

# A COMPUTER-AIDED SOUNDTRACK COMPOSITION SYSTEM DESIGNED FOR HUMANS

*Edwin Vane*

University of Waterloo  
revane@cgl.uwaterloo.ca

*William Cowan*

University of Waterloo  
wmcowan@cgl.uwaterloo.ca

## ABSTRACT

Film music is a well-defined compositional domain having specific features that are easily and usefully automated. To begin exploring automated soundtrack composition we have implemented a system that helps a composer to create a musical score matched to a film or video, for music composed in a minimalist style. The composer provides the musical themes and specifies a repetition pattern; the computer provides a collection of scores matching the composer's work to the film timing; and the composer chooses from among the proposed scores. We judge that this composition procedure leaves all significant creativity in the hands of the composer where it belongs, facilitates recomposition required by changes during film editing, and maintains the overall process of film music composition, while doing automatically the timing and tempo calculations that are distasteful to many composers.

## 1. INTRODUCTION

Most film and video is accompanied by a soundtrack, which complements and amplifies the action and emotion of its visual component. The soundtrack combines three components: a speech track, a sound effects track and a music track. The components of the soundtrack are synchronized with each other and with the visual component to create the final product. The theory and practice of automated tools to help a composer composing a music track is the subject of this paper.

Soundtrack composition differs in many ways from the composition of standalone music. Most importantly, standalone music is composed with enough structure and complexity to occupy the full attention of a listener, while the structure of soundtrack music is secondary to the structure of the visual component. In fact, overly structured soundtrack music can subtract from the overall effect by structural dissonance between the music and visual components [11]. Actual soundtrack music is found to be lightly structured, mainly using repetition with variation.

Second, standalone music plays without pause, or nearly so, while soundtrack music is intermittent, present in some scenes and not others. Each segment of soundtrack music is called a **cue**: it mainly relates to the visual component it accompanies, with secondary relationships to other cues in the score.

Third, standalone music sets its own time, while soundtrack music must synchronize with the times of events in the visual component. These events are measured in **clock time** (minutes and seconds), whereas the score is measured in **music time** (beats). Synchronization of the music track with the video track depends on tempo which expresses music time in terms of clock time.

Finally, the visual component changes as editing proceeds, often during soundtrack composition. Soundtrack music must then be changed to match new clock times. No such requirement exists for standalone music.

The presentation of these differences is necessary for considering the challenges facing a composer; the challenges that motivate the design of our computer-aided soundtrack composition system. When considering how these challenges can be assisted by computation we hold paramount the principle that the creativity of the composer should never be compromised: the computer should be a mindless assistant, who never makes creative decisions. Thus, the extensive research in algorithmic composition (such as [17, 6, 13, 8]) is not relevant to our research, which has a goal similar to that of QSketcher [1]: to be a tool that enhances, without in any way displacing, human creativity.

That being said, computers, which excel at numerical calculation, can take many of the temporal calculations off the hands of the composer. And when there are many score variations consistent with clock time constraints a computer can offer them to the composer with a guarantee that the collection is comprehensive. This capability is particularly effective in assisting with the third and fourth differences described above. To do so, it is necessary to define a shorthand for music specification, described in Section 3. The shorthand is, for ease of learning, very close to common practice notation.

Sections 4 and 5 then describe how this shorthand provides a complete solution for making a score conform to clock time constraints, and the subject of section 6 is a system we implemented, EMuse, which makes these capabilities available to a composer.

## 2. BACKGROUND

Composers have developed a set of methods that work well for soundtrack composition. Although these methods provide some inspiration for our research, the details

go far beyond the scope of this paper: Karlin and Wright [11] provide an excellent description of music track composition practices during the last century.

## 2.1. Commercial Software

Existing commercial products provide substantial computer support for music track composition. For example, a score writing program like **Sibelius** [18] covers aspects of composition shared by standalone music and music tracks, with particular emphasis on music typesetting. In contrast, **The Auricle** [3], is a specialized tool written and used by film composers which serves as a time calculator and synchronization tool.

