more PDG solutions that support barriers and their complexities, mainly when applied to TDML and SS dungeons. It may be inviting to explore the barrier mechanics applied in 3D structures. Finally, we need to explore further adaptive and mixed-initiatives approaches – not necessarily both together – because they present the potential to generate complex and challenging levels and increase the game designers’ creativity and productivity involved in the creation process.

Since the survey made by van der Linden et al. (2014) much has progressed in PDG research, and several works crafted excellent solutions. Nonetheless, as we have observed throughout this paper, the procedural generation of dungeons still remains a complex problem. In Section 4, we discuss some potential open problems and research suggestions that might help in expanding the field.

It is worth mentioning that some works were somehow difficult to classify or grasp due to the lack of information. To overcome these difficulties in the future, we encourage the

authors to: (1) define the game genre or the specific game for which the level will be generated; (2) define the level structure (TDML, TDCL, SS, or any other); (3) highlight the generative process pipeline; (4) classify the solution based on the Togelius's PCG taxonomy (Togelius et al., 2016), and; (5) define the generation problem mathematically, when possible. These suggestions will hopefully improve the overall understanding of the field and help us keep track of open problems, and map important avenues of research.

Acknowledgements

Breno Mauricio de Freitas Viana acknowledges the financial support of the National Council for Scientific and Technological Development (CNPq).

References

- Adams, C. and Louis, S. (2017). Procedural maze level generation with evolutionary cellular automata. In *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on*, pages 1–8. IEEE.
- Alvarez, A., Dahlskog, S., Font, J., Holmberg, J., and Johansson, S. (2018). Assessing aesthetic criteria in the evolutionary dungeon designer. In *Proceedings of the 13th International Conference on the Foundations of Digital Games*, page 44. ACM.
- Alvarez, A., Dahlskog, S., Font, J., and Togelius, J. (2019). Empowering quality diversity in dungeon design with interactive constrained map-elites. In *2019 IEEE Conference on Games (CoG)*, pages 1–8. IEEE.
- Antoniuk, I., Hoser, P., and Strzemiński, D. (2018). L-system application to procedural generation of room shapes for 3d dungeon creation in computer games. In *International Multi-Conference on Advanced Computer Systems*, pages 375–386. Springer.
- Ashlock, D. (2015). Evolvable fashion-based cellular automata for generating cavern systems. In *Computational Intelligence and Games (CIG), 2015 IEEE Conference on*, pages 306–313. IEEE.
- Ashlock, D., Lee, C., and McGuinness, C. (2011a). Search-based procedural generation of maze-like levels. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):260–273.
- Ashlock, D., Lee, C., and McGuinness, C. (2011b). Simultaneous dual level creation for games. *IEEE Computational Intelligence Magazine*, 6(2):26–37.
