Search space optimisation for procedural content generators

Harald De Bondt

Supervisor(s): Prof. Dr. Ir. Sofie Van Hoecke, Prof. Dr. Peter Lambert, Counsellors: Jelle Van Campen and Gaeétan Deglorie

Abstract— **The gaming market grows every year and in order to meet this demand, game content production needs to be able to scale. The domain of procedural content generation (PCG) offers a solution by automatically generating content. Current PCG approaches often suffer from two big issue. (1) Expressiveness, the ability to generate a large amount of diverse content which is not regarded as repetitive by the consumer. (2) Controllability, the ability of the user to understand the behaviour of input parameters and their influence on the variety of generated content. This paper proposes a new approach, called a "hyper-generator", made out of two process' , (1) the generative process and (2) the learning process. The generative process offers high expressiveness and controllability by combining two distinct types of control, bottom-up and top-down, in a flexible way. Other PCG approaches, exclude by design, possibly interesting varieties in the content in favour of reduced computation time. In contrast, the learning process of the hyper-generator instead tries to increase performance by optimising the probabilistic model for suitable content. The expressive range of the approach is extensively evaluated using a new metric based on visualisation of probabilistic densities. Five generalised scene layouts from architectural context have been extensively evaluated. Three of them have been successfully optimised.**

Keywords—**PCG, learning, controllability, expressiveness**

I. Introduction

The global gaming market is one of the fastest growing markets in the world with a revenue of 83.6B $ in 2014 and a forecasted 91.5B $ for 2015 [1]. The games industry is gaining in popularity and scale and with this comes a demand for games with diverse content. As the content production is becoming too expensive [2] and unscalable [3], we are facing a "content creation problem" . Automatic means of generating content - as found in the academic domain of procedural content generation (PCG) - could help resolve the content creation problem [4]. Not only can generators alleviate parts of manual authoring, they can also increase a game's replayability and unpredictability by presenting the user with a unique variety of the content. In literature, an often used metric for the contents variety is expressive range [5].

To meet the demand of more unique and complex graphics, the techniques driving generators have increased in complexity over the years. This makes them hard to use for the people in charge of content creation, for example non-experts that are not specifically trained to understand the behaviour of the underlying techniques. In current literature on PCG there are two distinct approaches for controlling over content generators, a bottom-up and top-down approach. A bottom-up approach requires the user to manipulate either low-level input parameters or the sequence of operations in the generation process. A top-down approach provides the user with a higher level of control by generating content iteratively and search for the most fitting result across generations. In order to increase controllability, both types of control should be combined in such a way that its respective disadvantages are minimized, without hurting the control type's useful characteristics or reducing the content generators expressive range. I will apply a learning algorithm to automatically optimise the search space of the iterative search process, without hurting the diversity in the search space. This reduces the number of required iterations to find the desired content and allows the use to declare the search space in less detail.

H. De Bondt is with the Computer Science Engineering, Ghent University (UGent), Gent, Belgium. E-mail: harald.debondt@UGent.be .

II. Related work

In PCG literature controllability has been referred to as how much a user understands the shift in resulting content relative to changes of the input parameters [6]. According to [5] there is a trade-off between controllability and expressiveness. A larger expressive range can require the user to tweak more content parameters, a larger range of possible values for these parameters or both. I will distinguish related work based on the techniques used to solve the problem of controllability. The chapter contains a discussion of the previous work in two domains: controllability of PCG and learning techniques applied to design problems in general.

A. Controllability in Procedural content generators

Constructive approaches, for example generative grammars, are a powerful way to describe an infinite amount of possible designs in a bottom-up way. A grammar for generating buildings is CGA++ [7] which in contrast to other grammar based techniques allows context sensitive tasks. However, Its formal, often textual, representation and recursive control logic is generally inadequate for artists. Other solutions try to avoid the text-based construction and increase controllability by using visualisation, more user-interaction or both. Visualising the internal structure of a generator can help the user better understand the generators configuration[8]. Tracery [9], enables novices to work with grammars that generate textual stories. They combine visualisation with iterative feedback by regenerating the story on edits. visualisation and user-interaction can speed up the design process and make it more intuitive, however the user still needs to manipulate most details of the content by hand.

