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

29th April 2020

Supervisor: Dr. Rob Alexander

To my players, may your next DM be less pedantic

Acknowledgements

This project would not have been possible without the amazing TRPG communities of the One Page Dungeon Contest, r/DnD, Dungeons and Dragons UK, Dungeons & Dragons Bulgaria, Dungeons and Dragons Society York and the nerds at r/SampleSize.

I would like to direct explicit gratitude to Dr. Rob Alexander for his expert guidance and for supporting every crazy idea in this project.

Contents

Executive Summary **vii**

1 Introduction **1**
1.1 Role-playing games 1
1.2 State of the Art Generative methods 2
1.2.1 Non-digital Generators 2
1.2.2 Digital Generators 3

2 Motivation and Aims **6**
2.1 Mission and Space 6
2.2 Aim 6

3 Bayesian Inference Networks for PDG **8**
3.1 What are Bayesian networks? 8
3.2 Why use Bayesian networks? 9

4 Methods and Implementation **10**
4.1 Data acquisition 10
4.1.1 Data selection 10
4.1.2 Extracting Features 11
4.2 Model selection 13
4.2.1 Bayesian Network topology 14
4.2.2 Parameter Learning 16
4.3 Implementation 17
4.3.1 Dungeon configuration 17
4.3.2 Converting from configuration to Space 18

5 Results **20**
5.1 Internal Validation 20
5.2 External Validation 21

6 Discussion **23**
6.1 Critique 24

7 Conclusion **25**
7.0.1 Future Work 25

A Appendix **27**

B Appendix **30**

List of Figures

4.1 Expert defined Network topologies 14
4.2 Learned Network topologies 15
4.3 Fully connected graph with heuristic causal flow 16
4.4 Pipeline for the data-driven PDG tool 17

5.1 CP performance 21

A.1 Parameter statistics 27

B.1 Loot table for gemstones taken from Dungeon Master Guide 5e 30
B.2 Steps for transcribing Sepulchre of the Abyss 2015 (dark red is CP) 31
B.3 Generated dungeons 32
B.4 category encoding (OPDC, ours, random) = (1,2,3) 32
B.5 Hypothesised vs Actual distribution 33
B.6 Choice distributions 34

List of Tables

5.1 Structure and parameter learning comparison. 21

Executive Summary

Tabletop Role-playing Games TRPGs are multiplayer, story-driven games which have been around for decades. They require a group of players taking up the role of fictional personas and being guided through a fictional world by another player, often referred to as the *Dungeon Master DM*. Game sessions are played by verbal interaction between the participants and the DM, whereby the DM has the non-trivial task of compiling a compelling narrative. Much like in other literary contexts, creating an interesting story is a difficult and highly creatively demanding job. That is why often the narrative is focused on one location within this fictional world that can be polished with virtually as much detail as wanted. These locations are referred to as *dungeons*. But even with focusing all the creativity on one location, it is still a very time consuming process that could benefit from automation.

Procedural Dungeon Generation PDG has been an object of interest of researchers and hobbyists alike due to the appetising prospect of alleviating time-consuming work from the DM in designing these key spaces. Current state-of-the-art digital and non-digital PDG methods are oriented in defining a unique generative philosophy which produces the skeleton of a Dungeon, but provides little support for assessing the qualities of the product. The lack of focus on the "goodness" of a dungeon is a drawback that is a characteristic shared by all dungeon generators apart of the human designer.

We suggest a data-driven approach to learning dungeon characteristics from human-made dungeons and in that way generating "good" dungeons by imitating the only dungeon generator that is proven to create compelling narratives. This formulation provides us with a natural evaluation criterion by asking *"are dungeons we produce indistinguishable from human-made ones?"*. The generative method proposed is split in two sections: firstly human-made dungeons properties are captured by statistical modelling and secondly we construct a generative algorithm that produces a usable dungeon layout given our model.

For the first step, formalising dungeon creation as a statistical modelling problem is done by training a Bayesian Network BN on winning dungeons from the One-Page Dungeon competition. We empirically compare different network architectures and learning algorithms in search of the model that best captures properties of the human-made dungeons. A statistical evaluation of the models is conducted by using them as classifiers to predict dungeon features from the data. We infer that the model that can most accurately predict the data has best learned its properties and therefore use it for the generation phase.

The output of the modelling phase is a trained BN which given a desired size of the dungeon can produce a set of informed dungeon features. The produced set can be viewed as a configuration or descriptor of a dungeon that corresponds to the winner dungeons from the OPDC. A natural way to view the set of descriptors is as constraints on what the produced dungeon has to have. Therefore the generation algorithm chosen is a Constraint Propagator, which attempts satisfy the produced configurations and in the process give us a performance metric of how long it takes to satisfy these constraints. Ultimately dungeons generated with the best performant model are presented to 95 TRPG players in a blind preference study to validate if our PDG method produces layouts that are indistinguishable from human-made dungeons.

This study discuses the various difficulties associated with formalising a creative task such as dungeon generation and how required assumptions can lead to lack of data and therefore to poorly performing models. We outline an end-to-end pipeline from data acquisition, through statistical modelling to physical dungeon generation and show that our model trained only on layout features fails to produce satisfactory results both in terms of statistical accuracy, speed and ultimately user preference. The implications of this performance strongly suggests that the highly variant domain of dungeon-making requires a more detailed analysis on feature selection and data acquisition in general.