- Ashlock, D. and McGuinness, C. (2014). Automatic generation of fantasy role-playing modules. In *Computational Intelligence and Games (CIG), 2014 IEEE Conference on*, pages 1–8. IEEE.
- Assadi, H., Jones, R., Swift, A. J., Al-Mohammad, A., and Garg, P. (2020). Cardiac mri for the prognostication of heart failure with preserved ejection fraction: A systematic review and meta-analysis. *Magnetic Resonance Imaging*.
- Baghdadi, W., Eddin, F. S., Al-Omari, R., Alhalawani, Z., Shaker, M., and Shaker, N. (2015). A procedural method for automatic generation of spelunky levels. In *European Conference on the Applications of Evolutionary Computation*, pages 305–317. Springer.
- Baldwin, A., Dahlskog, S., Font, J. M., and Holmberg, J. (2017a). Mixed-initiative procedural generation of dungeons using game design patterns. In *Computational Intelligence and Games (CIG), 2017 IEEE Conference on*, pages 25–32. IEEE.
- Baldwin, A., Dahlskog, S., Font, J. M., and Holmberg, J. (2017b). Towards pattern-based mixed-initiative dungeon generation. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*, page 74. ACM.
- Baron, J. R. (2017). Procedural dungeon generation analysis and adaptation. In *Proceedings of the SouthEast Conference*, pages 168–171. ACM.
- Blizzard Entertainment (1996). Diablo. Accessed in: 2020-11-02.
- Brace Yourself Games (2015). Crypt of the necrodancer. Accessed in: 2020-11-02.
- Charity, M., Green, M. C., Khalifa, A., and Togelius, J. (2020). Mech-elites: Illuminating the mechanic space of gvgai. *arXiv preprint arXiv:2002.04733*.
- Chomsky, N. (2006). *Language and mind*. Cambridge University Press.
- De Kegel, B. and Haahr, M. (2019). Procedural puzzle generation: a survey. *IEEE Transactions on Games*, 12(1):21–40.
- De Kok, T., Van Kreveld, M., and Löffler, M. (2007). Generating realistic terrains with higher-order delaunay triangulations. *Computational Geometry*, 36(1):52–65.
- Delaunay, B. et al. (1934). Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2.
- Digital Sun (2018). Moonlighter. Accessed in: 2020-07-25.
- Dormans, J. and Bakkes, S. (2011). Generating missions and spaces for adaptable play experiences. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):216–228.
- Forsyth, W. (2016). Globalized random procedural content for dungeon generation. *Journal of Computing Sciences in Colleges*, 32(2):192–201.
- Gellel, A. and Sweetser, P. (2020). A hybrid approach to procedural generation of roguelike video game levels. In *International Conference on the Foundations of Digital Games*, pages 1–10.
- Gendreau, M., Potvin, J.-Y., et al. (2010). *Handbook of metaheuristics*, volume 2. Springer.
- Goandy, H., Young, J. C., and Hansun, S. (2020). No escape: A 2d top-down shooting roguelike game embedded with drunkard walk algorithm. *International Journal of Advanced Trends in Computer Science and Engineering*.
- Gravina, D., Khalifa, A., Liapis, A., Togelius, J., and Yannakakis, G. N. (2019). Procedural content generation through quality diversity. In *2019 IEEE Conference on Games (CoG)*, pages 1–8. IEEE.
- Green, M. C., Khalifa, A., Alsoughayer, A., Surana, D., Liapis, A., and Togelius, J. (2019). Two-step constructive approaches for dungeon generation. In *Proceedings of the*

