A methodological approach for algorithmic composition systems' parameter spaces aesthetic exploration

Iván Paz, Àngela Nebot, Enrique Romero, Francisco Mugica and Alfredo Vellido
Soft Computing Research Group
Computer Science Department
Universitat Politècnica de Catalunya - Barcelona Tech
C. Jordi Girona 1-3, 08034 Barcelona, Spain
{ivanpaz, angela, eromero, fmugica, avellido}@cs.upc.edu

***Abstract*—Algorithmic composition is the process of creating musical material by means of formal methods. As a consequence of its design, algorithmic composition systems are (explicitly or implicitly) described in terms of parameters. Thus, parameter space exploration plays a key role in learning the system's capabilities. However, this task has surprisingly received little attention. Two main problems appear when working on exploring parameter spaces. First, depending on the system, the dimension of the output space maybe very large. And second, the produced changes on the human perception of the outputs, as a response to changes on the parameters, could be highly non-linear. The present work describes a methodology for the human perceptual (or aesthetic) exploration of generative systems' parameter spaces. As the systems' outputs are intended to produce an aesthetic experience on humans, audition plays a central role in the process. The methodology starts from a set of parameter combinations which are perceptually evaluated by the user. The sampling process of such combinations depends on the system under study and possible on heuristic considerations. The evaluated set is processed by a compaction algorithm able to generate linguistic rules describing the distinct perceptions (classes) of the user evaluation. The semantic level of the extracted rules allows for interpretability, while showing great potential in describing high and low-level musical entities. Previous work and the experiments that lead to the current methodology and algorithm are described in detail. As the resulting rules represent discrete points in the parameter space, further possible extensions for interpolation between points are also discussed. Finally, some practical implementations are presented together with paths of current and further research.**

Keywords—Algorithmic composition, rule extraction, parameter spaces exploration, fuzzy inductive reasoning.

I. Introduction

Algorithmic composition is the process of creating musical structures by using formal methods (e.g., formal grammars, statistical models, cellular automata, or mixed ad hoc combinations). The algorithms can be used to generate either the complete composition or some parts, and at different hierarchical levels. For a more detailed review on algorithmic composition the reader is referred to [1] and [2]. The algorithmic systems utilize parameters controlling the generation and manipulation of the musical data. These can be implicitly or explicitly established in the system. The parametric structure is a consequence of the sound and music parameterizations used within the semantic of the algorithms, or (at least) in the mapping of the produced data into the sonic (wave form) or symbolic outputs (midi) of the system (see, for example, [3]). Therefore, the exploration of different combinations of parameters is a key aspect in learning the system's capabilities. However, despite its relevance, little research has been carried out in parameter spaces' exploration. This is mainly due to the difficulties that arise when working with parameter spaces from a perceptual perspective. These are: (1) Depending on the algorithms, the dimension of the output space could be of intractable size [4, 5]; (2) as the outputs of the systems are intended to produce an aesthetic impression in the listener, human audition takes a central role in the data acquisition process, being in many cases the system's bottleneck; (3) the changes on human perception of the outputs produced as a response to changes on the parameters could be highly non-linear. Therefore, it is difficult to use interpolation techniques to infer the quality among known points in the parameter space to extend the system to unheard cases.

The present work describes a methodology for the human perceptual exploration of generative systems' parameter spaces. It is intended for the modeling of low and high-level musical entities. The methodology is based on a rule-extraction algorithm that starts from a pattern of input/output relationships and performs an iterative compaction process to obtain interpretable and flexible rules. The compaction methodology avoids the problem of variability of human perception of the outputs as it is restricted (in its basic process) to compact the information received.

The problem of finding sets of parameters that successfully describe low and high-level perceptual entities when used in an algorithmic composition system has been addressed by

Dahlstedt in [5] and Collins in [4, 6]. Both applied interactive evolution [7], which uses human evaluation as the fitness function of a genetic algorithm, for system parameter optimization. In the first case, this technique was applied to sound synthesis and pattern generation algorithms; in the second case, for searching successful sets of arguments controlling algorithmic routines for audio cut procedures. Our methodology is built upon these foundations, i.e. on the possibility of finding sets of parameters for algorithmic systems that create effective aural results for a listener.