Between these two extremes are software tools for creating full soundtracks. For example, **Soundtrack** [2], like other similar commercial products, facilitates soundtrack creation by providing a user with audio recording, editing, and mixing tools. These tools are often augmented to support synchronizing audio with a video track.

## 2.2. Previous Work

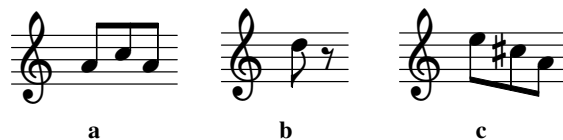
While it is useful to observe the capabilities of commercial software, the methods by which they are provided is hidden. Research in computer music, which includes many aspects of composition, such as music representation, music programming languages, and algorithmic composition, is more accessible.

Research in algorithmic composition falls into four categories: rule-based [17], stochastic [6], grammar-based [13], and genetic [8]. All of these algorithms focus on computer-generated music with little intervention from humans. More relevant to our interests are music programming languages, where the composer explicitly controls every sound produced. With sound synthesis languages like cSound [5], the composer can control music right down to individual waveforms. At a higher level of abstraction, compositional languages like Pla [16] try to support all the different practices of working composers.

The Concept Based Sequencer [10] and mkmusic [15] are examples of computer-generated soundtrack tools. In both cases, timings from a video track and a high-level description of music drive an algorithmic composition process resulting in computer-composed soundtracks. Evidently, our research on computer tools to support human-composed soundtracks fills a gap in research on computer music.

## 3. MUSIC SPECIFICATION

The musical style supported by our system is **minimal music**, which is described as a style using either a small amount of musical material or a restricted set of musical transformations [14]. Owing to the limited amount of material, repetition tends to be very important. Additionally, minimal music can be considered *music of the moment* as



**Figure 1.** Some simple motifs with their names.



**Figure 2.** Music produced by a sequence of motifs from figure 1 using the expression aabbcca.

listeners are not required to remember what has come before [14]. Thus, minimal music generally lacks large-scale structure making it well suited for music tracks.

Thus, in designing a method for music specification, we focus on minimal music. The music specification method defined in this section, and used in the following ones, describes music as the repetition of indivisible musical *atoms* called **motifs**, which most often are short sequences of notes and rests. The following points were the goals of the design.

- It must be easy to use the specification to create music.
- An algorithm must be able to interpret the specification without making creative decisions.
- The composer should have control at a range of granularities. Some composers like to write individual notes while others prefer to work in shorthand.

### 3.1. Expression Basics

The notes and rests that make up motifs are specified explicitly by the composer, subject to a single constraint: they must contain at least one note or rest so as to have a non-zero music-time duration. Motifs are repeated to fill durations so they must themselves have non-zero durations. Motifs are given unique names as shown in figure 1. Sequences of names then define sequences of motifs. Figure 2 shows the music resulting from a combination of motifs from figure 1.

As shown in figure 2, repetition of a motif is as simple as repeating its name in the expression. However, this form of repetition gives an algorithm no flexibility to choose or suggest a repetition count that makes the music represented by the expression fit a particular clock-time duration. To provide flexibility, we allow expressions to include **closure operators**: \*.

The closure operator is applied to the immediately preceding motif or *sub-expression* (a sequence of motif names enclosed in parentheses). It indicates that the preceding motif or sub-expression may be repeated any number of times. Closure operators in expressions give algorithms

Expression	Possible Expansion
$ab^*c$	abbbbc
$b(cd)^*b$	bcdcdb
$a^*b^*$	aaabbbbb

**Table 1.** Examples of valid expressions and their possible expansions.

the freedom to choose repetition counts that make the expression’s resulting music match a target clock-time duration. Only later, when a target clock-time duration has been specified, does the algorithm determine possible repetition counts from which the composer may choose. Table 1 shows several valid expressions and their *possible* expansions.

Expressions with closure provide the composer with varying levels of control: composers can specify an exact number of repetitions using explicit repetition, as in  $aaa$ , or a variable number using closure, as in  $a^*$ . Explicit repetition gives composers fine-grained control of the music while retaining the benefit of working with motifs. However, they give up using the ability of the computer to calculate repetition counts that match clock-time durations.

