EXPLORING SHAPE GRAMMAR OPTIMIZATION

AS A TOOL FOR AUTOMATED DESIGN

An Undergraduate Research Scholars Thesis

by

JORDAN ALEXANDER CAZAMIAS

Submitted to Honors and Undergraduate Research
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Research Advisor: Dr. Dylan Shell

May 2014

Major: Computer Science
Applied Mathematics

TABLE OF CONTENTS

ABSTRACT

Exploring Shape Grammar Optimization as a Tool for Automated Design. (May 2014)

Jordan Alexander Cazamias
Department of Computer Science & Engineering
Texas A&M University

Research Advisor: Dr. Dylan Shell
Department of Computer Science & Engineering

Shape grammars are quite effective at representing the structure of objects, thus raising the question of whether they could be utilized for design automation or related techniques like procedural generation. However, refining these grammars requires tediously adjusting its many hard-coded parameters. This research serves to answer whether a sub-optimal shape grammar could instead be adjusted using grammar induction and optimization techniques. A general optimization framework for shape grammars was defined to address this question. From this framework, a specific optimization process was also created and its effectiveness was tested in a pilot experiment. To carry out this experiment, a program was written to take a textual design grammar as input and, after several rounds of training by the user, adjust the grammar's parameters such that it outputs higher-quality designs. After collecting data on the mean grammar design quality per round, it was found that the quality of the designs was, in fact, significantly higher in later rounds than in earlier rounds. This provides an encouraging first step into the potential for applying this optimization framework to design grammars in general.

ACKNOWLEDGEMENTS

I would like to thank Dr. Dylan Shell; he has been a great mentor and guide throughout this process. I would also like to thank Ben Fine, my current graduate student mentor, for his encouragement and contribution to the inspiration for this project idea. Without both of these people's support, my project would not have taken the promising turn that it has so far.

The 3D building models in Figure 1 were created in Structure Synth and provided by yhancik at http://www.flickr.com/photos/yhancik/3179604858/ under a Creative Commons license.

CHAPTER I

INTRODUCTION

A staggering number of real-world objects, both natural and man-made, follow a certain structure in their design or form. Cars, for example, may be widely varied but share many commonalities in their overall structure and design: they all have headlights and tail-lights, wheels, windshields, etc., and all in relatively similar positions. This hidden, common structure is essentially a collection of *de facto* rules for how objects should be built, which allows them to be separated into classes. Understanding these rules, then, can help one to understand an object's design at its core. Taking this one step further, if there exists a way to encode these rules so that a computer could analyze them, these rules could be easily modified on the fly, which would potentially allow for novel ways to classify, generate, and even optimize object designs with minimal human interaction.

Many industries already employ the *shape grammar* (a set of primitive shapes and a set of rules for constructing a complex object out of primitive shapes) as a model for an object. It is a useful method for the procedural generation of urban landscapes, since buildings (while they are not uniform) follow a well-defined structure [1]. Shape grammars are also quite straightforward to build using existing software tools. Context Free (available at http://contextfreeart.org) has been designed to allow the creation of 2D art and designs by building shape grammars, and Structure Synth (available at http://structuresynth.sourceforge.net) is an equivalent program for 3D models. Both of these programs define a simple language used to create these shape grammars and only require basic programming experience. Figure 1 demonstrates some of the examples of user-generated art using this software, all of which are generated from relatively simple programs (on the order of tens of lines).

