Caring for the Instrumentalist in Automatic Orchestration

NICK COLLINS
Centre for Electronic Arts
Middlesex University
Cat Hill, Barnet, Herts, EN4 8HT
UNITED KINGDOM
n.collins@mdx.ac.uk   http://www.axp.mdx.ac.uk/~nicholas15/

Abstract: - As part of a general scheme for automatic orchestration for human instrumentalists this paper focuses on the eventual performer of algorithmically generated music. A cost function is presented to measure the material they must play in terms of fatigue and difficulty of performance on the instrument. This cost is independent of attempting to match the character of music to a given instrument, and the exposure of a given instrument in the piece as a whole. We consider a self contained algorithmic composition module for orchestration of pre-composed material, though the material is adaptable to other circumstances. Assumptions of the model are made explicit in the text wherever possible. The paper ends with an example implementation, and a set of suggestions for further work. Whilst the model has its flaws, the aim of this paper is as much to promote consideration for the instrumentalist playing an algorithmically composed score.

Key-Words: - automatic orchestration, algorithmic composition for acoustic instruments

1 Introduction

Orchestration is just one task in algorithmic composition, and has not always been a historically prominent one. It is pleasing then to see Lejaren Hiller in [5] discussing instrumentation in the second movement of his Algorithms 1 thus:

'I used an instrumentation process for this movement that interchanges instruments, giving the high wind instruments breathing space. The individual lines can be quite long and taxing otherwise'

Hiller understands that if scores for human players do not take account of the performer's fatigue they will not get an accurate performance. One can also consider the physical technique required for an instrument. Charles Ames' composition for piano *Gradient* [1] only utilises chordal spans that a human hand might comfortably reach.

In contrast, Koenig's Project One program was never intended to orchestrate, only to provide the material for the composer to fashion into a finished score- 'the balanced distribution of the material permits different interpretations' [6]. Project Two is a step into automatic orchestration and will produce a finished score ready to write up, though the amount of consideration for the performer is doubtful. Discussing Koenig's Projects One and Two, Laske [7] points out the parametric conflicts that writing for instruments causes- 'Only a limited number of pitch classes, octave registers, durations, amplitudes and modes of performance can be realised with any chosen instrument'. Yet this knowledge is only part of a consideration of the technique of an instrumentalist. It is not just what a performer can do for any one event, but the ease of performance of a sequence of events over the course of a piece.

Surveying the literature of algorithmic composition, it is interesting how rarely the instrumentalist is mentioned as an important concern. The difficult leaps of a pointillistic style abound in algorithmic composition (see the score examples in [2]). As an example of a less caring orchestration procedure, the piece Entre l'Absurde et le Mystere for chamber ensemble described in [9] had the following instrumentation process 'Finally the user specifies the instrumentation of the composition by associating each possible state of the Demon Cyclic Space with a General MIDI Instrument' An orchestration that favours the players will only occur by chance. On the other hand, this piece received a performance in 1995, but this doesn't mean every note is playable.

Perhaps it is tempting not to prioritise the instrumentalist because professional players can be expected to attempt to play almost continuously. A flautist confronted with one minute of semiquavers will breathe whenever they must to navigate the passage as successfully as humanly possible. Certain instruments, like strings or the piano, might be considered practically inexhaustible! Statistical distributions or other methods utilised in algorithmic composition can be fine-tuned to spread notes in a such a way that nothing too arduous for any one given performer occurs. The scaling of a process will avoid notes that are too quick to play accurately or too long to play at all. Xenakis's Stochastic Music Program ([11] pp 837-838) has a parameter for the longest duration playable by each instrument, and Xenakis is not adverse to tweaking output ([11] pp 845). Yet one can only remove or disfigure grossly difficult passages by hand if the material is not critical. In an orchestration of a pre-existing composition, or an algorithmically generated piece where the minutiae are still vital, this kind of post-process meddling will invalidate the orchestration process, a process which should really be rethought.