In our methodology, each successful combination of parameters represents a point in the space of possible combinations. After user classification, it 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 means of compaction algorithms such as Linguistic Rules in Fuzzy Inductive Reasoning methodology (LR-FIR) [8, 12]. Our particular interest in working with linguistic rules lays on its interpretability. Linguistic rules, in contrast with subsymbolic approaches (like neural net classifiers), are human-readable information, which makes them especially attractive for applications in the context of computer music.

The rest of the paper is structured as follows: Section 2 carefully describes the algorithm and the motivation of its functionalities. Section 3 presents a practical rule extraction example. Section 4 discusses the algorithm limitations. Finally, Section 5 presents the conclusions and possibilities for future work.

II. The Parameter Space Perceptual Exploration Algorithm

The general structure of the algorithm is shown in Figure 1. The methodology starts with an interactive sampling process (1) from which the different combinations of parameters are taken one at a time. As previously mentioned, the parameter combinations are valid configurations of the algorithmic system under study. Each set of parameters (instance) is presented to the user for its perceptual evaluation (2). The evaluation looks for a high-level musical characteristic, and classifies each example in one particular class depending on the perceived presence of such characteristic. For our purposes, high-level musical features (or characteristics) are those that are derived from the combination of the lower level information provided in the input data. In this case, from parameter combinations.

After the human evaluation process, the set of instances (evaluated instances (3) in Figure 1) are passed to the compaction algorithm (4). The module “strict compaction-all permutations” performs the compaction process (described below) in every permutation of the input data. This is the module that compresses the information contained in the instances by finding the parameters that do not determine the class of the system's output as long as we have particular values in the other parameters. The functionalities of this module are described below.

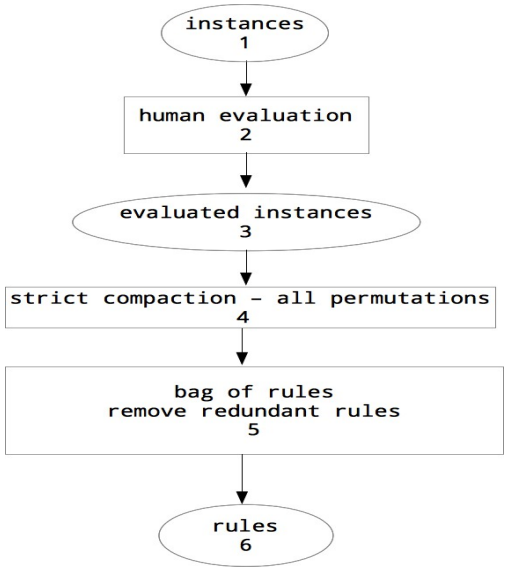

Fig. 1. Schematic representation of the parameter space perceptual exploration algorithm

Finally, the module “bag of rules, remove redundant rules” (5) grabs all the created rules from the previous process, and removes those that are redundant (rules that are contained in another -more general- rule). The results are presented to the user (6) through an interface that allows to select the different rules and use them directly in live performance.

A. *Strict compaction-all permutations*

1) *Strict compaction*

The strict compaction all permutations module takes the set of evaluated instances as input. Then, it performs an iterative process searching (in the given order) for each parameter (P_i), sets of instances containing all the possible values of that parameter, and sharing the same values in the rest of the parameters and in the evaluation. In that case, we can consider that the parameter P_i does not determine the class of the evaluation, given that, as long as all the other parameters have those specific values, the output will be the same. Then, the set of instances is compacted into one rule having a "-1" in the place of P_i, indicating that this parameter was compressed (or dismissed). A simple example of this process is shown in Tables 1 and 2. Parameters P_1 and P_2 can take {2} and {3} discrete values, respectively, and each set is evaluated in one of two classes {2}, which are considered the output of the system. In this

case, let us suppose that all instances were evaluated as 1. Throughout this paper the sets of parameters that do not contain "-1" are called instances, and those that contain "-1s" are referred to as rules.