Declarative modelling helps the designer to enforce consistency over the content properties in a top-down manner without having to worry about the details of each separate property. In [10]

they research the declarative modelling for scenes. They divide the scene modelling in a description phase and generation phase. In the description phase the external description given by the designer is translated into either a set of rules and facts, which are processed in the generation phase with a greedy algorithm. The Semantic content generation framework [11] uses knowledge about entities in the world and attributes they possess. However top-down control alone is insufficient in cases where control over detailed and independent properties is desired. Search based approaches can offer both bottom-up and top-down control, respectively implemented with a search space and fitness functions. In [12] several search based approaches based on evolutionary strategies are investigated to find "good" and "diverse" maps. Abstractions of area control, exploration and balance are used as fitness function. With a process called novelty search, the diversity of levels is enforced. The diagram 1, slightly adapted from Togelius et al. [13], situates the search based approach with respect to generate-and-test and constructive approaches.

Most search based approaches rely on stochastic optimisation and indirectly imply a probabilistic model for the generation process. Some research tries to explicitly formalise content generators in a more general way as probabilistic models from which content instances can be sampled. The mathematical properties of such models allow for significant optimisations in the iteration process and provides in some cases finite time theoretical guarantees for the generator. For example Yeh et al. present a novel Markov chain Monte Carlo (MCMC) algorithm to synthesize constrained open world layouts, layouts where the number of objects are variable. They design a MCMC algorithm tailored to handle the change in dimensionality of the problem due to an unknown amount of variables. The probabilistic model which defines the search space is declared in an imperative programming language augmented with two probabilistic primitives, random variables and constraints.

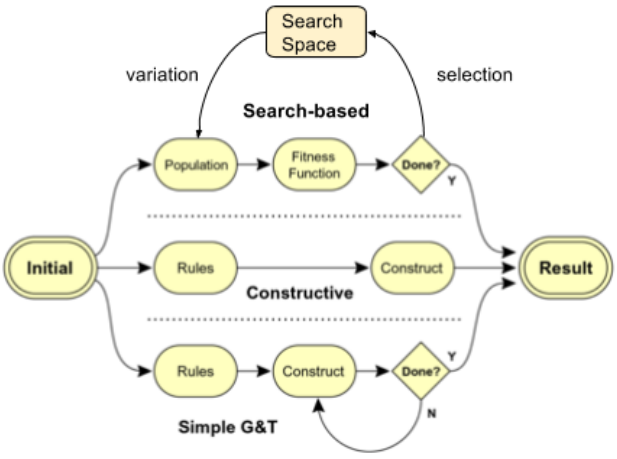


Fig. 1. PCG approaches

Learning based approaches can be used in order to further reduce the user workload using example data. According to Merrel et al. [14] the high-level requirements of building layouts, i.e. the form and placement of rooms and corridors, comes from complex considerations with respect to human comfort and social relationships. However, these considerations have resisted complete formalization. They try to tackle the lack of formalisation by learning them from data. The examples of high-level requirements in the data are extracted from an extensive catalogue of residential layouts.

B. *Solving design problems*

Content generation is not the only domain which tries to solve design problems. More specifically, I will discuss multi-objective (MO) design problems with possibly varying dimensionality which have been addressed using learning techniques. I focus on techniques that are relevant for the learning process in my methodology, and as such I will not address the limitations of their research but rather borrow its insights and apply them to the field of content generation. Karshenas et al. tackle the multi-objective search problem by estimating the search space density jointly with the objectives[15].They propose a multi-dimensional Bayesian network as the underlying probabilistic model for the joint distribution. Their motivation behind it, is to capture the dependencies between objectives and variables. Their experiments show significant results in comparison to state-of-the-art search algorithms based on evolutionary strategies.

III. Methodology

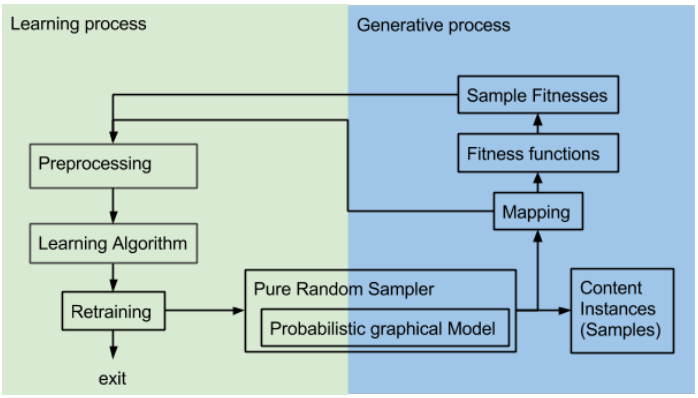


Fig. 2. Methodology overview

A generator with optimal expressive range would naively enumerate all possible varieties of a certain type of content, which results in a good expressive range, but poor performance. I want to maximise expressive range and still offer good performance. To do this, I optimise the search space, which is iterated in purely random way, of my procedural content generator. As mentioned in the introduction, a generator with both bottom-up and top-down control is desired. I define a new type of generator which offers both types of control, called the hyper-generator. It is made out of two process, (1) the generative process and (2) the learning process. In figure X is an overview of the main components in the hypergenerator.

