Non-overlapping, Time-coherent Visualisation of Action Commands in the AscoGraph Interactive Music User Interface

Grigore Burloiu
University Politehnica of Bucharest
Faculty of Electronics, Telecommunications
and Information Technology
gburloiu@gmail.com

Arshia Cont
MuTant Team-Project
IRCAM STMS UMR, CNRS, INRIA, UPMC
cont@ircam.fr

ABSTRACT

Integrated authoring and performing of mixed music scores, where musicians interact dynamically with computer-controlled electronics, is enabled by the *Antescofo* state-of-the-art software package. Composers are able to plan computerised actions through a dedicated programming language, and performances are then synchronised in real time. *AscoGraph* is the dedicated graphical interface that allows users to configure *Antescofo* behaviours and visualise their layout over a mixed music score. This paper presents developments in the direction of increased clarity and coherence of *AscoGraph*'s visualisation of computerised action scores. Algorithms for efficient automatic stacking of time-overlapping action blocks are presented, as well as a simplified model for displaying atomic actions. The paper presents the improvements in score readability achieved, as well as the challenges faced towards a complete representation of dynamic mixed scores in the *AscoGraph* visual environment.

1. INTRODUCTION

This paper describes a model of interactive visualisation for composition and performance of mixed music repertoire. Mixed music is commonly referred to as the live association of human musicians with interactive software/hardware during live music performance; as well as the authoring (composition) within this mixed medium. Among common practices of mixed music, *score following* has been an active line of research where the computer is equipped with a real-time machine listener that dynamically aligns a musician's performance to a pre-written music score and decodes performance parameters, which can be used to interpret and evaluate computerised actions. One can think of such a compositional paradigm as an extension of musical automatic accompaniment application, where accompaniment playback is replaced by programs acting on various aspects of sound and music computing.

Visualisation for mixed music is a challenging task for several reasons. The score is a joint combination of two main components: one that describes expected events from human musicians as extensions of classical musical notation; and one that describes computerised, or electronic actions. The two components are strongly-timed and most often aligned during authoring and synched dynamically during performance. Computerised actions in turn have heterogeneous time models: they can be discrete message passing, or continuous curves, or dynamic calculations. Their temporal ordering can be described as sequences of delays expressed in absolute or relative time; actions can be hooked sequentially (through delays inside a single sequence) or vertically (to an external event, condition or synchronisation pivot). Composers and performers are proficient at dealing with authoring and interpreting a variety of such parameters.

Enabling this level of expressivity in mixed music composition and performance is the goal of the *Antescofo* software, a state-of-the-art system for mixed music composition which integrates a score following engine [1] and a synchronous reactive programming language [2]. The user/composer is able to plan complex dynamic electronic actions which the system launches and controls during the performance, in sync with the live musicians' tempo. First described in [3], *AscoGraph* is the dedicated graphic development environment which enables visual feedback for authoring and performing *Antescofo* mixed scores.

The *AscoGraph* workspace consists of two main sections: the textual score editor and the graphical editor, which in turn is split into an instrumental section and an electronic actions section. The instrumental view and electronic view are coupled in musical time along a common horizontal timeline. During a performance, *Antescofo*'s score follower determines the position of *AscoGraph*'s graphical cursor along the timeline.

This paper presents updates to *AscoGraph*'s electronic action view, developed with two directions in mind: (1) a clearer and more time-coherent visualization of *Antescofo* scores, and (2) a step towards a complete, self-contained visual notation format for mixed music scores. Section 2 presents the problem of overlapping action blocks, recast as a subset of the two-dimensional strip packing problem. The following section shows the three proposed algorithms for re-arranging action blocks. Section 4 tackles the issue of coherence between block width and musical time. We conclude the paper with an evaluation of the present model and future perspectives.

This project was partially funded by the French ANR INEDIT Project.

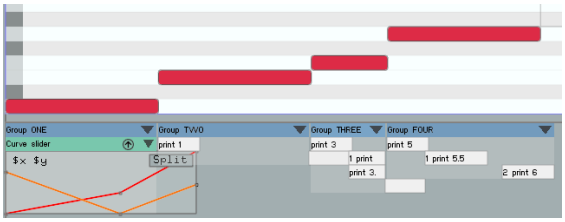


Figure 1: The "classic" action block display in *AscoGraph*. The musical timespans of Groups ONE, TWO and THREE are not represented clearly because of the overlapping of group blocks.