All dungeon data used is under a Creative Commons licence. All user data collected is anonymous. The study was conducted as an online survey and as such no hard copies of the data are produced and the digital entries are securely stored under the University of York Qualtrics suite.

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 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*) 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, but 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 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 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 TRPG 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 TRPG can evolve rapidly outside the planned narrative structure prepared by the DM and a simple way to quickly generate new adaptive story elements via content tables 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 treasures [7] entirely based on randomised content tables. An issue arises with using loot tables when a randomly selected element from one table is contradicted by an item selected from another. 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 TRPGs 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 input parameters and produces 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 structures 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 with corridors. An example that can accommodate both can be seen in 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 [13]. PDG has been described as a CSP on occasions where the variables and constraints define the layout, contents or both layout and content of a dungeon [14], [15]. Constraint Propagation (CP) is a particular method proven useful for solving CSP in the context of PDG. The algorithm

randomly 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 dungeon 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. Even thought CPs have solved some problems of the purely 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.

Context aware Binary Space Partitioning

Other methods proposed by Thrall [11] and Brown [12] use a Binary Partitioning Tree [16] 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 be contextually aware of the game's narrative. Examples are tailoring loot for specific players or noting that noise in a certain room will provoke monsters from nearby ones. Although this is a step towards narrative automation, these elements are very basic. This leads to the necessity of the DM to refine the result of the generation and occasionally ignore it completely. An example of a case where the narrative guide makes no sense would be a room labelled as provoking, yet no nearby rooms host monsters. Thrall and Brown's PDG can be viewed as a digital version of the classic TRPG 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 TRPGs takes on average 3-4 hours [17] making it very difficult to get a representative sample size in a short period of time. This implores considering the trade-off between environmental validity and practicality when designing a user study for evaluating PDG methods for tabletop RPGs.

Formal Languages

The natural encoding of contextual meaning and structure has made formal languages for PDG and object of interest to several researchers [18]–[20]. Generative grammars enable designers to specify production rules with arbitrary complexity that dictate the generation of both layout and contents of a dungeon. A critical consideration when using grammars is the issue of what vocabulary of dungeons is a particular grammar going to produce. Cadogan [19] created a baseline grammar based on some logical heuristics, but it does not differ in output from a parameterised random PDG (such as the donjon CA). Deery used a more analytical approach to grammar rule selection by manually analysing submissions to the One Page Dungeon Contest *OPDC* [21] and noting any common patterns. His aim was to transcribe them as grammar production rules in an attempt to make the generator produce dungeons with elements that exist in human-made ones. Although a very interesting proposition, it required several important restrictions to be practical. Namely he focused only on the narrative content i.e. what the players have to do to finish the dungeon and disregarded the level topology. The generator then maps the narrative content to a usable layout in a 1:1 ratio, resulting in a single task per room. It is trivial to see that the originals in OPDC do not impose such a restriction. A room can have multiple narrative elements (e.g. the key to unlocking the door is on the bandit's waist, Key + Encounter). Furthermore, Deery manually looks at ten competition winners and heuristically extract the grammar rules, which he highlights that they do not capture all possible patterns.

Data-driven approaches

Arguably one of the most pleasant experiences in TRPGs spring from exploring intricately structured dungeons and their mysteries. Their compelling features are a result of the human designer understanding the needs of his players. Researchers have acknowledged that the structure of human-made dungeons should be studied to better answer the question of "What makes a compelling dungeon". Summerville has shown that a machine learning ML model trained on data from man-made dungeons from the Legend of Zelda games can accurately capture and recreate layout and narrative features [22]. ML for PDG not only promotes informed content generation, but it can be used as an exploratory mixed-initiative/analysis tool for understanding dungeon structures [23]. The previously discussed approaches have seen applications for digital and tabletop RPGs, but data-driven PDG is largely unexplored. It is especially rare for TRPGs so much that we do not know of publications on applications of ML to TRPG PDG. Presumably this is due to TRPGs being virtually unbounded by functional constraints which makes finding a consistent dataset very difficult.

2 Motivation and Aims

2.1 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 in his definitions of *Mission* and *Space* [18]. 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 by the overlap of Mission and Space. As noted by Dormans, a *Mission* mapped on different layouts will result in drastically different exploration patterns. By understanding the necessities of the two separate entities we can model player experience better. Due to this separation we have seen advancements in generating adaptable to the player game environments [24] and even reverse engineering the creation of *Missions* and *Spaces* by learning structure from data as discussed in section 1.2.2.

2.2 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 recreating *good* man-made dungeons, we could potentially achieve higher levels of narrative automation. As highlighted by Deery's work discussed in section

1.2.2, extracting meaningful information from human-made dungeons and transcribing it in a way that a PDG tool can use it is a very powerful concept. The issues raised by his manual approach can be addressed by an automatic learning of the patterns. Programmatically learning a level from data has been an object of interest for computer based RPGs as seen in section 1.2.2, but has not been applied to TRPGs, 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 layout that is undistinguishable from human-made dungeon topologies. Choosing Space exclusively is due to it being more universally *consistent* than Mission. Consistency in this context means that two dungeons must be logically comparable. Space is comparable by looking at individual rooms, but without rigorous and lengthy efforts to define the meaning of comparing a mystery to a drama, it is not immediately clear how comparable Missions are in TRPG games. It should be noted that consistency as such is not a mandatory requisite for ML algorithms, but is necessary for managing the project scope in this already time constrained environment.