- 14th International Conference on the Foundations of Digital Games, pages 1–7.
- Gutierrez, J. and Schrum, J. (2020). Generative adversarial network rooms in generative graph grammar dungeons for the legend of zelda. *arXiv preprint arXiv:2001.05065*.
- Halatsch, J., Kunze, A., and Schmitt, G. (2008). Using shape grammars for master planning. In *Design Computing and Cognition '08*, pages 655–673. Springer.
- Harper, S. L., Cunsolo, A., Babujee, A., Coggins, S., Aguilar, M. D., and Wright, C. J. (2021). Climate change and health in north america: literature review protocol. *Systematic Reviews*, 10(1):1–13.
- Hell, J., Clay, M., and ELAarag, H. (2017). Hierarchical dungeon procedural generation and optimal path finding based on user input. *Journal of Computing Sciences in Colleges*, 33(1):175–183.
- Hilliard, N., Salis, J., and ELAarag, H. (2017). Algorithms for procedural dungeon generation. *Journal of Computing Sciences in Colleges*, 33(1):166–174.
- Johnson, L., Yannakakis, G. N., and Togelius, J. (2010). Cellular automata for real-time generation of infinite cave levels. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, pages 1–4.
- Karavolos, D., Liapis, A., and Yannakakis, G. N. (2016). Evolving missions for dwarf quest dungeons. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–2.
- Kimbrough, S. O., Koehler, G. J., Lu, M., and Wood, D. H. (2005). Introducing a feasible-infeasible two-population (fi-2pop) genetic algorithm for constrained optimization: Distance tracing and no free lunch. *European Journal of Operational Research*.
- Klei Entertainment (2013). Don't starve. Accessed in: 2020-11-02.
- Kreitzer, M., Ashlock, D., and Pereira, R. (2019). Automatic generation of diverse cavern maps with morphing cellular automata. In *2019 IEEE Conference on Games (CoG)*, pages 1–8. IEEE.
- Lavender, B. and Thompson, T. (2017). A generative grammar approach for action-adventure map generation in the legend of zelda.
- Liapis, A. (2017). Multi-segment evolution of dungeon game levels. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 203–210. ACM.
- Liapis, A., Holmgård, C., Yannakakis, G. N., and Togelius, J. (2015). Procedural personas as critics for dungeon generation. In *European Conference on the Applications of Evolutionary Computation*, pages 331–343. Springer.
- Liberati, A., Altman, D. G., Tetzlaff, J., Mulrow, C., Gøtzsche, P. C., Ioannidis, J. P., Clarke, M., Devereaux, P. J., Kleijnen, J., and Moher, D. (2009). The prisma statement for reporting systematic reviews and meta-analyses of studies that evaluate health care interventions: explanation and elaboration. *Journal of clinical epidemiology*, 62(10):e1–e34.
- Masotta, V., Dante, A., Marcotullio, A., Bertocchi, L., La Cerra, C., Caponnetto, V., Petrucci, C., and Alfes, C. M. (2020). The concept of high-fidelity simulation and related factors in nursing education: A scoping review. In *International Conference in Methodologies and intelligent Systems for Technology Enhanced Learning*, pages 119–126. Springer.
- McGuinness, C. and Ashlock, D. (2011). Decomposing the level generation problem with tiles. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 849–856. IEEE.
- McMillen, E. and Himsl, F. (2011). The binding of isaac. Accessed in: 2020-11-02.
- Melotti, A. S. and de Moraes, C. H. V. (2018). Evolving roguelike dungeons with deluged novelty search local competition. *IEEE Transactions on Games*, 11(2):173–182.
- Moher, D., Liberati, A., Tetzlaff, J., and Altman, D. G. (2009). Preferred reporting items for systematic reviews and meta-analyses: the prisma statement. *Annals of internal medicine*, 151(4):264–269.
- Moray, K. V., Chaurasia, H., Sachin, O., and Joshi, B. (2021). A systematic review on clinical effectiveness, side-effect profile and meta-analysis on continuation rate of etonogestrel contraceptive implant. *Reproductive Health*, 18(1):1–24.
- Motion Twin (2018). Dead cells. Accessed in: 2020-07-25.
- Mulrow, C. D. (1994). Systematic reviews: rationale for systematic reviews. *Bmj*, 309(6954):597–599.
- Nepožitek, O. and Gemrot, J. (2018). Fast configurable tile-based dungeon level generator.
- Nintendo (1986). The legend of zelda - franchise. Accessed in: 2020-09-04.
- Nintendo (1996). Pokémon - franchise. Accessed in: 2020-09-10.
- Olsen, J. (2004). Realtime procedural terrain generation.
- Pech, A., Hingston, P., Masek, M., and Lam, C. P. (2015). Evolving cellular automata for maze generation. In *Australasian Conference on Artificial Life and Computational Intelligence*, pages 112–124. Springer.
- Pech, A., Masek, M., Lam, C.-P., and Hingston, P. (2016). Game level layout generation using evolved cellular automata. *Connection Science*, 28(1):63–82.
- Pereira, L. T., Prado, P. V., and Toledo, C. (2018). Evolving dungeon maps with locked door missions. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.
- Prim, R. C. (1957). Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6):1389–1401.
- Pugh, J. K., Soros, L. B., and Stanley, K. O. (2016). Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3:40.
- Rozenberg, G. (1997). *Handbook of Graph Grammars and Comp.*, volume 1. World scientific.
- Sampaio, P., Baffa, A., Feijó, B., and Lana, M. (2017). 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)*, pages 10–18. IEEE.
- Santamaria-Ibirika, A., Cantero, X., Huerta, S., Santos, I., and Bringas, P. G. (2014). Procedural playable cave sys-