This method of music specification is based on **regular expressions** which have been studied as formal languages. Regular expressions are a basic concept of theoretical computer science. The expressions defined above are a small subset of regular expressions. Using larger subsets in a richer music specification system is an avenue for future work, limited by the problem of making them easily comprehensible.

### 3.2. Timed Regular Expressions

Expressions still lack a crucial dimension: to make an expression last for a given clock-time duration it is necessary to know the durations of its elements. Each motif has a fixed music-time duration based on the notes and rests it contains. However, when an expression includes a closure operator the number of repetitions of part of the sequence is known only to be an integer multiple of the duration of the argument of the operator. The multiple is called a **repetition unknown**. Solving for possible values of repetition unknowns is an important feature of our system.

The combination of an expression and its duration, which may include repetition unknowns, is called a **timed regular expression**. For notational purposes, a timed regular expression is made of two parts: a **repetition expression** which specifies how the motifs are combined and a **duration expression** which specifies how motif durations are combined. Table 2 provides some examples of timed regular expressions.

## 4. CHOOSING REPETITION SOLUTIONS

With a timed regular expression in hand, values for repetition unknowns can be sought. But, because repetition

Repetition Expression	Duration Expression
$abc$	$t_a + t_b + t_c$
$a^*$	$m_1 t_a$
$(bc)^*ba^*$	$m_1(t_b + t_c) + t_b + m_2 t_a$

**Table 2.** Timed regular expressions split into their repetition and duration expressions. The duration of motif  $i$  is specified by  $t_i$ . Repetition unknowns are represented by  $m_j$ .

unknowns must be integers, there is very rarely a set of repetition values, or **repetition solution**, for which the expression’s clock-time duration exactly matches its target clock-time duration. Composers normally make the match exact by varying tempo. Thus, it is appropriate for our system to provide a collection of approximate repetition solutions and, after one has been chosen, to allow the composer to vary its tempo until it matches the target duration exactly.

The target clock-time duration  $d$  is therefore provided with error bounds, which indicate the range of acceptable solution durations. That is, the composer provides  $\Delta_0$  and  $\Delta_1$  satisfying  $\Delta_0 \leq d \leq \Delta_1$  to define the range. For example, If the composer is more willing to slow the tempo down than to speed it up,  $\Delta_0$  is farther from  $d$  than  $\Delta_1$ .

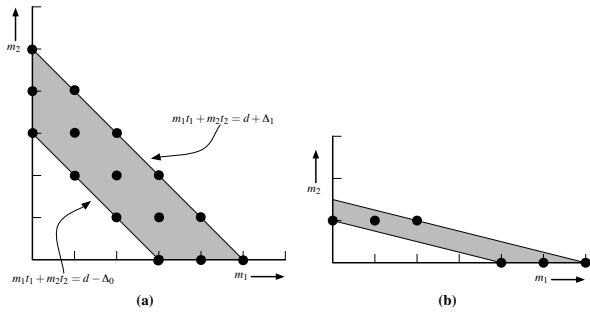
Given these values and the duration expression for a timed regular expression, repetition solutions are found by solving:

$$d - \Delta_0 \leq d_{exp} \leq d + \Delta_1$$

$$d_{exp} = k + \sum_i^n m_i t_i \quad (1)$$

In this equation,  $d_{exp}$  is the general form of a duration expression of which specific examples were given in table 2.  $k$  is the sum of the durations of all parts of the timed regular expression not affected by any closure operator.  $t_i$  is the duration of a sub-expression and  $m_i$  its repetition unknown. When  $n = 0$ , no closure operators are present and the existence or non-existence of a solution is the sole responsibility of the composer: the computer cannot help, so we assume  $n > 0$ . Then, there are usually many solutions since equation 1 is an *underdetermined system of equations*.

