Computational Challenges in Multiple Wavetable Interpolation Synthesis

Jonathan Mohr[1] and Xiaobo Li[2]

[1] Augustana University College, Camrose, Alberta, Canada T4V 2R3
mohrj@augustana.ca, http://www.augustana.ca/~mohrj/
[2] University of Alberta, Edmonton, Alberta, Canada T6G 2M7
li@cs.ualberta.ca, http://www.cs.ualberta.ca/~li/

Abstract. A new music analysis/synthesis algorithm, optimized multiple wavetable interpolation, poses significant computational challenges in the spectral matching stage, in which it searches for the subset of the available wavetables that best matches the spectrum at each breakpoint of a piecewise linear approximation of the spectral envelope of a recorded tone. Two methods of reducing the computational cost of spectral matching are presented: a multi-level pruned search and a caching genetic algorithm.

1 Introduction

Multiple wavetable interpolation [1] is a form of music analysis/synthesis that involves three basic steps: a recorded sound is reduced to a set of breakpoints by piecewise linear approximation of the spectral envelopes of its harmonics; the spectrum at each breakpoint is matched by determining weightings for a small number of wavetables; and the sound is resynthesized using multiple wavetable additive synthesis by interpolating between the weightings for each wavetable at consecutive breakpoints.

Multiple wavetable interpolation as introduced by Horner [1] incorporates the spectral matching techniques from Horner's previous work on multiple wavetable synthesis [2,3] into Serra, Rubine, and Dannenberg's earlier *spectral interpolation* method [4]. These techniques aim to reduce the computational cost of synthesis and to achieve data reduction through automated analysis techniques while retaining the perceptually salient features of the sound to be resynthesized. Recent studies have evaluated the audible effects of various types of data simplifications [5], compared a variety of sound analysis/synthesis systems [6], and tested a wavetable matching method that takes into account the characteristics of the human auditory system [7].

A recently introduced algorithm [8] that optimizes the use of available oscillators during the synthesis stage of multiple wavetable interpolation poses significant computational challenges in the spectral matching stage. After introducing the concepts and methods of analysis/synthesis by multiple wavetable interpolation in Section 2, methods of reducing the computational cost of spectral matching by deterministic and probabilistic search will be discussed in Section 3, and conclusions will be drawn in Section 4.

2 Multiple Wavetable Interpolation Synthesis

Analysis/synthesis [9] (or analysis/resynthesis [10]) is a general process in which a recorded sound is analyzed in such a way that a musician or sound technician can modify the analysis data and synthesize an altered sound from the modified data; alternatively, the goal of the analysis may be to find a data-reduced representation from which the original tone may be resynthesized economically but with high fidelity, typically by a commercial synthesizer ("keyboard").

Musical tones are commonly synthesized by *additive synthesis* using a number of *digital oscillators*: a single cycle of a discrete (sampled) sinusoidal waveform is stored in a lookup table; the digital oscillator then generates a sound by repeatedly scanning this *wavetable* and sending the samples through a digital-to-analog converter (DAC). The outputs of several digital oscillators may be summed to form a composite sound waveform with a spectrum of arbitrary complexity.

Multiple wavetable synthesis [2,3] is an additive synthesis technique based on the sum of fixed waveforms or periodic basis functions with time-varying weights. Unlike classical additive synthesis, in which the waveforms to be added are simple sinusoids, multiple wavetable synthesis loads each wavetable with one cycle of a waveform of arbitrary complexity. Typically, the waveforms to be added are themselves the fixed weighted sum of several harmonic sine waves; the spectrum produced by a particular set of harmonic weights is referred to as a *basis spectrum.*

The principal advantage of multiple wavetable synthesis is its efficiency, since the number of wavetables used is typically much smaller than the number of sine waves that would be used in classical additive synthesis. The principal difficulty is that, for an arbitrary small set of wavetables, most time-varying spectra cannot be approximated very closely by a linear combination of these wavetables, even if their weights are time-varying. Thus the basis spectra must be chosen carefully and their weights appropriately manipulated when synthesizing dynamic spectra.

