A rule-extraction algorithm for describing perceptual parametric subspaces in algorithmic composition systems

Iván PAZ [1], Àngela NEBOT, Enrique ROMERO, Francisco MUGICA
[a] *Llenguatges i Sistemes Informàtics, Soft Computing research group, Barcelona Tech.*

Abstract. The present work describes an algorithm for human perceptual exploration of algorithmic composition systems' parameter spaces. It works by considering the values of the system parameters, together with the perceptual user evaluation of the system output corresponding to such parameter configuration, as input-output relations. Then, the algorithm iteratively searches in the data to find combinations of parameters with the same classification (evaluation) and differing only in the values of one parameter. If the absolute difference between them is less than a pre-established threshold, the combinations are compressed into one rule. The rules have the if-then standard form. As the parameters commonly express different physical dimensions (like amplitude and frequency), the threshold is set independently for each one. Finally, an example applied to the parameter space of a band limited impulse oscillator is presented.

Keywords. Algorithmic composition, Rule extraction, Parametric systems, Perceptual exploration

Introduction

Algorithmic composition is the process of creating musical material by means of formal methods [1,2], ranging from sound synthesis and automated composition of pieces, to signal processors, etc. As a consequence of their design, algorithmic composition systems are (explicitly or implicitly) described in terms of parameters. The possible parameter combinations constitute the "lexicon" of the system. Therefore, their exploration plays a key role in learning the system's capabilities. A common practice of the performer/composer is to choose, out of all the possibilities, specific parameter configurations for particular results. The process of finding such configurations is, in many cases, performed by try and error or by heuristic system information (based on previous experience). Nonetheless, some methodologies for finding sets of parameters that successfully produce desired perceptual entities have been proposed. For example [3,4,5] applied interactive evolution [6], which uses human evaluation as the fitness function of a genetic algorithm for system parameter optimization. In [3], this technique was applied to sound synthesis and pattern generation; in [4,5], for searching successful sets of arguments controlling algorithmic routines for audio cut procedures. These results, i.e. the possibility to create methodologies for finding sets of parameters that create effective aural results for a listener, have inspired the proposed algorithm. It is designed to be able to work with data acquired during the auditory exploration process, where the parameter combinations used are selected and classified according to a particular purpose. An early version of the algorithm, along with some ideas for the exploration and processing methodologies, as well as a generative application of the results can be found at [7]. As the systems outputs are intended to produce an aesthetic experience on humans, audition plays a central role during the exploration. Each combination of parameters is seen as representing a point in the parametric space. The system outputs associated with the parameter combinations are perceptually classified by the user. After the classification, each pair (parameters, class) can be seen as an input/output relation, in the sense that this combination of parameters is associated with a particular output label representing a perceptual property. Such relations can be compacted to get interpretable rules, describing the knowledge contained in the instances, by finding regularities in the data. The rules can be used to "travel" within the perceptual predefined spaces (or classifications), allowing to produce variability in the outputs without stepping out of the described classes. The special interest in working with rules relays on its

[1] E-mail: {ivanpaz, angela, eromero, fmugica}@cs.upc.edu

interpretability. In contrast to subsymbolic approaches (like neural net classifiers), if-then rules are human-readable, which makes them especially attractive for applications in the context of computer music.

In the present text an algorithm for human perceptual exploration of algorithmic composition systems' parameter spaces is presented. It performs iterative searches in the data, to find combinations of parameters with the same classification and differing only in the values of one parameter. If the absolute difference between the values is less than a pre-established threshold, the combinations are compressed into one rule. There can be a different threshold for each parameter. The use of such thresholds allow to form subsets within the sets of parameter values. Different thresholds are introduced to handle the fact that the parameters usually represent different physical system's dimensions (either signals' characteristics or values for signal processing). Therefore, they are measured in different units and scales. Two main motivations suggests the grouping of some data into subsets: (1) Each variable can have lots of values (infinity in the limit of continuous variables). (2) The values may have structure (i.e., are ordered, there are areas that can be equivalent from a musical point of view, for example regions of beating, octaves, etc). The rest of the paper is structured as follows. Section 1 discusses some considerations as well as the algorithm. Section 2 presents an example of the rule extraction process for data taken from different users with a Band limited impulse oscillator. Finally, Section 3 presents the conclusions as well as ongoing and further work.