TABLE 1. SIMPLE EXAMPLE OF THE COMPACTION PROCESS. FOR SIMPLICITY WE ARE ASSUMING THAT ALL SETS OF PARAMETERS (INSTANCES) WERE CLASSIFIED OR EVALUATED AS CLASS 1

Num. Instance	P_1 {2}	P_2 {3}	EVALUATION {2}
1	1	1	1
2	2	1	1
3	1	2	1
4	1	3	1

Considering P_1, instances 1 and 2 form a set containing all possible values of P_1, and sharing the same values in P_2 and in the evaluation. Then, the set is compacted into rule -1 1 1. After that, it is not possible to compact any other set of instances considering the P_1 parameter.
The set of rules and instances is now written as:

TABLE 2. SET OF RULES AND INSTANCES AFTER THE COMPACTION PROCESS OVER THE FIRST PREMISE

Num. Instance	P_1 {2}	P_2 {3}	EVALUATION {2}
1-2	-1	1	1
3	1	2	1
4	1	3	1

When we consider P_2 in the resulting set, we can see that instances 3 and 4 share values in P_1 and the evaluation, however, rule 1-2 does not share the same value in P_1. Then, although all the possible values of P_2 are present in the set, this parameter cannot be compacted.

For compacting a parameter, all variables must be equal, including the -1; otherwise, unheard cases are included in the rules. To illustrate this, let us consider the following: If we assumed that the -1, as it contains all the values of the parameter, can be used for compacting the instances into one rule. Under the only condition that the set of all possible values described by the rule does not create contradictions with the original set of instances. Considering contradictions those cases for which two instances have the same values in the parameters and different evaluation. In this case, if we compact the second premise we end with the rule -1 -1 1. That describes (or can generate) the cases listed in Table 3:

TABLE 3. SET OF ALL POSSIBLE CONFIGURATIONS THAT CAN BE CREATED WITH THE RULE -1 -1 1

Num. Instance	P_1 {2}	P_2 {3}	EVALUATION {2}
1	1	1	1
2	2	1	1
3	1	2	1
4	2	2	1
5	1	3	1
6	2	3	1

None of these instances creates contradictions with the original data (showed in Table 1). However, if we follow this criterion we will include the unheard cases 2 2 1 and 2 3 1, i.e., we will be assuming that the combinations 2 2 and 2 3 will produce outputs evaluated as "1". However, when working within a sonic context, as discussed below, it is common for these cases to be perceived as another class (e.g., as class 2) reducing the precision of the rules. Therefore, we only allow the algorithm to compact two rules or instances if all the values in the parameters, including the "-1s", are equal. We call this condition "strict compaction".

2) All permutations

As pointed out before, it should be noted that the resulting set depends on the placing order of the parameters. For example, if we interchange the order of the parameters (P_1 and P_2) in the training set (Table 4).

TABLE 4. ORIGINAL SET OF INSTANCES INTERCHANGING THE ORDER OF P_1 AND P_2

Num. Instance	P_1 {2}	P_2 {3}	EVALUATION {2}
1	1	1	1
2	1	2	1
3	2	1	1
4	3	1	1

The resulting set applying the basic strict compaction is (Table 5):

TABLE 5. SET OF RULES AND INSTANCES SHOWED IN TABLE IV AFTER APPLYING STRICT COMPACTION

Num. Instance	P_1 {2}	P_2 {3}	EVALUATION {2}
1-3-4	-1	1	1
2	1	2	1

The process of interchanging parameters is equivalent to change the order in which we consider the parameters to perform the search of possible sets for compaction (see Tables 2 and 5). To refer to the different sets of rules obtained in each case, we will name the different orders of parameters by its permutation number, or explicitly by writing the permutation. In a formal way, we can say that, with our original data, we have two compaction orders (name [1,2] and [2,1]) depending on which parameter we consider first for searching the possible sets for compaction. Then, compacting first in order [1,2] and then in order [2,1], our example results in the following sets:

TABLE 6. SET OF RULES AND INSTANCES AFTER THE STRICT COMPACTION PROCESS FOLLOWING THE COMPACTION ORDER OF THE PERMUTATION [1,2]

Num. Instance	P_1 {2}	P_2 {3}	EVALUATION {2}
1-2	-1	1	1
3	1	2	1
4	1	3	1

TABLE 7. SET OF RULES AND INSTANCES AFTER THE STRICT COMPACTION PROCESS FOLLOWING THE COMPACTION ORDER OF THE PERMUTATION [2,1]

Num. Instance	P_1 {2}	P_2 {3}	EVALUATION {2}
1-3-4	1	-1	1
2	2	1	1

We are assuming that parameter combinations represent discrete points in the parameter space. Then, what is important for the aesthetic impression is the value of the parameters and not the order in which they are enumerated. So, $P_1 = 2$ and $P_2 = 1$ will be aesthetically perceived the same as $P_2 = 1$ and $P_1 = 2$, given that this is only a way to enumerate the values of the parameter setting in the algorithmic system. Then, we are saying that, as long as the composition system has the same values in the parameters, it will produce the same aural impression on the listener. It is important to note that the order in which the sets of parameters are presented to the listener could change the evaluation of each set, given that our ear and brain use past experience, or references, to "classify" the material. Therefore, the order in which the instances are auditioned should be taken into account during the sampling process.

If we look at the rules of Tables 6 and 7, it can be seen that the rules describe the same data (the information contained in Table 1) in different ways. While for some applications we can consider any set as the result presented to the user (sometimes the small set is preferred), for our purposes we wanted to keep all the sets as valid descriptions of the original information, and later decide how to use the different sets of rules. There are many reasons to do that. For example, if we classify each instance assigning the part of the piece in which the setting can be used as output class, e.g. part A, part B, part C, we can see the -1 as free parameters, i.e. parameters that can be changed to play or add variability to the part without stepping out of the desired part. In other words, we want to have the greatest palette of possibilities first, and later take decisions on this set. For that reason, the algorithm starts creating all the possible permutations in the input data, and it then applies strict compaction to each set. Therefore, the module is called "strict compaction-all permutations" and returns all rule sets of the different permutations compacted under the strict compaction condition.

We implemented this module in the SuperCollider programming language [11], which allows exploring and playing with the obtained set of rules. This implementation is available from the GitHub repository [13].

It is worth stressing that this approach works for algorithmic systems that, for the same parameter combination, exhibit consistent aural results, excluding, for example, systems that include complete random processes in the generation of the data in such a way that it is not possible to establish an aural correlation among the outputs and the parameter setting.

B. Bag of rules and removing redundant rules

From the different possible compactions resulting from the strict compaction–all permutations a "Bag of rules" is chosen. It consists in selecting all the rules, i.e. all the instances in which at least one parameter is a free parameter (having a -1), of each permutation. Then, these rules are analyzed and the redundant rules are eliminated. Redundant rules are those that are contained in a rule with greater number of free parameters. In the following section, a step-by-step example of the methodology is presented.

III. EXAMPLE: PARAMETER EXPLORATION OF A 16-STEPS GARAGE BEAT GENERATOR

As an example of the methodology, we present a quick parameter exploration of an algorithmic system intended to produce 16-steps beat patterns in the context of garage music. It is a simplified version of an algorithm presented in [10]. The implementation of the algorithm, together with a graphical user interface performing the evaluation and rule extraction processes, can be found in [14].