To achieve our goal, we aim to create an end-to-end generative pipeline covering stages from data acquisition, through modelling and generation to evaluation. Inspired by work on digital RPGs, we discuss the application of inference (Bayesian) networks to learning Space features from data. We discuss the availability of data, the meaning of *good* dungeons and how meaningful features can be extracted from them. 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 on preference to externally validate how well the generator has approximated human-made dungeon topologies.

3 Bayesian Inference Networks for PDG

3.1 What are Bayesian networks?

Bayesian Networks *BNs* [25] (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. In essence they are a graphical way to represent dependencies within a probabilistic context, allowing us to input knowledge for one RV and infer information for the others. More formally, 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 4.3.

To contextualise the inference process for TRPGs, we can look into Summerville's work on the ARPG Legend of Zelda dungeons [22]. After annotating the dungeon maps with elements of both Mission and Space (total number of rooms, treasures, monsters, the length of the optimal path from entrance to goal, i.e. the *critical path CP* and others), he extracts features for the BN by parsing them as RVs. Conditioning on total number of rooms by stating that there is 100% chance of having n rooms $P(NumRooms = n) = 1$ 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 is extracted. This *observe-then-infer* process can be done with any permutation of our RVs.

3.2 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 digital RPGs [22], [26]. 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 [20].

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. 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. An argument can be made 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. These limitations and our ways to work around them are discussed in detail in section 4.1.2.

4 Methods and Implementation

In this section we explore various issues with formalising the conversion of human-made dungeons to probability distributions for use with our Bayesian networks. The key assumption we propagate throughout this section is about the parallels between digital and tabletop RPGs we made in section 1.2.2. This assumption allows us to base feature selection and model architecture on previous work done by the ML community. We discuss Bayesian network architectures in three different levels of complexity and two different parameter learning algorithms.

4.1 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)
- A *point estimate* which is the probability of a particular value in a PD is denoted as $P(X = x)$

4.1.1 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 logically comparable.

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 *good* 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.)

4.1.2 Extracting Features

We base our feature selection on the work done by Summerville discussed in section 1.2.2. He showed that his Extreme Sparse model (seen in fig 4.1a) not only has the greatest predictive capabilities, but is also the least complex model, having only 10 features. As a corollary from our assertion that digital RPGs do not differ from tabletop RPGs in terms of dungeon design philosophy, we can assume that the predictive capabilities of the Extreme Sparse model will be similar if applied in TRPG context. As noted in section 2.2, we focus on Space related variables, disregarding Mission due to its inconsistency in TRPG dungeons. This crucial reduction can be detrimental to the predictive capabilities of the model, but it is a necessary step to introduce the prerequisite consistency for the ML task. Incorporating Mission features in our model would mean further reduction of the already small dataset.

The resultant selection uses the top five explanatory Space RVs split in two categories:

1. Dungeon (Global): total number of rooms R and how many rooms form the shortest path between the entrance and the goal (the *critical path CP*)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

The chance of a particular RV taking some value is dictated by an unknown PD. As our RVs are related, it is logical to describe our dungeon as the interaction of the different RV PDs. That means that we can describe this unknown PD for a full dungeon by composing the global and local parameters in a joint PD. More formally, 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

$$P(R, L, S, D, N) = P(R, L) \prod_{r}^{|R|} P(S_r, D_r, N_r) \tag{4.1}$$

where $|R|$ is the total number of rooms defined by the instantiation of R. This reads that the probability of having a particular dungeon is dictated by the interaction of the of the size and CP length with each individual room.

Each of the terms on the LHS is unknown and we need to *learn* them from the OPDC data. To prepare the data for the learning process, we extract these five features by transforming the artistic and graphically rich textual representations of the OPDC dungeons to a computer readable representation. Unfortunately automating feature extraction is very difficult due to the variant stylistic formatting and the prose like description of the player goals within the dungeon. 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. The process is as follows: note the Entrance and the Goal rooms according to the description given in the OPDC; connect them using the least amount of rooms (this will be the CP); connect all other rooms; calculate R, L, D, S, N by counting. Example abstract graph annotating can be seen in Appendix B 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 3. 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 [27] and exceptional results on small datasets [28]. 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 BNs. 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 [29, p45]. However this representation of the unknown will impose a restriction on what CPT learning algorithms we can use. We discuss them in section 4.2.2.

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 data statistics can be found in appendix A.

4.2 Model selection

With the data in suitable format, we start the learning process by defining two key attributes of the BNs:

1. The causal network's topology
2. The network parameters

We make the common ML assertion that the model that has best learned to predict the data is the model that has best captured its properties. Therefore we empirically compare different models in search of the one with highest statistical predictive capabilities. We evaluate three structures with two CPT learning algorithms against a uniform distribution (control) for a total of 7 models. The structures are the previously mentioned Extreme Sparse model (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