- tems based on voronoi diagram and delaunay triangulation. In *Cyberworlds (CW), 2014 International Conference on*, pages 15–22. IEEE.
- Sheffield, E. C. and Shah, M. D. (2018). Dungeon digger: Apprenticeship learning for procedural dungeon building agents. In *Proceedings of the 2018 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*, pages 603–610.
- Smith, A. J. and Bryson, J. J. (2014). 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*.
- Smith, T., Padget, J., and Vidler, A. (2018). 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*, page 52. ACM.
- Sony (2005). God of war. Accessed in: 2021-03-30.
- Summerville, A., Snodgrass, S., Guzdial, M., Holmgård, C., Hoover, A. K., Isaksen, A., Nealen, A., and Togelius, J. (2018). Procedural content generation via machine learning (pcgml). *IEEE Transactions on Games*, 10(3):257–270.
- Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
- Togelius, J., Justinussen, T., and Hartzen, A. (2012). Compositional procedural content generation. In *Proceedings of the The third workshop on Procedural Content Generation in Games*, page 16. ACM.
- Togelius, J., Shaker, N., and Nelson, M. J. (2016). Introduction. In Shaker, N., Togelius, J., and Nelson, M. J., editors, *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*, pages 1–15. Springer.
- Togelius, J., Yannakakis, G. N., Stanley, K. O., and Browne, C. (2011). Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(3):172–186.
- Toy, M. and Wichman, G. (1980). Rogue. Accessed in: 2020-07-25.
- Valtchanov, V. and Brown, J. A. (2012). Evolving dungeon crawler levels with relative placement. In *Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering*, pages 27–35. ACM.
- Valve Corporation (2009). Left 4 dead 2. Acessado em: 2020-11-02.
- van der Linden, R., Lopes, R., and Bidarra, R. (2013). Designing procedurally generated levels. In *Proceedings of the the second workshop on Artificial Intelligence in the Game Design Process*.
- van der Linden, R., Lopes, R., and Bidarra, R. (2014). Procedural generation of dungeons. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(1):78–89.
- Van Laarhoven, P. J. and Aarts, E. H. (1987). Simulated annealing. In *Simulated annealing: Theory and applications*, pages 7–15. Springer.
- Viana, B. M. and dos Santos, S. R. (2019). A survey of procedural dungeon generation. In *2019 18th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pages 29–38. IEEE.
- Voronoi, G. (1908). Nouvelles applications des paramètres continus à la théorie des formes quadratiques. deuxième mémoire. recherches sur les paralléloèdres primitifs. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1908(134):198–287.
- Whitehead, J. (2020). Spatial layout of procedural dungeons using linear constraints and smt solvers. In *International Conference on the Foundations of Digital Games*, pages 1–9.
- Wild Card Games (2013). Dwarf quest. Accessed in: 2020-08-10.
- Wolfram, S. (1983). Statistical mechanics of cellular automata. *Reviews of modern physics*, 55(3):601.
- Yu, D. (2008). Spelunky. Accessed in: 2020-08-10.