The algorithmic composer produces 16-steps beat sequences in the context of garage music for three instruments, i.e. kick, snare, and hi-hat, represented in the algorithm with values 1, 2, and 3 respectively. The silence is represented by the number 4. The algorithm determines which instrument plays at each one of the 16 places in order to produce a consistent style with some variability. The sequences have 11 unchanged variables, which define their constant part, and 5 changing parameters, which produce the different possible patterns. These parameters can also take different values, i.e. $P_1\{1,4\}$, $P_2\{1,4\}$, $P_3\{1,2,3,4\}$, $P_4\{2,4\}$ and $P_5\{1,2,3,4\}$. This information is detailed and described in [10]. Then, the

algorithmic system is able to produce different pattern variations of the musical style.

A. Sampling process and evaluation

The sampling process of the implementation has two modalities: random and manual. The random mode creates a random pattern out of all the possible patterns in the space. The manual mode allows the user to manually select the value for each parameter. In the experiments, we found that the random mode is useful when the space of parameters is small. For higher-dimensional spaces it is better to start from a seed in the manual mode and explore the space in the different directions, performing gradual changes in the parameters and returning to the original seed after exploring each parameter. For each parameters combination, the user perceptual evaluation is saved. The system allows having as much output classes as desired. Each classification corresponds to a specific aural property of the outputs selected by the user. Figure 2 shows the user graphic interface developed.

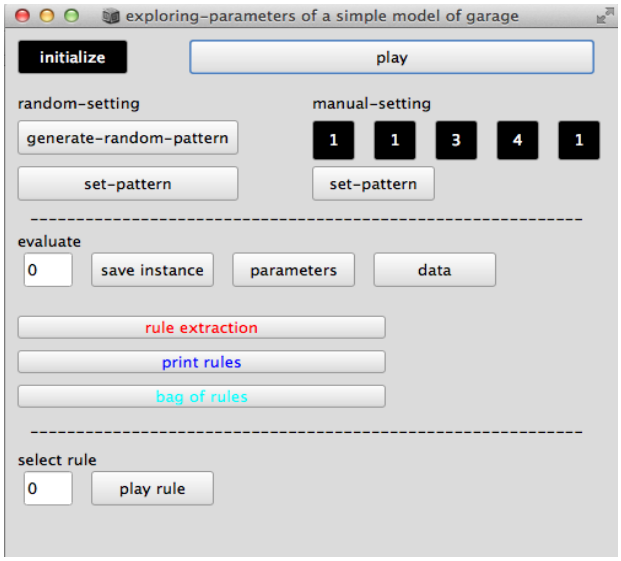

Fig. 2. Image of the graphical user interface developed for the sampling and rule extraction processes. The SuperCollider code is available at [14]

1) Data

For a quick and open exploration 28 different parameter combinations were selected by using the manual mode; they are listed in Table 8. The exploration started from a seed, chosen by using heuristic information of the algorithmic system. It is identified by the legend "original seed" in Table 8.

TABLE 8. PARAMETER COMBINATION OBTAINED USING THE MANUAL MODE OF THE GRAPHICAL USER INTERFACE

Num.	P_1	P_2	P_3	P_4	P_5	EVALUA
Instance						TION
1	1	1	1	2	1	1 ORIGINAL SEED
2	4	1	1	2	1	1
3	1	4	1	2	1	1
4	1	1	1	4	1	1
5	1	1	1	4	4	1
6	1	1	1	4	2	1
7	1	1	1	4	3	1
8	1	1	1	4	2	1
9	1	1	2	4	2	2
10	1	1	2	2	2	2
11	1	1	3	2	2	2
12	1	1	4	2	2	2
13	1	1	1	2	2	2
14	4	4	1	2	1	2
15	4	4	1	4	1	2
16	1	4	1	4	1	1
17	4	4	1	4	1	2
18	4	1	1	4	1	1
19	4	1	2	4	1	2
20	4	1	3	4	1	2
21	4	1	4	4	1	2
22	4	4	2	4	1	2
23	4	4	2	2	1	2
24	4	4	2	4	1	2
25	4	4	2	2	2	2
26	4	4	2	2	3	2
27	4	4	2	2	4	2
28	4	4	2	2	1	2

B. Strict compaction-all permutations

After the data acquisition process, the compaction module creates the rules obtained with the strict compaction for the permutations in the 5 variables (120 permutations). From all these sets, the ones that are unique, i.e., the ones that are not repeated, are selected. In this case, there are 41 different sets of rules. They are listed in the example described in [14].