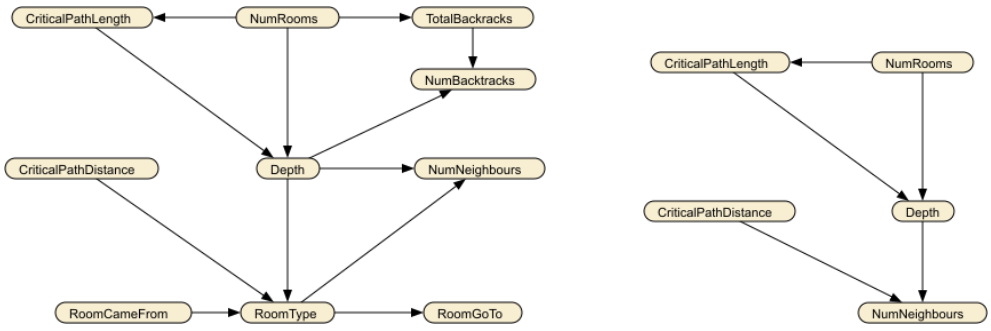
and in assumptions. The comparison is an exploratory experiment aiming to see which pair of complexity and assumptions fits the OPDC data best. In regards to parameter learning, the CPT learning algorithms will be Expectation Maximisation *EM* and Gradient Descent *GD* due to their ability of handling missing data. Implementation of all BN related algorithms is done in the Netica Bayesian Network development software [30].

4.2.1 Bayesian Network topology

Summerville's Extreme Sparse Model *SESM*

In section 4.1.2 we discussed the usefulness of the high accuracy SESM for small datasets (Summerville's original model was trained on 38 dungeons with 1031 rooms). The first BN topology we therefore consider is a flattened to Space features only SESM. The flattening is done by reconstructing the SESM in Netica and applying the `AbsorbNodes_bn` [29, 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 4.1. The resulting model describes the following factorisation over our RVs:

$$P(R,L,S,D,N) = P(R)P(S)P(L \mid R)P(D \mid R,L)P(N \mid D,S) \quad (4.2)$$



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

Figure 4.1: Expert defined Network topologies

Tree Augmented Naïve Bayes

In the above approach we applied specialist knowledge from Summerville's experiments to create the BN structure. For that network to be valid, we made two assumptions: equivalence between TRPG 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 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* [31].

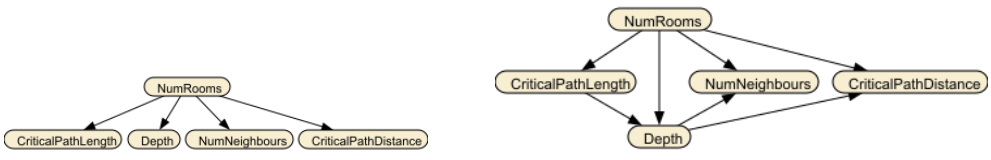
Informally the learning procedure can be defined as finding the most likely BN structure that *agrees* with our data. That means we are searching for the network structure that is most likely to have been produced given the data. TAN finds the most likely network structure in two steps:

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 choose the class to be R due to its contextual importance. The naïvity assumption implies that

$$P(R, \boldsymbol{X}) = P(R) \prod_{x}^{\boldsymbol{X}} P(x \mid R) \tag{4.3}$$

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 [32], 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 4.2.



(a) Naïve Bayes (b) TAN

Figure 4.2: Learned Network topologies

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 4.1, but we 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 4.3.

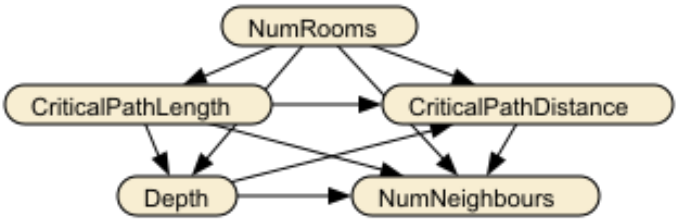


Figure 4.3: Fully connected graph with heuristic causal flow

4.2.2 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 $P(\mathcal{N})$ [29, pp. 46–48]. This is in fact the common maximum likelihood estimation 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 [33] 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, whereby it maximises $P(\mathcal{N} \mid \mathcal{D})$ by computing the steepest gradient, taking in account the previous one [29, p47]. Both EM and GD risk getting stuck in local optima.

4.3 Implementation

Obtaining a dungeon is done in three steps, seen in fig 4.4. 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 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.

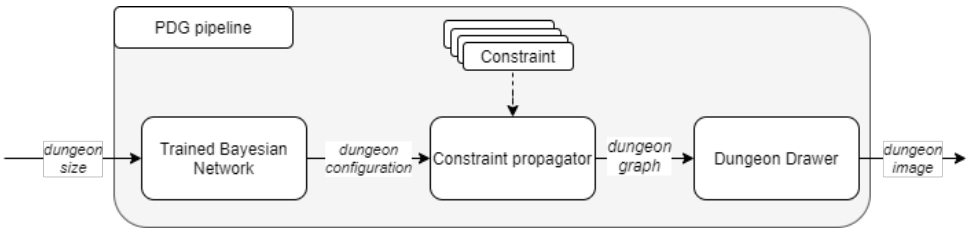


Figure 4.4: Pipeline for the data-driven PDG tool

4.3.1 Dungeon configuration

As we discussed in chapter 3, generating a dungeon with five rooms is a matter of fixing (*observing*) the R 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 this experiment, we start by observing only the size of the dungeon via R. Obtaining a dungeon configuration from our model is done in two steps:

1. Infer the global RVs and validate them (dungeon sampling)
2. Infer the local RVs and validate them (room sampling)

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. The validation step is required as although no data entry would ever show an invalid dungeon, the stochastic factor of the sampling could occasionally produce such results. 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 can be seen in Appendix A algorithm 1.