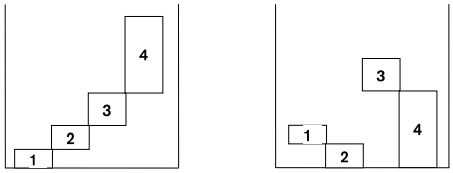
2. PROBLEM DEFINITION

We distinguish between physical time (measured in seconds) and musical time (measured in beats). The amount of physical time elapsed between actions depends on the tempo detected during performance, and on the active synchronisation strategies [2]. Meanwhile, *Antescofo* scores are specified in musical time. Since *AscoGraph* was primarily designed as a score visualisation tool, it employs a musical timeline. When a physical time unit is specified in a score (e.g. "after 2 seconds"), in order to display it *AscoGraph* must first translate it to an ideal musical time (e.g. "after 4 beats at 120bpm").

Fig. 1 shows an example of the original *AscoGraph* action block arrangement style. Here, each of the four notes (drawn in red in the instrumental view's piano roll) has one corresponding action group block. Durations can – and often do – differ between the length of a note and that of its associated electronic actions. While actions within a single group (e.g. Group FOUR) are stacked consecutively downwards, when two different action groups are partially concurrent, the second group is drawn over of the first. Consequently, the first group's duration is no longer clearly shown; things become even more confusing when overlapping automation curves (e.g. the one in Group ONE) are involved.

In order to rectify this loss of coherence and clarity, the need arises to stack action groups in downward non-overlapping order, similarly to how elements *within* groups are arranged. As the challenge becomes one of efficient management of 2D space, it is useful to describe it as a two-dimensional *strip packing* problem. A subset of *bin packing*, strip packing is used in areas ranging from optimizing cloth fabric usage to multiprocessor scheduling [4]. Algorithms seek to arrange a set of rectangles within a 2D space of fixed width and bottom base, and of infinite height. In our present case, the width of the strip corresponds to the total duration of the piece, and the rectangles to be placed are the action group blocks.

A particular constraint separates our problem from the rest of the bin packing literature. Unlike in existing bin packing problems, all *AscoGraph* action blocks must retain their X coordinate along the time axis. Since we are not allowed to "nudge" blocks horizontally, relying on existing packing algorithms becomes impractical.



(a) FF (b) OPT (FFD)

Figure 2: Horizontally constrained strip packing. Boxes are numbered in temporal (horizontal) order. This layout is arranged by FF in (a) and optimised by FFD in (b).

3. PACKING ALGORITHMS

We introduce three new algorithms for stacked action group display in *AscoGraph*'s graphical editor. The user can switch between one of them and the original display style through the application's *View* menu. The appropriate option will depend on score complexity and the user's personal taste.

Please note: following bin packing convention, we shall consider the rectangles as being placed *on top of* the strip base. Naturally, in the *AscoGraph* environment the situation is mirrored and we build *downwards* starting from the upper border.

3.1 First Fit (FF)

The first option is the trivial solution of placing the blocks in the first space they will fit, starting from the base. The benefits of this option are speed and predictability: blocks are placed in the order in which they appear in the source code text, which is also their scheduled temporal order.

The downside can be intuited from Fig. 2a and b. We propose a worst-case scenario: a set of blocks with increasing heights and, for simplicity, all equal widths. While FF would stack them on top of each other (Fig. 2a), the optimal method would stack them two by two (Fig. 2b), so that the maximum height is given just by the final two elements.

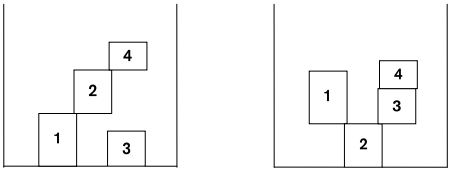
3.2 First Fit Decreasing (FFD)

Note that in the previous case, the optimal configuration can be reached by simply reordering the blocks by height. This insight lies at the root of the classic FFDH strip packing algorithm [5]. In our case, the FFD algorithm orders the blocks by non-increasing height, after which the First Fit process is applied.[1] Fig. 3a shows an FFD arrangement, along with the optimal solution at Fig. 3b.

3.3 First Fit Decreasing Towers (FFDT)

Again, the optimal configuration in the previous example points towards the next algorithm. We propose a greedy heuristic that builds upon FFD while tackling *AscoGraph*-specific situations like one action block sharing time with several blocks on both sides of it. The basic goal is to

[1] The difference to the classic FFDH algorithm is the absence of horizontal levels. New blocks are stacked at the minimum possible altitude rather than a common level.



(a) FFD (b) OPT (FFDT)

Figure 3: Horizontally constrained strip packing. Boxes are numbered in temporal (horizontal) order. This layout is arranged by FFD in (a) and optimised by FFDT in (b).

minimise gaps, such as the one between blocks 2 and 3 in Fig. 3a.

