Audio Engineering Society

Convention Paper

Presented at the 126th Convention
2009 May 7–10 Munich, Germany

Automatic Adjustment of Off-the-Shelf Reverberation Effects

Sebastian Heise[1], Michael Hlatky[1], and Jörn Loviscach[2]

[1] Hochschule Bremen (University of Applied Sciences), 28133 Bremen, Germany
Sebastian@h3e.eu, mhlatky@acm.org

[2] Fachhochschule Bielefeld (University of Applied Sciences), 33602 Bielefeld, Germany
jl@j3l7h.de

ABSTRACT

Virtually all effect units that process digital audio—software plug-ins as well as dedicated hardware—can be controlled digitally. This allows subjecting their settings to optimization processes. We demonstrate the automatic adaptation of reverberation plug-ins to given room impulse responses. This facilitates replacing computationally expensive convolution reverberation units with standard ones, which also are amenable to easier parameter tweaking after their overall setting has been adjusted through our method. We propose optimization strategies for this multi-dimensional non-linear problem that need no adaptation to the particularities of each effect unit, are sped up using multi-core processors and networked computers. The optimization process evaluates the difference between the actual response and the targeted response on the basis of psychoacoustic features. An acoustic comparison with effect parameter settings crafted by professional human operators indicates that the computationally optimized settings yield comparable or better results.

1. INTRODUCTION

The automatic adjustment of audio effect units is a difficult task due to product-specific algorithms. The controls offered often do not stick to a standard selection of parameters. And even when they do, the actual meaning of numeric settings of parameters such as "Reverberation Density" varies from unit to unit. To make automatic adjustments, one could try to learn the specific details of any unit at hand—a gargantuan task. We propose instead to analyze the effect of the unit on the audio signal and apply optimization routines that treat the unit as a black box.

The particular application considered in this paper concerns artificial reverberation. This bears practical relevance: Convolution reverberation units, which are based

on measured impulse responses, are known to reproduce the acoustics of a room with great fidelity. However, they often incur a huge computational load, are not easy to tweak, and may produce a sterile, static sound if the impulse response is kept strictly constant. The question arises what the best setting of a standard, filter-and-delay-type reverberation unit is to simulate a given room impulse response.

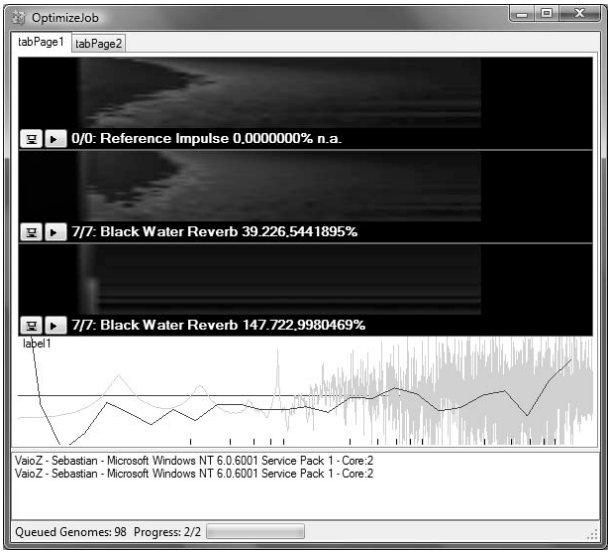

Fig. 1: The optimizer application adjusts the parameters of VST-based audio effects through comparing their impulse responses to a user-defined reference using a simplified psychoacoustic model.

Our prototype is based on the VST application-programming interface for audio effect plug-ins. The program aims at optimizing the settings of reverberation plug-ins to mimic a given room impulse response, see Fig. 1. The software can handle a virtually unlimited number of different plug-ins in parallel, finding an optimal one for the given task. In our tests, we used more than a dozen commercial or freely available reverberation plug-ins. The automated comparison employs the Euclidean distance of vectors of Mel-Frequency Cepstral Coefficients (MFCCs) [1].