All durations in equation 1 are expressed in units of music time.  $t_i$  cannot be expressed in clock time because the clock-time duration of an expression depends on tempo which varies over time. Expressed in clock time,  $t_i$  would be a function of time making this equation much more difficult to solve. However,  $d$ ,  $\Delta_0$ , and  $\Delta_1$  can be expressed in music time using time frame conversions described in section 5. Solving equation 1 using music-time quantities is made possible because the function mapping clock time to music time is monotonic: tempo is a positive quantity.



**Figure 3.** Example of 1D hyperplanes (i.e. lines) in a 2D space. The bounding hyperplanes due to the inequalities in equation 1 are shown. Integer-valued repetition solutions are marked by circles. In Figure b, the solution space has changed due to an increase in duration  $t_2$ .

#### 4.1. Solution Space Structure

Given many possible repetition solutions for a timed regular expression, how does the system present them to the composer? A naïve solution presents repetition solutions in a linear list. A composer then scans the list to find suitable solutions. However, among the solutions there are patterns and structure that a linear list obscures. If the number of solutions is large, this structure must be shown to the composer.

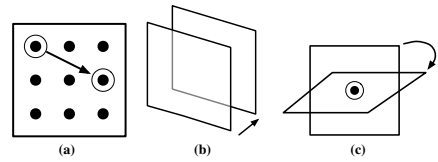
Thinking mathematically about the repetition solutions provides some insight. A repetition solution is an ordered  $n$ -tuple of integers satisfying equation 1. Repetition solutions thus lie in an  $\{n\}$ -dimensional space. By defining each vector in the  $\{n\}$ -dimensional standard basis to represent a repetition unknown, we discover that repetition solutions for one timed regular expression form a subset of an infinite lattice.

Equation 1 is an  $\{n\}$ -dimensional linear equation. That is, it describes an  $\{n-1\}$ -dimensional hyperplane. The inequalities give the hyperplane “thickness”. That is, repetition solutions lie between two parallel hyperplanes. Figure 3 provides an example in two dimensions which also demonstrates how the orientation of the hyperplanes depends on  $t_i$ .

#### 4.2. Presenting Solutions

Given the space of repetition solutions and its structure, we must find a visualization method that is easy for the composer to understand and use. Fortunately, the nature of the solution space makes OSA PlaneSight [12] an attractive solution.

OSA PlaneSight is a tool for choosing colours from the OSA uniform colour space. OSA PlaneSight enables an artist to easily choose a colour from a collection of colours most of which are not visible at any given time. That is, OSA colours and repetition solutions exist in a space whose dimensionality is greater than that of the display surface. OSA PlaneSight solves the problem by providing simple navigation controls that control a 2D slicing plane. Only the colours falling on the slicing plane are displayed



**Figure 4.** Illustrations of RepChooser navigation operations: a) in-plane motion, b) out-of-plane motion, and c) reorientation.

at any time.

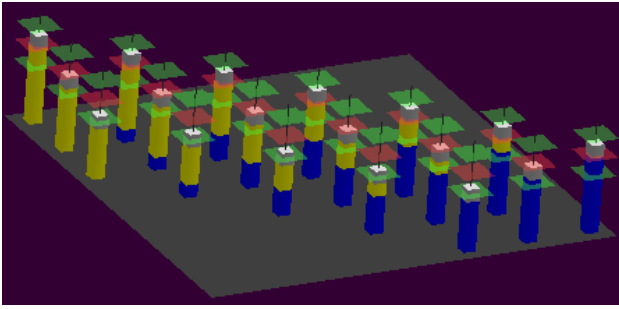
**RepChooser** presents repetition solutions to the composer using the ideas of OSA PlaneSight. It uses 2D slicing plane orientations that are parallel to any two standard basis vectors and perpendicular to all the rest. Thus, there are  $\binom{n}{2}$  possible plane orientations. RepChooser provides three navigation operations, described below and illustrated in figure 4 to change the orientation and position of the slicing plane within the repetition solution space.

