Department of Computer Science

UNIVERSITY
of York

Submitted in part fulfilment for the degree of BEng.

Networks and Dragons: A data-driven approach to procedural dungeon generation

Angel Simeonov

1st April 2020

Supervisor: Dr. Rob Alexander

<insert dedication>

Acknowledgements

<insert acknowledgements>

Contents

List of Figures

List of Tables

Executive Summary

>At most two (2) pages, aimed a non-specialist, knowledgeable authorial peer. The summary must: –state the aim of the reported work, –motivate the work, –state methods used, –state results found, and –highlight any legal, social, ethical, professional, and commercial issues as appropriate to the topic of study (if none, then this should be explicitly stated).

1 Introduction

1.1 Role-playing games

Role-playing game (RPG) is a broad term encompassing a multitude of different games with often distinct mechanics and media of interaction. The common factor between all RPGs is that the player(s) portray a fictional character and is involved in a fictional world (or a subset of one). The interaction of the players with this world is governed by rules, defined by the media. To better understand the structure of RPGs, we can view it in terms of two sets:

1. Rules defining how we play the game which describe the allowed actions for the player at any given moment in the game.

2. Narrative elements which provide the *purpose* of the players to interact with the fictional world.

The first set can be viewed as "How I interact with the environment" (functional) and the second as "What is the meaning of the environment" (narrative).

Computer RPGs like the Action RPG (ARPG) Legend of Zelda [1], Diablo [2], Fallout [3] and many alike have both sets of rules defined by the game designers. A functional rule in an RPG like Skyrim [4] is that you can attack with the left mouse button and a story element is that you are a hero with a quest. This quest is defined by the writers and designers of the game and as such exploring the narrative in a digital RPG can be comparable to an interactive reading of a fiction book. Good computer RPGs often have the ability to relax the narrative [5], allowing for players to have more freedom in the exploration of the world and as such create the sensation that they have impact on this pre-programmed fictional environment.

The ability for a player to influence the narrative is one of the defining features of tabletop RPGs (TRPG a.k.a pen-and-paper PnP) like Dungeons and Dragons [6]. In them, the functional rules are usually defined by a rulebook (like the Player's Handbook) and players verbally describe their interactions with the environment. The narrative is an ever evolving

amalgam between the input of players and the Dungeon Master (DM). The DM's task is to create a narrative outline and guide the player interaction. Because of the verbal nature of the game, the narrative does not suffer the limitations of its digital counterparts. But because there are no hard constraints to how the narrative is told, the DM has the non-trivial task of introducing consistency and outlining a structure for the story that would result in a compelling and ideally immersive experience for the players. This is often achieved by focusing the adventure's act on a particular and detailed location. These locations are often referred to as Dungeons and in practice can be anything from the villain's mansion or a beast's cave to a city under siege. A good dungeon design is crucial for creating a compelling narrative. The creative task of making the Dungeon is a laborious process. To facilitate that, academics and various gaming communities have been exploring different ways of automating Dungeon creation. We will refer to this process as Procedural Dungeon Generation (PDG). Arguably the greatest problem posed by automating PDG is answering the question of "What is a compelling Dungeon?". In the next section we will review different PDG methods and their associated limitations in an attempt to answer this question.

1.2 State of the Art Generative methods

1.2.1 Non-digital Generators

The designers of the classic PnP modules acknowledge the issue of Dungeon creation and often incorporate *Loot* or *Encounter* tables in the rule-books. The tables are a reference over a set of treasures or monsters respectively, which can be chosen by rolling the specified die (example in appendix B fig B.1). As noted before, a PnP RPG can evolve rapidly outside the planned narrative structure prepared by the DM and a simple way to quickly define new adaptive story elements via a loot table can be useful. Some early modules even provided a step-by-step guide for creating a full campaign from the plot, villains' obsessions and story setting to the various encounters and and treasures [7] entirely based on randomised content tables. An issue arises when using loot tables when a randomly selected element from one table is contradicted by an item selected from another. **[[find example of contradiction]]**. It is up to the DM to resolve such disparities. Although this process provides some degree of creative assistance, it does little to facilitate the laborious nature of creating an end-to-end compelling narrative. Furthermore, as these methods are occupied with providing inspiration for global narrative elements, they pay no attention to providing guidance of what would mean a "good" Dungeon.

1.2.2 Digital Generators

It is important to note that PnP RPGs and the various genres of computer RPGs share the same narrative goal despite differing in mechanisms of interaction [8]. Therefore we will not limit ourselves to looking only at existing tabletop solutions. Unlike the original non-digital generators, computerised ones rarely allow for human input during the generation process. A generative algorithm would provide a DM with an interface that takes a set of parameters and produce a template of a game. The degree of complexity of this template is dependent on the complexity of the algorithm itself. Because they aim to exclude the human designer from the process, the digital generators tend to focus on creating elaborate Dungeon spaces and either provide basic narrative content or omit it entirely.

Cellular Automata

Cellular automata (CA) utilises a procedural generation mechanism in which a $N \times M$ grid space is mutated incrementally by a set of agents. The implementation of these mutation operators define if the CA will simulate erosion [9] or man-made structures such as rooms and corridors. The latter is the case of the donjon dungeon generator [10]. Its simplicity and degrees of customisations of layout and style has made donjon a popular generator choice. At time of writing, it has more than 2500 generated dungeons in the last 12 hours and 100 donations in the last 3 months. Despite its popularity, donjon (as well as other CA PDGs) implores purely random generation techniques which are only evaluated based on the reachability (i.e. is there a path from the start to each room). Furthermore, although we have control over the input parameters for the topological randomness, the narrative elements (the contents, rather than the structure) of the dungeon are entirely arbitrary and often contradictory [11], [12]. In terms of the aforementioned reachability, donjon does not take in account the randomly allocated locked doors as the key allocation is left at the DM's discretion. That ultimately limits the usefulness of the tool as a narrative generator.

Constraint Propagators

Constraint Satisfaction Problems (CSPs) define a discrete set of variables with corresponding constraints that restrict the possible variable instantiations **[[cn of CSPs]]**. PDG has been described as a CSP on occasions where the variables and constraints define the layout **[[cn]]**, contents [13] or both layout and content [14] of a dungeon. Constraint Propagation (CP) is a particular method proven useful for solving CSP in the context of PDG. The algorithm selects a set of possible values for a particular variable and

propagates the result to the rest of the variables, adjusting their possible values accordingly. If the selected value narrows another variable's possible values to the empty set, we infer that this instantiation is suboptimal and we backpropagate to the last viable solution and retry. Furthermore it can be generalised that a level is solvable if an instantiation exists such that no variables are reduced to the empty set. Utilising this formalisation of the *solvability* of a level we can populate a level with varying degrees of complexity from ensuring that keys are always spawned before the door that they unlock to adjusting difficulty by measuring survivability metrics between rooms **[[cn]]**. Even thought CPs have solved some problems of the completely random generators, formulating a numerical constraint for a narrative element can be difficult. The complication is both in the translation of a story element to a numerical representation and in the computational expense that comes with computing large amounts of permutations.