We have implemented and studied four optimization strategies:

- Evolutionary [2]: A population of genotypes each of which represents the complete setting of a plug-in is subjected to mutation, crossover, and selection. This approach is easily parallelizable.

- Nelder-Mead [3]: The Nelder-Mead approach tries to fit a simplex body into the problem space by moving it or shrinking its size. This method is inherently non-parallel, does not deal well with discontinuities, and is easily trapped in local minima.

- Nelder-Mead with brute-force parallelization: All steps of the standard Nelder-Mead optimizer are computed in parallel, which leaves some of the computational results unused. Although this speeds up the computational time, the remaining restrictions of the non-parallel implementation still apply.

- Particle Swarm Optimization [4]: In this optimization strategy we used artificial swarm intelligence to find an optimal solution. This approach is also easily parallelizable.

The best performing plug-ins that we have tested lead to a clear progress after 25 to 50 fitness calculations in the optimization. However, depending on the difference of the plug-in's initial settings to the intended result, sometimes 100 steps are required to achieve acoustic similarity. One step requires about a quarter second on a single processor core. The largest part of this time is consumed by the computation of the spectral envelopes.

2. RELATED WORK

Being a linear and (theoretically) time-invariant phenomenon, the effect of reverberation on a signal is relatively easy to analyze, as opposed to the effect of, say, a dynamics compressor. This, too, makes reverberation a good starting point for automated optimization. The simpler task of automatic adjustment of equalizers with genetic algorithms has already been addressed in a multitude of research papers [5][6][7][8].

In previous work [9], one of us already used a genetic algorithm to find equalizer settings so that the magnitude response curve matches anchor points defined by the user. Since the genetic string used in that work is relatively short and comparing the curve to the settings

could be implemented effectively, it was possible to find proper settings nearly in real time.

In further previous work [10], one of us already dug into VST setup libraries and searched for statistical coherence. In this work we concentrated on automated setup of synthesizer effects, where usually a lot of parameters show coherence in their settings. Applied to VST reverberation effects, one could search for a relation between different parameters, say, decay and level, and simplify the interface in combining these parameters in one knob.

Extra et al [11] compared different reverberation devices in a listening test to judge their perceived sound quality. They also compiled a set of analysis tools that help compare and describe aspects and phenomena of reverberation. The result of their test was that none of the proposed objective analysis tools could be used to replace a listening test in judging the reverberation sound quality.

3. IMPLEMENTATION

The optimization software applies its specific search strategy to analyze the effect of the plug-in's settings through measuring the first seven seconds of its impulse response. Since there is no real-time audio output necessary, the plug-ins can compute this signal in a fraction of the actual playback duration. The development of the prototype had to address three major issues: how to compare different reverberation settings, how to distribute the workload onto several processor cores and/or several computers, and how to conduct the optimization.

3.1. Comparing Reverberation

The basis of all optimization strategies is formed by a distance function that estimates the similarity of a sound sample to a reference. Such a comparison cannot be undertaken on a sample-per-sample basis: What counts is the perceived similarity of the sound. In the application at hand we capitalize on the fact that most of the interesting impulse responses sound noisy. Only in extreme cases such as a small room with hard walls we will get harmonic structures through clearly isolated reflections. Noisy audio snippets can properly be described by MFCCs, which ignore the fine structure of the frequency spectrum. To capture the temporal development of the impulse response into account we extract the MFCC vectors over time (26 MFCCs, sampling frequency: 44,100 Hz, frame size: 2048 samples, overlap: 50%).