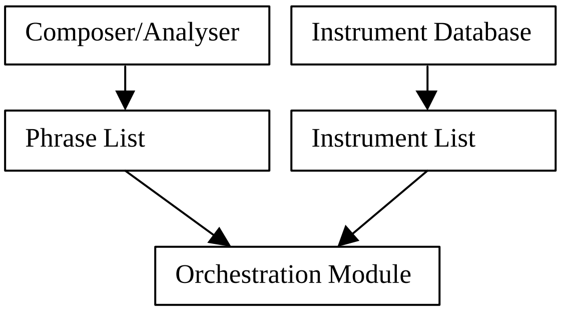
In this paper a cost function is presented to measure the difficulty of monophonic phrases for particular instruments in terms of how tiring those phrase are to play, and how technically demanding, though without specialisation to the sound production methods of each instrument. This function is supposed to raise awareness of the eventual performer of the algorithmically composed score and make the performer a more central concern of an algorithmic composition.

The study connects to the area of expert systems for orchestration. A composer's assistant for orchestration is discussed in [10]. The IOS system described therein searches out phrases and allows allocation of features of phrases to given instruments, but has no provision for the instrumentalist themselves (to be fair, Roads' work is also geared towards digital sound realisation). Automated orchestration is definitely a current topic. In a recent survey of developments at IRCAM ([16], also [3]), Hughes Vinet claims that researchers are working on automated assistance for orchestration within the OpenMusic project, though no precise details are given.

In the following, the orchestration module is separate to the musical material, and it is assumed that the material is already parsed into a particular format. Considering the act of orchestration independently of the act of composition of musical material is a terrible simplification for real composition straight to orchestra, but still an instructive one for the act of arrangement and orchestration itself. There is much work on segmentation and analysis, and algorithmic composition ([4], [14], [15]), relevant to what would be required to produce an input for the orchestrator. Finally, a good review of algorithmic composition from the perspective of connectionism is in [8].

2 The Model

2.1 The Orchestrator Module



The orchestrator model described here is relatively simple. The input is a list of monophonic phrases in a suitable format, comprising the entire musical material of the piece, and a list of instruments to which allocation of individual phrases must be carried out. As each phrase is assigned to a given instrument, it is removed from the list of phrases to be allocated. There is no attempt to consider co-temporality of phrases, or any hierarchical structure of musical phrases. The allocation is entirely based on a measure of the UNSUITABILITY of the current phrase for each potential instrument that could play it. The instrument with the least UNSUITABILITY 'wins' the phrase.

Our aim is to set up some sensible measure of the difficulty of a given phrase. In order to do so, we separate the measure of the phrase difficulty from notions of character matching and instrument exposure within the piece.

UNSUITABILITY= SOMEFUNCTION (FATIGUE, OUTOFCHARACTER, OVEREXPOSURE)

For our purposes, let SOMEFUNCTION be a linear weighted sum. Then UNSUITABILITY equals

W_F* FATIGUE + W_C*OUTOFCHARACTER + W_E*OVEREXPOSURE.

where for normalisation the weights W_F, W_C and W_E sum to 1.0, and FATIGUE, OUTORCHARACTER and OVEREXPOSURE are all functions of a given instrument, the current phrase to be allocated, and all previously allocated phrases, taking value from 0.0 to 1.0.

FATIGUE measures how difficult a given phrase is to play on the instrument in terms of physical technique required and cumulative tiredness of the performer as they play. It might also be called CANITBEPLAYED? It depends on what that instrument has been previously allocated to play over the course of the piece (the player may already be tired before they play this new phrase). The function depends on the instrument to be tested only.

OUTOFCHARACTER is a measure of how suitable a given phrase is to a given instrument. IOS [10] allows a composer to specify particular phrase characteristics to map to given instruments. Alternatively, there might be a connectionist model [15] to map particular phrases to given instruments based on a long training period over examples in the literature. This function only depends on the current phrase, not previous allocations.

OVEREXPOSURE is a measure of how much a given instrument has been used already, and in the extreme case, whether it is available at all at that moment in time according to the design of the piece (the orchestrator module may assign such restrictions itself before it begins to try to allocate phrases to instruments). Some kind of statistical scheme would be the most likely candidate for the OVEREXPOSURE function. This function depends on all previous allocations over all instruments.