1. Considerations and design of the algorithm

The range of architectures (systems) used for algorithmic composition is extremely wide, and so the different dimensions of the parameters that control it. Some of them describe physical dimensions (like frequency), and others represent non-physical dimensions, like the seed of a random number generator. The architectures range from sound synthesis (e.g AM, FM, or granular) and audio signal processors (like filters or reverbs), to sequencers (of melodies and rhythms), as well as those for the automatic composition of small structures or entire pieces. As a consequence, the way in which our perception changes as a response to changes in the system parameters is highly variant from one system to another. While for some of these the variation of our perception can be almost homogeneous or linear, in some cases (for example in the case of FM synthesis) the desired aural characteristic may be scattered in subspaces, and sometimes even in specific points. The algorithm was designed embracing this condition in order to develop a flexible tool, applicable to a wide variety of systems.

1.1. Acceptable difference for "close values"

As said, since the system parameters represent different dimensions, the amount of variation that produces a substantial perceptual change can be different for each parameter. In the algorithm, the user can define how close the values within the rules will be for each parameter by selecting the distance at which two values are considered "close enough" for being grouped. These values are introduced by the user at the **acceptableDifference** vector. Thus the obtained rules can be handled more easily, for example, when used to construct upper hierarchical levels.

1.2. Algorithm

The inputs for the algorithm are the **acceptableDifference**: Array containing, for each parameter, the distance (threshold) at which two values can be considered "close" to be grouped into one set. And **data**: $\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_m\}$, where each $\mathbf{a}_i = ((x_{i1}, x_{i2}, ..., x_{in}), y_i)$. In this notation x_{ij} denotes the value of the jth parameter of the ith combination explored, and y_i denotes the user classification. Index i satisfies that $1 \leq i \leq m$ and n is the number of parameters. The algorithm pseudocode is shown below. It is currently implemented in the SuperCollider programming language [8]. As the compression process depends on the order of the parameters, the algorithm starts by calculating the permutations of their indexes, to perform all the possible ways of compression. Then, for each index it excludes the corresponding column from the data and searches for identical rows. Then, it takes the **values** of the excluded column for the selected rows. These values are sorted from min to max. The threshold is the entrance of the **acceptableDifference** with the index of the excluded column. The **values** are used to form subsets. Each subset is composed by the subsequent values whose absolute difference is less or equal that the threshold (distance). The algorithm creates one rule for each subset. The rules have the values of the selected rows in all the parameters and an array with all the values of the subset in the index of the excluded column. For example, if the acceptable difference for the parameter in turn is 0.2 and we have **values** = [0.1, 0.3, 0.5, 0.8, 0.9, 1.1] we will have the sets: [0.1, 0.3, 0.5] and [0.8, 0.9, 1.1]. The new rules created, if we have excluded column k, will be : $x_1, x_2, ..., x_{k-1}, [0.1, 0.3, 0.5], x_{k+1}, ...x_n, y_k$, and $x_1, x_2, ..., x_{k-1}, [0.8, 0.9, 1.1], x_{k+1}, ...x_n, y_k$. Finally the selected rows are eliminated and the rules are added to **data**.

Algorithm pseudocode

Algorithm inputs:
data: Array containing the (input, output) entries. Repeated entries are not allowed.
acceptableDifference: Array containing, for each parameter, the distance at which two values of that parameter can be considered "close" to be grouped into one set.

1. permutations = create an array with the possible permutations of the input parameter indexes (excluding the output).

2. For each permutation in permutations:
    temporal_data = data
    For each j in permutation (j from 0 to # parameters):
        Exclude column permutation[j] from temporal_data
        For each row (in temporal_data):
            Look for identical rows
            collect identical rows values located at the excluded column
            create subsets and new rules with the collected values [*1]
            eliminate the identical rows and add the new rules to temporal_data
    add temporal_data into sets_of_rules
return sets_of_rules

[*1] subroutine for create sets and rules
subroutine inputs:
distance = acceptableDifference[j]
values: array with the values of the identical rows located at the excluded column
row: current row of temporal_data being compared

```
array = [ ] // array to store the created sets
1. sort values form min to max
2. For (i=0, length(values) - 1){
    append values[i] to temporal_array
    if absolut_value(values[i] - values[i+1] )<= distance:
        append values[i+1] to temporal_array
    else:
        if length(temporal_array)>1:
            append temporal_array to array
}
return array
//create a new rule for each set
rules = [ ] //array to store the rules
for each element of array:
    new rule = row
    row[j] = element
    append row to rules
return rules
```