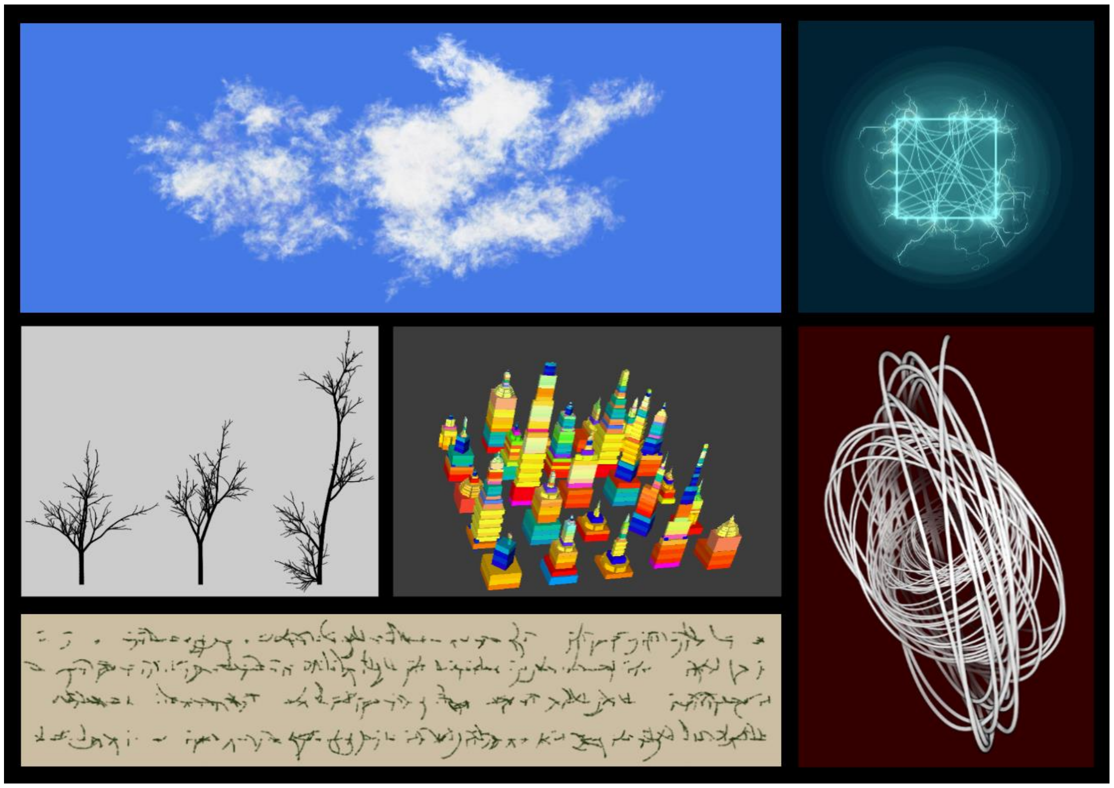


Figure 1: Examples of designs created by users of the Context Free and Structure Synth programs. All of these designs are created with random variation, and so will be different every time they are generated.

Not only are these grammars powerful, but they are fairly straightforward to build, at least with regard to their basic structure. However, refining these grammars, or creating more complex grammars, typically requires manually adjusting the grammar's numerous obscure parameters, a time-consuming and frustrating process of trial and error (see Figure 2). Furthermore, since the objects in question follow a structured generation process, it begs the question of whether shape grammars could instead be improved by a computer, through optimization techniques. If so, the benefits of using shape grammars as a model for structured objects, and even as a tool for Computer-Automated Design, could be made much more accessible to designers, artists, and programmers alike.

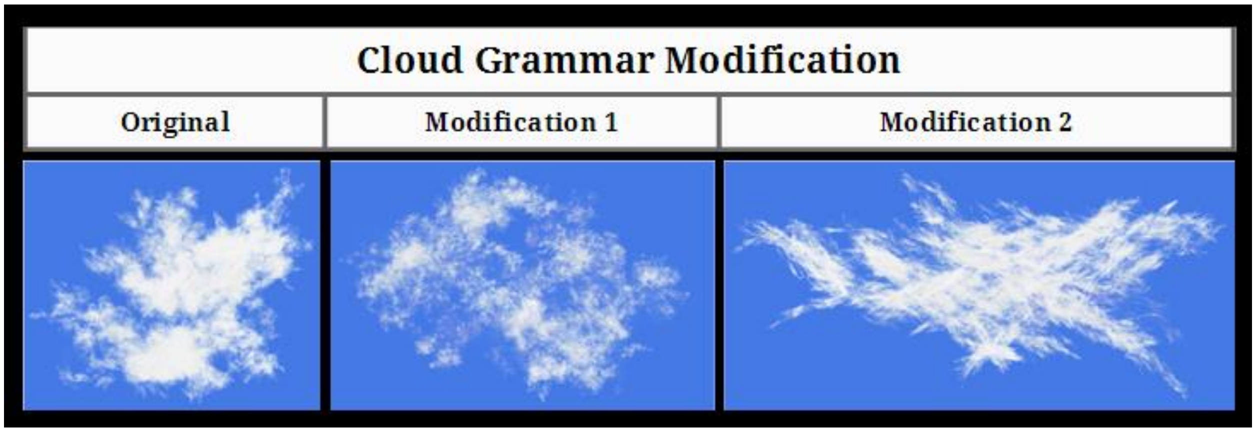


Figure 2: Making a few small changes to the rule probabilities, or other attributes, of a shape grammar can result in drastic changes in the resulting designs' structure. The effect that changing these attributes will actually have, however, is typically hard to predict.

A process, related to this research, has been developed to take fully formed objects as input and construct a grammar that can regenerate these input objects. This is known as *grammar induction.* This induction process assumes that there exists some optimal grammar that captures the input objects and tries to find a grammar as close to this optimal grammar as possible. At the moment, the lion's share of grammar induction research has been applied to textual grammars, though recent work has been performed on the application of grammar induction to shape grammars. Most of this research, however, is geared toward specific applications. Several of the referenced papers [1,2,5] use induction specifically for the generation of urban buildings and facades. Talton et al. [3] take a large step towards a more general approach by inducing grammars to capture certain design patterns, but the objects in question—buildings, trees, and HTML documents—are still rather specialized and typically require complex starting pieces. Furthermore, there is a large drawback to all the aforementioned induction strategies: they not only require training examples, but the examples themselves must be labeled in a way that illustrate their derivation process [7]. This task of labeling may not be much trouble to a resident expert in the fields of formal grammars and natural language processing, but it is a high barrier to entry for any common user. Without labels, however, creating a grammar from scratch using induction becomes an enormously difficult challenge.