In contrast to multiple wavetable synthesis, which uses all of the selected basis spectra in each match, *multiple wavetable interpolation* uses only a subset of the basis spectra at each match point (breakpoint). If the subset of the wavetables used at one breakpoint differs from the subset used at the next breakpoint, two or more oscillators must be used to *crossfade* the changing wavetables between match points in order to avoid audible clicks and spectral discontinuities [4].

Figure 1 illustrates several possible assignments of wavetables to oscillators at two adjacent breakpoints (B_i, B_{i+1}), assuming that four oscillators are available for use in synthesis. Part (a) represents the case in which the same set of wavetables is used at both breakpoints; the weighting of each wavetable may vary from one breakpoint to the next, but all four wavetables remain in active use. Part (b) illustrates the fade-in of two wavetables from zero amplitude; one or more wavetables may similarly be faded out. Parts (c) and (d) show the simultaneous fade-in and -out of multiple wavetables from one breakpoint to the next.

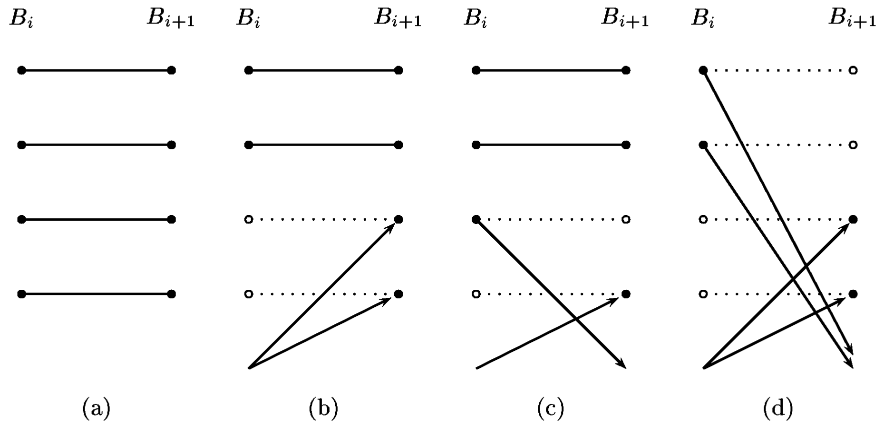

Fig. 1. Examples of possible oscillator assignments with four oscillators.

Horner [1] dealt with the need to crossfade between changing wavetables by imposing two simple restrictions: if synthesis is to be performed with N oscillators, only $N - 1$ wavetables may be used at each match point, and only one wavetable may be changed from one match point to another. His method starts by selecting by enumeration the best combination of $N - 1$ wavetables to use at the breakpoint with the peak RMS amplitude, then works backward and forward to neighboring match points, changing at most one of the wavetables, using enumeration to decide what change to make, if any.

Mohr [8] introduced a method that is not subject to the constraint of selecting $N - 1$ wavetables at each breakpoint; it finds a globally optimal set of weighted wavetable matches across all breakpoints, given a particular error measure and a specified method of choosing an initial best match of a specified size to the spectrum at each breakpoint. As illustrated in Figure 2, the algorithm uses a three-stage process.

In the first stage of the breakpoint matching algorithm, an initial match of a user-specified size is found for each breakpoint. The size of the match (that is, the number of different wavetables used in the match) can maximally be the same as the number of oscillators to be used in the synthesis stage, but may also reasonably be less than the number of oscillators, since the set of wavetables to be considered for final use at a given breakpoint may be augmented with additional tables in later stages of the matching algorithm.