Table 2. Solution strategy data collected from the found papers. We could classify some solutions because some of them did not fit in an umbrella classification; therefore, we called them ‘exclusive’ solutions. The ‘*’ occurrences mean that the work only performed a translation from a generated sketch to a level layout.

| Work | Game Genre | Level | Dimensionality | Representation | Algorithm Strategy | |
|----------------------------------|--------------------------|-----------------------------|----------------|--------------------------|----------------------------------|--|
| | | | | | Macro | Micro |
| Johnson et al. (2010) | - | layout | 2D | Grid | - | CA |
| Ashlock et al. (2011a) | - | layout | 2D | Grid | EA | GA |
| McGuinness and Ashlock (2011) | - | layout | 2D | Grid | EA | GA |
| Ashlock et al. (2011b) | - | layout | 2D | Grid | EA | GA |
| Dormans and Bakkes (2011) | Action-Adventure | sketch + layout | 2D | Graph + Shape | Hybrid | Graph and Shape Grammars |
| Valtchanov and Brown (2012) | - | sketch | *2D | Tree | EA | GP |
| Togelius et al. (2012) | Rogue-like | layout | 2D | Grid | Hybrid | EA + ASP |
| van der Linden et al. (2013) | Action-Adventure | sketch | *3D | Graph | Grammar | Graph Grammar |
| Ashlock and McGuinness (2014) | RPG | layout | 2D | Graph + Grid | EA | GA |
| Smith and Bryson (2014) | Rogue-like | sketch + layout | 2D | Answer Set + Grid | Hybrid | ASP + exclusive |
| Santamaria-Ibirika et al. (2014) | - | layout | 2D + 3D | Shape | Hybrid | Voronoi Diagram and Delaunay Triangulation |
| Pech et al. (2015) | - | layout | 2D | Grid | Hybrid | GA + CA |
| Liapis et al. (2015) | Rogue-like | layout | 2D | Grid | EA | FI2Pop GA |
| Ashlock (2015) | - | layout | 2D | Grid | Hybrid | GA + CA |
| Baghdadi et al. (2015) | Platform Rogue-like | sketch + layout | 2D | Grid | EA | GA |
| Lavender and Thompson (2017) | Action-Adventure | sketch + layout | 2D | Graph + Grid | Hybrid | Graph and Shape Grammars |
| Pech et al. (2016) | - | layout | 2D | Grid | Hybrid | GA + CA |
| Karavolos et al. (2016) | Adventure RPG | sketch | *3D | Graph + Grid | EA | - |
| Forsyth (2016) | Rogue-like | sketch | - | Graph | - | exclusive |
| Hell et al. (2017) | - | sketch | - | Graph | - | exclusive |
| Hilliard et al. (2017) | - | layout | 2D | Grid | - | 2 exclusives |
| Baron (2017) | - | layout | 2D + *3D | Grid | - | 5 exclusives |
| Liapis (2017) | - | sketch + layout | 2D | Grid | EA | FI2Pop GA |
| Baldwin et al. (2017a) | - | layout | 2D | Grid | EA | FI2Pop GA |
| Baldwin et al. (2017b) | - | layout | 2D | Grid | EA | FI2Pop GA |
| Sampaio et al. (2017) | - | layout | 2D | Grid | - | exclusive |
| Smith et al. (2018) | Action-Adventure | sketch | - | Graph | - | ASP |
| Alvarez et al. (2018) | - | layout | 2D | Grid | EA | FI2Pop GA |
| Pereira et al. (2018) | Action-Adventure | sketch | *2D | Tree | EA | GP |
| Nepožitek and Gemrot (2018) | - | sketch + layout | 2D | Graph + Shape | - | Simulated Annealing |
| Sheffield and Shah (2018) | Rogue-like and Adventure | sketch | *2D | Grid | RL | Digger Agents (RL Agents) |
| Antoniuk et al. (2018) | - | layout | 3D | Shape | Grammar | L-system |
| Melotti and de Moraes (2018) | Rogue-like | layout | 2D | Grid | EA | DNLSC |
| Kreitzer et al. (2019) | - | layout | 2D | Grid | Hybrid | GA + CA |
| Alvarez et al. (2019) | - | layout | 2D | Graph + Grid | EA | Constrained MAP-Elites |
| Green et al. (2019) | Rogue-like | layout | 2D | Grid | - | CA + 2 agents + 3 exclusive |
| Goandy et al. (2020) | Rogue-like | layout | 2D | Grid | - | exclusive |
| Gutierrez and Schrum (2020) | Rogue-like | Level: sketch, Room: layout | 2D | Level: Graph, Room: Grid | Hybrid. Level: Grammar, Room: ML | Level: Graph Grammar, Room: GAN |
| Gellel and Sweetser (2020) | Rogue-like | sketch + layout | 2D | Graph + Grid | Hybrid | Grammars + CA-inspired. |
| Whitehead (2020) | - | layout | 2D | Shape | - | SMT Solver |
| Charity et al. (2020) | - | layout | 2D | Grid | EA | Constrained MAP-Elites |

