Department of Computer Science

Submitted in part fulfilment for the degree of BEng.

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

Angel Simeonov

9th March 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 platforms of interaction (the media). 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. Describe the allowed actions for the player at any given moment in the game.
2. Narrative elements. 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, Diablo, Fallout and many alike have both sets of rules defined by the game designers. A functional rule in an RPG like Skyrim is that you can attack with the left mouse button and a story element is that you are a Dragonborn 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 [1], allowing for the player to have more freedom in the exploration of the world and as such create the sensation that the player has some 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 [2]. 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 generative methods for Dungeons 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 **need figure of loot/encounter table**. 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 [3] 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. **show an example from the ADND module**. 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 [4]. 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 naturally 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 semantic content or ommit 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**[citation needed for CA and fig. showing agents in action?]]**. The implementation of these mutation operators define if the CA will simulate erosion [5] or man-made structures such as rooms and corridors. The latter is the case of the donjon dungeon generator [6]. Its simplicity and degrees of customisations of both layout and style has made donjon a popular generator choice with more than 2500 generated dungeons in the last 12 hours and 100 donations in the last 3 months **[[citation needed?]]**. 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 semantic elements (the contents, rather than the structure) of the dungeon are entirely arbitrary and often contradictory [7], [8]. 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]]**. Dungeon generation has been described as a CSP on occasions where the variables and constraints define the layout **[[cn]]**, contents [9] or both layout and content [10] of a dungeon. Constraint

Propagation (CP) is a particular method proven useful for solving CSP in the context of dungeon level generation. 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 if an instantiation for all variables that does not narrow any variable to the empty set exists, the level is solvable. 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 Partitioning

Formal Languages

Data-driven approaches

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 [11] in his definitions of *Mission* and *Space* in relation to the ARPG Legend of Zelda game series. 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 interaction of these two concepts. 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 [12] and even reverse engineering the creation of *Missions* and *Spaces* by

learning structure from data [13].

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 made the first steps by manually analysing submissions to the One Page Dungeon Contest *OPDC* [14] to extract a graph grammar for Mission generation [15]. 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 [13], 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 various 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* [16] (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 [13]. 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 [13], [17]. 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 [15].

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. 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 [18] and exceptional results on small datasets [19]. 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

3.1 Data acquisition

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 parameters are allowed to change 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 [17] by selecting Space related parameters from his Extreme Sparse model due to its simplicity and effectiveness. We extract five parameters 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 parameters is defined by the joint PDs respectively $P(R, L)$ and $P(S, D, N)$. 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 parameters, 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 parameters as shown in **[[insert figure of 2015 Sepulchre of the Abyss transcriptions]]**.

TODO: Filling in the data gaps. Linear interpolation and artificial datapoints

The final dataset is saved in a CSV format and consists of 92 organic dungeon + 1193 organic room entries and 10 linearly interpolated dungeon entries. 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. Conditional probability tables *CPTs* for each RV

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 three CPT learning algorithms for a total of nine models. The structures are the previously proposed Extreme Sparse model [17] (modified for Space only), an automatically learned structure using the Tree Augmented Naïve Bayes and a fully connected model. The CPT learning algorithms will be a simple joint occurrence count, Expectation Maximisation *EM* and Gradient Descent *GD*. Implementation of all BN related algorithms is done in the Netica Bayesian Network development software [20].

3.2.1 Bayesian Network topology

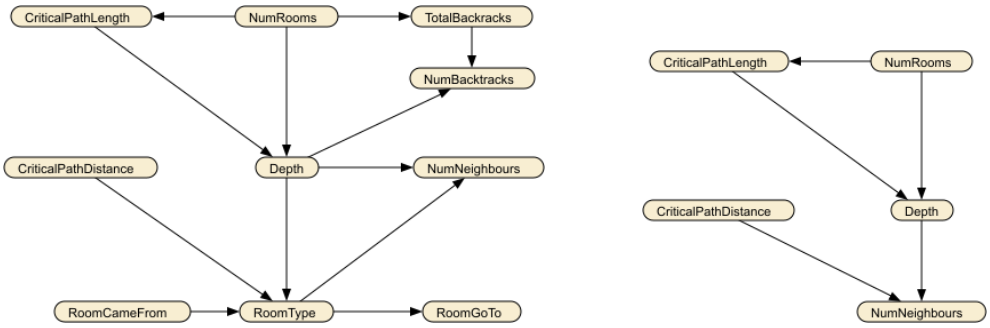
Summerville's Extreme Sparse Model *SESM*

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 parameters. The flattening is done by reconstructing the SESM in Netica and applying the `AbsorbNodes_bn` [21, 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 parameters. The transformation can be seen in figure 3.1. The resulting model describes the following factorisation over our RVs

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

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 asserted two assumptions: equivalence between PnP and ARPG dungeons and that flattening Mission to Space retains the same operational



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

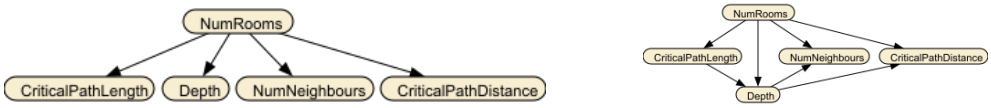
Figure 3.1: Expert defined Network topologies

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* [22].