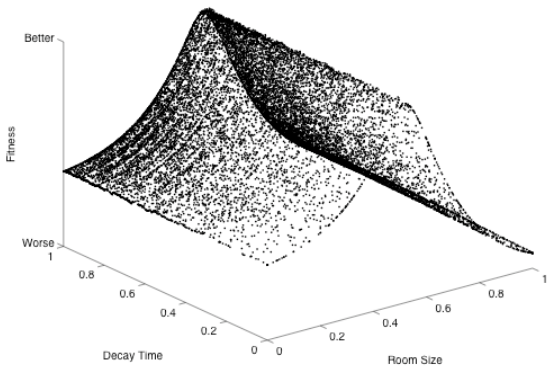

Fig. 2: 10,000 random settings of the Ambience[1] VST reverb's parameters "Room Size" and "Decay Time" versus the computed fitness against a given room impulse response. The fitness scale is arbitrary derived from the Euclidean distance between the MFCC vectors and therefore displayed without scale.

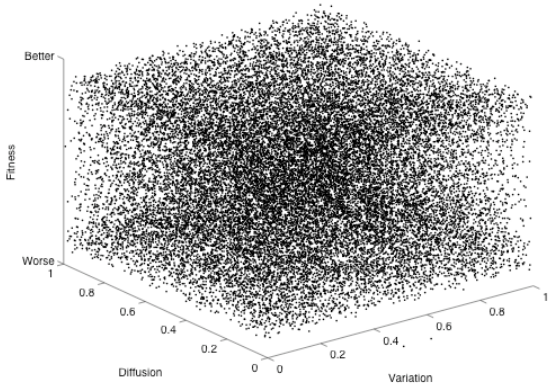

Fig. 3: 10,000 random settings of the Ambience plug-in's parameters "Variation" and "Diffusion" versus the computed fitness against a given room impulse response. The fitness scale is arbitrary derived from the Euclidean distance between the MFCC vectors and therefore displayed without scale.

[1] http://magnus.smartelectronix.com/

The distance between two MFCC vectors is computed in the straightforward Euclidean fashion. To compute the total distance between two audio snippets, we lay them side by side—capitalizing on the fact that we know the precise starting points—and sum all distances between the MFCC vectors in corresponding frames of both files.

Fig. 2. illustrates the behavior if the fitness function in a real case where we only tuned the parameters "Room Size" and "Decay Time" of the Ambience plug-in. Not all parameters of a plug-in have the same acoustic impact. In this comparison the effect of "Room Size" is marginal compared to that of "Decay Time."

Most parameters such as "Room Size" result in a continuous change in the fitness. Some parameters such as "Reverberation Type", however, cause jumps in the fitness. This makes automated adjustment harder; especially with optimization approaches that tend to get stuck in local maxima and do not take the whole solution space into account. Fig. 3. illustrates a very idiosyncratic behavior that defies most optimization algorithms: The parameters "Variation" and "Diffusion" of the Ambience plug-in are intended to create a random behavior.

3.2. Network Rendering

The computation concerning the audio spectrum is particularly time-consuming. For speeding up this process we implemented a distributed method: The user has to decide which of optimization method is to be applied and to choose a reference impulse audio file and the dry impulse recording that has been used to create the reference file. These settings are bundled into a job description and stored in the server component of the software. On any machine connected to the network one can now start a small background application once per processor core. This so-called "Cruncher" application registers with the server and waits for data. If a new optimization process has been started, the plug-in executables to be optimized and the reference recording are copied to every machine registered.

The Cruncher applies the VST-based plug-in to the audio file, it determines the MFCCs as well as—for display purposes—the MPEG-7 spectral envelope, and computes the similarity to the reference recording. This number and the spectral envelope are sent back to the server.

The optimization algorithm runs within the server. If a Cruncher sends back its result, the parameter settings can be changed depending on the optimization strategy and queued back in the service pipe to be sent to the next available crunching slot. The Crunchers request a new set of parameters as soon as they run idle. The service component then distributes the new settings.

In every update step of the optimization algorithm, the server queues all new settings to be computed next, and the network service distributes the tasks to the cruncher machines. The more machines join the queue, the more parameter settings can be processed in parallel.