Instead of relying on full grammar induction, this research proposes a framework for grammar *optimization* that draws from induction techniques. This framework could serve as a viable tool for anyone with basic programming knowledge because it does not require any domain-specific expertise, unlike a pure induction strategy. It should also be oriented towards a more general goal: that of creating a grammar that can generate objects to achieve a predefined utility or purpose. This purpose may be any that the designer deems necessary for the designs to have, be it a capability to perform a certain task, a fundamental design pattern, an appealing aesthetic, or a combination of several features. Most importantly, the utility does not have to actually relate to an object's shape, or even be any measure that is easily quantifiable by a computer, so long as an object can be tested for the utility in some way. For example, if a designer has created a shape grammar that generates images of clouds, such as in Figure 2, an appropriate criterion for the cloud designs is that they appear to be of a certain cloud type, such as cirrus or cumulonimbus. This criterion is not easily measured by a computer without various machine learning techniques (this point will be further discussed in Chapter IV). However, a human could certainly distinguish between clouds that fit a certain type and clouds that do not.

In theory, a more general treatment of shape grammar optimization will cover many more applications, some of which are inconceivable today. The ultimate goal for this research (a goal that is outside the scope of this thesis) is to make the idea of automated design a reality for anyone with the most basic programming knowledge, even that which is reasonable for artists and designers to be able to learn with minimal hassle. Whether this is possible, and how well this could be carried out with the proposed system, will be the focus of exploration in this project.

Objectives

The primary goal is to propose a strategy for achieving shape grammar optimization in an induction context. I hypothesize that shape grammar optimization is not only possible, but can be used to create grammars of a quality that surpasses that of grammars created manually within a reasonable amount of time.

CHAPTER II

METHODS

One particular challenge with relating grammar induction to shape grammar optimization is that most induction research has been focused on textual grammars rather than more design-oriented grammars, and applying the same assumptions and principles from textual grammar induction would not be the ideal way to approach design grammar optimization. Both types of grammars are used for vastly different purposes. Traditional textual grammars are built mainly for the purpose of parsing natural language (or to formally define programming languages). Essentially, the goal of these grammars is to closely mimic a human's ability to recognize the structure of valid sentences within the language. This is a task with a clear metric of quality (i.e. a grammar can be considered high-quality if it can parse a sentence almost exactly how that sentence would have been parsed by a human). Accuracy is the key goal, and this accuracy is easily measured. Few textual grammars are created with the purpose of generating natural human-sounding sentences, although there are some notable exceptions [6]. Design grammars, on the other hand, are used primarily to generate new designs rather than parse existing designs. Accuracy is still important—a design for a house is no good when the goal is a design for a car, for instance—but it is not paramount and it is more loosely defined. Essentially, the key goals for design grammars are novelty and creativity, which are not as easily quantifiable.

Optimization Framework Proposal

Figure 3 illustrates the top level operation of the created optimization framework:

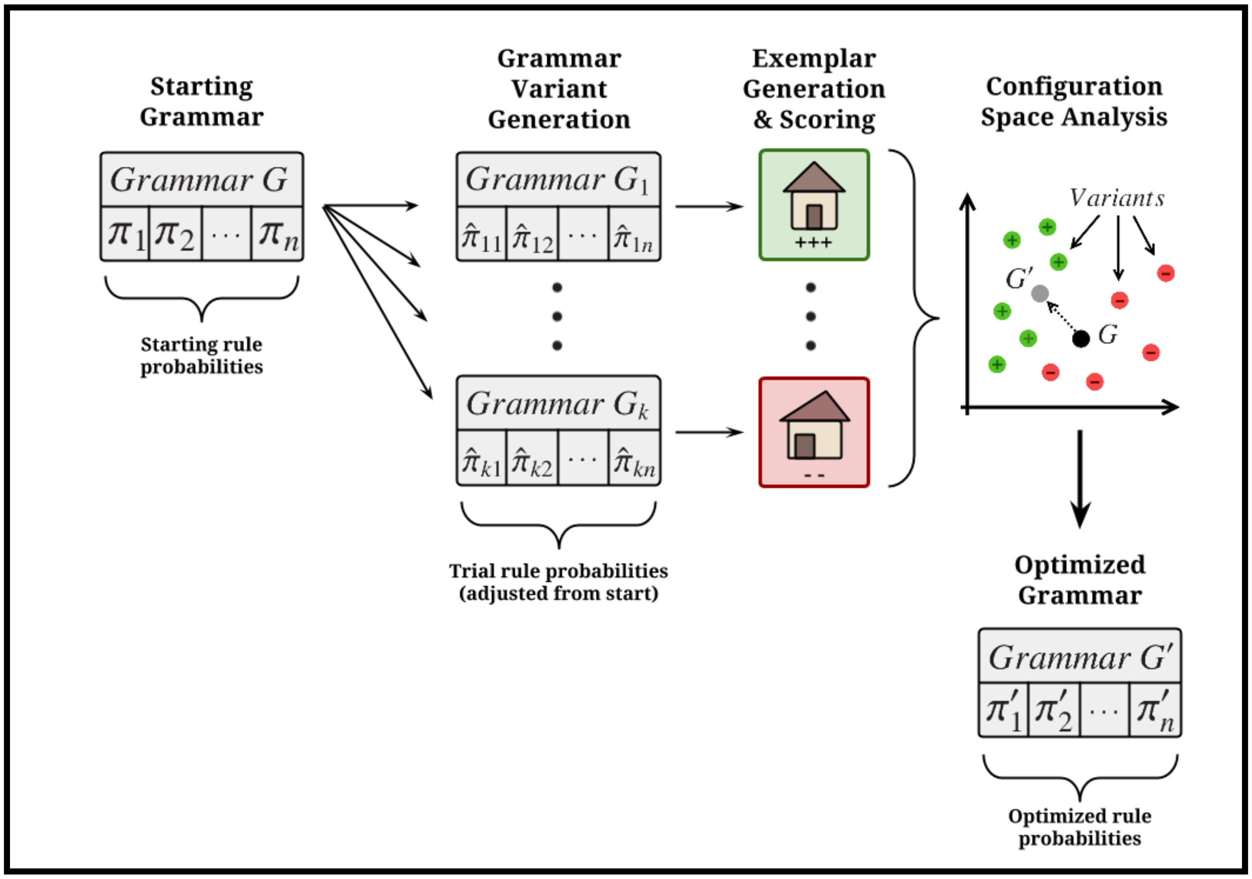

Starting Grammar
Grammar G
π_1 π_2 $\cdots$ π_n
Starting rule probabilities
Grammar Variant Generation
Grammar G_1
$\hat{\pi}_{11}$ $\hat{\pi}_{12}$ $\cdots$ $\hat{\pi}_{1n}$
Grammar G_k
$\hat{\pi}_{k1}$ $\hat{\pi}_{k2}$ $\cdots$ $\hat{\pi}_{kn}$
Trial rule probabilities (adjusted from start)
Exemplar Generation & Scoring
+++
- -
Configuration Space Analysis
Variants
G'
G
Optimized Grammar
Grammar G'
π'_1 π'_2 $\cdots$ π'_n
Optimized rule probabilities

Figure 3: Top level design of the induction-based grammar optimization framework

As input, a grammar *G* is provided. This grammar has various attributes that can be changed by the user, which should change the output of the grammar. In this case, Context Free uses a stochastic grammar, and so each rule has an associated **rule probability** that can be set by the user. For this project, the only grammar attributes that will be considered for modification are each of the grammar's rule probabilities $\pi_1, \dots, \pi_n$. These parameters affect how often a rule will be selected, and can drastically change the structure of the resulting designs. An appropriate next step in this research should be expanding the optimization algorithm to allow for other attributes to be modified, such as the shapes' transformation parameters.

From grammar *G*, the **grammar variants** $G_1, \ldots, G_k$ are created. These are copies of grammar *G* where some of the rule probabilities have been modified. How these rule probabilities are modified is one of the many points of generalization within this optimization framework; there are many ways to approach this question, and anyone using the framework should be able to customize how it occurs. Perhaps the simplest (and most likely fastest) approach is to scramble the rule probabilities for each grammar variant. However, for more fine-tuned control, the rule probabilities can be more systematically changed. For instance, grammar G_1 can be a variant of *G* where only π_1 is modified, and so on. This would also help to isolate the rule probabilities that make the largest impact on the quality of the grammar. After the grammar variants are created, each variant generates one or more **exemplar designs**, all with potentially different levels of quality. This design "quality" may or may not be easily quantifiable by a computer. However, the criteria used to judge exemplars should be well-defined enough that a human user should definitely be able to make judgments on an exemplar's quality. Therefore, these exemplars will be scored by a human user. The structure of this scoring system is another point of generalization, though it is suggested to include two important features:

- The score should be *weighted*, meaning that a user should be able to segregate exemplars that have a weakly-positive quality and a strongly-positive quality; i.e. the scoring is not a binary yes/no answer.

- The score should be *signed*, meaning that negative scores should be possible. This would allow a user to isolate exemplars that are so low-quality that they would weaken the quality of the overall grammar and, thus, should be discouraged. The score may not need an actual negative value, so long as some balance point between “good” and “bad” scores exists; for example, a scoring system from 1 to 5 can be used, with 3 as the balance point.