Binary Space Partitioning

Other methods proposed by Thrall [11] and Brown [12] use a Binary Partitioning Tree [15] to create the layout of the dungeon. The simplicity in randomly dividing a 2D space until we have the desired number of rooms allows them to focus on the narrative content of the dungeon. Apart of just specifying layout parameters as is the case for the donjon generator, they attempt to generate a more contextual information for the game's narrative. Example is tailoring loot for specific players or noting that noise in a certain room will provoke monsters from nearby ones. Although narratively oriented, they are very basic guides for what the DM should then refine and often ignore (example would be a room labelled as provoking, yet no nearby rooms host monsters). These can be viewed as a digital version of the classic PnP generators we discussed in section 1.2.2 and as such incur the same issues. Furthermore, Thrall and Brown have shown that evaluating the *goodness* of a PDG method is a complicated task. A purely HCI evaluation of the qualities of a dungeon is understandable, but a single game of PnP RPGs takes on average 3-4 hours [16] making it very difficult to get a representative sample size. That is why the often preferred approach is instead of participants playing a game with a dungeon generated by the tool, to ask a static survey of which dungeon layout looks better.

Formal Languages

Formal languages and namely generative grammars have been used for PDG as they have a natural way of describing narrative structures [17]–[19].

Data-driven approaches

Arguably one of the most pleasant gameplay experiences in tabletop RPGs spring from exploring intricately structured dungeons and their mysteries. Their compelling features are due to the fact that they are made by a human designer that understands the needs of the players. Researchers like Dormans have acknowledged that the structure of human-made dungeons should be studied to better answer the question of "What makes a compelling dungeon". Sumerville [20] has shown that extracting features for layout and narrative from the Legend of Zelda games can be automated by training a machine learning ML model on data from the existing, man-made dungeon levels. ML for PDG not only promotes informed content generation, but it can be used as an exploratory mixed-initiative/analysis tool for formalising dungeon structures [21]. The approaches we have discussed have seen applications for both digital and tabletop RPGs, but data-driven PDG is largely an unexplored area. They are especially rare for tabletop RPGs so much that we do not know of publications on ML for PnP RPGS, presumably because of the lack of consistent datasets.

1.3 Mission and Space

An attentive reader would have noted that distinguishing between the level layout and contents is a common occurrence in popular PDG algorithms. This discrimination was formalised by Dormans [17] in his definitions of *Mission* and *Space*. The *Space* of a dungeon represents its topological structure and can be encoded as an undirected cyclic graph where each vertex is a room and each edge is a door or corridor. The *Mission* is the set of narrative tasks the player must accomplish in order for him to complete the dungeon. It can be encoded as a directed graph where each vertex is the objective and the edge directions show the required sequence of completion. How players interact with the level is governed entirely by the overlap of Mission and Space. As noted by Dormans, a *Mission* mapped on different layouts will result in drastically different exploration patterns and it is by understanding the necessities of the two separate entities that we can model player experience better. Due to this separation we have seen advancements in generating adaptable to the player game environments [22] and even reverse engineering the creation of *Missions* and *Spaces* by learning structure from data [20].

1.4 Aim

The dungeon's topology (Space) and contents (Mission) are correlated and embody the narrative of the game. As we have seen, dungeon generators usually implore bottom up approaches in which they apply rules for topology and then introduce dungeon content in an attempt to provide the structure for a captivating narrative. The various algorithmic methods have clear strengths and shortcomings in achieving the complex goal of creating an interesting story. From these observations it can be argued that that the ultimate *dungeon generator* is the human designer. Dungeons and therefore narratives produced by humans are the most compelling out of all created dungeons. If we implore a top-down approach of analysing what makes a good man-made dungeon, we could potentially achieve higher levels of narrative automation.

Deery [19] made the first steps by manually analysing submissions to the One Page Dungeon Contest *OPDC* [23] to extract a graph grammar for Mission generation. The Mission graph was then mapped to a physical Space in a 1:1 ratio. The result was that each room was limited to a single Mission element. It is trivial to see that the originals in OPDC do not impose such a restriction. One room can have multiple Mission elements (e.g. the key to unlocking the door is on the bandit's waist, Key + Encounter). Furthermore, Deery's approach was to manually look at 10 competition winners and heuristically extract the grammar rules, which he highlights that they do not capture all the possible patterns. An automated approach to learning would potentially solve that issue. Programmatically learning a level from data has been an object of interest for computer based RPGs [20], but has not been applied to PnP RPGs, presumably because of the lack of a consistent dataset.

In this paper we investigate if applying a data-driven approach to learning the Space of the dungeon can produce a map that is undistinguishable from human-made dungeon topologies. We discuss how we can use inference (Bayesian) networks to learn Space features from data. A review of the availability of data is conducted and complemented by assessing how meaningful features can be extracted. Subsequently we compare different network structures and learning algorithms and internally assess which model has the greatest statistical predictive capabilities using two scoring rules. We examine different methods that we can use for generating a dungeon topology from the best performant inference model. We conclude with a user study to externally validate how well the generator has approximated human-made dungeon topologies.

2 Bayesian Inference Networks

What are Bayesian networks? Bayesian Networks *BNs* [24] (also referred to as Inference Networks, Belief Networks or Directed Acyclic Graph *DAG* Models) are a graphical probabilistic model in which vertices are random variables *RVs* whose edges indicate causal beliefs. BNs assert that our domain is defined in terms of a joint probability distribution *PD* $P(X_1, \ldots, X_n)$ over a finite set of RVs X, each of which has a set of potential values it can take $X_i(\Omega) = \{x_1, \ldots, x_j\}$ with an associated PD. The network structure shows how the full join distribution factorises given the causal dependencies. This property gives us a way to encode our prior domain knowledge about the relationships between our RVs. Once we have encoded the RV interaction, using a BN as an inference tool is a matter of querying with a conditional set of RVs and noting how the conditional probability tables *CPTs* change given our new knowledge. In practice, conditioning is done by *observing* (i.e. instantiating) an RV to a particular value. Extracting information from our newly formed conditional distributions is done by sampling. Sampling involves simulating our network with respect to the observations in order to obtain point estimates. A detailed discussion of the sampling methodology will be provided in section 3.3.

To contextualise the inference process for PnP RPGs, we can look into Summerville's work on the ARPG Legend of Zelda dungeons [20]. After annotating the dungeon maps with elements of both Mission and Space (total number of rooms, treasures, monsters, critical path CP, etc.), he extracts features for the BN by parsing them as RVs. Conditioning on total number of rooms by stating $P(NumRooms = n) = 1$ results in a change of the PD for the possible CP values. An example logical outcome would be that the CPs that are greater than the total number of rooms would become impossible $P(CriticalPathLength > n \mid NumRooms = n) \to 0$. Then by sampling the network a point estimate for critical path length given that number of rooms in the dungeon is extracted. This *observe-then-infer* process can be done with any permutation of our RVs.