2. Rule extraction process example

2.1. Band limited impulse oscillator

In this section an example of the rule extraction process is presented. We worked with a single Band limited impulse oscillator “Blip” that produces a fundamental frequency to which a certain number of upper harmonics, all with equal amplitude, are added [9]. It is available as a unit generator in SuperCollider [8]. It has three arguments. These

are: *freq*, the fundamental frequency, *numharm* which controls the number of upper harmonics added, and *amp*, defining the output's amplitude. The parameters can range in the following way: the fundamental frequency can take values from 0 Hz, with no audible result, to 20, 000 Hz which is the upper audible limit. The number of added harmonics range from 0, which leads to a pure tone frequency of the chosen fundamental, to any (theoretical) number of harmonics. However, considering the aliasing processes, this number is limited by the sample rate and the chosen fundamental frequency. Finally the signal amplitude is normalized between 0 and 1. Although this is a simple system, the different aural possibilities produced by its parameter combinations clearly define distinct perceptual subspaces [10]. For the experiment we worked with three of these perceptual properties. These comprised outputs perceived as rhythmic, rough, and pure-tone. For further reference of the Blip generator the reader is referred to [10]. Generally speaking, outputs perceived as rhythmic are characterized by low fundamental frequencies (less than 15 Hz) and high number of upper harmonics (more than 200 depending on the fundamental frequency) together with high amplitudes (> 0.5). Outputs perceived as rough have fundamental frequency values approximately between 15 and 35 Hz, and upper harmonics approximately above 5. Outputs perceived as pure-tones are characterized by frequencies with low number of upper harmonics (for example, less than 5). However, the considerations mentioned above are general and may vary according to the perception of each user as well as with the relative relations between the parameters.

2.2. Classification process

For the experiment, data from five different users were considered. The users were provided with a buttons and knobs interface for controlling the system parameters. They were asked to listen to the different parameter combinations, and classify the outputs in the corresponding class if they perceived it as belonging to one of these classes. The classes were assigned in the following way: Class1 contains the rhythmic outputs. Class 2 the rough ones, and Class 3 those perceived as pure-tone. Each user classified 50 different combinations.

2.3. Methodology

1. Set the acceptable difference (**acceptableDifference**) for each parameter. The selection of these values establishes the accepted amount of output variation allowed for each parameter (while keeping the others constant) for compress two (or more) instances in one. These quantities can be adjusted depending on the desired compactness (understood as the maximum possible difference among two adjacent values) of the created subsets. It is possible to try different values and choose the ones that produce the rules with best results. However, a common starting choice is to set these values equal to the absolute difference between the minimum and maximum values for each parameter.
2. Compress the data using the algorithm. The collected instances are compressed. As the algorithm produces a set of rules for each parameter permutation, a bag of rules containing all the different rules for those compressions is created.
3. Compress the bag of rules by using the algorithm. The bag or rules is compressed using the algorithm allowing the compression of rules created at different permutations.
4. Unify rules. The extracted rules are processed by the unification function (available at [11]). It searches and unifies the rules having the same values in all the parameters as well as in the classification, but with different subspaces in the remaining parameter. In this process the redundant rules are eliminated.

2.4. Obtained rules

For the experiment the acceptable difference was set, for each user's data, equal to the absolute difference between the minimum and maximum values for each parameter. The numerical means for all users are: freq 1999.8, numharm 10000, and amp 0.89. As the algorithm is intended for the structuring of the information without adding unseen cases, the obtained rules in the experiment share ranges and tendencies, although they can be different. For example, the frequency region producing rhythmic outputs is contained in the open interval (0, 10). In Table 1, the rules obtained after the compaction process for one of the analyzed cases are shown. The ranges values for each parameter for the obtained rules for the different data sets $\pm$ SD are presented in Table 2.

The results in Table 2 show the role of each parameter, as well as their interactions, in the construction of the perception. For example, in Class 1 the frequency and amplitude are contained in small ranges, while the number of upper harmonics, although is kept high, shows great variation. It is clear that, to have an output perceived as rhythmic, the frequency is the determinant parameter. It has to be less that 10 or 15 Hz, depending on the values of the number of upper harmonics and the amplitude. In Class 3 the frequency can variate, while the number of upper

Table 1. Final sets of rules, grouped by classes, obtained for one of the experiments. The rules express the possible values for frequency, number of harmonics, and amplitude, associated with one classification. For example [[1, 2], 400, 0.6, 1] should be read as: IF frequency is 1 OR 2 AND number of harmonics is 400 AND amplitude is 0.6 THEN class is 1.