Using the scored exemplars, the grammar *G* and all of the grammar variants will be plotted within the **configuration space** for grammar *G*. The configuration space is the *n*-dimensional vector space where a grammar *G* is represented as the vector containing all of its non-fixed parameters; in this case, $\langle \pi_1, \pi_2, \ldots, \pi_n \rangle$. In essence, the configuration space contains all of the possible variants of grammar *G*. The goal for optimization using the configuration space is to take grammar *G* and calculate an offset to move *G* to a new position in the space, in order to find a more optimal grammar *G'*. The details for how to calculate this offset is yet another point of generalization, and perhaps the most important one. One potential solution involves each grammar variant exerting a "force" on the offset grammar *G*. In other words, each grammar variant G_i will contribute an influence to the offset of grammar *G*, dependent on the vector $\delta_i = k\frac{G-G_i}{\|G-G_i\|}$, where $k \in \mathbb{R}$ and $\frac{G-G_i}{\|G-G_i\|}$ is the unit vector starting at G_i and oriented toward *G*. Whether k, the weight of this offset vector, is positive or negative depends on whether the grammar variant G_i was positively or negatively scored, and the magnitude of k will depend on many factors including the score of G_i, its distance from grammar *G*, etc. In general, positively-scored grammar variants will exert an attractive force on grammar *G*, encouraging grammar G' to be closer to these variants. The opposite goes for negatively-scored variants.

All of these offset vectors are combined into a total offset $\Delta = \sum \delta_i$, which is then added to the configuration vector for grammar *G*. This new vector will be considered a new grammar G'. This new grammar should have rule probabilities that are closer to those of the "high-quality" grammar variants than those of the "low-quality" variants. Thus, if the model is well-formed, grammar G' should be more likely than grammar *G* to create high-quality designs, and should therefore be considered more optimal.

Pilot Experiment

The goal of creating an entire software package with full 2D shape grammar support was not feasible given the allotted time. To demonstrate the optimization potential of this framework, however, a pilot experiment was created to serve as a precursor. This experiment is essentially a specific implementation of the aforementioned optimization framework, with the following extra simplifications:

- Rather than optimize a shape grammar, the experiment will attempt to optimize a text grammar. One important distinction to make, however, is that this text grammar will be used to generate words rather than parse human text. So, this grammar will be considered a design grammar rather than a textual grammar (based on the definition of “textual grammar” mentioned earlier).

- Instead of a full configuration space analysis, the grammar variants will simply be derived from the highest-scoring grammar variant to date, and simulated annealing will be performed so that the variants converge on a locally optimal grammar [8].

Again, a design grammar should be optimized based on how well its designs achieve the criteria of the designer. These criteria may be virtually any factor, so long as a generated design can easily be judged by a human scorer based on how well it fits these criteria. For this experiment, the criterion was chosen to be that something that is easily determined by humans and also something most people are already familiar with.

The criterion chosen was: **How well would this word fit as the name of a spell or incantation from the Harry Potter series?** Besides having the previously mentioned benefits, the Harry Potter criterion is a favorable one because it has a distinct, but not binary, success state. Most incantations from Harry Potter are derived from Latin; as such, words that sound like they are derived from Latin or related languages should fit the criterion much better than words derived from unrelated languages such as Japanese. One may ask, then, *why not make the criterion "How well does the word sound like Latin or one of its derivative languages?"* The entire Latin vocabulary, however, encompasses an enormously wide and varied palette of words, which would make judgments along this criterion ambiguous and more difficult to keep consistent. Harry Potter spells, on the other hand, draw from a much more targeted subset of Latin words; as a result, they have a more unified feel, which will hopefully aid human scorers greatly in the judging process and reduce unwanted variance.

It was specifically hypothesized for this experiment that the grammar variants created in later rounds of optimization would have a higher quality overall than those created in the earlier rounds. To test this hypothesis, a textual grammar interpreter program was created to enable the generation of words from a simple, user-defined context free text grammar. A feature was also added to allow the user to specify which rules they would like to recalculate probabilities for using optimization and which rules should remain fixed. Other than this feature, however, the program's function was designed to closely match that of Context Free in order for the experiment's results to be as relevant as possible to discussions on the potential of optimizing shape grammars, especially those created using Context Free.

Several of the most commonly used Harry Potter spells were analyzed to determine how the grammar for a typical "spell" should be structured. In the grammar, a spell was restricted to a single word that consisted of three parts: the beginning segment (from 0 to 2 syllables long), the penultimate (i.e. second-to-last) syllable, and the final syllable. Each of these parts had their own probability distribution for individual letters as well as for the types of syllable used. For instance, most words started with a vowel or consonant and simply alternated VCVC… or CVCV…, respectively, while some words occasionally contained a syllable with a double consonant or double vowel. The rules that determined the balance of syllable types were marked for probability recalculation, and the rest were left fixed.

To optimize the grammar, a simplified version of the aforementioned process was used.