The second stage of the breakpoint matching algorithm is intended to provide more flexibility in the subsequent optimization stage which, as part of its task of assigning wavetables to oscillators, must decide when to fade a wavetable in or out of use. The wavetable sets chosen in the first stage are overlapped with the wavetable sets at preceding and following breakpoints before the optimizer makes oscillator-assignment decisions in the following stage. The greater the overlap

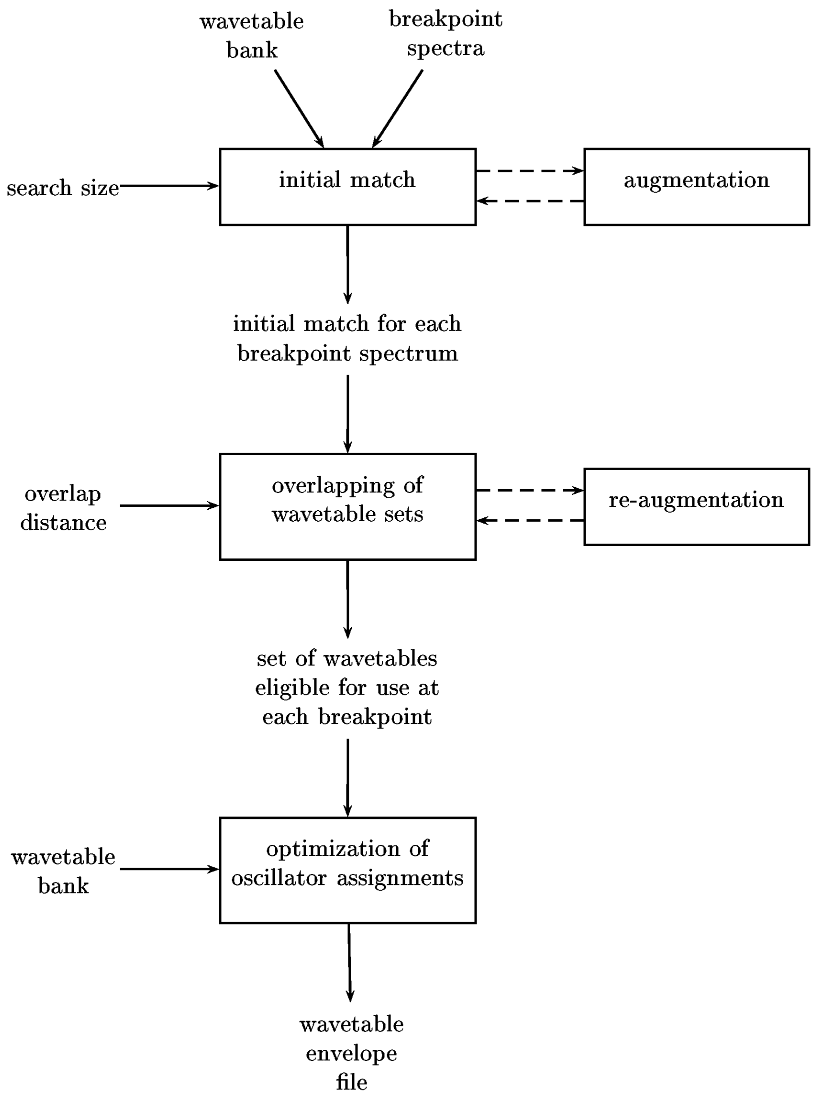
wavetable bank
breakpoint spectra
search size
initial match
augmentation
initial match for each breakpoint spectrum
overlap distance
overlapping of wavetable sets
re-augmentation
set of wavetables eligible for use at each breakpoint
wavetable bank
optimization of oscillator assignments
wavetable envelope file

Fig. 2. The stages of the globally optimal breakpoint matching algorithm.

distance, the more possibilities which must be evaluated by the optimizer, so the amount of overlap is best limited to distances from one to three.