1. In-plane motion: A composer selects a solution on the current slice as a new focus of attention. The currently selected solution serves as a center of rotation for the third operation below.
2. Out-of-plane motion: The slicing plane maintains its orientation but moves along one of the orthogonal basis vectors. In  $n$  dimensions there are  $n - 2$  orthogonal basis vectors.
3. Reorientation: A composer selects a new orientation for the slicing plane by choosing two basis vectors. The slicing plane rotates to that new orientation around the currently selected solution. Therefore the selected solution is present in both the old and new orientations.

Each navigation operation is presented to the composer in terms of the timed regular expression and its closure operators. In-plane motion selects a visible solution. Out-of-plane motion presents all solutions obtained by increasing or decreasing the value of one repetition unknown. Reorientation is achieved by the composer choosing two closure operators whose repetition counts they wish to see vary across the plane. Because the plane is defined by two basis vectors, only two repetition counts vary across the plane and patterns are easily visible.

In addition to these navigation tools, a visual representation of repetition solutions is required. Common practice notation is an option, but it is poor for providing information at a glance or for comparing large numbers of solutions together. RepChooser uses the simpler method of visualization shown in figure 5.

Each bar on the 2D slicing plane shows the music corresponding to a repetition solution. A bar is segmented into different colours, where segment size indicates the musical duration of a closure-repeated expression. This representation combines both the base duration of a repeated expression and the value of its repetition unknown.



**Figure 5.** Screenshot of the RepChooser prototype. Solutions are arrayed on a 2D plane. The floating red markers indicate the target duration  $d$ . The green markers represent  $\Delta_0$  and  $\Delta_1$ .

The total height of each bar is total musical duration of a repetition solution; the height of each coloured segment is the music-time duration of one repeated sub-expression. Even with many bars presented on the slicing plane together, relationships between repetition solutions are readily visible.

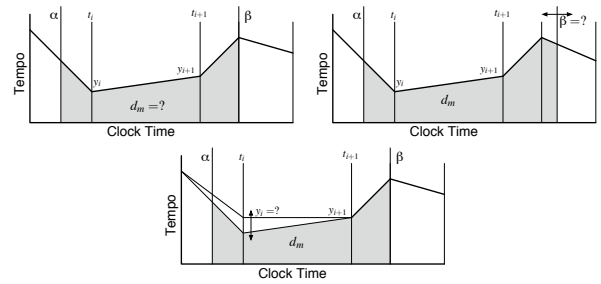
## 5. TEMPO

Tempo is important in all music. In soundtrack music it is especially important because only a precisely chosen tempo makes music-time durations match clock-time ones. The clock-time duration of a solution chosen by a composer is unlikely to exactly match the target duration necessitating tempo adjustment. Here the system helps the composer by calculating required tempi. In addition, the system automatically performs other calculations based on tempo, such as conversions between music time and clock time.

To perform these calculations while allowing the composer to control tempo, we need a tempo representation a composer can easily understand. Several tempo representations exist in the literature [4, 7, 9] and we make use of the representation more closely matching a musician’s understanding of tempo: rate of beats as a function of score location. However, instead of being a function of score location, we represent tempo as a function of clock time to help the composer map music onto visual events. Tempo is represented as a piecewise linear function, the corners of which are set and edited by the composer. An illustrative tempo function is shown in figure 6.

This tempo representation has a useful feature: the area under the curve between any two points in clock time is the music-time duration. Thus, clock-time durations are converted into music time by integration. Integration converts the target clock-time duration  $d$  into music time to solve equation 1. Music-time durations are converted to clock time also by integration. In this case, the music-time duration is known and  $\beta$  – see figure 6 – must be found.  $\beta$  being found, the clock-time duration is  $\beta - \alpha$ .

The last calculation needed defines the tempo value  $y_i$  for a given tempo marker  $i$ . This calculation provides the



**Figure 6.** Illustrations of the three tempo calculations. Each segment is represented by a linear function. The tempo value at an adjustment point is denoted by  $y_i$ . The clock-time position of adjustment points is denoted with  $t_i$ . The shaded region  $d_m$  represents the area under the curve from  $\alpha$  to  $\beta$ .

exact tempo to make a music-time duration,  $d_m$ , fit a given clock-time duration,  $\beta - \alpha$ . It is useful for helping the composer set a tempo that fits a repetition solution to a target duration. All three calculations are illustrated in figure 6.