4.3.2 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 whose distance from it is zero ($S = 0$). The current constraints are as follows:

1. The dungeon is exactly of size R
2. The i_{th} room must be exactly D_i rooms away from the entrance
3. The i_{th} room must have exactly N_i connections
4. The i_{th} room must be exactly S_i rooms away from the critical path
5. Newly added rooms must not alter the predefined CP
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 [34] 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 use Nepožitek's PDG tool [35] 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 [36].

5 Results

5.1 Internal Validation

As we are modelling probability distributions, we can pose the question of "How well do we reduce uncertainty". An answer is obtained by using the BN as a classifier in which the predictor variable is the user input (i.e. the size of the dungeon R) and the target variables are the rest of the features as described in the feature extraction section 4.1.2. Due to the small dataset, we construct a test where we measure how well the different models estimate $P(L, S, D, N \mid R)$ by using the same data for training and testing. Reusing train data for testing will ultimately result in our model's performance being generally more optimistic due to it predicting the exact data it has already observed. We note that this can lead to poor generalisation, but it is a necessary compromise compared to reducing our already small dataset even further in order to get distinct training and testing representatives. We use two metrics: the classification error rate E and quadratic loss (Brier score). The error rate is used due to its natural meaning and the quadratic loss is used due to it being a well known and robust metric for evaluating statistical performance [37]. It should be noted that Netica uses the original formulation of the quadratic loss which is bounded between zero and two. The comparison between our different models can be seen in table 5.1. We observe that the TAN and fully connected networks both have the highest predictive capabilities. In favour of simplicity, we choose TAN with EM to continue forward to the generative process.

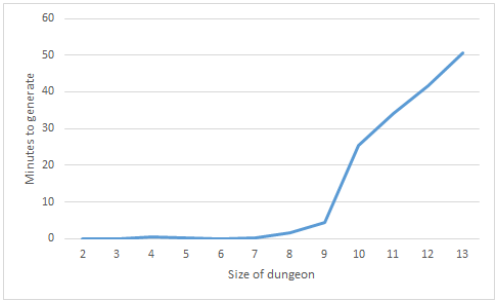
To assess the computational expense of the generation, 15 layouts each were generated for each size and the average time is taken. Generation time increases exponentially with dungeon size with a factor that makes dungeons above 13 rooms unfeasible. Figure 5.1b shows the increase in time and table 5.1a shows the time and number of unsatisfiable dungeons sampled (retries) before finding a single dungeon that satisfies our physical constraint requirements described in section 4.3.2 for the top three most common dungeon sizes (8, 10 and 13).

	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 5.1: Structure and parameter learning comparison.

Size	time (min)	retries
8	3	105
10	25	451
13	50	314

(a) Common sizes generation



(b) Generation time plot

Figure 5.1: CP performance

5.2 External Validation

The ultimate validation for a successful PDG tool is undoubtedly a user study assessing how players use the generator for their games. Unfortunately as discussed in section 1.2.2, the nature of TRPGs makes acquiring a representative sample size within the timeframe of the project very difficult.

To work around this issue while still obtaining meaningful information from users, we draw inspiration from previous work on aesthetics of composition. Two experiments looking at the composition of the contemporary artists Piet Modrian and Damien Hirst were conducted to see if people can discriminate between originals and facsimiles of their work. When people are presented with original and computer generated artwork by those artists and are simply asked for their preference between the two, the human-made originals are consistently preferred [38], [39]. The artistic nature of creating a dungeon allows us to draw a parallel between the work done on these two artists and the user evaluation of our generator. We utilise this phenomenon to answer the question of are our dungeons undistinguishable from human-made dungeons and in that way answering the question of have we made *good* dungeons by virtue of proxy.

We conduct a preference study in which 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 containing the possible combinations *(OPDC,ours), (OPDC,random), (ours,random)* are randomly permuted for the common 10, 13 and 8 room dungeons as highlighted in section 5 (example Appendix B fig B.3). Pairs in ratio of [1:1:2] respectively for *(OPDC,ours), (OPDC,random), (ours,random)* are presented in random order and random orientation (i.e. each dungeon type has equal chance on appearing either on the left or right) and the users are simply asked which one they prefer. The decision of having more representatives of the pair between our PDG and the control group was due to our ability of generating indefinite amounts unlike the OPDC dungeons. Each participant is presented with 24 pairs (8 per dungeon size) followed by 7 demographics questions assessing their experience with TRPGs.

After the data is collected, the preference from each question is collated in three columns for the different pairs and a one-sample binomial test is executed for each pair. The [1:1:2] ratio does not invalidate our nonparametric binomial significance. We hypothesise an even preference for all three pairs with a Bonferonni corrected significance threshold of 0.0167. The test reports that the preference for all three pairs are very significantly different $p < 0.001$. Inspecting the choice distributions shows that inline with the literature, the human-made dungeons are consistently preferred and random dungeons are preferred over our PDG.

The participants in this experiment were selected to be adults with at least some experience in TRPGs. 95 participants sourced from the following online hobbyist forums: r/DnD, r/SampleSize, Dungeons and Dragons UK, Dungeons and Dragons Bulgaria, Dungeons and Dragons Society York (latter three being facebook groups). The experiment was done as an online survey and results can be seen in Appendix B.