The task of the final phase of the breakpoint-matching algorithm is to assign a weighted wavetable to each available oscillator at each breakpoint such that the overall error is minimized, taking into account the need to fade a wavetable in or out when it begins or ceases to be used. The optimization of oscillator

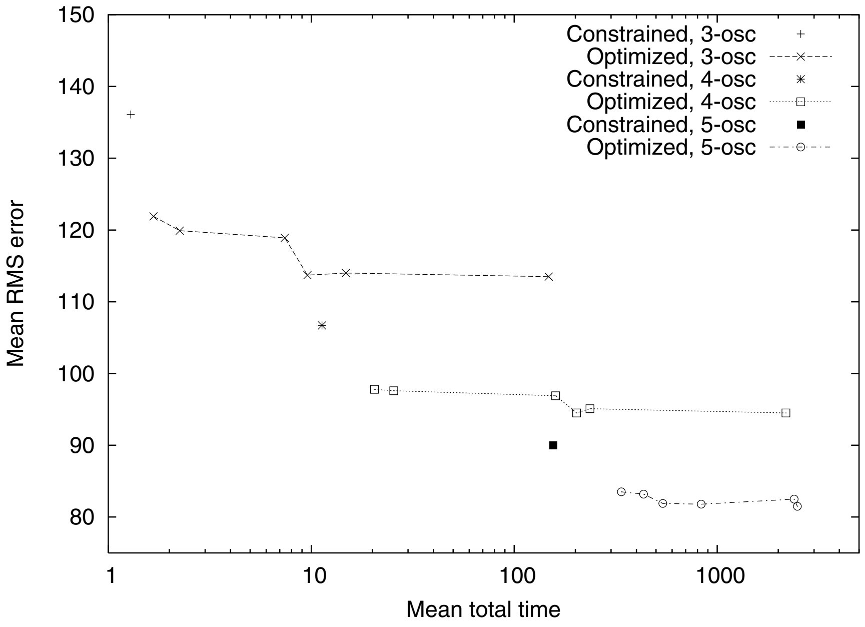

Fig. 3. Comparison of Horner's constrained matching with optimized multi-level pruned search results for a group of tones.

assignments is achieved by modeling the problem as a vertex-weighted directed acyclic graph (DAG) and using the single-source acyclic weighted shortest path algorithm [11].

Figure 3 compares the average results of Horner's constrained matching method with those of optimized matching (beginning with various types of search for initial matches) using 3, 4, and 5 oscillators for matching a group of 43 low-pitched tones played by various instruments, with matches selected from a bank of 48 wavetables.

While Horner's constrained matching method is faster than any of the types of multi-level pruned search optimization for a given number of oscillators, the error levels produced by Horner's method are significantly higher than those of the optimized matches, and are closer to those achieved by optimization with one fewer oscillators.

3 Search Techniques for Spectral Matching

Given a particular set of basis spectra (a *wavetable bank*), a set of breakpoint spectra representing a piecewise linear approximation (PLA) of the spectral envelope of a particular tone, and the number of oscillators (N) to be used in resynthesis, the spectral matching algorithm selects, by index, at most N wavetables from the bank which, in weighted combination, best match the spectrum at each breakpoint according to some error measure.

As Horner, Beauchamp, and Haken have shown [3], the problem of determining the weightings (amplitude factors) of a set of basis spectra which provide the best match to a particular spectrum in a least-squares sense is a linear problem, and can be solved using matrix arithmetic by use of the *normal equations* [12]:

$$(\mathbf{A}^T \cdot \mathbf{A}) \cdot \mathbf{c} = \mathbf{A}^T \cdot \mathbf{b} \ . \tag{1}$$

This system can be solved by finding an LUP decomposition of the transpose-product $\mathbf{A}^T \cdot \mathbf{A}$ where $\mathbf{A}$ is a particular set of wavetables (basis spectra) selected from the wavetable bank. The LUP decomposition needs to be calculated only once per selection and used multiple times to evaluate how well that set of wavetables can approximate the spectrum at each breakpoint or at some subset of the breakpoints.

Even so, the computational demands of finding the best possible wavetable match to the spectrum at each breakpoint can be prohibitive. The number of LUP decompositions performed in the course of an exhaustive search for an m-table match where the tables are drawn from a wavetable bank of size N_{WT} is $\binom{N_{\mathrm{WT}}}{m}$ and the number of least-squares solutions to be found is $\binom{N_{\mathrm{WT}}}{m} N_{\mathrm{bkpt}}$ where N_{bkpt} is the number of breakpoints in the current tone. A single LUP decomposition takes about the same amount of time as a single least-squares solution, and both are dependent on the number of harmonics in the basis spectra of the wavetable bank.