Conversely, we could measure suitability as a combination of NOFATIGUE, CHARACTER, UNDEREXPOSURE.

In the following, CW = EW = 0.0, and FW = 1.0. It is not that a general orchestrator module would not find a character measure important, or would not consider how much a particular instrument has been heard, just that herein we concern ourselves specifically with FATIGUE. For a more general orchestrator module, the UNSUITABILITY measure depends on the instrument, and for strings might weight fatigue lower than exposure, but for woodwind, rate fatigue a higher concern.

2.2 Assumptions of the orchestrator module

It is thought wise to make a list of some of the many assumptions that the model presented here takes.

- The orchestrator holds no psychoacoustic data on such topics as masking and loudness curves, and cannot confirm the likely audio output of players realising a score. The orchestrator cannot test the balance of instruments at a given time.
- All dynamic indications are on an absolute dynamic scale. Human instrumentalists will vary their interpretation of dynamic markings relative to the orchestral forces playing at the time, and their pitch within the register of their instruments.
- The psychology of music is not a solved subject area, so we cannot assume understanding of the mental state a performer will be in after viewing a given piece of music!
- The allocation of a given phrase is independent of any other phrase. There is no consideration of simultaneity. There will be no grouping of instruments or seeking of homogenity of tone or particular texture at a particular point.
- The instrumentalists act as soloists. So two flutes would be independent instruments, or one instrument receiving exactly the same material. There is no attempt to sustain a long note by staggering entries of two flutes, say.
- Phrases and Instruments are monophonic
- Pitch is 12 tone equal temperament
- Pitch does not vary over a given note (no portamento and glissando)
- Rests during a phrase do not allow any recovery (with corollary- the cost of a phrase is independent of the fatigue level before you began the phrase)
- Ornamentation and articulation are not modelled
- Stresses naturally given by players by strength of each beat in a given bar are not taken into account.
- The fatigue measure is currently most applicable to woodwind, brass, and voices, that is, where breath is the deciding factor. The possibility of circular breathing is ignored.

2.3 Timing information for a piece

The basis of the piece has already been composed, or extracted by some analysis; the list of phrases. We require timing information in seconds. Estimates of how long a player can blow must be made in seconds, not beats! We ignore time signatures by ignoring the position of a note within a bar (ignoring any stress a player would place on a particular beat), and assume tempo changes are already known and taken account of in the timings. It is envisaged that given timings in beats, plus the list of tempo changes, a pre-processing step turns all timing information into seconds.

2.4 Definitions of basic musical objects

A **dynamic** is in the range 1 to 8, where 1 is ppp and 8 is fff.
A **dynamic curve** is a continuous function of dynamic over time. The dynamic curves used herein are dynamic keyframe sequences over time with linear interpolation.

Example A crescendo from ppp to mp followed by a steady dynamic over 10 seconds might be represented by keyframes {(0,1), (5,4), (10,4)}

A **note event** has the attributes of: pitch, taking values 0-127 equivalent to the MIDI pitch scale and duration in seconds.

A **phrase** consists of a map over time of non-intersecting note events and a dynamic curve defined over the same span as the map of notes. The condition of non-intersection forces monophonicity of phrases.

2.5 Instruments

Each instrument is given particular values of the following parameters:

Table 1 Instrument parameters for the FATIGUE cost function

Shorthand	Parameter	Type
LBRANGE	Lower bound of instrument range	MIDI PITCH (0-127)
UBRANGE	Upper bound of instrument range	MIDI PITCH
LBCOMF	Lower bound of comfortable register	MIDI PITCH
UBCOMF	Upper bound of comfortable register	MIDI PITCH
MAXLEAP	Maximum comfortable leap	MIDI PITCH
MINDUR	Minimum comfortable duration	TIME in seconds
MAXPLAY	Maximum phrase length playable	TIME
RECRATE	Recovery rate during rests	positive real

As well as the standard range of a professional player, we model a 'comfortable range' being that portion of the instrument where the player is most at ease in playing. The maximum comfortable leap and minimum duration give a measure of the agility and speed capabilities of the instrument. The MAXPLAY parameter measures how long a performer can play for on the given instrument with a single sustained tone at average dynamic (mf). It is highly pertinent to breathing. For percussive instruments which require no energy to sustain, or string players who could continually bow for half an hour, this parameter is interpreted as the longest the orchestrator wants to let an instrument play a succession of even pulses above the MINDUR rate.