Class 1, rhythmic freq / numharm / amp / class	**Class 2, rough** freq / numharm / amp / class	**Class 3, pure-tone** freq / numharm/ amp / class
[[1, 2], 400, 0.6, 1]	[[20, 25], 800, 0.1, 2]	[[50, 60], 5, 0.06, 3]
[1, [500, 10000], 0.9, 1]	[[25, 35], 200, 0.1, 2]	[[20, 30, 40], 0, 0.5, 3]
[2, [100, 200, 400, 10000], 0.6, 1]	[40, [40, 50, 60], 0.06, 2]	[[500, 1000], 0, 0.009, 3]
[1, [400, 10000], 0.6, 1]	[60, [30, 100], 0.06, 2]	[100, [0, 1], 0.1, 3]
[1, 10000, [0.6, 0.9], 1]	[25, [200, 800], 0.1, 2]	[500, [0, 2, 4], 0.01, 3]
[[0.2, 1, 2, 3, 4], 500, 0.9, 1]	[15, 400, [0.2, 0.3], 2]	[500, 0, [0.009, 0.01], 3]
[[1, 2, 3], 10000, 0.6, 1]	[25, 200, [0.06, 0.1], 2]	[500, [2, 4], 0.01, 3]
[[2, 4], [100, 200], 0.6, 1]	[[11, 12, 15, 18], 400, 0.2, 2]	

Table 2. Ranges of the values for each parameter for the obtained rules for the different data sets $\pm$ SD.

mean freq $\pm$ SD	mean numharm $\pm$ SD	mean amp $\pm$ SD
Class 1		
1.8875 $\pm$ 1.1265	3771.4285 $\pm$ 4819.4204	0.7 $\pm$ 0.15
Class 2		
25.0769 $\pm$ 13.5059	273.3333 $\pm$ 276.6794	0.128 $\pm$ 0.0795
Class 3		
300 $\pm$ 320.7802	1.6363 $\pm$ 1.9116	0.0998 $\pm$ 0.1798

harmonics remains low, given that every frequency is perceived as a pure tone without upper harmonics. In the rules for Class 2, the frequency remains in the rough window (approx. 15 to 35 Hz) while the number of upper harmonics is in the mid range.

3. Conclusions and further work

Throughout the text, the design and implementation of a rule-extraction algorithm, intended to describe perceptual subspaces within the parameter spaces of algorithmic composition systems, are presented. Its design was based on the restrictions imposed by the systems, as well as considering the exploration process during which the selection/classification of the aural material is performed. As discussed in Section 2.4, the obtained rules are interpretable instances, from which we can infer meaningful information about how the parameters and their interactions are related with our perceptual classification. In a specific example, the algorithm was used to extract rules representing the perceptual space of a band limited pulse oscillator. Three well defined perceptual properties were used. Although the information after the compression is only a smaller representation of the original data, it is clearly more interpretable than the original one. Also, the rules can be turned into generative systems, allowing the user to select combinations of parameters exhibiting the selected properties. A fine-tune of the rules can be made by experimenting different compression distances for each parameter. Furthermore, the rules can also be the starting point to build generative systems capable of suggesting new unheard material. For example, we can consider as valid all the values that are contained between the extreme values of the subsets (if any). While the algorithm can produce different rules from one user to another, since it was designed for only structuring the regularities in the input data, rules from the different users show trends in the aural regions found. Ongoing research includes methodologies to extend the rules to generate unheard examples (considering particular constrains for different type of systems), as well as the use of fuzzy techniques for managing non-crisp classification of the outputs.

References

[1] Nierhaus, G. 2009. “Algorithmic Composition. Paradigms of Automated Music Generation.” Springer Wien, New York.

[2] Fernández, J. and Vico. F. (2013) “AI Methods in Algorithmic Composition: A Comprehensive Survey.” Journal of Artificial Intelligence Research 48 (2013) 513-582.

[3] Dahlstedt, P. 2001 “Creating and exploring huge parameter spaces: interactive evolution as a tool for sound generation”. Proceedings of the International Computer Music Conference, Habana, Cuba.

[4] Collins, N. 2002. “Interactive Evolution of Breakbeat Cut Sequences”. Proceedings of Cybersonica, Institute of Contemporary Arts, London, June 5-7, 2002.
[5] Collins, N. 2002 “Experiments With a New Customisable Interactive Evolution Framework”, Organised Sound 7(3): pp 263-273. Cambridge University Press.
[6] Dawkins, R. 1986. The Blind Watchmaker, Essex: Longman Scientific and Technical.
[7] Paz, I. Nebot, À. Romero, E. Francisco, M. and Vellido A. (2016) A methodological approach for algorithmic composition systems' parameter spaces aesthetic exploration. Proceedings of the IEEE Word Conference on Computational Intelligence 24-29 July 2016, Vancouver, Canada.
[8] SuperCollider programming language http://supercollider.github.io
[9] Blip, Band limited impulse oscillator. https://github.com/supercollider/supercollider/blob/master/HelpSource/Classes/Blip.schelp
[10] Roads, C. 2001. Sound Composition with Pulsars. Journal of Audio Engeeniering Society Vol 49. No 3. March 2001.
[11] Rohrhuber, J and Paz I. (2015) Exploring-parameters GitHub repository https://github.com/musikinformatik/exploring-parameters