Table 3. PCG taxonomy data collected from the found papers. Classifications: ON (online); OFF (offline); P (parameters); RS (random seeds); G (generic); A (adaptive); S (stochastic); D (deterministic); C (constructive); GT (generate-and-test); AT (automatic), and; MI (mixed-initiative).

| Work | Choice Method | Generation Time | Generation Control | Generality | Generation Method | Content Authorship |
|----------------------------------|---------------|-----------------|--------------------|------------|-------------------|--------------------|
| Johnson et al. (2010) | OFF | P | G | S | C | AT |
| Ashlock et al. (2011a) | OFF | P | G | S | GT | AT |
| McGuinness and Ashlock (2011) | OFF | P | G | S | GT | AT |
| Ashlock et al. (2011b) | OFF | P | G | S | GT | AT |
| Dormans and Bakkes (2011) | OFF | P | G | S | C | AT |
| Valtchanov and Brown (2012) | OFF | P | G | S | GT | AT |
| Togelius et al. (2012) | OFF | P | G | S | GT | AT |
| van der Linden et al. (2013) | OFF | P | G | S | C | AT |
| Ashlock and McGuinness (2014) | OFF | P | G | S | GT | AT |
| Smith and Bryson (2014) | OFF | P | G | S | GT | AT |
| Santamaria-Ibirika et al. (2014) | OFF | P | G | S | C | AT |
| Pech et al. (2015) | OFF | P | G | S | GT | AT |
| Liapis et al. (2015) | OFF | P | G | S | GT | AT |
| Ashlock (2015) | OFF | P | G | S | GT | AT |
| Baghdadi et al. (2015) | OFF | P | G | S | GT | AT |
| Lavender and Thompson (2017) | OFF | P | G | S | GT | AT |
| Pech et al. (2016) | OFF | P | G | S | GT | AT |
| Karavolos et al. (2016) | OFF | P | G | S | GT | AT |
| Forsyth (2016) | OFF | P | G | S | C | AT |
| Hell et al. (2017) | OFF | P | G | S | C | AT |
| Hilliard et al. (2017) | OFF | P | G | S | C | AT |
| Baron (2017) | OFF | P | G | S | C | AT |
| Liapis (2017) | OFF | P | G | S | GT | AT |
| Baldwin et al. (2017a) | OFF | P | G | S | GT | MI |
| Baldwin et al. (2017b) | OFF | P | G | S | GT | MI |
| Sampaio et al. (2017) | OFF | RS | G | D | C | AT |
| Smith et al. (2018) | OFF | RS | G | D | GT | AT |
| Alvarez et al. (2018) | OFF | P | A | S | GT | MI |
| Pereira et al. (2018) | OFF | P | G | S | GT | AT |
| Nepožitek and Gemrot (2018) | OFF | P | G | S | GT | AT |
| Sheffield and Shah (2018) | OFF | P | G | S | GT | AT |
| Antoniuk et al. (2018) | OFF | P | G | S | C | AT |
| Melotti and de Moraes (2018) | OFF | P | G | S | GT | AT |
| Kreitzer et al. (2019) | OFF | P | G | S | GT | AT |
| Alvarez et al. (2019) | OFF | P | A | S | GT | MI |
| Green et al. (2019) | OFF | P | G | S | C | AT |
| Goandy et al. (2020) | OFF | RS | G | D | C | AT |
| Gutierrez and Schrum (2020) | OFF | P | G | S | GT | AT |
| Gellel and Sweetser (2020) | OFF | P | G | S | GT | AT |
| Whitehead (2020) | OFF | P | G | S | GT | AT |
| Charity et al. (2020) | OFF | P | G | S | GT | AT |

Table 4. Content-related data collected from the found papers. Level Structures: TDML (Top-down mansion-like); TDCL (Top-down cavern-like); and SS (Side-scrolling dungeon). The ‘*’ occurrences mean that the paper was not clear in the respective classification.