Why use Bayesian networks? The reasoning behind choosing BNs is that the probabilistic causal structures have been shown to perform well in capturing structural properties of dungeons from data for PDG tasks in ARPGs [20], [25]. Furthermore they give us a natural description about the dependencies in our model, which can be used as a supplemental

analysis tool for investigating human-made dungeons. As a PDG tool itself, the *observe-then-infer* pattern gives the user precise control over the desired properties of his dungeon without loss of automation. If a user observes a dungeon with 5 rooms and critical path of length of 3, they will be guaranteed in the final product. This solves the problem previous graph grammar data-driven approaches encountered, whereby the dungeon size input was used as an approximation for the actual number of rooms in the dungeon [19].

The downside of BNs is that their generalisation capabilities are highly dependent on the availability of data. If conditioning on an *undefined* set (e.g. there was never a data entry that showed a 5 room dungeon with CP of 3), the inference will fail. Fortunately there are ways to work around this problem which we will discuss in section 3.1.2.

A proposed way to solve this issue is by creating artificial data entries for the undefined cases. We will use linear interpolation due to its simplicity and the fact it has been shown to produce good results in general [26] and exceptional results on small datasets [27]. This can potentially solve the problems of undefined dungeons within the range of sizes of our dataset, but it cannot extrapolate to infer arbitrarily large dungeons. We would like to argue that this generalisation restriction is not a limitation of our algorithm, but of the data. Different dungeons exhibit different properties and it is not feasible to consider learning all possible existing dungeon configurations from a single dataset. This limitation is reflected in the dataset selection criteria in the next section.

3 Methods and Implementation

In this section we explore various issues with formalising graphical representations of human-made dungeons for use with our Bayesian networks. We make key assumptions about how to isolate Space features and We discuss Bayesian network architectures in three different levels of complexity.

3.1 Data acquisition

Note on notation *Features* or *RVs* are denoted by capital letters X where a boldface $\boldsymbol{X}$ is a shorthand for a set of RVs $\{X_1, \ldots, X_n\}$. *Parameters* refer to the Bayesian network parameters, i.e. the conditional probability tables CPTs defined by the different RV dependencies $P(Y_1, \ldots, Y_n \mid X_1, \ldots, X_n)$. Joint probability distributions $P(X_1, \ldots, X_n)$. Calligraphic $\mathcal{N}$ is a shorthand for the parameters for a given network. Calligraphic $\mathcal{D}$ is a wrapper for all the RVs (i.e. our data).

3.1.1 Data selection

Unsurprisingly the first priority of a data-driven method is the acquisition of a suitable dataset. The criteria we have chosen for our data is the following:

1. Data must be consistent. Dungeons have to be representatives of a similar and comparable category.

2. Data must be available under a research or similar open-source license.

The *OPDC* supplies a dataset that satisfies our requirements. All dungeons are in the same category of a one-session long dungeon and each dungeon is issued under the Creative Commons CC license. Furthermore due to the fact that it is a competition, we have an exemplar answer of our main question "What makes a compelling dungeon?" in the form of the competition winners. We apply the first filter and review only entries by competition winners. Due to the creative freedom the human designers have, it is naïve

to assume that the dungeon Space is necessarily a dominant factor in all dungeons. In order to narrow down to dungeons that do have a consistent representation of Space as an important asset, we apply a second filter by omitting dungeons that have:

1. no map (no Space)

2. exclusively linear topologies

3. non-standard Space traversal (the map topology is ignored due to a functional mechanism e.g. time-travel, game of political control, king of the hill, etc.)

4. inconsistent map (our Space features change as the game progresses e.g. intrigue campaigns where the objective constantly changes locations)

5. no goal defined by the designer (introduces the same problem as 4.)

3.1.2 Extracting Features

We will base our feature selection on the work done by Summerville [25] by selecting Space related variables from his Extreme Sparse model due to its simplicity and effectiveness. We extract five features in two categories:

1. Dungeon (Global): total number of rooms R and how many rooms are on the critical path L

2. Room (Local): depth D (how far away the room is from the entrance), critical path distance S (how far away is the room from a critical path room, 0 indicates it is on the CP) , total number of neighbours N

Each global and local set of features is defined by the joint PDs respectively $P(R, L)$ and $P(S, D, N)$, where each variable is discrete. A full dungeon is therefore characterised by

$$P(R, L, S, D, N) = P(R, L) \prod_{r}^{|R|} P(S_r, D_r, N_r) \tag{3.1}$$

where $|R|$ is the total number of rooms defined by the instantiation of R. To extract the five features, we need to transform the artistic and graphically rich textual representations of the dungeons in our dataset. Unfortunately automating feature extraction is very difficult due to the variant stylistic formatting and the prose like description of the tasks. Because of this, we establish a systematic way of manually annotating the Space of the dungeon as an abstract undirected graph from which we can calculate all the different features (see Appendix A for example B.2).

The dungeon sizes in our dataset range from two to 27, but due to our restrictions combined with chance, dungeons of some sizes were never recorded. Namely dungeons with sizes $\{4, 23\}$. We could conclude that lack of dungeons from those sizes means they are highly improbable to the point where we should not be concerned with their existence. This is what we do with dungeons that are outside of the range of sizes for one-page adventures. But in terms of utility of our tool we would like to be able generate any dungeon within that range. This issue of undefined data is the main drawback of inference network as outlined in chapter 2. Our proposed solution is to linearly interpolate *LI* dungeons that are close to the missing ones in order for us to get estimates for the global RVs. We have chosen LI due to its simplicity and the fact it has been shown to produce good results in general [26] and exceptional results on small datasets [27]. Room (local) RVs could potentially also be linearly interpolated by using all the rooms from the neighbouring dungeons but this poses several questions about what does it *mean* to interpolate 4 rooms and 6 rooms. A better and more intuitive way to handle them is by leveraging the properties of the BNets. We can represent the rooms for the interpolated dungeons as unknown values to be inferred from the known data. We are treating the data itself as a parameter to be estimated from our other observations. This is a convenient way that inference networks treat missing values or sometimes even missing variables [28, p45]. However this representation of the unknown will impose a restriction on what CPT learning algorithms we can use. We discuss them in section 3.2.3.

The final dataset is saved in a CSV format `DR_data.csv` and consists of 92 dungeon + 1193 room entries and 10 linearly interpolated dungeon entries with 189 unknown rooms. Common statistics of the dataset can be found in appendix A.