Informally the learning procedure can be defined as finding the most likely BN structure that agrees with our data. **[[more formal description of maximising the posterior in respect to P(Network|Data)?]]**. The TAN algorithm starts with the usual naïve Bayes model structure which states 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 naïve 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, in practice, have a BN structure that we can use. But although the induced bias towards the class is often shown to be practically useful, in our case assuming that the room parameters are independent from each other 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 final structure which can be seen in 3.2 is the original naïve Bayes with the extra added augmenting edges, resulting in a model that has an informed causal structure and is alleviated from the restrictions of the naïve assumption. It should be noted that the we did not discuss the *tree* nature of TAN, because it is related to practical optimisations for learning large graphs with which we are not concerned.



(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.

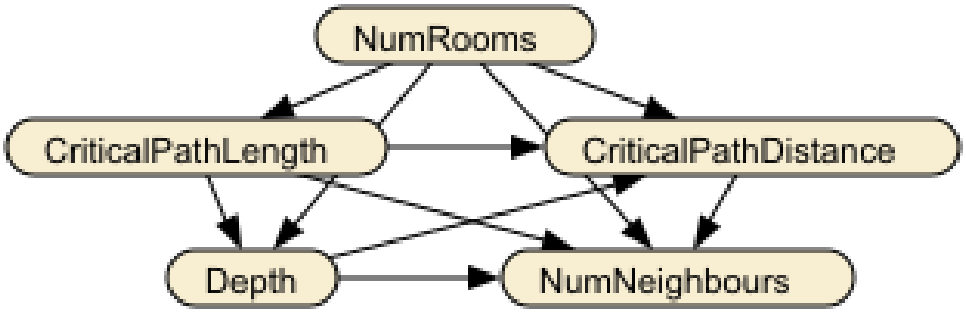


Figure 3.3: Fully connected graph with heuristic causal flow

3.2.3 Parameter Learning

Similarly to structure learning from data, 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 given our prior $P(\mathcal{D} \mid \mathcal{N})P(\mathcal{N})$ [21, pp. 46–48]. This is in fact the usual Bayesian updating in which Netica will consider the prior to have a Dirichlet distribution and the conditional probabilities (likelihood) are independent of each other. To learn the likelihood we review three algorithms Netica has in its toolkit.

Count [21, pp. 48–50]

The counting learning states that the probabilities in a single node are defined by how often a particular value has been seen in the data. This notion is defined as the *experience* of that value. The network starts in a state of complete ignorance with uniform probabilities across all RVs. As we iteratively load in the data, the CPT are updated by increasing the probability for a particular value every time it is seen. Specifically, the experience is updated by the previous experience and the degree factor of that data point (i.e how often is it repeated, usually 1) $e' = e + d$. The CPT values are computed as a ratio of the previous and newly observed experience $P'(X = x) = \frac{P(X=x) \times e}{e'}$. The counting algorithm is recommended due to its speed and simplicity, but it has a substantial drawback in the fact that it needs the data to be fully defined. Theoretically we should be able to use the counting algorithm as we have no latent (undefined) RVs nor missing data.

Expectation Maximisation EM

Gradient Descent GD

3.3 Implementation

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 [23] 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 [24].

Dungeon Sampling We can define characteristics of the desired dungeon by leveraging the inference nature of BNs. Generating a dungeon with five rooms is a matter of fixing (*observing*) the `NumRooms` parameter. The user can choose virtually any permutation of parameters to be fixed or inferred. For consistency of this experiment, we are observing only the size of the dungeon via `NumRooms`.

Room Sampling

Converting from samples to Space

4 Results

4.1 Internal Validation

As we are modelling uncertainty, we can pose the question of "How well do we reduce uncertainty".