| Work | Reward | Challenge | Puzzle | Barrier | Difficulty | Area/Room distinction | Level Structure |
|----------------------------------|--------|-----------|--------|---------|------------|-----------------------|-----------------|
| Johnson et al. (2010) | - | - | - | - | - | - | TDCL |
| Ashlock et al. (2011a) | - | - | - | - | - | - | TDCL |
| McGuinness and Ashlock (2011) | - | - | - | - | - | - | TDCL |
| Ashlock et al. (2011b) | - | - | - | - | - | - | TDCL |
| Dormans and Bakkes (2011) | ✓ | ✓ | - | ✓ | ✓ | - | TDCL |
| Valtchanov and Brown (2012) | - | - | - | - | - | ✓ | TDML |
| Togelius et al. (2012) | - | ✓ | - | - | - | - | TDCL |
| van der Linden et al. (2013) | ✓ | ✓ | - | ✓ | ✓ | - | TDML |
| Ashlock and McGuinness (2014) | ✓ | ✓ | - | - | - | ✓ | TDCL |
| Smith and Bryson (2014) | ✓ | ✓ | - | - | - | ✓ | TDCL |
| Santamaria-Ibirika et al. (2014) | ✓ | ✓ | - | - | - | - | TDCL |
| Pech et al. (2015) | - | - | - | - | - | - | TDCL |
| Liapis et al. (2015) | ✓ | ✓ | - | - | - | - | TDCL |
| Ashlock (2015) | - | - | - | - | - | ✓ | TDCL |
| Baghdadi et al. (2015) | ✓ | ✓ | - | - | ✓ | ✓ | SS |
| Lavender and Thompson (2017) | ✓ | ✓ | - | ✓ | ✓ | - | TDML |
| Pech et al. (2016) | - | - | - | - | - | - | TDCL |
| Karavolos et al. (2016) | ✓ | ✓ | ✓ | ✓ | ✓ | - | TDML |
| Forsyth (2016) | - | - | - | - | ✓ | - | - |
| Hell et al. (2017) | ✓ | - | - | - | ✓ | - | - |
| Hilliard et al. (2017) | - | - | - | - | - | - | TDCL |
| Baron (2017) | - | - | - | - | - | - | TDCL |
| Liapis (2017) | ✓ | ✓ | - | - | - | ✓ | TDCL |
| Baldwin et al. (2017a) | ✓ | ✓ | - | - | ✓ | - | TDCL |
| Baldwin et al. (2017b) | ✓ | ✓ | - | - | ✓ | ✓ | TDCL |
| Sampaio et al. (2017) | ✓ | ✓ | - | - | ✓ | - | TDCL |
| Smith et al. (2018) | ✓ | ✓ | ✓ | ✓ | - | ✓ | - |
| Alvarez et al. (2018) | ✓ | ✓ | - | - | ✓ | ✓ | TDCL |
| Pereira et al. (2018) | - | - | - | ✓ | - | - | TDML |
| Nepožitek and Gemrot (2018) | - | - | - | - | - | - | TDCL |
| Sheffield and Shah (2018) | ✓ | ✓ | - | ✓ | - | - | TDML |
| Antoniuk et al. (2018) | - | - | - | - | - | - | - |
| Melotti and de Moraes (2018) | - | ✓ | - | - | - | - | TDCL |
| Kreitzer et al. (2019) | - | - | - | - | - | ✓ | TDCL |
| Alvarez et al. (2019) | ✓ | ✓ | - | - | ✓ | ✓ | TDCL |
| Green et al. (2019) | ✓ | ✓ | - | - | - | - | TDCL |
| Goandy et al. (2020) | - | ✓ | - | - | - | - | TDCL |
| Gutierrez and Schrum (2020) | - | ✓ | ✓ | ✓ | - | - | TDML |
| Gellel and Sweetser (2020) | - | ✓ | - | ✓ | - | ✓ | TDML |
| Whitehead (2020) | - | - | - | - | - | - | TDCL |
| Charity et al. (2020) | ✓ | ✓ | - | * | - | - | - |