## 6. THE SYSTEM

The previous sections introduced concepts that provide a framework for computer-aided soundtrack composition. To show that they adequately support soundtrack composition, we implemented a complete system based on the following workflow. The composer starts by defining a collection of motifs. Then points of interest are marked on a clock-time timeline: cue starts, cue ends, and other visual events to which the music should refer. Using these points as a guideline, regions of clock time are defined where each region has its music defined by a timed regular expression. Timed regular expressions are created using the motifs defined earlier.

The system then calculates a set of repetition solutions which are presented to the composer via RepChooser. The composer browses the space of repetition solutions and chooses one, the clock-time duration of which is calculated using the current tempo. This duration is then shown on a time-line, along with the target duration. The composer then adjusts the tempo function to make the two durations match. The tempo is changed at adjustment points which the composer can add and remove at any time. For any adjustment point the composer is working with, the system calculates a tempo value required to make the clock-time durations of the chosen solution and the region match. The result of this calculation is provided as a hint to the composer.

Once a tempo function is defined the music may be auditioned, likely suggesting changes and thus initiating the edit/test cycle that is characteristic of most creative work. The composer is free to edit any part of the music at any time: the motifs, the choice of repetition solution, or the tempo function.

Should clock-time durations later change, perhaps be-

Motif	Name	Beats
	<b>a</b>	1
	<b>b</b>	2
	<b>c</b>	1
	<b>d</b>	4

**Table 3.** Four motifs, with their names and music-time duration given in beats assuming a tempo of 120 beats per minute.

cause the visual component is re-edited, the composer adjusts the duration of affected regions to match new timings. The system then automatically recalculates repetition solutions while showing timing discrepancies on the time-line. The composer is free to choose a new repetition solution, adjust the tempo function, or even define a new timed regular expression.

EMuse [19], the prototype composition system described above, was implemented for Mac OSX using the Quick-Time framework to synchronize digital video and MIDI-generated music synthesized with Core Audio. EMuse was tested by using it to compose soundtrack music for several short films. We particularly examined the composer’s interaction with RepChooser, interested to see if it adequately supports a convergent process in which the composer discovers a sequence of ever better repetition solutions, culminating in a final selection.

## 7. AN EXAMPLE: GENERATING SOLUTIONS

To help clarify the mechanics of what the composition system does for the user when calculating repetition solutions, we now present a simple example. In this scenario, a user wishes to fill ten seconds of time with music. The user decides it is acceptable if the music is up to one second shorter or longer than this target duration. During this interval, the tempo is a constant 120 beats per minute. The user has defined four named motifs, as shown in table 3, with which to fill this interval. The timed regular expression describing how these motifs are to be combined is  $a^*b^*c^*d$ .

Given the above input, the system can now calculate repetition solutions. First, clock-time durations are converted to music time. At 120 beats per minute, ten seconds is twenty beats with a two beat error tolerance. Next, the timed regular expression is interpreted as a duration expression;

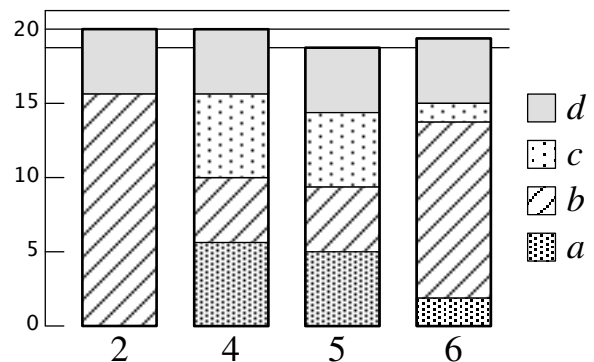
$$a^*b^*c^*d$$

becomes

$$18 \leq 1n_a + 2n_b + 1n_c + 4 \leq 22$$

#	$n_a$	$n_b$	$n_c$	Total Beats
1	14	0	0	14
2	0	8	0	16
3	0	0	17	17
4	6	2	6	16
5	5	2	5	14
6	3	5	2	15

**Table 4.** A small selection of valid repetition solutions chosen from the many possible solutions to equation 2.



**Figure 7.** 2D illustrations of repetition solutions 2, 4, 5, and 6 from table 4 similar to those presented by RepChooser. The beat duration of each solution is measured using the meter on the left.