6 Discussion

All models performing better than the uniform (base) model is a good statistical indication that there is a non-random pattern in human-made dungeons. However the above 50% error rate reveals that our assumptions (section 4.1.2) for making the problem more concrete have removed variables that can give us insight into this pattern. The success of Summerville's Mission + Space model (sections 1.2.2, 4.1.2) and the low predictive capabilities of our Space only model are indicators that Mission parameters or some other unmodeled hidden set of parameters are critical when trying to approximate human-made dungeons. In practice this disparity indicates that a human-designer does not decouple the narrative from the location.

Additionally these results are optimistic due to the fact that we used the same dataset for both training and testing the model. Although that was necessary due to the lack of data, the bias incurred is a sure indicator of poor generalisation. Furthermore the constraint satisfier struggles to generate the sampled dungeons even with our optimisations. The time it takes to produce a layout grows exponentially with the number of rooms and larger dungeons could take hours to produce. We would like to argue that a permissible generation time would be one that is less than the usual session preparation time for a DM. Preparation times vary between DMs but common suggestions usually mention up to an hour [40]. For the most common dungeon size (10 rooms) the inference procedure takes on average 25 minutes which is within reason.

Another observation is that less common dungeon sizes often lack variety and produce similar dungeons, meaning that the model has learned only a small set of viable parameters due to the dungeon being less common.

Although statistically our generator is better at imitating human-made dungeons than a purely random one, the fact that users prefer the random dungeons over our PDG is an indicator that the underlying pattern of human-made dungeons has not been recreated successfully.

6.1 Critique

Data

Having small number of data instances is detrimental to a data-driven method. We restricted the dataset to using winning entries which are considered as a rule-of-thumb *good* dungeons. This does not imply that non-winning entries are examples of *bad* dungeons. The *goodness* of a dungeon was not discussed on a spectrum but only on a binary scale of is it a winner in the competition, ultimately leading to a very small sample, albeit high quality.

The extraction of data has also been a manual process. Although due diligence is paid, noise introduced from human error is inevitable.

Methodology

In chapter 3 we highlighted drawback is that our PDG method cannot generalise to unseen cases. We can interpolate for missing data, but we cannot learn dungeons outside of our size range. This drawback is not unique to Bayesian networks, but to ML methods in general. This limitation is permissible in the context of our PDG method trying to approximate one-page dungeon adventures, designed for a single game session.

In terms of the choice for a generative algorithm, constraint propagators that are not built with performance in mind will be impractical for a generative tool. For this experiment, the additional metric for satisfiability is more valuable than how fast we produce a dungeon.

Validation

Conducting a study with a high environmental validity is difficult due to multiple confounding factors that result from the nature of tabletop RPGs. Not only does a one-session adventure usually take on average 3-4 hours, but they are extremely variant between groups of players and DMs. Our preference study is a reasonable compromise given the trade-off between the environmental validity and practicality.

7 Conclusion

We conceptualised the benefit of applying data-driven methods to TRPG PDG tasks and systematically identified practical issues and related assumptions. Although our best performant model has unsatisfactory performance to be used as a PDG tool, we outlined an end-to-end pipeline that allows for rigorous building and evaluation of statistical models for PDGs.

7.0.1 Future Work

Data Acquisition

The OPDC is conducted every year, so simply re-running the model with new data is one simple avenue for further work. More importantly a way of automating feature extraction such as building an optical character recognition OCR tool would allow for a more robust way of collecting data. Additionally, other datasets can be considered. We've use OPDC due to its CC license, but a plethora of paid and copyrighted modules exist.

Feature extraction

Future work should focus on solving the limitations of the reduced feature set. Incorporating Mission parameters or an analytical approach of assessing TRPG specific features could be utilised.

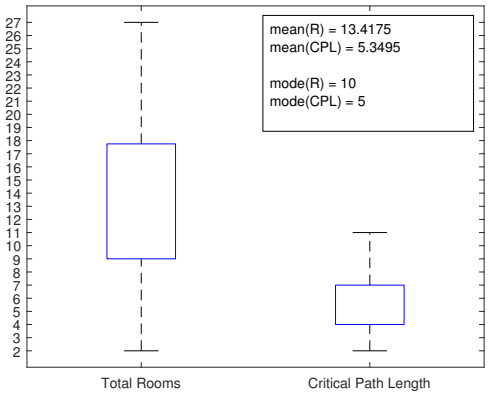
Generator optimisations

A multitude of optimisations exist for CSPs which were not considered in this project, due to time constraints. Formulating the internal representation of the problem to be used by generic linear solvers would reduce the generation time.

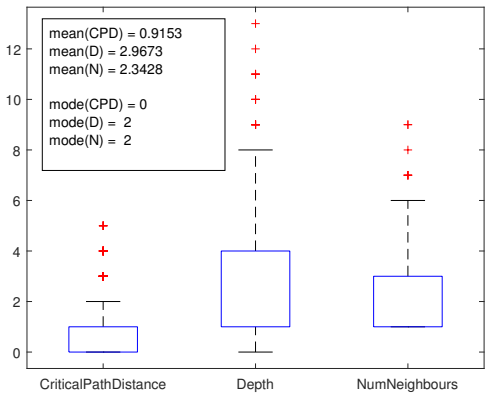
Validation

Provided a more extensive time period, an experiment with greater environmental validity than our preference study could be conducted. Furthermore this experiment was focused on answering the question if our generated dungeons are indistinguishable from the human-made dungeons. 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.