3.3. Optimization

We implemented four different optimization strategies in the software prototype, and compared their performance.

Upon starting the optimization, one half of the candidates' initial settings to be optimized are taken as a random selection from the factory settings delivered with the plug-in. Often these settings cover a wide range of different reverberation sounds and therefore already provide appropriate settings. The other half of the candidates' initial settings are populated with random values. Although the random settings often produce uninteresting results, we experienced factory settings that had their wet/dry balance set to zero, therefore not producing any effect.

3.3.1. Evolutionary Optimization

Evolutionary optimization algorithms try to find a maximum of a fitness function by tweaking a set of candidates—the population—according to principles found in biology. In our case, each individual of the population is the setting of a reverberation effect. The fitness is determined as described before to mimic the acoustic similarity of the reverberation effect's response to the given reference.

The Cruncher network computes the fitness for every individual in the population. After this is finished, the best individual is recombined with the rest of the population and mutated. To this end, one randomly picked third of an individual's is changed by random, and another third is replaced with the corresponding values from the best one.

3.3.2. Nelder Mead Optimization

In broad strokes, the Nelder-Mead algorithm, also known as downhill simplex algorithm, repeatedly takes the worst individual from a set of solutions and mirrors it across the centroid of that set. If it does not find a better solution there, it also tries to mirror other known points. If no better solution becomes apparent, the algorithm assumes that it has encircled a local minimum in the function and shrinks all points towards the centroid. Although this approach is an efficient way to find minima in a continuous problem space, the algorithm will immediately get stuck in a local but non-global extrema, which is particularly vexing with non-continuous fitness functions. To counteract this behavior, we set the initial settings of the candidates as to govern a wide part of the problem space.

3.3.3. Brute-Force Nelder-Mead Optimization

The simplex algorithm described above cannot be parallelized efficiently due to several steps in the algorithms where choices are made upon recalculating the fitness. We parallelized the algorithm by simply calculating all steps at once and leaving part of the results unused due to choices in the algorithm: All possible interim points are queued for network rendering.

3.3.4. Particle Swarm Optimization

The fourth technique we tested was an optimization algorithm devised by Eberhardt and Kennedy [4]. This so-called Particle Swarm Optimization (PSO) is inspired by the social behavior of bird flocking or fish schooling. The PSO approach shares many similarities with the evolutionary algorithm described before. A population of particles is initialized by the system. Each particle carries a vector containing the plug-in settings, and an offset vector, which is randomly initialized. In every update step the particles are moved through the problem space by adding the offset to the current setting. Each particle keeps track of its coordinate in problem space that has generated the best fitness so far: the local best setting. Furthermore, the optimizer keeps track of the population's best setting ever: the global best setting To update a particle, its position is shifted toward a random blend of the local and the global best settings, see Fig. 4.

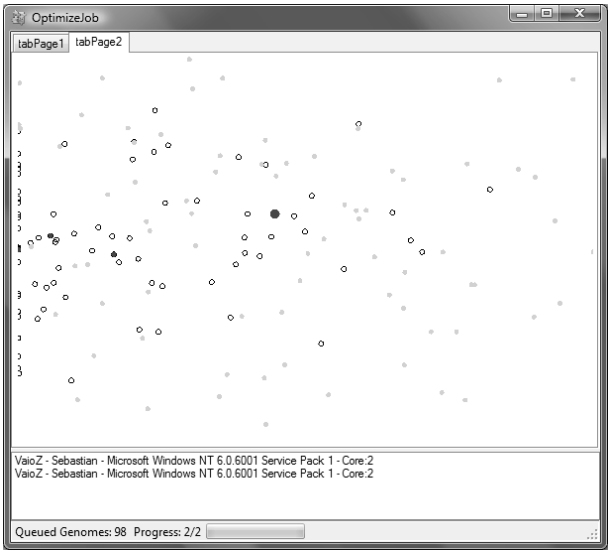

Fig. 4: Visualization of the particles' current settings data mapped to a 2D grid (white circles). The particles "fly" around between their own local best setting (light grey), and the global best setting (black circle).