For example, in testing the optimized multiple wavetable interpolation algorithm on a set of 198 tones played by sixteen different instruments, spanning the range from A1 to B6 by minor thirds, wavetable banks containing as many as 74 different basis spectra were used to match the spectra at an average of 45.6 breakpoints per tone. An exhaustive search to find the best possible four-wavetable match to each of the 48 breakpoint spectra of the PLA of an English horn tone, given a 74-wavetable bank, required 1,150,626 LUP decompositions and 5.5×10^7 least-squares solutions, which took an average of 2.3 hours per tone on a 500 MHz Intel Pentium II Celeron-based system.

Two alternative search strategies were tested in order to reduce the cost of finding an initial weighted wavetable match to the spectrum at each PLA breakpoint: a multi-level pruned search and a genetic algorithm.

3.1 Multi-level Pruned Search

One way to reduce the cost of a search is to focus the search by pruning the search tree. This can be done in the present case by performing an exhaustive search for the best matches of some size less than N_{osc} and then extending the first-level search by a second level which seeks to augment only those sets of wavetables which provided a best match to at least one breakpoint spectrum in the first-level search. For example, if an eventual 4-oscillator match is desired, the search performed at this stage could search for the best 3-wavetable matches in the first level and augment those sets with a fourth wavetable in the second-level search (a “3+1” search). Alternatively, the first-level search could seek only 2-wavetable matches which would be augmented with two additional wavetables (a “2+2”search) or even a single additional wavetable (a “2+1” search) in the second-level search.

A “3+1” search executes about an order of magnitude faster than an exhaustive search of depth 4, yet yields about the same or better error rates, on average, after optimization. A “2+1” search is another order of magnitude faster than a “3+1” search, at the cost of an increased average error of about 50%; however, after optimization, the difference in average matching error is reduced to between 2% and 5%. For example, the average time for a “3+1” search on the same English horn tones referenced above was just over six minutes; a “2+1” search took just 17.7 seconds on average.

3.2 A Caching Genetic Algorithm

A genetic algorithm (GA) [13,14] is a form of probabilistic search that is guided by a strategy based on genetic inheritance. Beginning with a randomly generated *population* of potential solutions to the problem, the GA repeatedly generates a new *generation* by probabilistically “breeding” new individuals from pairs of existing individuals; by giving preference to individuals which are more *fit* (as determined by an *objective function*), the quality of the population is likely to increase (*evolve*) over the generations. The process is typically terminated after a specified number of generations or when some level of convergence of the population or the fitness scores of the best individuals has been reached.

A genetic algorithm was implemented as an alternative to the first level of the multi-level search discussed above; that is, if an initial search of size N_{osc} was specified, the genetic algorithm was used instead of exhaustive search, but if an initial search of size less than N_{osc} was specified, a second-level pruned search could be used to augment the matches found by the genetic algorithm.

Testing of this approach confirmed that, when used as a first-level search in combination with an exhaustive second-level search, the genetic algorithm could find matches about as good as those found by exhaustive or pruned search, but savings in time were only realized relative to the larger exhaustive searches, and then only if the objective function cached the results of calls to the LUP decomposition and least squares evaluation functions.

When the objective function was invoked by the GA on an individual of the current generation, the genes of that individual were deemed to be an ordered list of wavetable sets, each to be used as an initial match at its respective breakpoint, and were inserted into a multimap which mapped each wavetable set to the breakpoint(s) at which it was used. The multimap was then traversed with an iterator so that the basis spectra corresponding to each different wavetable set used in the current individual were referenced as a matrix and analyzed by LUP decomposition; the LUP decomposition was then used as in equation 1 to fit that wavetable set in a least-squares sense to the spectrum at each breakpoint specified by the genes of the current individual, again using the multimap iterator. Without caching, this implied that LUP decompositions were being repeatedly performed on the same sets of wavetable spectra, since it was highly likely that the same wavetable sets would be used by multiple individuals in the current generation and repeatedly from generation to generation; similarly, the same wavetable sets would likely be used as matches to the same breakpoint spectra in various individuals and across generations.