The generative process has two main components, (1) the probabilistic model (bottom-up) and (2) the fitness functions (top-down) which can be defined by the user. In addition, the generative process requires a mapping which maps samples from the probabilistic model to a representation which can be evaluated by the fitness functions.

A. Generative process

The probabilistic model is a hierarchical structure where each node represents an asset type. For each asset type it is possible to define the range of its random variables. I focus on generating open-world layouts, therefore number of generated asset types is variable as well. The hierarchical structure is directed acyclic graph where each node only has a single parent and each parent can, as aforementioned, have variable amount of child nodes. The structure dictates the order for sampling the asset types, parent is sampled before child. The function which relates parent and child variables, called functional relation, can be used to declare an affine transformation between the variables.

In the hyper-generator the mapping is a simple one-on-one transformation of the samples into a representation for the fitness function to evaluate.

The sampled content is constrained with a set of fitness functions. These fitness function can be any injective function of the mapped representation to a real value. It informs the learning process which sampled layouts are desirable and which are not. I distinguish fitness function based on their type of hierarchical relation between instances; (1) pairwise between a parent and child node, (2) relative between siblings or (3) absolute over the complete layout.

In order to properly learn the search space distribution I need an unbiased set of samples, representative for the actual search space. The sampling is a recursive process, which starts with sampling the root parent node, followed by picking a number of children (n) from the user-defined range. Afterwards, n child nodes are sampled and the functional relation between parent and child is calculated.

B. Learning process

For the learning process I formulate the root learning problem of my approach in order to learn from the generation process. I want to learn two types of relations.

The first is between the variables of a parent and its child nodes, the second between the variables of sibling nodes, child nodes with a common parent. the aforementioned relations will be modelled as conditional probabilities, called conditional relations. The conditional relation models the relation between two nodes as the probability for the variables of a node given the variables of the other node. This fits in the generation process because the required order of first sampling the parent is not disturbed. The number of variables in my model can change due to the variable number of children. However, the probability of a random variable is defined by a probability density function, also called densities, which has a fixed dimensionality. To address this issue I will separate samples from the model into independent groups, each only containing samples with a certain amount of children.

To estimate the densities of the variables such that it has a higher probability for values which result in higher fitness I will use a learning method called supervised density estimation (SDE). Supervised learning, does not solely try to learn the structure of the data but requires additional labels for its data.

In contrast to the probabilistic model initialised in the generation process the densities are no longer uniform but defined by a Gaussian Mixture Model (GMM). I chose for a GMM because it is commonly used as model for the probability density of continuous measurements and it is cheap to calculate its affine transformation, necessary in the generation process, and its conditional and marginal distribution, both required for the learning process.

A single probability density function is not enough to learn the densities of variables in the generative process. However, training a density is an expensive operation. Therefore I propose two optimisation techniques to reduce the number of required training operations. The first, called variable sibling order, relies on the independence assumption between the variables of a child and its siblings past a certain order. The second, called marginalisation, assumes that the density which models the variables of a child can be marginalised from the density with higher dimensionality.

To estimate the parameters of a statistical model, a GMM in my case, I will use the most widely used parameter estimating method in machine learning called maximum likelihood estimation (MLE). The algorithm for finding the MLE I use is Expectation-maximization (EM). EM is commonly used for density estimation. However in order to use it for supervised density estimation, it needs additional methods. I will use two different methods. The first is called weighted density estimation (WDE), which estimates density given a dataset and their corresponding weights. These weights reflect the importance of the respecitve sample in the dataset[16]. The weights for WDE cannot be multi dimensional. The second technique I use, which does not suffer from this limitation, is called probabilistic regression. It estimates the density of layout variables V given their fitness F by calculating their conditional probability $(P(V|F))$ [17].