The FFDT algorithm first orders all blocks as in FFD. Then, action group *towers* are defined at the time-axis intersections between two or more group blocks. Their height is equal to the sum of the heights of their component blocks. For instance, in Fig. 3a and d the rectangle 2 is part of four towers: $\mathsf{T}_a\{r_1, r_2\}, \mathsf{T}_b\{r_2\}$ [2], $\mathsf{T}_c\{r_2, r_3\}$ and $\mathsf{T}_d\{r_2, r_3, r_4\}$.

The entire width being now split along these virtual vertical strips (towers), we are able to refine the ordering of the blocks. The first criterion is the decreasing *maximum height* among the *towers* each block is a member of. If this maximum tower height is equal for two blocks, then the second criterion is decreasing *number of towers* each block is a member of. If this number is equal as well, we leave the FFD ordering (non-increasing block height) untouched.

By definition, the maximum tower height is a definite lower bound of any *AscoGraph* strip packing configuration. Therefore, in the FFDT heuristic the tallest tower will always be placed first, in an attempt not to overshoot this lower bound. Among its component blocks, the ones that are shared with many other towers are dropped closer to the base - the intention again being to maintain tower integrity as much as possible. Lastly, as with FFD, tall blocks are prioritised so as to fill gaps efficiently.

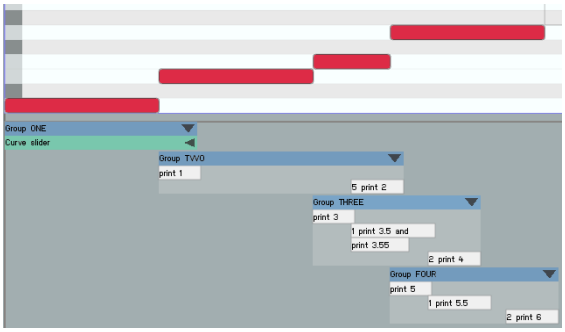
An *AscoGraph* use-case comparison of the three algorithms can be seen in Fig. 4. In it, we illustrate the orderliness of FF and the compactness of FFDT, with FFD a potentially useful compromise between the two.

4. TIME COHERENCE OF ATOMIC ACTIONS

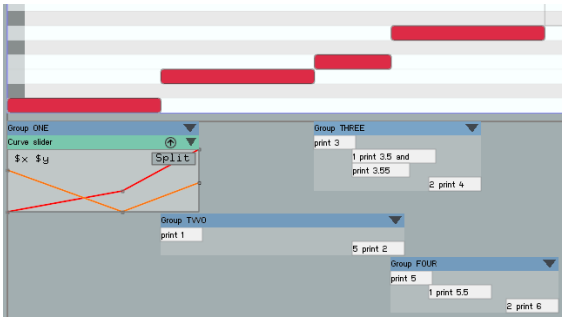
A basic element in *Antescofo*'s reactive language is the atomic action. Atomic actions can be part of larger dynamic constructs, but often they are simple messages to be triggered at a specific point in time. Since they are instantaneous, their visual representation taking up horizontal space on the action timeline is discordant. Moreover, as figures 1 and 4 show, they often clutter the workspace unnecessarily.

Our solution to more accurately represent action messages is to group all instances from a specific hierarchical level and display them on a single line as small circles, or conceptual *points*. When the mouse hovers over such a

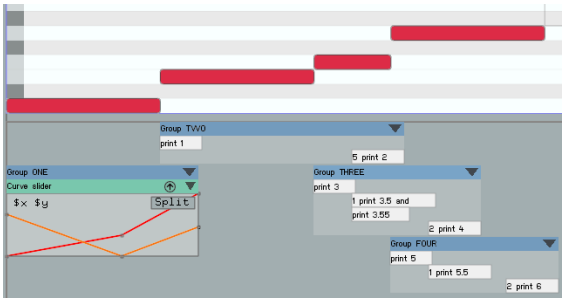
[2] A minimal tower only contains one action block.



(a) FF - action blocks are placed as close to the baseline as possible, in the order in which they appear in the code.



(b) FFD - blocks are ordered by height before being placed.