- First, the designer decides to hold i rounds of optimization.
- During each round, j grammar variants are generated. These variants are derived from the most optimal grammar variant that has been found up to this point (or, for the first round, the original grammar). These grammar variants have their non-fixed rule probabilities randomized, but in a way that is centered on the corresponding rule probability from their derived grammar. More specifically, for every probability π_i in the derived grammar, the variant probability π_i' should be chosen from some probability distribution where π_i is the median value.
- To evaluate each of these variants, k exemplar designs are generated by each variant and are subsequently evaluated by a human scorer. The overall score for a variant is simply the average of its exemplars' scores.
- After scoring each variant in the round, they are all compared to the most optimal grammar found. If one of these variants has a higher overall score than that of the most optimal grammar found, it becomes the new optimal grammar.
- With subsequent rounds, simulated annealing is employed to decrease the random variation in the variants' rule probabilities. Thus, the variants should converge towards the most optimal grammar found.
- After all the rounds, the most optimal grammar is returned as *G'*, the new grammar.

For this experiment specifically, 5 optimization rounds were carried out, each with between 5 and 8 variants. Each variant was used to generate 25 words, which were rated on a 1-5 scale by human evaluators (a score of 0 was also allowed if the word contained a nonsensical combination of letters; for instance, *ccaneva* was an exemplar that was given a score of 0).

To emulate the function of Context Free as closely as possible, the text grammar interpreter utilized the same approach towards rule probabilities by using a weight system. Therefore, for a nonterminal symbol N_1, suppose rules $r_1, \ldots, r_n$ are all the rules where N_1 is on the left hand side, meaning they encompass are all the possible ways N_1 can be converted to other symbols during the design generation process. Each rule r_i has an associated weight $w_i > 0$. Then, the probability π_i that r_i will be selected whenever one of N_i's rules is chosen is:

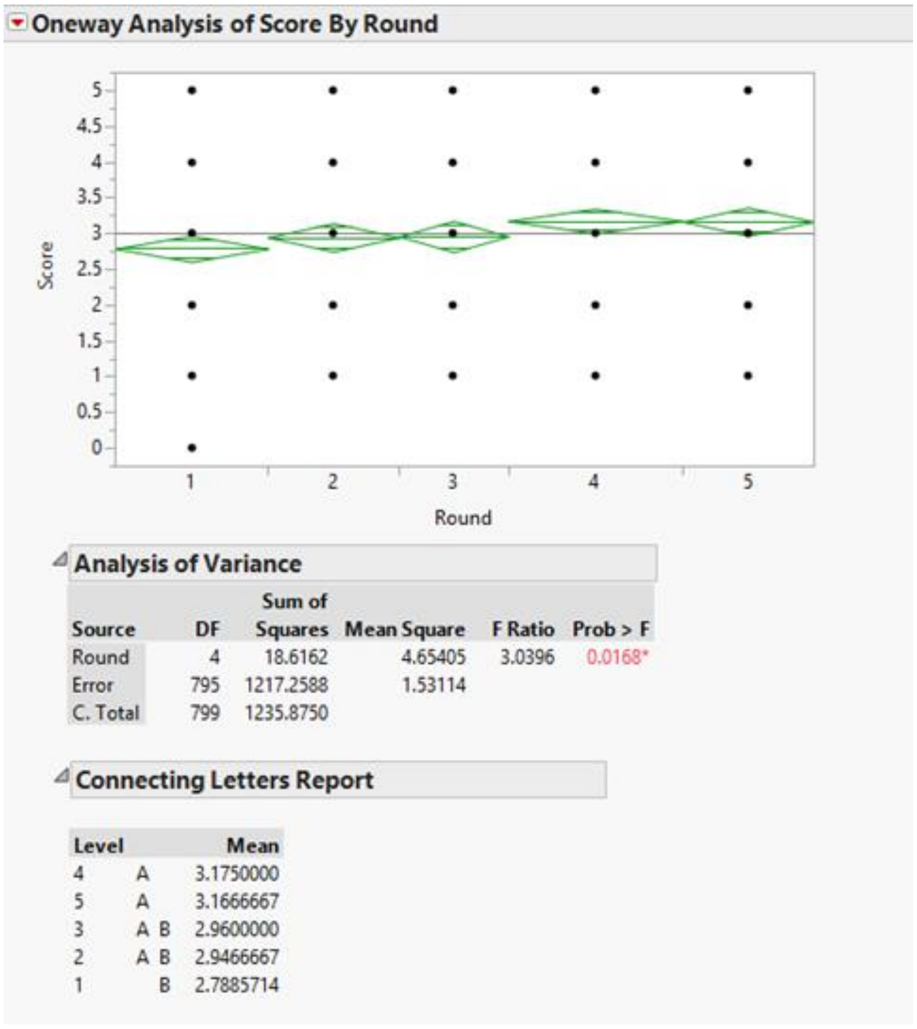
$$\pi_i = \frac{w_i}{\sum_{j=1}^{n} w_j}$$

The only requirement for a weight is that it is positive, so it can be any positive floating point number. However, weights are typically relatively small (on the order of 1 or 2 digits), and changes in smaller weights have a larger impact than changes in larger weights. Thus, for a weight w_i, the new weight w_i' was determined by drawing a random variable from the lognormal distribution with a mean value of $\mu = \ln(w_i)$ and a predetermined standard deviation σ. Because of the value of μ, the distribution will have a median of w_i. To optimize the grammars via simulated annealing, the value for σ was decreased with each round, eventually converging to 0.