Example a piccolo
LBRANGE=74, UBRANGE=108
LBCOMF=76, UBCOMF=96
MAXLEAP=17, MINDUR=0.05,
MAXPLAY=10.0, RECRATE=1.0

The piccolo is an agile instrument, and the small body means it is not too tiring to play (though it is a tricky instrument to control). The bottom Eb and D are very tough to play whereas the E is easy. The real difficulties of the instrument occur going in cold for high notes. The model does not cover the case that leaps up are easier than the equivalent leap down.

2.6 Defining the FATIGUE cost function

2.6.1 FOVERP (Fatigue over a phrase) cost function

FOVERP =

$$\frac{w * DURCOST + x * LEAPCOST + y * SPEEDCOST}{MAXPLAY}$$

where w, x, y are weights. DURCOST(duration cost) is given by the formula

$$\sum_{i=1}^{N} DYNINT(i) * registerfn(pitch(i))$$

where DYNINT(i)=

$$\int_{start(i)}^{duration(i)} dynfn(dc(t)) * instrdynfn(dc(t), pitch(i))\,dt$$

LEAPCOST=

$$\sum_{i=1}^{N-1} leapfn(abs(pitch(i+1) - pitch(i)))$$

SPEEDCOST=

$$\sum_{i=1}^{N} speed\cos tfn(duration(i))$$

The summations are over the N notes of the phrase. The rests in a phrase will be naturally ignored. Functions of i are properties of the notes of the phrase- pitch(i), start(i) and duration(i) are respectively the pitch, start time in seconds and duration in seconds of the i^{th} note. abs is the absolute value of the integer argument. dc(t) is the value of the dynamic curve at time t and the functions dynfn, instrdynfn, registerfn, leapfn and speedchangefn are given concrete forms below utilising the instrument data from table 1.

dynfn(dc(t))=
1.0 if dc(t) ∊ [4.0, 5.0] or otherwise

$$1.0+\frac{VC*(abs(dynamiccurve(t)-4.5))}{3.5}$$

The assumption is that a very soft dynamic is the same effort as a very loud one. VC is a constant, taken as 1.0 for testing. The DURCOST measure is integrating dynamic effort over the durations of all notes.

instrdynfn(d,p)= 1.0 for all d, p (constant function)

This assumes there is no difference in dynamic difficulty at different points of a range. This is not true for a horn for example (mf only is practicable at the extreme top of the range) but is a helpful simplification. The summand DYNINT(i) is now independent of the instrument and only needs to be calculated once for each phrase.

registerfn(p)=
1.0 if p ∊ [LBCOMF, UBCOMF]
1.0+ (LBCOMF-p)/(LBCOMF-LBRANGE)
if p ∊ [LBRANGE, LBCOMF)
1.0+ (p-UBCOMF)/(UBRANGE-UBCOMF)
if p ∊ (UBCOMF, UBRANGE]
infinity if p<LBRANGE or p> UBRANGE

registerfn(p) is linear outside the comfortable range to the limits of the instrument range.

leapfn(p)= 0.0 if p ≤ MAXLEAP otherwise

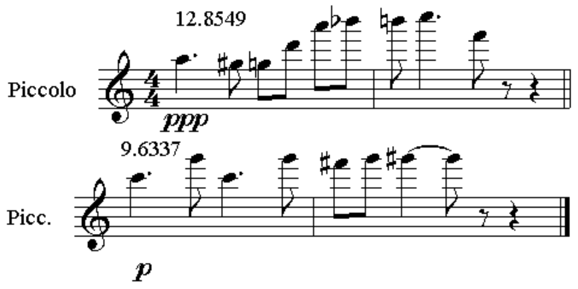
$$1.0-\frac{p-MAXLEAP}{UBRANGE-LBRANGE-MAXLEAP}$$

There is no need to check whether a leap stays in range for registerfn already deals with this case.