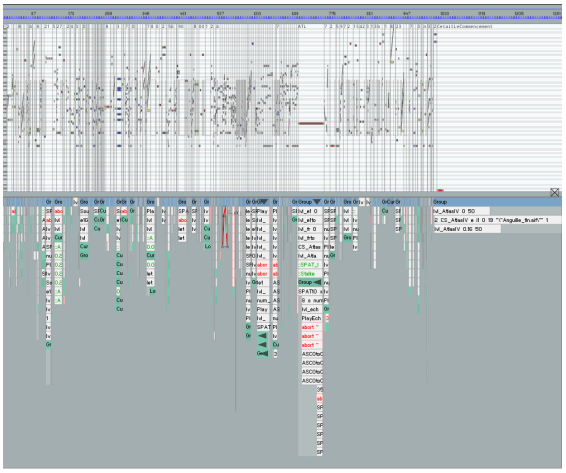
(c) FFDT - blocks are ordered according to a gap-minimisation heuristic before being placed.

Figure 4: The three *AscoGraph* packing options.

point, a list of the messages it contains is shown. Fig. 5 shows the expanded list for Group THREE; the messages are set at 3 different points in time, which is why 3 points are present in the message line.

Our new model is fully time coherent and considerably clearer than before. The user experience improvement over the classic model becomes most obvious when dealing with complex scores with many messages - see Fig. 6. With the timeline fully zoomed out, the old model offers a less accurate overview of the activity in the electronic score. Action durations are impossible to estimate; the most egregious problem being at the final note of the score, where, with

Figure 5: Time-coherent message circles display



(a) old model: blocks overlap, messages occupy horizontal space

(b) new model: blocks are stacked, messages are grouped in time-coherent points

Figure 6: Comparison of old and new *AscoGraph* models over a complex score.

nothing to stop them, musically instantaneous message actions take up an inordinate amount of space in the timeline. Meanwhile, the new model neatly groups messages together and offers a clear view of action block distribution in time and individual durations.

5. CONCLUSIONS AND FUTURE WORK

We have shown an improved layout mechanism for electronic action groups over a musical timeline in *AscoGraph*. By stacking action group blocks we ensure information integrity and coherence, while expanding the vertical real estate used. The most basic stacking method, First Fit, is also the most easily readable option for scores of moderate depth. We also proposed two increasingly efficient stacking algorithms, FFD and FFDT, for scores containing larger concentrations of actions per time unit. While superior algorithms are technically conceivable (possibly a metaheuristic scheme built on top of FFDT), the present options were deemed appropriate for the practical use and the processing overhead of the *AscoGraph* software.

Finally, we have introduced a method of displaying related messages on a single line which preserves group hierarchy. The main advantages are time coherence and vertical compactness. Still, this model can be seen as a compromise in our quest for a completely specified, self-contained visual notation format which we proposed in the introduction. Dynamic constructs from the *Antescofo* language are in a similar situation. For instance, a Curve whose duration is a dynamic variable: in this case, *AscoGraph* cannot know its exact plot over time before execution.

Therefore, one direction of future research is a *performance simulation mode*, decoupled from the compositional display described thus far, in which all messages, Loops and other dynamic constructs are represented as they "happen" in an offline simulation. This function is currently in prototype form, having been first described in [3].

However, the need remains for a graphic compositional model that clearly describes dynamic behaviour and action results. With the growing crystallisation of *Antescofo*'s language into a mature, stable package, the path is now open for research in this direction.

6. REFERENCES

[1] A. Cont, "Antescofo: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music." in *International Computer Music Conference (ICMC)*, Belfast, Ireland, Aug. 2008, pp. 33–40. [Online]. Available: http://hal.inria.fr/hal-00694803

[2] J. Echeveste, A. Cont, J.-L. Giavitto, and F. Jacquemard, "Operational semantics of a domain specific language for real time musician-computer interaction," *Discrete Event Dynamic Systems*, vol. 23, no. 4, pp. 343–383, Aug. 2013. [Online]. Available: http://hal.inria.fr/hal-00854719

[3] T. Coffy, J.-L. Giavitto, and A. Cont, "AscoGraph: A User Interface for Sequencing and Score Following for Interactive Music," in *ICMC 2014 - 40th International Computer Music Conference*, Athens, Greece, Sep. 2014. [Online]. Available: https://hal.inria.fr/hal-01024865

[4] R. Thöle, "Approximation algorithms for packing and scheduling problems," Ph.D. dissertation, Christian-Albrechts-Universität zu Kiel, 2008.

[5] J. Coffman, E. G., M. R. Garey, D. S. Johnson, and R. E. Tarjan, "Performance bounds for level-oriented two-dimensional packing algorithms," *SIAM J. Comput.*, no. 9, pp. 808–826, 1980.

[6] J.-L. Giavitto, A. Cont, and J. Echeveste. Antescofo a not-so-short introduction to version 0. x. [Online]. Available: http://support.ircam.fr/docs/Antescofo/AntescofoReference.pdf