3.4. Comparison of the Optimization Algorithms

Neither of the two implemented simplex algorithms was able to solve our problem satisfactorily. Thus, to cover a wider range of settings in problem space we chose a genetic optimization method. Genetic algorithms can easily be parallelized and are also easier to implement than the Nelder-Mead approach, as the genomes can be computed asynchronously. All we have to track is the global best candidate.

The main advantage of the Particle Swarm Optimization compared to Nelder-Mead and the simple evolutionary approach is that while the particles are flying through the problem space they automatically deal with the non-continuity of the function and are able to escape a local but non-global maximum of the fitness function, unlike the Nelder-Mead strategy. They do so with less computational load than the simple evolutionary algorithm.

To compare the performance of the different optimization algorithms, we recorded the fitness values over 10,000 parameter changesin a population of 20 candidates ten times for the most promising optimization strategies: the Genetic Optimization (see Fig. 5) and the Particle Swarm Optimization (see Fig. 6) applied to the

"Medium Rooms – Classroom" impulse response taken from Sony's Acoustic Mirror[2] Software to be modeled with the Ambience VST reverberation plug-in. One can see that both optimization algorithms work well in optimizing the parameter settings.

In all cases both algorithms optimize the fitness without any protruding values. As the starting values may have a high difference due to the random initialization of the first 10 candidates (the population's size was 20), the resulting fitness values have a higher deviation in the beginning of the optimization process (see Fig. 6) and "jump" to a better fitness value after the randomly initialized candidates are evaluated (see Fig. 5).

4. LISTENING TEST

To test the performance of the prototype, we carried out a listening test. We conducted a MUSHRA-type comparison test to compare hand-crafted plug-in settings with settings retrieved from the optimizer. Two professional sound engineers were asked to set the Ambience Reverb and the Black Water Reverb[3] as close as possible to given reference recordings, which were created using dry recordings processed with the Sony Acoustic Mirror plug-in and the "Medium Rooms – Classroom" impulse response. The human listeners were presented with correspondingly reverberated music recordings processed again with Sony's Acoustic Mirror. We used a drum track and an acoustic guitar track[4], each of 15 seconds' length, as acoustic material to be processed. With the percussive drum track one can clearly perceive the reverberation time and falloff characteristic, whereas with the guitar recording the reverberation's timbre dominates the acoustic experience.

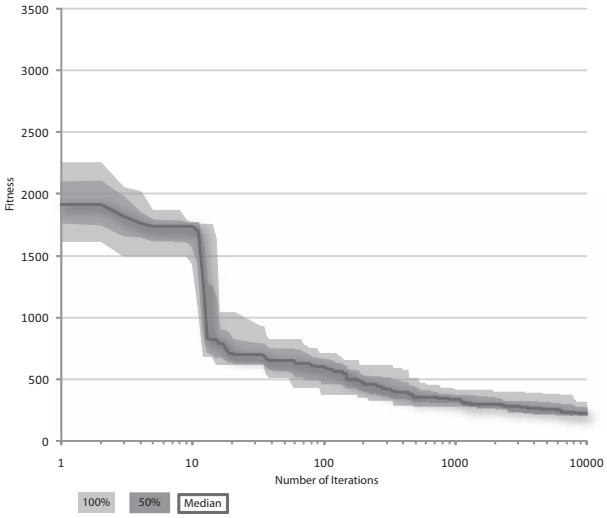

Fig. 5: Statistical analysis of ten evolution processes over time using the genetic optimization algorithm. Light grey: region containing the best fitness values of each respective population Dark grey: region containing the inner 50 % of the best fitness values of each respective population. Black Line: Median fitness of all populations.