3.2 Model selection

What we need to do to use Bayesian Networks? We need to define two key attributes of the BN:

1. The causal network's topology

2. The network parameters

In this paper we will empirically compare different models in search of the one with highest statistical predictive capabilities. We will evaluate three structures with two CPT learning algorithms against a uniform distribution (control) for a total of 7 models. The structures are the previously proposed Extreme Sparse model [25] (modified for Space only), an automatically

learned structure using the Tree Augmented Naïve Bayes and a fully connected model. The three architectures are chosen as such because each one differs in complexity and in assumptions. The comparison is an exploratory experiment aiming to see which pair of complexity and assumptions fits the OPDC data best. In regards to parameter learning, the CPT learning algorithms will be Expectation Maximisation *EM* and Gradient Descent *GD* due to their ability of handling missing data. Implementation of all BN related algorithms is done in the Netica Bayesian Network development software [29].

3.2.1 Bayesian Network topology

Summerville's Extreme Sparse Model *SESM*

In exploring data-driven PDG for the Legend of Zelda franchise, Summerville found out that when dealing with a small dataset (38 dungeons with 1031 rooms), the SESM is able to capture high level properties of both Mission and Space with the highest accuracy in respect to the learning criteria and the lowest complexity. Following our assertion that digital RPGs differ from PnP RPGs only in mechanisms of interaction, we will base one of our BN topologies on Summerville's work. In order to do so, we flatten down the graph to use only the Space features. The flattening is done by reconstructing the SESM in Netica and applying the `AbsorbNodes_bn` [28, pp. 62–63]. The function removes the unused Mission RVs by effectively treating them as constants and propagating any implied dependencies from the other nodes. In this way we remove the unused RVs and maintain the joint distribution dependencies between our Space features. The transformation can be seen in figure 3.1. The resulting model describes the following factorisation over our RVs

$$P(R, L, S, D, N) = P(R)P(S)P(L \mid R)P(D \mid R, L)P(N \mid D, S) \quad (3.2)$$

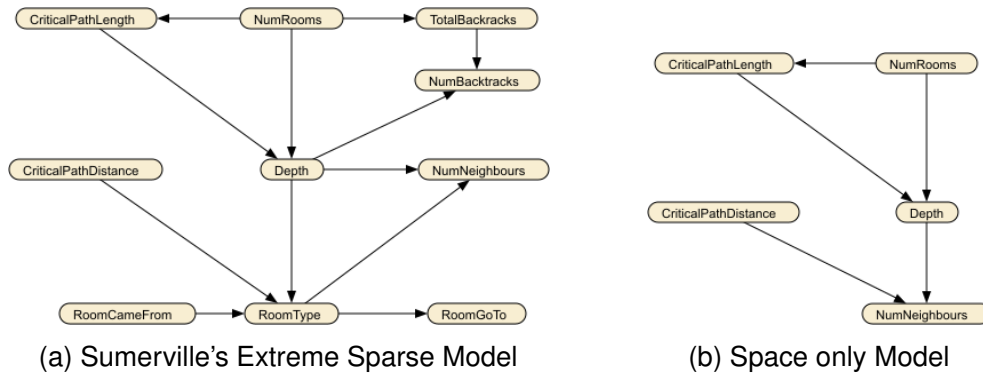(a) Sumerville's Extreme Sparse Model   (b) Space only Model

Figure 3.1: Expert defined Network topologies

Tree Augmented Naïve Bayes

In the above approach we applied specialist knowledge from Sumerville's experiments to create the BN structure. For that network to be valid, we made two assumptions: equivalence between PnP and ARPG dungeons and that flattening Mission to Space retains the same operational meaning due to the BN retaining the same conditional dependencies. To contest these somewhat ambitious assumptions, we will also create a BN by ignoring any domain knowledge we have and infer the structure purely based on the data using Tree Augmented Naïve Bayes *TAN* [30].

Informally the learning procedure can be defined as finding the most likely BN structure that *agrees* with our data. That means we are searching for the network structure that is most likely to have been produced given the data. TAN finds the most likely network structure in two steps:

1. Start with a universal model (Create a naïve Bayes model)

2. Find correlated variables in the data and connect them with causal edges

The universal model TAN chooses is the standard naïve Bayes. It asserts that all RVs are conditionally independent given one specific RV (the class RV). We will choose the class to be R due to its contextual importance. The naïvity assumption implies that

$$P(R, \boldsymbol{X}) = P(R) \prod_{x}^{\boldsymbol{X}} P(x \mid R) \tag{3.3}$$

where $\boldsymbol{X} = \{L, S, D, N\}$. With this very restrictive assertion we have a usable BN structure. The conditional independence given the class variable is useful in practice and has been shown to produce competitive results [31], but the operational meaning of this assumption might cause issues if we use it as is. Namely we are assuming that as long as we know how big the dungeon is, the room variables (depth, critical path distance and neighbours) do not influence one another, which does not make much sense. For precisely this reason, the TAN algorithm analyses our data and creates *augmenting edges*. They are causal links connecting RVs that are found to have correlations. The resultant network has an informed causal structure and is alleviated from the restrictions of the naïve assumption. The networks resulting from each of the two steps can be seen in figure 3.2.
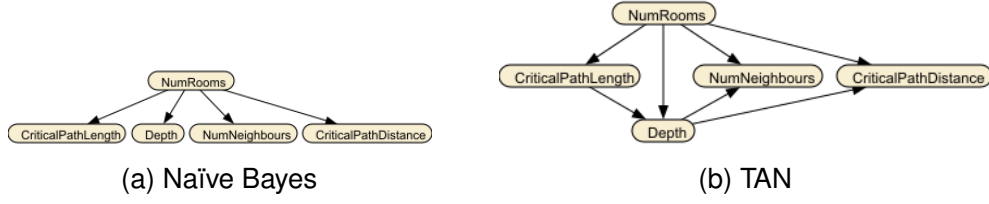
(a) Naïve Bayes   (b) TAN

Figure 3.2: Learned Network topologies

3.2.2 Fully Connected

Due to the fact that we have a small number of nodes, we can afford the computational expense of running a model with no independence assumptions. In effect we're solving the original model 3.1, but we will include a heuristic assumption about the direction of the causality in order to maintain the BN requirement of having an acyclic graph. Namely we assert that our global parameters cause our local parameters. The model can be seen in figure 3.3.
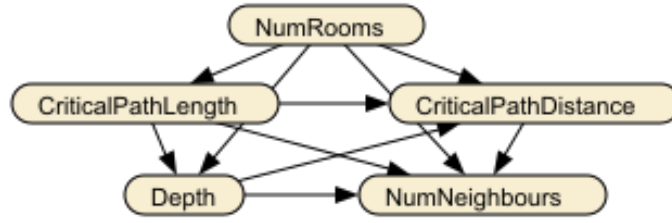

Figure 3.3: Fully connected graph with heuristic causal flow

3.2.3 Parameter Learning

When we learn parameters (CPTs for each RV) we are trying to find the most likely PD for each RV that agrees with our data. More formally if $\mathcal{N}$ is the network's CPTs and $\mathcal{D}$ is the data then from Bayes's rule we know that $P(\mathcal{N} \mid \mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{N})P(\mathcal{N})}{P(\mathcal{D})}$. We can formulate the parameter learning as an optimisation problem where we find the best values for the posterior $P(\mathcal{N} \mid \mathcal{D})$ by maximising the likelihood $P(\mathcal{D} \mid \mathcal{N})$ given our prior $P(\mathcal{N})$ [28, pp. 46–48]. This is in fact the common maximum likelihood estimation MLE **[[cn for MLE]]**, but because we have missing data, $P(\mathcal{D})$ is undefined and therefore the MLE is also undefined. To work around this issue and learn the likelihood we review two algorithms Netica has in its toolkit.