which simplifies to

$$14 \leq 1n_a + 2n_b + 1n_c \leq 18 \quad (2)$$

The repetition unknowns for motifs  $a$ ,  $b$ , and  $c$  are  $n_a$ ,  $n_b$ , and  $n_c$  respectively. There is no repetition unknown for motif  $d$  as the user has explicitly given a count of one in the timed regular expression. The coefficient of each repetition unknown is the music-time duration for the corresponding motif. The linear system represented by equation 2 is now solved producing sets of values for repetition unknowns, some of which are listed in table 4. Each set of values represents a valid repetition solution fitting the duration requirement of the user.

The function of RepChooser is to allow the user to efficiently navigate a large number of valid repetition solutions in order to choose the most ideal solution. Figure 7 illustrates how a few of the solutions from table 4 might appear in RepChooser. Each repetition solution in figure 7 shows total music-time duration of the solution, including motifs that have fixed repetition counts. Figure 8 shows the same repetition solutions typeset in common practice notation.

## 8. CONCLUSIONS

The successful implementation of the EMuse system shows that music specifications, repetition solutions and tempo calculations together form a complete music track creation and editing system. It has been used by one of the authors



**Figure 8.** Repetition solutions 2, 4, 5, and 6 from table 4 typeset in common practice notation.

(EV) to produce several scores that accompany animated videos created by students in the Fine Arts Department at the University of Waterloo. These scores show the system to be able to produce scores that amply complement the videos, reinforcing mood that varies from cue to cue, even within the limitations of minimalist musical style.

Three questions about the system come immediately to mind. Can a professional composer of film music construct a large scale score using such a system? Unfortunately, such a question cannot be answered with the current prototype implementation. Two very large obstacles would have to be overcome to do so. First, the system would require extension and polishing that goes many person years beyond the effort available for our research. And second, the composer would have to spend many months experimenting with the system before being able to evaluate its full expressive potential. EV is a skilled performer with little experience composing, but has an intimate knowledge of the capabilities of the system, gained while implementing it. He can use the system without extensive training but lacks the training to stretch its limits.

In the hands of a professional composer, would the use of EMuse shorten the time to compose? EMuse is designed to help composers with particular mechanical aspects of soundtrack composition. With these mechanical tasks in the hands of EMuse, one would expect the composer to have more creative time. However, does EMuse alter the creative process enough to affect the efficiency of composition?

Clearly, incorporating the capabilities of EMuse into an existing composition system is the best way to evaluate its ability to assist professional composers. In such a context, its capabilities may be used to support only part of the film composition process: generating a time-accurate baseline from which a more complex score can be composed.

The last question comes from the other extreme: can a musical novice compose satisfying music for something like a home video using the system? Composing short note sequences is a first act of musical creativity and creating repetition patterns is straightforward with the system. RepChooser is easy to use when the number of motifs is small, and previous experience with OSA PlaneSight showed that proficiency grows quickly with use. With MIDI playback, the time from making a change to hearing

the result is short. Thus, as long as the user has the musical taste to recognize a successful score getting good results from the system with little training seems very likely.

Ordinary users easily composing personal music to accompany on-line video opens up a new dimension for web services like YouTube. Democratizing music creation, as various web tools have democratized writing and video, may well be the most important impact of tools like EMuse. To be sure, high musical culture is unlikely to be much enriched by such activity, but an analogy to electric guitars in garages and bedrooms fifty years ago is not out of place.

### 8.1. Future Work

EMuse, as presented, is an extremely simple composition tool. Its implementation intentionally neglects, for reasons of scope, several key musical issues. EMuse is the result of the presented study on how to programmatically mitigate some of the basic challenges facing a soundtrack composer. Future work will study ways to extend EMuse in more musical directions such that the composer has creative control while benefitting from automation.