speedchangefn(d) =

0.0	if d≥ MINDUR
1.0- d / MINDUR	otherwise

Example contrasting two phrases for piccolo (Weights w = x = y = 1.0, functions as above)



These two phrases are marked with the the value of FOVERP. The values are either side of the MAXPLAY cutoff of possibility (10.0), so according to the cost function, the top phrase is unplayable, whereas the bottom one is. We can see quickly why the higher phrase is assessed with the higher FOVERP score, for it is at dynamic ppp at the very top of the piccolo's range. A piccolo player would probably demonstrate that this measure does not correlate to real life by playing the top phrase to us! However, the cost function is hardly precisely tuned at the moment (see the further work section). The weights allow scaling to set up the cost function to the composer's specification, and there is no law requiring us to use MAXPLAY as a strict cutoff!

FOVERP returns infinity where a phrase is definitely impossible on the instrument (perhaps out of range). FOVERP is normalised with respect to MAXPLAY. Values below 1.0 have not exhausted the player beyond their capabilities.

2.6.2 FATIGUE after allocation of a new phrase

The FATIGUE function used by the orchestrator function must take account of all phrases previously allocated to an instrument. So for a given phrase P, FATIGUE(P)=

$$\sum_{Q<P} FOVERP(Q)-\frac{RECTIME*RECRATE}{MAXPLAY}$$

The summand is over all phrases earlier in time than phrase Q. RECTIME (recovery time) is the time in seconds that the instrument has not been playing since the beginning of the piece up to the beginning of phrase P. There are two provisos. The first is that FATIGUE(P) returns infinity if the allocation of P would intersect with any previously allocated phrase Q. The second is that FATIGUE(P) returns infinity if the allocation of P to the instrument would mean that any phrase later than P now becomes over-fatiguing. This proviso only matters if phrases are being allocated free of linear time (otherwise, we already know phrase P to be allocated cannot occur before an earlier allocated phrase).Checks for FATIGUE are easy, since the FOVERP measure can be made once only for a given phrase on the instrument and then stored.

3 The Model in Practice

3.1 Search Procedures

I use terms from the work of Charles Ames ([2]). The orchestration process as given here always critically depends on the phrase list order.
A comparative search is not practical. If each phrase could only be allocated to two possible instruments on average, the number of possibilities over N phrases is 2^N, which will become too large for large N.

If all phrases must be allocated exactly as in the list, a constrained search with backtracking can be used to hunt through the allocations to find a solution. We might have to ignore the best fit instruments for a given phrase at certain stages to avoid over-fatiguing particular players.

Everything becomes more complicated if the orchestrator can make compositional decisions. If a phrase cannot be allocated to any instrument, we might allow the module to split the recalcitrant phrase in half, or even to make a split based on the amount a given phrase instruments can play before they become over fatigued. Further, in an orchestration, the orchestrator may assign the same phrase multiple times to a pertinent instrumental combination.

3.2 Implementation

A C++ program was constructed to test the cost function. This program also included a composition module for the phrase list. The output was a MIDI file which was imported into the score editor Sibelius. Automatic orchestration using the fatigue cost function alone was used to produce Palette 3 of the author's *24 Palettes*. The score can be viewed on the author's web site. It is not reproduced here for reasons of space, and since the FATIGUE measure will shows itself over the long term.

3.3 Further work

The cost function here is drastically flawed as a model of real instruments and performers, but is the beginning of what one believes is a laudable study. It is obvious that an empirical survey of instrumentalists and consultation with orchestrators would be a natural next step. Measurement of exactly how long instrumentalists can play at a given dynamic level on a given note, articulation and rate of pulsation without exhaustion would be extremely pertinent, as well as difficulties of particular changes of notes, dealing separately with leaps up and down between any two tones. Real orchestrations are a natural source of knowledge.

The instrument model here is weak. A hierarchy of classes could be introduced to cope with different instrumental families and specific instruments. Whilst each instrument would present a common interface to the orchestrator, they could calculate cost of phrases in an independent way. This would make it easier to model specific articulations like sul ponticello. The breaks on a flute occur between C#5 and D5, C#6 and D6 etc. The leap cost for a flute would ideally take this kind of transition into account. The model also assumes that leaps up are the same difficulty as an equivalent leap down, another problem that is solved by costs for each possible leap. The work of Sayegh [13] is extremely pertinent to the production of music for stringed instruments lying comfortably on the instrument.