For example, a three-table GA match[1] to a 24-breakpoint PLA of a bassoon A♯1 tone converged after 247 generations, during which it performed 269,632 LUP decompositions and 269,688 least-squares solutions and error calculations in about 146 seconds on the test platform. Augmenting the matches with a fourth wavetable and optimizing the final oscillator allocation took an additional 35 and 34 seconds, respectively, for a total time of 215 seconds. By comparison, an exhaustive search of depth 3 performed $\binom{48}{3} = 17{,}296$ LUP decompositions and 24 times that many (415,104) least-squares and error evaluations in just over 110 seconds; augmentation required only 6.5 seconds and optimization, 22 seconds, for a total of about 139 seconds.

Caching was implemented by introducing two mappings as static data members of the objective function so that the contents of the mappings would be preserved across the evaluations of all individuals in all generations. The first maps from sets of wavetables to the results of LUP decompositions, the second, from breakpoint number and wavetable set to the corresponding least-squares solution. When iterating across the multimap, the first map is checked for each wavetable set in the current individual, and LUP decompositions are performed only for those wavetable sets not already in the map; similarly, when iterating across the breakpoints at which a given wavetable set is used in the current individual, the second map is searched for a pre-existing least-squares solution and error level.

As a result of this form of caching, a GA search for a three-table match to the same bassoon tone discussed above now performs only 7,331 LUP decompositions and 9,248 least-squares and error evaluations in just over 16 seconds, for a total time of 85 seconds. While this particular invocation of the GA[2] tried only about 42% of the possible three-wavetable combinations, each at an average of 1.26 breakpoints, the final result (after augmentation and optimization) had an average error rate only two-thirds of a percent worse than that found by a "3+1" pruned search.

4 Conclusion

The results of multi-level pruned search and GA search are summarized in Figure 4, which plots root-mean-square (RMS) error levels vs. mean search times

[1] Matches were selected from the wavetable bank for the lowest-pitched group of tones tested, which has 48 wavetables, each consisting of 146 harmonic amplitudes. The GA used a population size of 50, and terminated upon convergence of 99% over 25 generations.

[2] Invocations with other random seeds would yield different results.

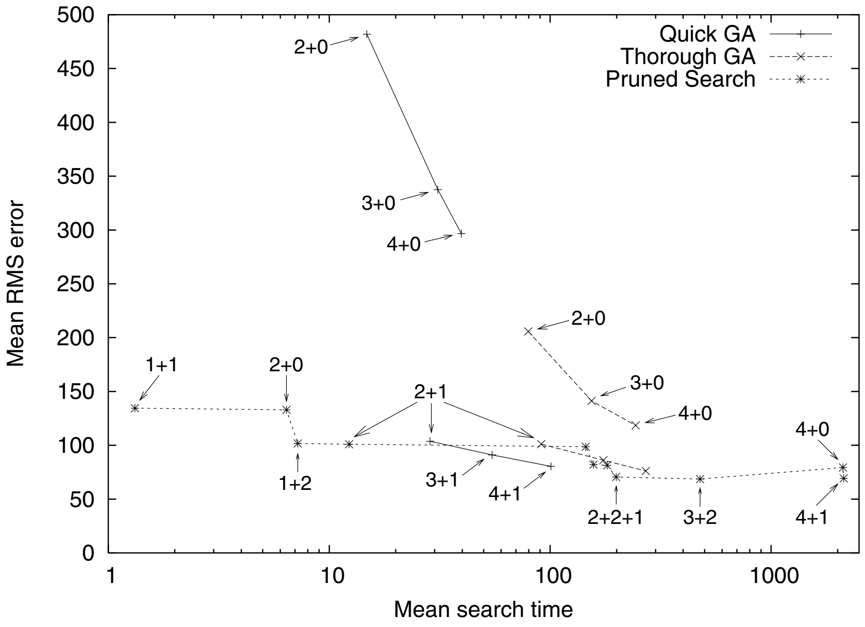

Fig. 4. Graph of multi-level pruned search and GA search results for a group of low-pitched instrumental tones. The search type is indicated for most data points.