CHAPTER III

RESULTS

After gathering the scores for each of the variants (with 800 words evaluated in total), the overall round scores (i.e. the mean of all the variant scores per round) for the Harry Potter criterion were compared. The results are shown below.



Oneway Analysis of Score By Round

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio	Prob > F
Round	4	18.6162	4.65405	3.0396	0.0168*
Error	795	1217.2588	1.53114		
C. Total	799	1235.8750			

Connecting Letters Report

Level			Mean
4	A		3.1750000
5	A		3.1666667
3	A	B	2.9600000
2	A	B	2.9466667
1		B	2.7885714

Figure 4: Overall mean scores per round, with ANOVA and Tukey's Procedure included

The graph shows the mean diamonds for each of the overall round scores; they demarcate the upper and lower bounds for the 95% confidence intervals for each of these scores. Using One-way ANOVA to compare the overall round scores, it can be stated with almost 99% confidence that these mean scores are not all the same. To determine which round scores can be considered different, Tukey's Procedure was used to compare each pair of scores. According to Tukey's, there is a significant difference between the scores for the elements in pairs (Round 1, Round 4) and (Round 1, Round 5) with at least 95% confidence. Drawing from this result, it can be inferred that the quality of the words in the later rounds (4 and 5) is higher overall than that of the words in the first round. This suggests that the optimization process is improving the quality of the grammar variants over several rounds.

CHAPTER IV

DISCUSSION

The data obtained from the pilot experiment suggests that the optimization process used was indeed successful in improving the Harry Potter grammar over time, and is an encouraging first step. Perhaps with further improvements to the algorithm, such as more optimization rounds and more fine-tuned changes in the random variation used, the variants could have the opportunity to converge even further and achieve even more optimized results. Regardless, it will be said that the hypothesis was supported.

This pilot experiment, of course, is only the first step in exploring the potential for using this framework to optimize shape grammars based on a designer’s criteria. There are a few directions that future research can take from here, many of which could be studied independently. The first and foremost of these directions is expanding the use of the optimization framework to actual shape grammars in Context Free to further support its effectiveness. This would also tie into the need to develop an open-source optimization tool for designers, as well as researchers, to use. With 2D shape grammars, other parameters are introduced other than just the rule probabilities; these parameters should also be studied for their potential in improving the structure of the generated designs. From 2D shape grammars, the natural extension would be to explore the possibility of applying similar techniques to 3D shape grammars. This should not pose too much difficulty once 2D grammar optimization is achieved, as 3D grammars are essentially 2D grammars with a few extra parameters such as new transformations.

Another direction to take, perhaps just as important as the first, is attempting to use machine learning techniques to replace the need for human evaluation. Grammar optimization is a data-oriented goal, and as such, its major limitations will primarily stem from the availability and reliability of this data. Human evaluation, while fairly reliable en masse, is a much slower process and one that is prone to much more variance than a purely computer-based technique. For instance, say a designer wanted to optimize a shape grammar that generates images of clouds and, instead of relying on human evaluators to rate the cloud exemplars, he or she could compile a training set made up of real images of clouds and have the computer compare the exemplars to the training designs. This would be more challenging to achieve, but would enormously speed up the evaluation time needed to optimize the grammar and eliminate much of the variance that arises from human-centered studies. Plus, in order for the designer to test the exemplars for specific criteria (for example, if the designer wanted the cloud grammar shown in Figure 1 to only generate cumulonimbus clouds), the training corpus would simply need to consist of designs that strongly achieve these criteria (in this case, it would simply need pictures of cumulonimbus clouds).

From the ability to quickly mine this data, other opportunities could open to further improve the optimization process used. For example, the standard deviation for the random variable used to scramble rule probabilities is currently the same for all rules; it only decreases with each successive optimization round. However, some rules may be much more sensitive to alterations in their probabilities than others. If a grammar contains some of these high-precision rules, and all of the rules' probabilities are changed a similar amount, then the effect of changing the highly sensitive rules could overshadow that of other, less sensitive rules. To avoid this effect and allow for more fine-tuned control of the grammar variant creation process, the standard deviation could be customized for each rule to account for its sensitivity. To do so quickly would require a computer evaluating the quality of exemplar designs rather than humans. Assuming this was possible, then the computer could run a set of preliminary evaluation trials where the rule probabilities are systematically changed and their effects on the resulting designs are measured. From these measurements, a gradient can be approximated to describe the "sensitivity" of rules and recommend appropriate standard deviations for the optimization trials.