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

$isDungeonValid \leftarrow false$
while $\neg isDungeonValid$ **do**
  $R \leftarrow$ BNetModel.`Observe(`$observed\_R$`)`
  $L \leftarrow$ BNetModel.`Sample()`
  $isDungeonValid \leftarrow ValidateDungeon(R, L)$
  if $isDungeonValid$ **then**
    for $r \in R$ **do**
      $temp\_R \leftarrow R$
      $temp\_L \leftarrow L$
      $isRoomValid \leftarrow false$
      while $\neg isRoomValid$ **do**
        $(S_r, D_r, N_r) \leftarrow$ BNetModel.`Sample()`
        BNetModel.`ClearObservations()`
        $R \leftarrow$ BNetModel.`Observe(`$temp\_R$`)`
        $L \leftarrow$ BNetModel.`Observe(`$temp\_L$`)`
        $isRoomValid \leftarrow ValidateRoom(S_r, D_r, N_r)$
      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

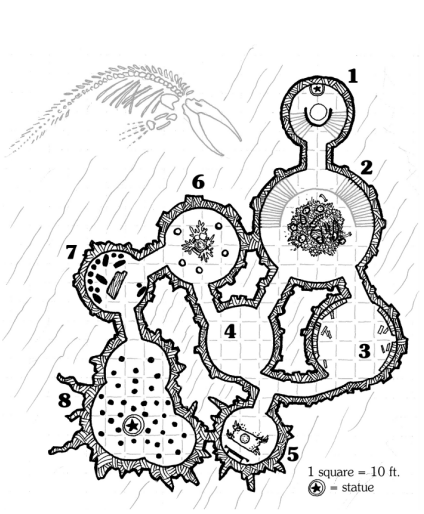
```
Map()

Function Map():
    for r ∈ 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

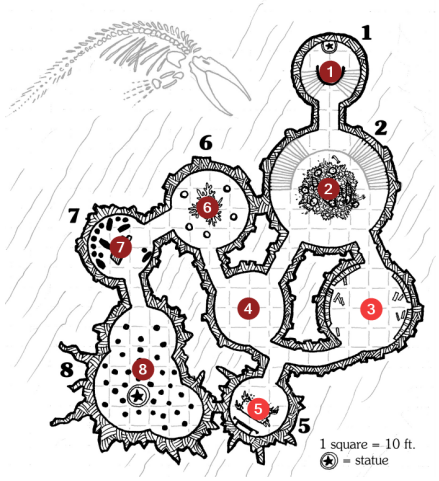
10 GP GEMSTONES

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)

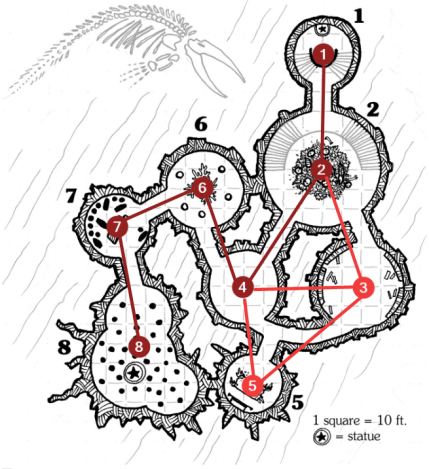
Figure B.1: Loot table for gemstones taken from Dungeon Master Guide 5e



(a) select a dungeon map

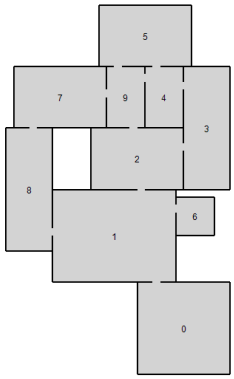


(b) annotate (or reannotate) rooms for convenience

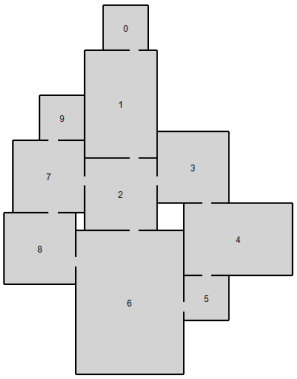


(c) connect rooms

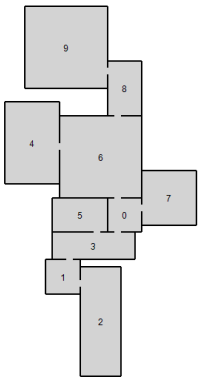
Figure B.2: Steps for transcribing Sepulchre of the Abyss 2015 (dark red is CP)



(a) 10 room Bayesian PDG



(b) The Vermin Hollow OPDC 2011



(c) 10 room Random PDG

Figure B.3: Generated dungeons

Hypothesis Test Summary

	Null Hypothesis	Test	Sig.	Decision
1	The categories defined by Original_BPDG = 1 and 2 occur with probabilities .500 and .500.	One-Sample Binomial Test	.000	Reject the null hypothesis.
2	The categories defined by BPDG_RAND = 3 and 2 occur with probabilities .500 and .500.	One-Sample Binomial Test	.000	Reject the null hypothesis.
3	The categories defined by Original_RAND = 3 and 1 occur with probabilities .500 and .500.	One-Sample Binomial Test	.000	Reject the null hypothesis.