(in seconds) for several types of search of a 48-wavetable bank for weighted combinations of wavetables to match the breakpoint spectra of 43 low-pitched tones (in the range A♯1 to C♯3) played by nine different instruments. The figure includes the results for both a thorough GA search, with a population size of 100 and termination on convergence after 50 generations, and a quick version of the GA search, with population size 50 and 25 generations to convergence.

The graph confirms that, in general, a search which takes more time (i.e., traverses more of the search tree) produces better results, but there are some cases which do not conform to this general model. For example, for a 5-table match, a "3+2" search is to be preferred to a "4+1" search, since it produces approximately the same or better results in significantly less time.

It is also clear that a GA search by itself results in significantly higher error levels than a GA search augmented by a second-level exhaustive search.

It should be noted that lower matching errors at this stage do not necessarily yield lower matching errors after oscillator assignment optimization. The primary reason that there is no direct relationship between the quality of initial matches and the quality of oscillator assignments is that matches which are highly specific to their respective breakpoint spectra are less likely to be used in a final set of matches than more general ones, due to the need to fade in and out any wavetables which change from one breakpoint to the next.

These results show that the high computational cost of an exhaustive search for the best initial wavetable match to breakpoint spectra may be avoided by use of a multi-level pruned search without a significant increase in error. A genetic algorithm may be useful as the initial stage of a multi-level search strategy, but only if caching of intermediate results is implemented; even then, some form of multi-level exhaustive search will likely yield better matches in less time than an augmented GA search of the same total depth.

References

1. Horner, A.: Computation and memory tradeoffs with multiple wavetable interpolation. Journal of the Audio Engineering Society **44** (1996) 481–496
2. Horner, A.: Spectral Matching of Musical Instrument Tones. PhD thesis, University of Illinois at Urbana-Champaign (1993)
3. Horner, A., Beauchamp, J., Haken, L.: Methods for multiple wavetable synthesis of musical instrument tones. Journal of the Audio Engineering Society **41** (1993) 336–355
4. Serra, M.H., Rubine, D., Dannenberg, R.: Analysis and synthesis of tones by spectral interpolation. Journal of the Audio Engineering Society **38** (1990) 111–128
5. McAdams, S., Beauchamp, J.W., Meneguzzi, S.: Discrimination of musical instrument sounds resynthesized with simplified spectrotemporal parameters. Journal of the Acoustical Society of America **105** (1999) 882–897
6. Wright, M., Beauchamp, J., Fitz, K., Rodet, X., Röbel, A., Serra, X., Wakefield, G.: Analysis/synthesis comparison. Organised Sound **5** (2000) 173–189
7. Wun, C.W., Horner, A.: Perceptual wavetable matching for synthesis of musical instrument tones. Journal of the Audio Engineering Society **49** (2001) 250–261
8. Mohr, J.: Music Analysis/Synthesis by Optimized Multiple Wavetable Interpolation. PhD thesis, University of Alberta (2002)
9. Cann, R.: An analysis/synthesis tutorial. In Roads, C., Strawn, J., eds.: Foundations of Computer Music. MIT Press, Cambridge, MA (1985) 114–144 Originally printed in *Computer Music Journal* 3(3):6–11, 1979; 3(4):9–13, 1979; and 4(1):36–42, 1980.
10. Roads, C.: The Computer Music Tutorial. MIT Press, Cambridge, MA (1996)
11. Cormen, T.H., Leiserson, C.E., Rivest, R.R.: Introduction to Algorithms. MIT Press, Cambridge, MA (1990)
12. Press, W.H., et al.: Numerical Recipes in C: The Art of Scientific Computing. second edn. Cambridge University Press, Cambridge, MA (1992)
13. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI (1975)
14. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA (1989)