Expectation Maximisation EM

EM [32] is a numerical algorithm that can handle missing data. It assigns random probabilities to our missing values and iteratively alternates between maximising the parameters of the model given the data (Maximisation step) and finding better estimations of the missing data (Expectation step). It converges after a certain set of predefined iterations or when no new updates are happening.

Gradient Descent GD

GD is another algorithm that is able to find parameters for missing data. Netica specifically uses conjugate gradient descent **[[cn]]** whereby it maximises $P(\mathcal{N} \mid \mathcal{D})$ by computing the steepest gradient, taking in account the previous one [28, p47]. Both EM and GD risk getting stuck in local optima.

3.3 Implementation

Obtaining a dungeon is done in three steps, seen in fig 3.4. Initially we run the inference process and obtain a set of estimates for all of our features given the size of the dungeon set by the user. We will call the set of estimates our *dungeon configuration*. Subsequently we convert the configuration to an abstract graph by applying a Constraint Propagation algorithm. Finally we convert the abstract graph to a usable dungeon image.
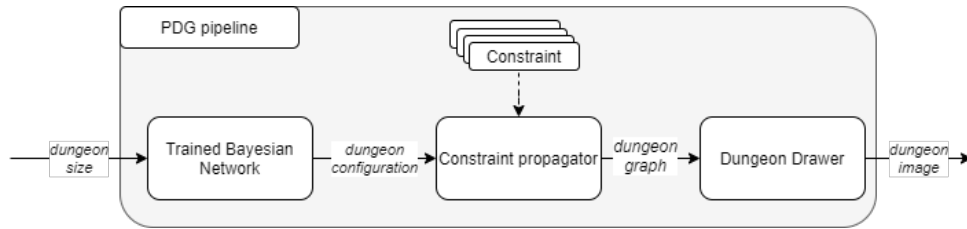

Figure 3.4: Pipeline for the data-driven PDG tool

3.3.1 Dungeon configuration

As we discussed in chapter 2, generating a dungeon with five rooms is a matter of fixing (*observing*) the `NumRooms` parameter and inferring the rest. When an RV is observed, Netica updates the CPTs for all dependent RVs with the newly added belief. We can then draw a value from the newly

formed distributions. This process of taking a single value from a PD is known as *sampling* the PD. After a value for a previously unknown RV has been sampled, it is added as an observed RV for any subsequent inference. The user can choose virtually any permutation of variables to be fixed or inferred. For consistency of this experiment, we start by observing only the size of the dungeon via `NumRooms`. Obtaining a dungeon configuration from our model is done in three steps:

1. Infer the dungeon RVs and validate them

2. Infer the room RVs and validate them

3. Convert our fixed RVs to a graphical dungeon

Dungeon Sampling   As the size of the dungeon is observed, we only require to sample and validate the CP length. The validation is a sanity check to see if the value for the CP length is not greater than the size. If the validation is successful we continue the inference process. Otherwise we redraw the sample. Formally what we are doing is obtaining a point estimate for the CP length $L = l$ from the PD $P(L \mid R = r)$ where $l \leq r$.

Room Sampling   Once the size and CP length of the dungeon have been observed we can infer the specific room parameters in the same way as before. The validation for room samples is that the depth and number of neighbours cannot exceed the size of the dungeon. If a sample is invalid, we again retry. This is repeated until we have drawn a valid sample for each room in that dungeon. Formally speaking, for each room i we obtain a point estimate for the room features $(D_i, S_i, N_i) = (d_i, s_i, n_i)$ from the PDs $\prod_i^r P(D_i, S_i, N_i \mid R = r, L = l)$ where $\forall d, \forall n \leq r$. The sequence of operations for dungeon and room sampling can be seen in Appendix A algorithm 1.

3.3.2 Converting from configuration to Space

We considered the different PDG methods as discussed in section 1.2.2 for this task and chose Constraint Propagators CPs. The reason behind this decision is that the samples we have drawn provide an intuitive description of the constraints. More importantly, unlike other PDG methods, CPs can be used as an analytics tool as they give us an intuition of how easy it is to *find* a layout that satisfies the configuration. As highlighted before, CPs' computational expense grows exponentially with the increase of variables and constraints. To alleviate this issue we have coded two practical optimisations: we operate on connection tuples that are of predetermined length

and we predefine the critical path by connecting all rooms with $S = 0$. The current constraints are as follows:

1. The dungeon is exactly of size R

2. The ith room must be exactly D_i rooms away from the entrance

3. The ith room must have exactly N_i connections

4. The ith room must be exactly S_i rooms away from the critical path

5. CP must not be altered

6. Resultant dungeon must be planar, i.e. no corridors cross

7. Every room must be reachable i.e. there is a path from the start to each room

The algorithm is an adapted version of Horswil's CP for the 2013 GDC conference [33] and can be seen in detail in Appendix A, algorithm 2.

The platform of choice for this project is .NET with the language of implementation being C#. We chose this as Netica has native support for C# via its COM API and because of the availability of useful third-party utilities. Namely we will be using Nepožitek's PDG tool [34] which allows us to input an abstract graph for the desired topology and produces a rendered image that is ready for player usage. The reason behind using a third-party solution for the dungeon rendering is due to time constraints and the fact that the visualiser is not a centre piece of this experiment. Our sole requirements are that the rendering is consistent and robust enough to handle our produced topologies. Nepožitek's PDG satisfies these requirements [35].

4 Results

4.1 Internal Validation

As we are modelling uncertainty, we can pose the question of "How well do we reduce uncertainty". This question can be answered by using the BNet as a classifier in which we only know the user input parameters (i.e. the size of the dungeon) and see how well we predict the rest of the parameters. We construct a within sample test where we measure how well the different models estimate $P(L, S, D, N \mid R)$. We will use two metrics: the classification error E due to its natural meaning in conjunction with the well known quadratic loss (Brier score) [36]. It should be noted that Netica uses the original formulation of the quadratic loss which is between zero and two. The comparison between our different models can be seen in table 4.1. We observe that the TAN and fully connected networks both have the highest predictive capabilities therefore we will choose TAN to continue forward as it is the simpler structure. Our constraint propagation algorithm further reports that on average (sample size of 15) it takes 451 retries and 25 minutes to sample a single 10 room dungeon that satisfies our physical consistency requirements described in section 3.3.2.