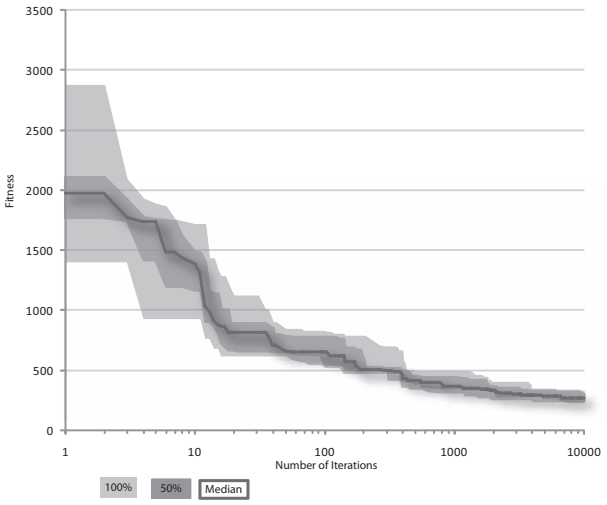

Fig. 6: Statistical analysis of ten evolution processes over time using the particle swarm optimization algorithm. Light grey: region containing the best fitness values of each respective population Dark grey: region containing the inner 50 % of the best fitness values of each respective population. Black Line: Median fitness of all populations.

[2] DirectX plug-in provided with Sony's Soundforge 9, http://www.sonycreativesoftware.com/soundforge

[3] Black Water Reverb, http://www.apulsoft.ch/ from Black Water Music is based on the open source Freeverb3 originally developed by Dreampoint.

[4] Sound Check – The Professional Audio Test Disk, by Alan Parsons & Stephen Court, track 62 and track 77.

Thus, we arrived at four sets for comparison by ear: a Black Water Reverb and an Ambience set for both the drum and the guitar track. For the automatically optimized settings, we took snapshots after 1000 and after 10,000 fitness evaluations for each plug-in. These two automatically generated settings, the two hand-crafted settings, and the reference sample itself constituted the sound material to be compare within each of the four sets.

We invited 25 subjects, 12 female and 13 male, 18 of them are students within the media technology field, 7 musicians or sound engineers. For all four sets, the subjects were asked to rank the five sound samples (presented in random order and without identifying labels) against the reference file after their perceived similarity in reverberation. Each example could receive 0 through 4 points, where 4 points represent the best match in the respective set.

Fig. 7: The computed MPEG-7 spectral envelopes of the compared Ambience settings mapped to grayscale. The numbers behind the settings' names indicate the resulting distance to the reference.

All subjects remarked that the differences between the examples were small. Even trained listeners noted that it was hard to discern the samples. Nonetheless, as can be seen in the box-and-whisker plot in Fig. 9, most users successfully identified the reference sound itself as the one closest to the reference recording, thus the box's upper and lower quartile are both on the 4 points level.

Fig. 8: The computed MPEG-7 spectral envelopes of the compared Black Water Reverb generated settings mapped to grayscale. The numbers behind the settings' names indicate the resulting distance to the reference.

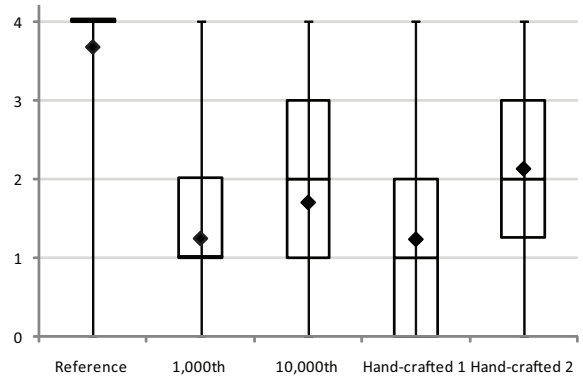

Fig. 9: The box-and-whisker plot of the combined quality test results of all four sound sets indicates that the automated settings can keep up with hand-crafted settings. For the vertical axis, refer to the text.