C. Bag of rules and removal of redundant rules

With the 41 sets of rules obtained after the strict compaction-all permutations process, the bag of rules is built. Then, the redundant rules are eliminated. The resulting set is shown in Table 9.

TABLE 9. BAG OF RULES AFTER REMOVING REDUNDANT RULES

Num. Instance	P_1	P_2	P_3	P_4	P_5	EVALUATION

1	-1	1	1	-1	1	1
2	1	1	1	4	-1	1
3	1	-1	1	-1	1	1
4	4	-1	2	4	1	2
5	1	1	-1	2	2	2
6	1	1	2	-1	2	2
7	4	4	2	-1	1	2
8	4	4	1	-1	1	2
9	4	4	2	2	-1	2

There are free parameters for four out of the five possible parameters in the rules describing the “class 1”, the same goes for class 2. Notice that this number (the number of free parameters in the bag of rules) is always greater or equal that the number of free parameters of the independent rule sets. It is clear that, given that as we are considering all the possible compactions, this number has to be greater than or equal to the number of free parameters of each of the permutations. In other words, the bag of rules is the set with the greatest number of free parameters. Also, as a consequence of its construction, it contains the more general rules. Then, it is the smallest representation of the most general description.

The bag of rules can also be analyzed to infer or derive knowledge about how the user is correlating the input parameters with the perceptual space. In Table 9, for example, it can be seen that all rules describing class 1, have “1” at P_3. In contrast, only 2 out of the six rules describing class 2 have a “1” at P_3. While this is not conclusive information, because it is clearly related with the value of the other parameters, it can be used to create further extensions for the algorithm, and to guide the rest of the space exploration.

Within the analysis of the rules, different levels of musical information can be extracted. For this particular case, patterns for a specific part of a piece, with highly "rhythmic" content, were chosen. Class 1 describes the patterns that can be used in that part, and class 2 those that cannot. In the rules reported in Table 9 the presence of different instruments in the different classes can be seen. For example, there is only one "4" in the rules describing class 1, and eight in rules describing the class 2. This information can be used to build a system's heuristic. For example, we can perform a new exploration beginning from a new seed using this information. We can also see that P4 parameter is the increased presence of "-1". Following this, the rules can also be analyzed at a higher level to derive new sets of more general or directed rules.

It can be seen that the rules describe the information present in the original set of instances (Table 8) and that they cannot generate any instance out of it. In that sense, the rules represent a close system representing the original information (close system refers to the fact that no “instance” out of the instances in the training set can be generated with the rules.). Given such property, the precision of the rules is 100%. We looked for this, to be able to place, in two separate processes, the strict description of space (from which we can analyze the relations among the parameters and the perception) and the methods for going into the unexplored space. The rules were tested into live performance, giving the expected results. Also, the perceived contrast between the class separations respects the user's selection of the input data.

IV. Algorithm limitations

Recapitulating, the current limitations of the algorithm are the following:

1) Consistent input/output relation

As previously mentioned, the application of the method is restricted to algorithmic systems showing consistent aural results for the same parameter combination, therefore excluding systems in which it is not possible to establish an aural correlation among the outputs and the parameter setting. In such cases, although the numeric construction of the rules is possible, when these are used, the results will not be perceived as the evaluation.

2) Permutations

The second limitation is related to the computational cost of perform all possible permutations. Depending on the machine and the amount of data, the running time of the algorithm varies. For practical live performance applications the system has been tested with a maximum of 8 parameters.