In order to use the learning algorithms, the samples from the generation process need to be transformed in proper training data with a fixed format. First, I extract the variables, which are part of the conditional density, from the hierarchical samples and format them in a flat vector. Secondly, I generate the fitness data, which are the labels for the SDE, by calculating the fitness values for all children of the samples and format them in a vector as well. After generating the training data I transform this data. First, I filter out samples with fitness values below a certain threshold. Afterwards, I apply a polynomial order to

the fitness values which affects the influence of fitness data on the learning process. The exact influence will be evaluated. Because the learning algorithm weighted density estimation only accepts a single weight, or label, per sample I need to reduce the dimensionality of the fitness vector to a single value, i define this type of dimensionality reduction as “single fitness dimension”. I additionally add two other types of dimensionality reduction as parameters to be evaluated when using probabilistic regression, I define as “sample fitness dimension” and “ sample fitness dimension”.

The retraining component, repeats the training of the probabilistic model, with two different loops called retraining trials and iterations. For two reasons; first, to avoid sampling bias. Second, to increase the performance of the generator even further by repeating the training on an already trained probabilistic model. The sampling process is the sampling discussed in the generation with an additional ordering for generating the child samples. Because after updating the probabilistic model in the training process, each child is conditionally dependent on its previously sampled siblings. One additional problem when sampling a trained density is that its samples can exceed the user defined range of a variable. In order to respect the user defined ranges, a rejection sampling method is employed, which is is still purely random.

IV. Evaluation and discussion

Scene layouts in architectural context, which are a suitable type of content because it has both bottom-up and top-down properties. The constraints of the scene, defined as fitness functions, are inspired by the works of Yeh et al. and Merrell et al. An exhaustive analysis of the possible influence of each fitness function on the learning process is out of scope of this thesis. I will however evaluate the most commonly used constraints and compare them, based on their characteristics necessary to distinguish them in terms of behaviour relative to their input. I will evaluate the fitness functions minimum polygon overlap, target surface ratio, target distance, minimum centroid difference and closest side alignment.

Because expressiveness is a hard to measure quality [6], I will evaluate it with several different approaches based on visualisation of the density functions in the probabilistic model. However due the high dimensionality of the problem most of these require a more ad-hoc approach on low-dimensional use-cases (< 3). For one use-case I will do an exhaustive analysis of visualisations in two parts; (1) in terms of variance of the density and how much of the variance covers actually desired content and (2) in terms of bias towards particular values. In the first part I visualise the expressiveness in figure X, while considering the fitness function, first by comparing heat-maps between naive rejection sampling, also called generate and test, and rejection sampling with our method.

I visualise the density of the trained GMMs in two dimensions in figure X. This is done by draw their outer ellipses which contain the two order confidence level (95%) of its Gaussian components, called Gaussian surface area (GSA). Green is the true positive area, the area covered by both the trained model and the optimal search space, cyan the false positive area and red the false negative.

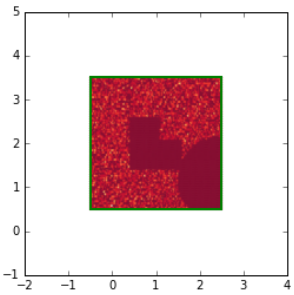


Fig. 3. Heat-map for naive sampling

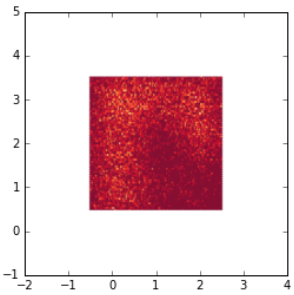


Fig. 4. Heat-map for sampling from trained model

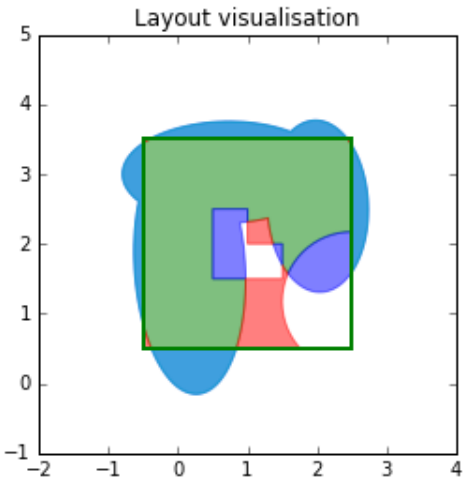


Fig. 5. GMM surface area coverage

The visualisation is used to calculate an analytical score for the expressiveness. The true positive area covers 89% and the true negative 85%. The second part visualises the expressiveness bias. I will do this by marginalising the variables from the joint distribution that models them.