As the median shows, the automatically optimized settings can easily keep up with the hand-crafted ones. When comparing the distancevalues in Figs. 7 and 8 to the mean values from the listening test in Fig. 9 one can

see that the ranking between the different processed sounds is coherent, at least for these examples.

The professionals whom we asked to perform the task of adjusting a VST plug-in to resemble a convolution reverberation unit all pointed out that the more parameters a reverberation algorithm provides, the more difficult it gets to set it up. In the case of the Ambience plug-in, which offers 21 parameters, the hand-crafted results were produced in not less than an hour. Even though the optimizer runs unattended, this number can be compared with the computing time of the automatic optimizer. This time depends on the speed and number of available processor cores. A 2.5 GHz dual-core notebook computer takes around 30 minutes for 10, 000 distance computations with the Ambience VST plug-in.

5. CONCLUSION AND OUTLOOK

We presented a method to adapt a common VST effect plug-in to match a given reference impulse comparing different optimization algorithms. A listening test showed that the results of our tool can easily keep up with handcrafted setting in quality, and outperform hand-crafted settings when it comes to counting the time to set up a complex reverberation plug-in. Even if it is possible for an experienced sound professional to find a proper setup for the effect, it might take a long time and therefore incur huge costs.

The tool could also be used to compare the sound quality of different VST plug-ins. For tasks where limited environments compel sound engineer to use simpler effect algorithms, such as integrated DSPs with little memory, the effect setting could be optimized with our tool and then implemented on the integrated module.

The hugest computational load in the optimization process consists in the computation of the difference function. In a future implementation, this task may be sped up by porting it to the graphics card.

6. REFERENCES

[1] Kim, H.-G., Moreau, N. and Sikora, T.: MPEG-7 Audio and Beyond - Audio Content Indexing and Retrieval. John Wiley & Sons Ltd (West Sussex, England 2005) 79, 2005.

[2] Fogel, L. J.; Owens, A. J. and Walsh, M. J.: Artificial Intelligence through Simulated Evolution (John Wiley, 1966 New York, NY, USA), 1966.

[3] Nelder, J. A. and Mead, R. A.: A Simplex Method for Function Minimization. Computer Journal, 1965 no. 7, 308-313, 1965.

[4] Kennedy, J. and Eberhart, R.: Particle swarm optimization. Presented at the IEEE International. Conference on Neural Networks, (Perth, Australia, November 27 – December 1, 1995), 1995.

[5] Rimell A. and Hawksford, M.: The application of genetic algorithms to digital audio filters. Presented at the 98th AES Convention (Paris, France, February 25–28, 1995), 1995.

[6] Uesaka, K. and Kawamata, M.: Evolutionary synthesis of digital filter structures using genetic programming. In IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing 50, 2003, no. 12, 977–783, 2003

[7] Tsai, J.-T., Chou, J.-H. and Liu T.-K.: Optimal design of digital IIR filters by using hybrid Taguchi genetic algorithm. In IEEE Transactions on Industrial Electronics 53, 2006, no. 3, 867–879, 2006.

[8] Yu Y.; Xinjie, Y.: Cooperative coevolutionary genetic algorithm for digital IIR filter design. In IEEE Transactions on Industrial Electronics 54, 2007, no. 3, 1311–1318, 2007.

[9] Loviscach, J.: Graphical Control of a Parametric Equalizer. Presented at the 124th AES Convention, (Amsterdam, The Netherlands, May 17-20, 2008), 2008.

[10] Loviscach, J.: Programming a Music Synthesizer through Data Mining. Presented at the 8th NIME Conference, (Genova, Italy, June 5-7, 2008), 2008.

[11] Extra, D., Simmer, U., Fischer, S., and Bitzer, J.: Artificial Reverberation: Comparing algorithms by using monaural analysis tools. Presented at the 121st AES Convention, (San Francisco, CA, USA October 5-8. 2006), 2006.