3) Human bottleneck

As previously stated, audition plays a central role in the methodology as the outputs produced by the algorithmic composition systems are intended to produce an aesthetic experience. The audition process is clearly a “human bottleneck”, and a system limitation, in the sense that is it not possible to cover huge parameter spaces, or to have big training sets. Also, the evaluation has to be consistent (otherwise, the rules could be perceived as chaotic).

4) “No surprises”

As the constructed rules represent a close system capturing the knowledge contained in the input data, no new patterns (out of the training set) are obtained. The system only allows us to organize the information of the explored space, representing it in a compact set of rules the regions of space that we had labeled. However, the shape of the rules allows us to establish new relations and to analyze how the labels of the space depend on the combinations of the parameters.

In further extensions, we will develop a module capable of suggesting new valid patterns.

V. Conclusions and further work

The obtained rules are able to represent low- and high-level musical entities . Also, the approach of seeing the compressed parameters as free parameters, thus using them for traveling within our perceptual predefined spaces, allows

the generation of variability in the outputs without stepping out of the described classes. This approach leads to good results in a live performance context.

However, given that the rules form a close representation of the data in the sense that they do not create new patterns, the methodology has some restrictions. Nevertheless, it works out to explore the parameter space of the algorithmic system creating subspaces that can be accessed later throughout the rules. But again, these rules are restricted to the explored places.

When working in computer music, we want the methodologies to help to extend the composer capacities. From this perspective, the system helps to create new relations among the parameters and the aural perception, but the inclusion of a module for extending the rules and to explore new places in the space appears necessary. Here is where the interpretability of the rules becomes important. As mentioned, the information extracted from the analysis of the rules can be used to guide the exploration of space helping the user to find new places close to those described by the rules. On the other hand, thinking differently, we can also use this information to analyze, for example, "distant spots" away from the ones described by the rules. Again, the obtained information is only the starting point to avoid a blind exploration of space.

Considering the discussion presented above, several lines for further work are proposed:

The current interface provides two scanning modes, random and manual. While the random model is quite inefficient (it requires many points), the manual mode requires heuristic considerations and some degree of expertise in the algorithm used. For that reason, we are currently working in a module that takes the user through an efficient way to explore the space.

The second line for future work includes an algorithm for analyzing the rules contained in the bag, allowing the extraction of more general rules from this point and to guide a further exploration of the space.

REFERENCES

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

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

[3] H. Taube. 2013. Notes form the Metalevel: An Introduction to Computer Composition. Hoboken: Taylor and Francis.

[4] N. Collins, 2002. "Interactive Evolution of Breakbeat Cut Sequences". Proceedings of Cybersonica, Institute of Contemporary Arts, London, June 5-7, 2002.

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

[6] N. Collons 2002 "Experiments With a New Customisable Interactive Evolution Framework", Organised Sound 7(3): pp 263-273. Copyright © Cambridge University Press.

[7] R. Dawkins, 1986. The Blind Watchmaker, Essex: Longman Scientific and Technical.

[8] F. Castro, À. Nebot, and F. Mugica, 2011. "On the extraction of decision support rules from fuzzy predictive models". Applied Soft Computing, 11 (4), 3463-3475.

[9] F. Mugica, I. Paz, À. Nebot, E. Romero. 2015. "A Fuzzy Inductive Approach for Rule-Based Modelling of High Level Structures in Algorithmic Composition Systems". In proceedings of IEEE international conference on fuzzy systems.

[10] N. Collins, 2003 "Algorithmic Composition Methods for Breakbeat Science" ARiADA No.3 May 2003

[11] SuperCollider programming language http://supercollider.github.io/

[12] A. Nebot and F. Mugica. (2012). "Fuzzy Inductive Reasoning: a consolidated approach to data-driven construction of complex dynamical systems," International Journal of General Systems, vol. 41(7), pp. 645-665.

[13] Github repo of the "strict compaction -all permutations SuperCollider implementation https://github.com/musikinformatik/exploring-parameters

[14] Graphical user interface repository https://github.com/ivan-paz/exploring-parameters-simple-garage-beat-generator