Asymptotic significances are displayed. The significance level is .017.

Figure B.4: category encoding (OPDC, ours, random) = (1,2,3)

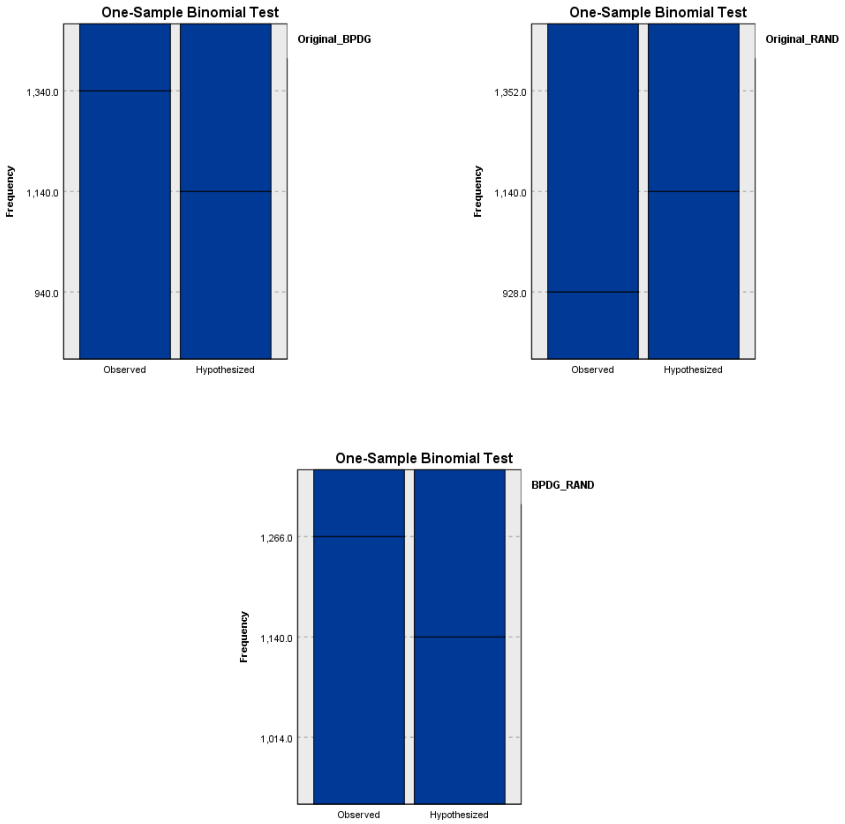


Figure B.5: Hypothesised vs Actual distribution

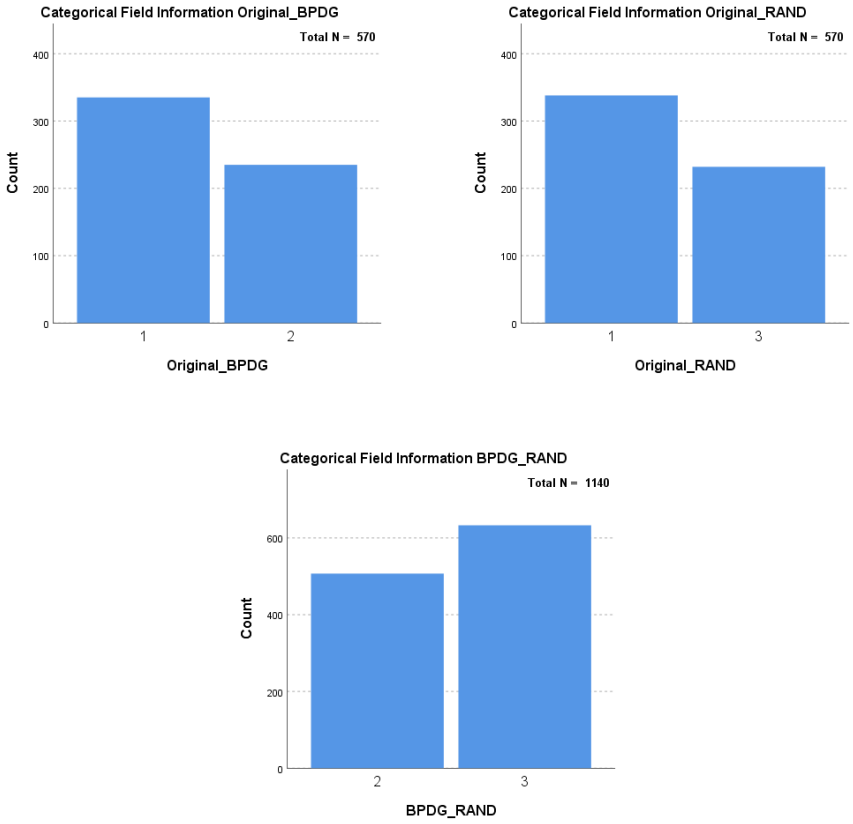


Figure B.6: Choice distributions

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] V. Kumar, 'Algorithms for constraint-satisfaction problems: A survey,' *AI magazine*, vol. 13, no. 1, pp. 32–32, 1992.

[14] 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.

[15] 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.

[16] 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.

[17] *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.

[18] 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.

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

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

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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

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

[27] 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.

[28] 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.

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

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

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

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

[33] 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.

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

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

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

[37] 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.

[38] 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.

[39] 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.

[40] *Rpg session prep – how much time is needed?* https://www.dungeonsolvers.com/2018/04/30/how-much-time-you-should-spend-prepping-an-rpg-session/, Accessed: 02/04/2020.