On the figures with marginalised variables I show that the bias towards particular values is limited, the peaks in the trained density are close to the uniform distribution. To evaluate all methodology parameters I focus each time on one or two parameters, let them vary between a range, and analyse the results in order to be able to discuss their influence on the metrics. In these experiment expressiveness will be visualised by marginalising each trained variable separately, in terms of GSA. The performance of a trained model will be measured as the difference in average fitness before and after training, called average fitness gain (AFG). The computation time is measured as the amount of seconds to execute the learning method. I will average the results from these metrics over 5 probabilistic model each with a different fitness function, as mentioned in the beginning of this section. I summarise the evaluated parameters grouped according to their location in the methodology.

- Retraining: number of trials and iterations
- Data generation parameters: number of data samples, Pois-

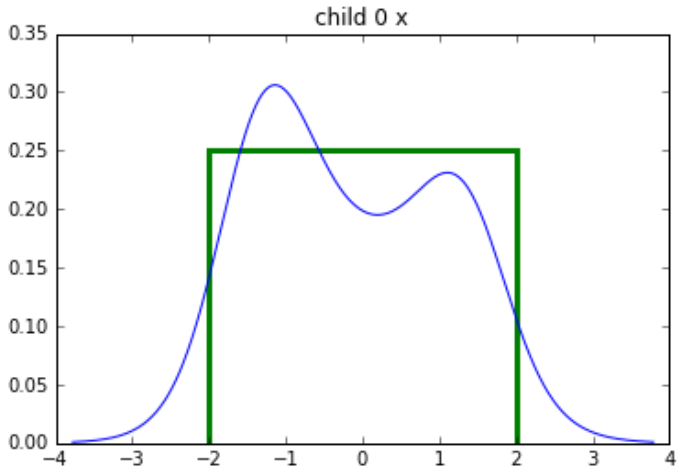
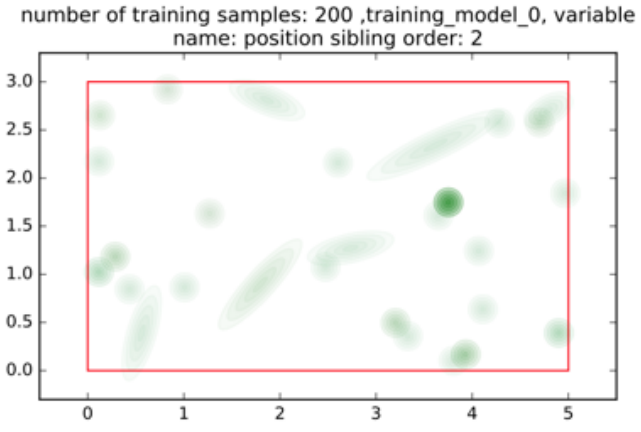


Fig. 6. Marginalised distribution for the x coordinate of child 0

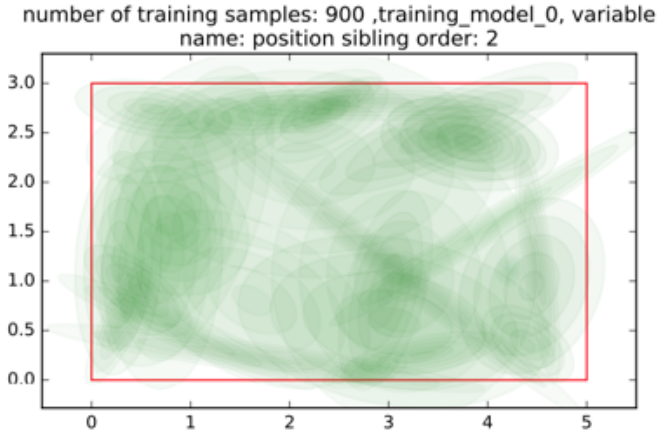
son disk sampling and fitness dimensionality reduction in probabilistic regression

- Fitness parameters: fitness polynomial order, fitness target, fitness hierarchical relation
- Hierarchical model parameters: maximum sibling order, GMM marginalisation
- Training model configuration: number of training variables

The last experiment is not a parameter of the methodology but evaluates the influence of removing one of the variable in probabilistic from the training data I will give the most significant results of these experiments and summarise how well each aforementioned fitness function has been optimised. Increasing the number of samples in the training data for the learning algorithm, significantly increases performance and expressive range. While only increasing computation time with 50% when quadrupling the number of samples. Probabilistic regression significantly outperforms weighted density estimation as supervised density estimation in terms of expressive range of the content, with similar results in terms of performance and computation time.