Finally, another potential direction deals with addressing a caveat with the proposed optimization framework. Namely, the proposed method relies on constructing a configuration space for all possible variants of grammar G and finding an optimal place for grammar G' within the configuration space, which makes the assumption that the positive and negative exemplars are located in easily separable clusters. If this assumption is met, then Grammar G' should be located closer to the cluster of positive variants than negative variants. However, what happens if not all of these variants are easily separable?

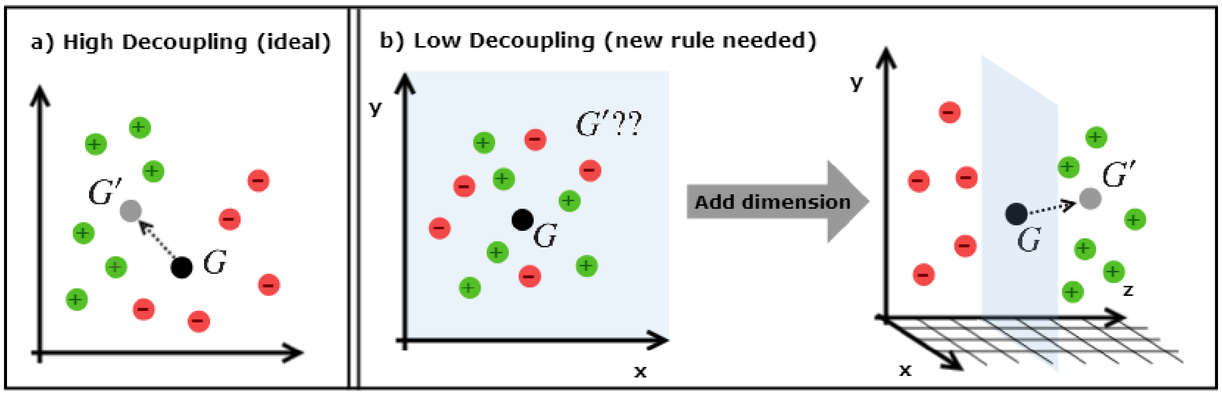

a) High Decoupling (ideal)
G'
G
b) Low Decoupling (new rule needed)
y
x
G'??
G
Add dimension
y
x
z
G
G'

Figure 5: Examples of possible configuration space outcomes. a) *High Decoupling:* the positive and negative variants are in easily separable clusters, and thus the direction that the properties of Grammar *G* need to be adjusted are apparent. b) *Low Decoupling:* the positive and negative variants are not easily separable, given the current dimension of the configuration space. This indicates that another dimension needs to be added, which is achieved by adding another rule (and thus another parameter) to Grammar *G*. With a well-formed new rule, these variants should become more easily separable.

If the positive and negative grammar variants are not easily separable, as shown in Figure 5b, then the "force" model will not work very well, because the forces from the positive and negative variants partially cancel each other out. A way to address this is to change the configuration space by changing the structure of grammar *G* itself. This would require the addition, deletion, and restructuring of rules. Adding a rule to grammar *G*, for example, would add another rule probability and thus another variable to *G*, increasing the dimension of grammar *G's* configuration space. Then, variants that were hard to separate before would hopefully be more easily separable in this new dimension. Figuring out which new rules to add, modify, or delete, however, is a tricky problem, and one that requires much further study to address.

REFERENCES

[1] A. Martinović and L. Gool. Bayesian Grammar Learning for Inverse Procedural Modeling. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, June 2013.

[2] O. Teboul, I. Kokkinos, L. Simon, P. Koutsourakis, and N. Paragios. Shape Grammar Parsing via Reinforcement Learning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition,* 2011.

[3] J. Talton, L. Yang, R. Kumar, M. Lim, N. Goodman, and R. Měch. Learning Design Patterns with Bayesian Grammar Induction. In *ACM Symposium on User Interface Software and Technology*, 2012.

[4] K. Vanlehn and W. Ball. A Version Space Approach to Learning Context-free Grammars. In *Machine Learning*, 2(1), 1987.

[5] J. Beirão, J. Duarte, and R. Stouffs. An Urban Grammar for Praia: Towards Generic Shape Grammars for Urban Design. In *Conference on Education and Research in Computer Aided Architectural Design in Europe*, 2009.

[6] SCIgen - An Automatic CS Paper Generator. Retrieved March 3, 2014, from Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology: http://pdos.csail.mit.edu/scigen/.

[7] A. Hwa. Supervised Grammar Induction Using Training Data with Limited Constituent Information. In *ACM Computing Research Repository,* 1999.

[8] D. Bertsimas and J. Tsitsiklis. Simulated Annealing. In *Statistical Science,* 8(1), 1993.

[9] M. Christensen. Structural Synthesis using a Context Free Design Grammar Approach. In *Generative Art International Conference*, 2009.