The first group of extensions stay within minimalist music. The most obvious is enlargement of the music notation, to include dynamics, articulation and polyphony. In addition, musical transformations more complex than repetition can be included. The challenge in such additions is to maintain the simplicity of the interface, to keep the system open to users without musical training.

Another sort of extension would open the system to other musical styles, which is more challenging. Minimalist music was chosen for this project because it is easier to formalize than other styles of music. However, many styles are used in practice, not all of them relying on repetition as overtly as minimalism. The challenge is to include the musical transformations of other styles in EMuse's music notation without falling into the trap of requiring note by note composition. Meeting and overcoming this difficulty holds the future of EMuse.

## 9. ACKNOWLEDGMENTS

We gratefully acknowledge the funding for this research provided by the University of Waterloo and the Natural

## 10. REFERENCES

- [1] S. Abrams, R. Bellofatto, R. Fuhrer, D. Oppenheim, J. Wright, R. Boulanger, N. Leonard, D. Mash, M. Rendish, and J. Smith. Qsketcher: an environment for composing music for film. In *C&C '02: Proceedings of the 4th Conference on Creativity & cognition*, pages 157–164, New York, NY, USA, 2002. ACM Press.
- [2] Apple Computer, Inc. Apple – Final Cut Studio – Soundtrack Pro [online, cited November 24, 2005]. Available from: <http://www.apple.com/finalcutstudio/soundtrackpro/>.
- [3] Auricle Control Systems. Welcome To The Auricle [online]. 2005 [cited November 1, 2005]. Available from: <http://www.auricle.com>.
- [4] J. Bilmes. A model for musical rhythm. In *International Computer Music Conference*, pages 207–210, San Francisco, 1993.
- [5] R. Boulanger. *The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing, and Programming*. MIT Press, 2000.
- [6] M. Farbood and B. Schoner. Analysis and synthesis of palestrina-style counterpoint using Markov chains. In *International Computer Music Conference*, 2001.
- [7] H. Honing. From time to time: the representation of timing and tempo. *Computer Music Journal*, 35(3):50–61, 2001.
- [8] B. L. Jacob. Composing with genetic algorithms. In *International Computer Music Conference*, 1995.
- [9] D. Jaffe. Ensemble timing in computer music. *Computer Music Journal*, 9(4):38–48, 1985.
- [10] M. O. Jewell, M. S. Nixon, and A. Prügel-Bennet. Cbs: a concept-based sequencer for soundtrack composition. In *Proceedings of 3rd International Conference on Web Delivering of Music*, pages 105–108, 2003.
- [11] F. Karlin and R. Wright. *On the Track: A Guide to Contemporary Film Scoring*. Routledge, New York, 2nd edition, 2004.
- [12] J. W. Lai. Implementation of colour design tools using the OSA uniform colour system. Master’s thesis, University of Waterloo, 1991.
- [13] J. McCormack. Grammar based music composition. In *Complex Systems: From Local Interactions to Global Phenomena*. ISO Press, 1996.
- [14] W. Mertens. *American Minimal Music*. Alexander Broude Inc., New York, 1983.
- [15] S. Mishra. The “mkmusic” system - automated soundtrack generation for computer animations and virtual environments. Master’s thesis, University of Glasgow, Glasgow, Scotland, May 1999.
- [16] W. Schottstaedt. Pla: A composer’s idea of a language. In C. Roads, editor, *The Music Machine*. MIT Press, 1989.
- [17] W. Shottstaedt. Automatic counterpoint. In *Current directions in computer music research*, pages 199–214. MIT Press, Cambridge, MA, USA, 1989.
- [18] The Sibelius Group. Sibelius [online]. 2005 [cited November 1, 2005]. Available from: <http://www.sibelius.com/>.
- [19] R. E. Vane. Composer-centered computer-aided soundtrack composition. Master’s thesis, University of Waterloo, 2006. Available from: <http://etd.uwaterloo.ca/etd/revane2006.pdf>.