I was able to increase the average fitness value over all parameter configurations for the fitness functions target distance (22%), minimum polygon overlap (100%) and minimum centroid difference (45%). The other fitness functions; closest side alignment and target surface ratio, were harder to optimise and only reached respectively 5 and 10% improvement in the best of cases over all parameters.



V. Conclusion

I have presented a new approach which is able to optimise the search space of a content generator for some fitness functions. This is achieved by using a learning algorithm which automatically adapts a user-defined search space, with a minimal decrease in expressive range of the generated content. I have introduced a new metric for the expressive range of content generators based on the probability density function of the random variables in the search space. The probabilistic model in my methodology has two limitations; (1) it is restricted to continuous random variables, (2) its acyclic and directed structure cannot model cyclic generation process' like grammars or undirected generation process' like sampling from a factor graph. I think that my methodology, the hyper generator, of combining a probabilistic model, high level constraints and a learning algorithm which adapts the first to the lather, is sound and would be a strong alternative to other existing approaches for PCG. However, the results from current implementation are not significant enough to choose my method over standard search algorithms without adaptive search space, for example Markov chain Monte Carlo sampling. In order for the hyper generator to be a considered as alternative, I envision a more complex and suitable probabilistic model, which would be capable of capturing a wider range of fitness function with high accuracy. Therefore, I also suggest to drop any optimisations which reduce the model's complexity, in favour of reduced computation time.

References

[1] Newzoo, "Global games market will grow 9.4% to $ 91.5bn in 2015," 2013. Consulted on 25-May-2016 via http://www.newzoo.com/insights/global-games-market-will-grow-9-4-to-91-5bn-in-2015/.

[2] Y. Takatsuki, "Cost headache for game developers," 2007. Consulted on 25-May-2016 via http://news.bbc.co.uk/2/hi/business/7151961.stm.

[3] A. Iosup, "POGGI: Puzzle-based online games on grid infrastructures,"

Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 5704 LNCS, pp. 390–403, 2009.

[4] G. Kelly and H. McCabe, "A survey of procedural techniques for city generation," *ITB Journal*, vol. 14, pp. 87–130, 2006.

[5] G. Smith and J. Whitehead, "Analyzing the expressive range of a level generator," *Proceedings of the 2010 Workshop on Procedural Content Generation in Games PCGames 10*, 2010.

[6] N. Shaker, G. Smith, and G. Yannakakis, "Evaluating content generators," *Proced. Content Gener. Games A Textb. an Overv. Curr. Res.*, pp. 211–218, 2015.

[7] M. Schwarz, "Advanced Procedural Modeling of Architecture," 2015.

[8] P. B. Silva, P. Müller, R. Bidarra, and A. Coelho, "Node-Based Shape Grammar Representation and Editing," *Pcg*, pp. 1–8, 2013.

[9] K. Compton, B. Filstrup, and M. Mateas, "Tracery : Approachable Story Grammar Authoring for Casual Users," *Pap. from 2014 AIIDE Work. Intell. Narrat. Technol. (7th INT, 2014)*, pp. 64–67, 2014.

[10] D. Plemenos and G. Miaoulis, *Visual Complexity and Intelligent Computer Graphics Techniques Enhancements*. 2009.

[11] M. Bennett, "Semantic content generation framework for game worlds," in *2014 6th Int. Conf. Games Virtual Worlds Serious Appl. VS-GAMES 2014*, vol. 35, pp. 352–363, 2015.

[12] M. Preuss, A. Liapis, and J. Togelius, "Searching for good and diverse game levels," in *IEEE Conf. Comput. Intell. Games, CIG*, pp. 1–8, 2014.

[13] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-Based Procedural Content Generation: A Taxonomy and Survey," *IEEE Trans. Comput. Intell. AI Games*, vol. 3, no. 3, pp. 172–186, 2011.

[14] P. Merrell, E. Schkufza, and V. Koltun, "Computer-generated residential building layouts," *ACM Trans. Graph.*, vol. 29, no. 6, p. 1, 2010.

[15] H. Karshenas, R. Santana, C. Bielza, and P. Larranaga, "Multi-objective Estimation of Distribution Algorithm Based on Joint Modeling of Objectives and Variables," *IEEE Trans. Evol. Comput.*, no. c, pp. 1–1, 2013.

[16] "Pomegranate, probabilistic modelling library for Python."

[17] S. Ghahramani, *Fundamentals of Probability with stochastic process*. Pearson Education, 1994.