	NumRooms		CPLength		CPDistance		Depth		Neighbours		
	$loss^2$	E	$loss^2$	E	$loss^2$	E	$loss^2$	E	$loss^2$	E	
Uniform (base)	0.9615	99.83%	0.9	97.52%	0.8333	57.86%	0.9286	91.97%	0.8889	70.51%	
Extreme Sparse	0.7819	67.09%	0.5294	41.79%	0.6173	51.54%	0.7657	61.54%	0.7174	59.15%	
TAN	0.6871	54.87%	0.5294	41.79%	0.5812	41.62%	0.6581	50.51%	0.7054	49.74%	count
Fully Connected	0.6012	44.53%	0.3391	22.14%	0.4205	17.44%	0.5814	35.98%	0.7029	25.64%	
Extreme Sparse	0.7522	66.67%	0.4974	41.2%	0.6237	54.19%	0.7354	63.25%	0.693	59.15%	
TAN	0.6081	50.77%	0.4974	41.2%	0.4968	41.62%	0.562	45.73%	0.6024	49.74%	EM
Fully Connected	**0.4297**	**39.23%**	**0.2097**	**18.72%**	**0.1909**	**16.92%**	**0.3683**	**33.76%**	**0.284**	**25.64%**	
Extreme Sparse	1.07	75.98%	0.7947	50.34%	0.6644	57.26%	0.7951	66.75%	0.7368	62.65%	
TAN	1.036	64.19%	0.8043	52.31%	0.5632	45.38%	0.7532	54.19%	0.735	55.73%	GD
Fully Connected	1.046	69.57%	0.701	44.7%	0.4416	38.8%	0.6256	53.25%	0.5295	45.38%	

Table 4.1: Structure and parameter learning comparison.

4.2 External Validation

User study design Get the topologies for three sets of dungeons. Real OPDC dungeons, our inferred dungeons and a control group containing the random donjon dungeons. All formatted in the same style. Preference study by providing randomly selected pairs from these three sets. 10, 13 and 8 room dungeons were used as 10 and 8 are the most common sizes and 13 is the mean dungeon size.

5 Discussion

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 approach 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

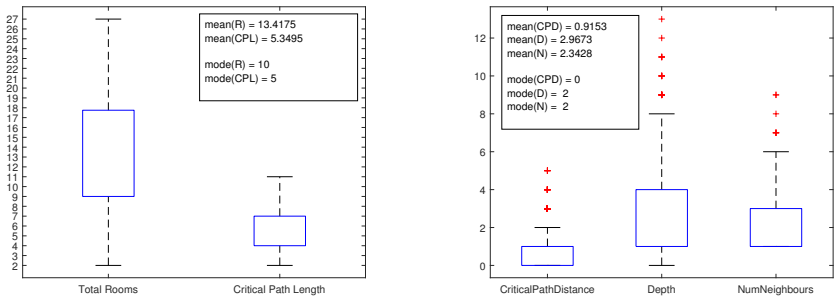
than just discriminating between different dungeons topologies would give us more insight in the success of the recreation of a human-made dungeon.

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. 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 $(R, L, \{S_r, D_r, N_r\})$

$R \leftarrow$ BNetModel.`Observe(`$observed\_R$`)`
$L \leftarrow$ BNetModel.`Sample()`
for $r \in R$ **do**
  $temp\_R = R$
  $temp\_L = L$
  $(S_r, D_r, N_r) \leftarrow$ BNetModel.`Sample()`
  BNetModel.`ClearObservations()`
  $R \leftarrow$ BNetModel.`Observe(`$temp\_R$`)`
  $L \leftarrow$ BNetModel.`Observe(`$temp\_L$`)`
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 ∈ R do
        r_v ← all possible values for r
    end
    MapOne()

Function MapOne():
    if ∀r | IsSingletonSet(r_v) then
        return
    end
    r' ← ChooseRand(r | IsSingletonSet(r_v))
    for v ∈ r'_v do
        try
            Reduce(r'_v, {v})
            MapOne()
        catch Failure
            Undo()
    end

    Function Reduce(r_v, set):
        if set = ∅ then
            throw Failure
        end
        if r_v ≠ set then
            r_v ← set
            for c ∈ C do
                Propagate(c, r_v)
            end
        end

    Function Propagate(c, r'_v):
        for r_v ∈ c_r do
            if r_v ≠ r'_v then
                Reduce(r_v, c, r_v)
            end
        end
```

B appendix

Use this section for questionnaires and external validation support

Bibliography

[1] 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.

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

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

[4] 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.

[5] *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.

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

[7] H. Thrall, 'Procedural generation of a dungeon for dungeons and dragons,' Master's thesis, University of York, 2017.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] C. Deery, 'Generating dungeons & dragons dungeons that make visible sense,' Master's thesis, University of York, 2019.

[16] 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.

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

[18] 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.

[19] 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.

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

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

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

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

[24] O. Nepozitek and J. Gemrot, ‘Fast configurable tile-based dungeon level generator,’ 2018.