The time and effort required for non-sounding actions like adding mutes, switching harp pedals or re-tuning timpani would also be factored into the cost function calculation, perhaps by adding extra time onto the front of a phrase as preparation time within the instrument's cost calculation. However, if we find ourselves considering time for page turns we might be taking this scheme too far!

There is much investigation to be done into different definitions for dynfn et al in the general FOVERP measure.

The model allows the possibility of altering the phrases themselves in difficulty. This is tantamount in some cases to recomposing. One might envisage a phrase being handed back to the composer module to redo in a way compatible with the composition, perhaps to a different dynamic or a less pointilistic pitch distribution (whilst preserving pitch classes). If the phrase cannot be changed in a consistent manner, then we get into difficulty, for removing the phrase from the orchestration destroys the integrity of the piece!

The orchestration process can be used by the composed module to do work for it- by deliberately composing difficult material, the composer lets the orchestrator break up that material into more likely units.

There are many factors to take into account, and the cost function would never become perfect, but at least we would be thinking of the performer more than without it.

3.4 Conclusions

The wise composer will be aware of the limits of the technique of the professional performer, and the wise algorithmic composing program can also attempt to embody such knowledge. The amount of knowledge associated with orchestration is very large, considering the great number of acoustic instruments, particularly through world cultures! This paper only describes one aspect of setting up an independent orchestrator module. As long as musical material conforms to the input specification, the orchestrator might be acting on novel algorithmically composed material, attempting to set a piano piece for full orchestra, reduce a piece for full orchestra to a piano piece or rework a piece for different instrumental forces.

The reader may believe that this paper is only relevant to the case of automatic composition for acoustic instruments. However, the material could be adapted to 'humanising' synthesised instruments. Finally, another curious application of this material might be to create algorithmic composition for amateur musicians within a certain difficulty level!

3.5 Acknowledgements

Many thanks to Gordon Davies for his observations and a tutorial on woodwind and brass instruments.

References

[1] Charles Ames, Stylistic Automata in *Gradient, Computer Music Journal*, Vol 7, No 4, 1983, pp 45-56

[2] Charles Ames, Automated Composition in Retrospect 1956-1986 *Leonardo*, Vol 20, No 2, 1987, pp 169-186

[3] Gerard Assayag et al, Computer Assisted Composition at IRCAM: From PatchWork to OpenMusic *Computer Music Journal*, Vol 23, No 3, 1999, pp 59-72

[4] Dennis Baggi (Ed), *Readings in Computer Generated Music,* IEEE Computer Soc Press, 1992

[5] Lejaren Hiller, Composing With Computers, A Progress Report , in [12]

[6] Gottfried Michael Koenig, Aesthetic Integration of Computer-Composed Scores , in [12]

[7] O Laske Compositional Theory in Koenig's Project One and Project Two, in [12]

[8] D. Gareth Loy, Connectionism and Musiconomy in [15]

[9] Kenneth McAlpine, Eduardo Miranda and Stuart Hoggar, Making Music with Algorithms: A Case Study System, *Computer Music Journal*, Vol 23, No 2, 1999

[10] Curtis Roads, Interactive Orchestration Based on Score Analysis, *Proceedings of the 1982 International Computer Music Conference 1983*, pp 703-717

[11] Curtis Roads, *The Computer Music Tutorial*, MIT Press, 1995

[12] Curtis Roads (ed), *The Music Machine,* MIT Press, 1989

[13] Samir I Sayegh, Fingering for String Instruments with the Optimum Path Paradigm, in [15]

[14] Schwanauer and Levitt (eds), *Machine Models of Music*, MIT Press, 1993

[15] Todd, Peter M and Loy, D.Gareth (eds), *Music and Connectionism*, MIT Press,1991

[16] Hughes Vinet. Recent Research and Development at IRCAM *Computer Music Journal*, Vol 23, No 3, 1999, pp 9-17