	CPLength		CPDistance		Depth		Neighbours		
	loss	E	loss	E	loss	E	loss	E	
Uniform (base)	0.9	96.74%	0.8333	58.17%	0.9286	92.04%	0.8889	70.08%	
Extreme Sparse	0.5524	48.63%	0.6744	58.17%	0.8184	74.43%	0.7486	67.64%	
TAN	**0.5524**	**48.63%**	**0.6462**	**54.82%**	**0.8184**	**74.43%**	**0.7124**	**60.6%**	EM
Fully Connected	**0.5524**	**48.63%**	**0.6464**	**54.82%**	**0.8184**	**74.43%**	**0.7131**	**60.6%**	
Extreme Sparse	0.8132	52.1%	0.7004	64.63%	0.8689	78.46%	0.7563	67.48%	
TAN	0.816	53.11%	0.6751	57.84%	0.8815	78.46%	0.7686	65.88%	GD
Fully Connected	0.8177	50.58%	0.6855	57.84%	0.877	79.3%	0.7597	66.39%	

Table 4.1: Structure and parameter learning comparison.

4.2 External Validation

User study design   Previous work the aesthetics of composition has shown that people can discriminate between computer generated and

original artwork [37] [38]. To validate if our model is able to generate dungeons that are undistinguishable from human-made dungeons we will conduct a preference study. Dungeons from the OPDC, our inferred set and a random control group are selected and formatted in the same visual style. Three sets of pairs are randomly permuted and contain an even distribution of the possible combinations *(OPDC,ours),(OPDC,random),(ours,random)*.

   10, 13 and 8 room dungeons (examples in Appendix B fig B.3) were used as they are the most common and additionally 13 is the mean dungeon size. The experiment was done as an online survey.

Demographics   The participants were selected to be adults with at least some experience in PnP RPGs.

5 Discussion

As expected, all models surpass the uniform distribution control. However the above 50% error rate reveals that with the assumptions we made in section 3.1.2 for making the problem description more concrete, we have removed crucial explanatory variables. The low predictive capabilities for the Space model is an indicator that the Mission parameters or some other unmodeled hidden set of parameters are critical when trying to approximate human-made dungeons. Furthermore the constraint satisfier struggles to generate the sampled dungeons even with our optimisations. As seen if figure **[[fig for elapsed time for net train]]**, the time it takes to produce a layout grows exponentially with the number of rooms. For the most common dungeon size (10 rooms) the inference procedure takes on average

The EM algorithm consistently outperforms GD, albeit not by a large margin.

5.1 Critique

1. Data

    a) availability and extraction.

        i. Could extend to include even non-winning entries for the sake of having a greater sample size. More datasets can be considered. We've use OPDC due to its CC license, but paid modules exist.

        ii. Data extraction has been a manual process. Although due diligence is paid, noise introduced from human error is inevitable.

2. Method

    a) Cannot generalise to unseen cases. We can interpolate for missing data, but we cannot extrapolate. Argue that this is an issue of all ML approaches, not just BNets, because we're trying to capture properties of One-Page dungeon.

    b) This paper is considering only Space gen. A more robust ap-

20

proach would be needed to extract consistent and meaningful parameters for Mission. For Legend of Zelda there is a discrete subset of things you can do so that is why Mission extraction is possible. Deery has shown that formalising Mission in PnP RPG's is difficult due to the variant and creative nature of the human-made dungeons.

3. Validation

   a) We have decided to approach the external evaluation in a quantitative, rather qualitative measuring. That is due to the fact that conducting a study with a high environmental index is difficult due to multiple confound factors that are due to the nature of a tabletop RPG session. Not only does a one-session adventure usually take around 3 hours **[[cn]]**, but they are extremely variant between groups of players and DMs. An experiment that analyses how players use the generated dungeon rather than just discriminating between different dungeons topologies would give us more insight in the success of the recreation of a human-made dungeon.

6 Conclusion

6.0.1 Future Work

1. Rerun with more data (OPDC is conducted every year)

2. Automate the Feature Extraction process (OCR + NLP)

3. Incorporate Mission parameters as well

4. Analyse PnP specific dungeons and find more parameters that might be useful. E.g. size of rooms, multiple CPs,

5. Although not considered in this paper, practical optimisations like using general linear solvers can be done.

6. Conduct a study with greater environmental impact (e.g. actually get people to play instead of using an aesthetics study)

A Appendix
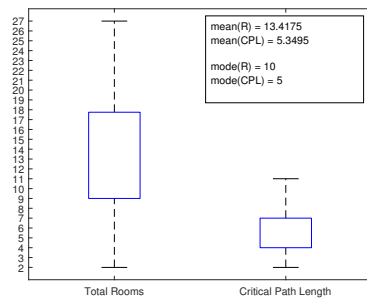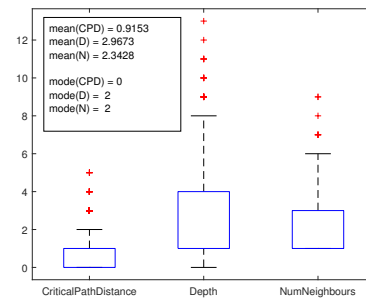
(a) Global (dungeon)   (b) Local (Dungeon)

Figure A.1: Parameter statistics

Algorithm 1: Dungeon and Room sampling algorithm

---

input :BNetModel, int *observed_R*

output :int estimates for the dungeon and each room parameter
$(R, L, \{S_r, D_r, N_r\})$

---

isDungeonValid ← *false*

while ¬*isDungeonValid* **do**

    R ← BNetModel.Observe(*observed_R*)

    L ← BNetModel.Sample()

    isDungeonValid ← *ValidateDungeon*(R, L)

    if *isDungeonValid* **then**

        for $r \in R$ **do**

            temp_R ← *R*

            temp_L ← *L*

            isRoomValid ← *false*

            while ¬*isRoomValid* **do**

                (S_r, D_r, N_r) ← BNetModel.Sample()

                BNetModel.ClearObservations()

                R ← BNetModel.Observe(*temp_R*)

                L ← BNetModel.Observe(*temp_L*)

                isRoomValid ← *ValidateRoom*(S_r, D_r, N_r)

            end

        end

    end

end

Algorithm 2: Constraint Propagation algorithm

---

input : set of rooms to be connected $\mathcal{R}$, set of constraints for the
dungeon and rooms $\mathcal{C} = (R, L, \{S_r, D_r, N_r\})$

output : A connected dungeon graph

```
Map()
```

Function `Map()`:

> **for** $r \in \mathcal{R}$ **do**
> > $r_v \leftarrow$ all possible values for r
>
> **end**
> `MapOne()`

Function `MapOne()`:

> **if** $\forall r \mid$ `IsSingletonSet`(r_v) **then**
> > **return**
>
> **end**
> $r' \leftarrow$ `ChooseRand`$(r \mid$`IsSingletonSet`$(r_v))$
> **for** $v \in r'_v$ **do**
> > **try**
> > > `Reduce`$(r'_v, \{v\})$
> > > `MapOne()`
> >
> > **catch** *Failure*
> > > `Undo()`
>
> **end**
>
> **Function** `Reduce`(r_v, set):
> > **if** $set = \emptyset$ **then**
> > > throw Failure
> >
> > **end**
> > **if** $r_v \neq set$ **then**
> > > $r_v \leftarrow set$
> > > **for** $c \in \mathcal{C}$ **do**
> > > > `Propagate`(c, r_v)
> > >
> > > **end**
> >
> > **end**
>
> **Function** `Propagate`(c, r'_v):
> > **for** $r_v \in c_r$ **do**
> > > **if** $r_v \neq r'_v$ **then**
> > > > `Reduce`(r_v, c, r_v)
> > >
> > > **end**
> >
> > **end**

B Appendix

10 GP GEMSTONES	
d12	**Stone Description**
1	Azurite (opaque mottled deep blue)
2	Banded agate (translucent striped brown, blue, white, or red)
3	Blue quartz (transparent pale blue)
4	Eye agate (translucent circles of gray, white, brown, blue, or green)
5	Hematite (opaque gray-black)
6	Lapis lazuli (opaque light and dark blue with yellow flecks)
7	Malachite (opaque striated light and dark green)
8	Moss agate (translucent pink or yellow-white with mossy gray or green markings)
9	Obsidian (opaque black)
10	Rhodochrosite (opaque light pink)
11	Tiger eye (translucent brown with golden center)
12	Turquoise (opaque light blue-green)

Figure B.1: Loot table for gemstones taken from Dungeon Master Guide 5e

(a) select a dungeon map

(b) annotate (or reannotate) rooms for convenience
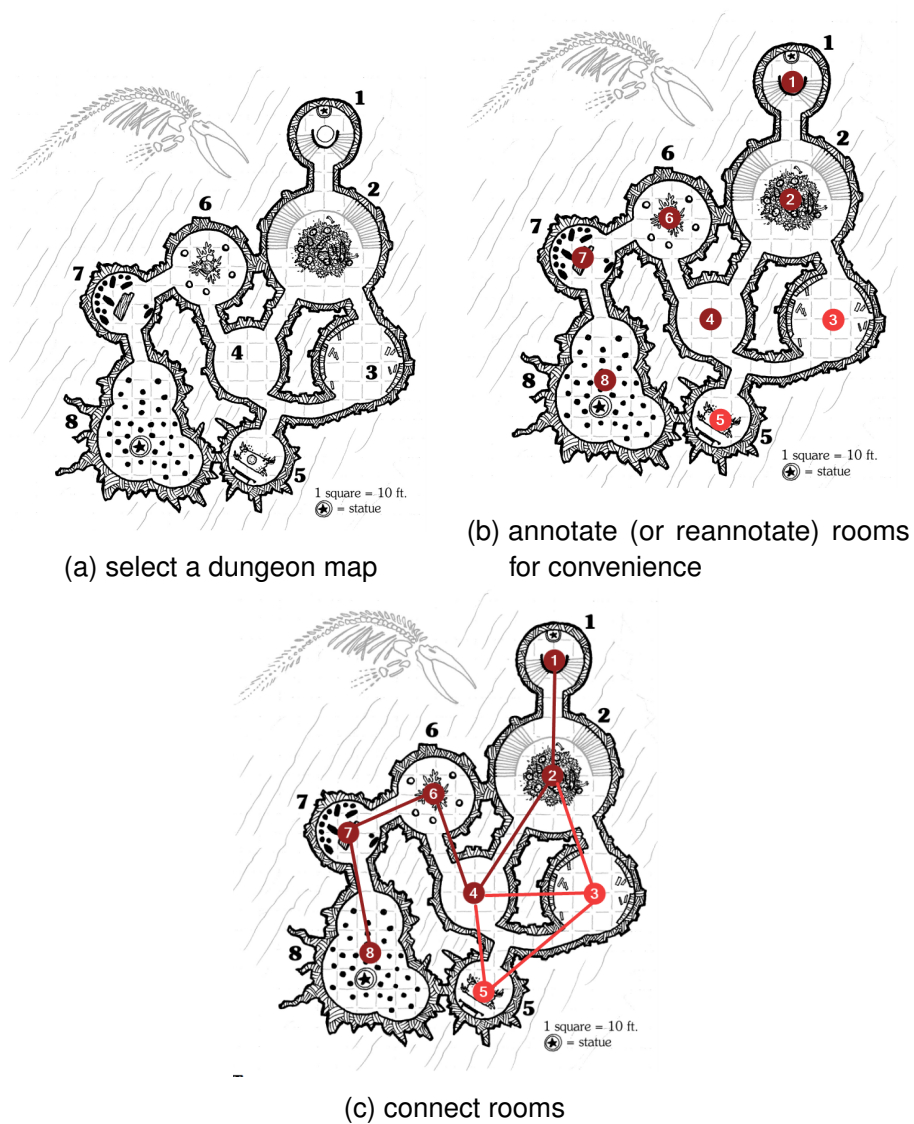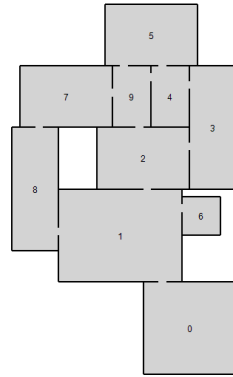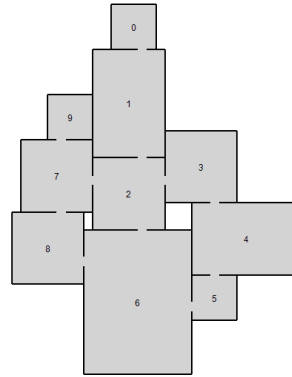

(c) connect rooms

Figure B.2: Steps for transcribing Sepulchre of the Abyss 2015 (dark red is CP)

(a) 10 room Bayesian PDG

(b) The Vermin Hollow OPDC 2011

(c) 10 room Random PDG

Figure B.3: Generated dungeons

Bibliography

[1]   *The legend of zelda*, [NES SNES], Kyoto Japan: Nintendo, 1986.

[2]   *Diablo*, [PC CD-ROM], California: Blizzard Entertainment, 1997.

[3]   *Fallout: A post nuclear role playing game*, [PC CD-ROM], Los Angeles: Interplay Productions, 1997.

[4]   *The elder scrolls v. skyrim*, [PC CD-ROM], Rockville, MD: Bethesda Softworks, 2013.

[5]   A. Tychsen, M. Hitchens, T. Brolund and M. Kavakli, 'The game master,' in *Proceedings of the Second Australasian Conference on Interactive Entertainment*, ser. IE '05, Sydney, Australia: Creativity & Cognition Studios Press, 2005, pp. 215–222, ISBN: 0975153323.

[6]   G. Gygax and D. Arneson, *Dungeons and dragons*, 1974.

[7]   H. Johnson and A. Allston, *Dungeon master's design kit (1e)*, Reading, Massachusetts, 1988.

[8]   A. Tychsen, 'Role playing games: Comparative analysis across two media platforms,' in *Proceedings of the 3rd Australasian Conference on Interactive Entertainment*, ser. IE '06, Perth, Australia: Murdoch University, 2006, pp. 75–82, ISBN: 869059025.

[9]   *Cellular automata method for generating random cave-like levels*, http://roguebasin.roguelikedevelopment.org/index.php?title=Cellular_Automata_Method_for_Generating_Random_Cave-Like_Levels, Accessed: 27/01/2020.

[10] *Donjon 5e dungeon generator*, https://donjon.bin.sh/5e/dungeon/, Accessed: 27/01/2020.

[11] H. Thrall, 'Procedural generation of a dungeon for dungeons and dragons,' M.S. thesis, University of York, 2017.

[12] S. Brown, 'Prototyping software for the complex human task of creating dungeons and dragons dungeons with narratives, using user centred design and procedural generation,' M.S. thesis, University of York, 2018.

[13] I. Horswill and L. Foged, 'Fast procedural level population with playability constraints,' in *Proceedings of the Eighth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, ser. AIIDE'12, Stanford, California, USA: AAAI Press, 2012, pp. 20–25.

[14] M. C. Green, A. Khalifa, A. Alsoughayer, D. Surana, A. Liapis and J. Togelius, 'Two-step constructive approaches for dungeon generation,' in *Proceedings of the 14th International Conference on the Foundations of Digital Games*, ser. FDG '19, San Luis Obispo, California: Association for Computing Machinery, 2019, ISBN: 9781450372176. DOI: 10.1145/3337722.3341847. [Online]. Available: https://doi.org/10.1145/3337722.3341847.

[15] H. Futchs, Z. Kedem and B. Naylor, 'On visible surface generation by a priori tree structures,' vol. 14, Dec. 1988, pp. 39–48. DOI: 10.1145/800250.807481.

[16] *Poll: How long do your sessions usually run?* https://www.reddit.com/r/DnD/comments/8cxhcy/poll_how_long_do_your_sessions_usually_run/, Accessed: 29/03/2020.

[17] J. Dormans, 'Adventures in level design: Generating missions and spaces for action adventure games,' in *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, ser. PCGames '10, Monterey, California: Association for Computing Machinery, 2010, ISBN: 9781450300230. DOI: 10.1145/1814256.1814257. [Online]. Available: https://doi.org/10.1145/1814256.1814257.

[18] H. Cadogan, 'Procedurally generating dungeons and dragons content using context-free grammars,' M.S. thesis, University of York, 2019.

[19] C. Deery, 'Generating dungeons & dragons dungeons that make visible sense,' M.S. thesis, University of York, 2019.

[20] A. J. Summerville, M. Behrooz, M. Mateas and A. Jhala, 'The learning of zelda: Data-driven learning of level topology,' in *Proceedings of the 10th International Conference on the Foundations of Digital Games*, Jun. 2015.

[21] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgård, A. Hoover, A. Isaksen, A. Nealen and J. Togelius, 'Procedural content generation via machine learning (pcgml),' *IEEE Transactions on Games*, vol. PP, Feb. 2017. DOI: 10.1109/TG.2018.2846639.

[22] J. Dormans and S. Bakkes, 'Generating missions and spaces for adaptable play experiences,' *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 216–228, Sep. 2011, ISSN: 1943-0698. DOI: 10.1109/TCIAIG.2011.2149523.

[23] *One page dungeon contest*, https://www.dungeoncontest.com/, Accessed: 21/02/2020.

[24] J. Pearl, 'Bayesian networks: A model of self-activated memory for evidential reasoning,' in *Proceedings of the 7th Conference of the Cognitive Science Society, University of California, Irvine, CA, USA*, 1985, pp. 15–17.

[25] A. J. Summerville and M. Mateas, 'Sampling hyrule: Sampling probabilistic machine learning for level generation,' 2015.

[26] P. H. Ibargüengoytia, U. A. García, J. Herrera-Vega, P. Hernandez-Leal, E. F. Morales, L. E. Sucar and F. Orihuela-Espina, 'On the estimation of missing data in incomplete databases: Autoregressive bayesian networks,' in *ICONS 2013*, 2013.

[27] J. Yu, V. A. Smith, P. P. Wang, A. J. Hartemink and E. D. Jarvis, 'Advances to bayesian network inference for generating causal networks from observational biological data,' *Bioinformatics*, vol. 20, no. 18, pp. 3594–3603, 2004.

[28] N. S. Corp., *Netica api programmer's library reference manual*, English, version Version 4.18+, Norsys Software Corp., 236 pp., October 18, 2010.

[29] Norsys Software Corp., *Netica*, version 6.07, 1995. [Online]. Available: https://www.norsys.com/index.html.

[30] N. Friedman, D. Geiger and M. Goldszmidt, 'Bayesian network classifiers,' *Machine Learning*, vol. 29, pp. 131–163, Nov. 1997. DOI: 10.1023/A:1007465528199.

[31] I. Rish, 'An empirical study of the naïve bayes classifier,' *IJCAI 2001 Work Empir Methods Artif Intell*, vol. 3, Jan. 2001.

[32] A. P. Dempster, N. M. Laird and D. B. Rubin, 'Maximum likelihood from incomplete data via the em algorithm,' *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1, pp. 1–38, 1977, ISSN: 00359246. [Online]. Available: http://www.jstor.org/stable/2984875.

[33] *How to build a constraint propagator in a weekend*, unpublished draft, Northwestern University, 2013.

[34] O. Nepozitek, *Procedural level generator*, version 1.06, 2018. [Online]. Available: https://ondrejnepozitek.github.io/ProceduralLevelGenerator/.

[35] O. Nepozitek and J. Gemrot, 'Fast configurable tile-based dungeon level generator,' 2018.

[36] J. Pearl, 'An economic basis for certain methods of evaluating probabilistic forecasts,' *International Journal of Man-Machine Studies*, vol. 10, no. 2, pp. 175–183, 1978, ISSN: 0020-7373. DOI: https://doi.org/10.1016/S0020-7373(78)80010-8. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0020737378800108.

[37] I. C. McManus, B. Cheema and J. Stoker, 'The aesthetics of composition: A study of mondrian,' *Empirical Studies of the Arts*, vol. 11, no. 2, pp. 83–94, 1993. DOI: 10.2190/HXR4-VU9A-P5D9-BPQQ. eprint: https://doi.org/10.2190/HXR4-VU9A-P5D9-BPQQ. [Online]. Available: https://doi.org/10.2190/HXR4-VU9A-P5D9-BPQQ.

[38] A. Furnham and S. Rao, 'Personality and the aesthetics of composition: A study of mondrian and hirst.,' *North American Journal of Psychology*, vol. 4, pp. 233–242, Jan. 2002.