Comparison of Heuristics for Generating All-partition Arrays in the Style of Milton Babbitt

Bemman, Brian; Meredith, David

Published in:
Proceedings of the 11th International Symposium on Computer Music Multidisciplinary Research

Publication date:
2015

Document Version
Peer reviewed version

Comparison of heuristics for generating all-partition arrays in the style of Milton Babbitt

Brian Bemman and David Meredith

Aalborg University
{bb,dave}@create.aau.dk

Abstract. An all-partition array is a mathematical structure developed by Milton Babbitt (1916–2011) for organizing the pitch classes in many of his twelve-tone works. Constructing an all-partition array requires finding a sequence of ℓ aggregate-forming subsets of a matrix of pitch classes called a *PCMatrix*. Each of these ℓ subsets corresponds to a distinct integer partition and ℓ is the total number of distinct integer partitions possible, given the number of rows in the PCMatrix. Collectively, the ℓ subsets (or integer partitions) form a partition of the PCMatrix. In some types of all-partition arrays, where $\ell = 58$, the PCMatrix does not contain all 12ℓ pitch classes. In these types, Babbitt developed a method for adding to the PCMatrix what we call *outer-aggregate repeated pcs* (OARPs). A *self-contained sequence* of integer partitions is one in which each integer partition either contains a complete aggregate or an incomplete one that can be made complete by adding OARPs. It is noteworthy that, when constructing an all-partition array, Babbitt started out with a *non-self-contained* sequence of partitions. In this paper, we use a known self-contained sequence as a basis for forming two heuristics that select integer partitions likely to have been chosen by Babbitt. We suggest these heuristics will select integer partitions more likely to produce a self-contained sequence and we present it as a means for efficiently searching the space of possible sequences. We apply our heuristics to both types of sequences and conclude with a simple method for evaluating the results.

Keywords: Babbitt, all-partition array, heuristic, city block distance

1 Introduction

The all-partition array is a structure used by Milton Babbitt (1916–2011) to organize the pitch class content of many of his twelve-tone compositions. While the details of its structure have been written about extensively (see, in particular,[4]), we will focus on the decisions Babbitt made in constructing it, offering some new definitions and insights into this process. Before considering the basic construction of the all-partition array, we present some preliminary definitions. We use the term *aggregate* in the usual way to mean the universe of pitch classes— that is, the set $\{0, 1, \ldots 11\}$. A *tone-row*, $A = \langle p_1, p_2, \ldots p_{12} \rangle$, is then an ordered set of pitch classes that contains each element in the aggregate exactly once—that is, $\bigcup_{i=1}^{12} \{p_i\} = \{0, 1, \ldots 11\}$.

Our following definitions apply specifically to the structure of the all-partition array in a way that will become evident below. An *integer partition*, which we denote by $\text{IntPart}(s_1, s_2, ..., s_k)$, is a representation of an integer $n = \sum_{i=1}^{k} s_i$, as an unordered sum of k positive integers, $s_1 \dots s_k$. For example, if $n = 12$ and $k = 6$, then one possible integer partition is $\text{IntPart}(3, 3, 2, 2, 1, 1)$. An *integer composition*, denoted by $\text{IntComp}(s_1, s_2, ..., s_k)$, is a representation of an integer $n = \sum_{i=1}^{k} s_i$, as an *ordered* sum of k positive integers. For example, if $n = 12$ and $k = 6$, then $\text{IntComp}(3, 3, 2, 2, 1, 1) \neq \text{IntComp}(3, 2, 1, 3, 2, 1)$. We define a *weak* integer composition, $\text{WIntComp}(s_1, s_2, ..., s_k)$, to be a representation of an integer $n = \sum_{i=1}^{k} s_i$, as an *ordered* sum of k *non-negative* integers, (i.e., including 0). For example, if $n = 12$ and $k = 6$, then $\text{WIntComp}(6, 6, 0, 0, 0, 0)$ is a weak integer composition. We further define two relations, *partitionally equivalent* and *partitionally distinct*. Two integer compositions, c and d, are *partitionally equivalent* if and only if $[c] = [d]$, where $[c]$ and $[d]$ denote the integer partitions associated with, c and d, respectively. Two integer compositions, c and d, are *partitionally distinct* if and only if $[c] \neq [d]$.

From the definitions above, we state that constructing an all-partition array requires finding a sequence of ℓ partitionally distinct integer compositions that exhaustively (and, typically, almost exclusively) partition a matrix of pitch classes we call a *PCMatrix* into complete aggregates. In this paper we will focus on the so-called *six-part* all-partition arrays that have the following characteristics [4]:

1. the dimensions of its PCMatrix are 6×96;
2. it is constructed from 48 twelve-tone rows;
3. integer compositions where $n = 12$ and $k = 6$ are used; and
4. $\ell = 58$.

Figure 1 shows two partitionally distinct integer compositions and how each is used in a selection of a PCMatrix to form regions containing a complete aggregate in the first and an incomplete aggregate in the second.

2	9	10	8	4	3	0	7	1	11	6	5
7	0	11	1	5	6	9	2	8	10	3	4
3	4	9	11	5	10	1	2	6	8	7	0
8	1	0	2	6	7	10	3	9	11	4	5
6	5	0	10	4	11	8	7	3	1	2	9
7	8	1	3	9	2	5	6	10	0	11	4

Fig. 1: $\text{WIntComp}(3, 3, 2, 2, 2, 0)$ forming a complete aggregate (indicated by the solid line) and $\text{WIntComp}(1, 0, 4, 3, 0, 4)$ forming an incomplete aggregate (indicated by the dashed line) in a selection of a PCMatrix. Note that the incomplete aggregate is missing pitch class 4 and has two occurrences of pitch class 8.

Each composition in Figure 1 consists of 6 segments, one in each row, whose lengths are equal to its summands. For example, in the first composition, WInt-

Comp(3,3,2,2,2,0), these segments have lengths 3, 3, 2, 2, 2 and 0. One will note, however, that the first composition forms a complete aggregate while the second composition (indicated by the dashed line boundary) forms an incomplete aggregate as it is missing pitch class 4 and contains two occurrences of pitch class 8. Babbitt's method for dealing with such a composition was by finding pitch classes from the first composition that, when repeated and pushed inside the region formed by the second composition, would form a complete aggregate. These repeated pitch classes we call *outer-aggregate repeated pcs* (OARPs). As the PCMatrix contains only 576 pitch classes (6×96), there must be an additional 120 of these OARPs added to it in order to have all 12ℓ required pitch classes. Figure 2(a) shows how this incomplete aggregate formed by the second composition in Figure 1 is made complete with OARPs.

2	9	10	**10**	8	4	3	0	7	1	11	6	5
7	0	11	1	5	6	9	2	8	10	3	4	
3	4	**4**	9	11	5	10	1	2	6	8	7	0
8	1	0	2	6	7	10	3	9	11	4	5	
6	5	0	10	4	11	8	7	3	1	2	9	
7	8	1	3	9	2	5	6	10	0	11	4	

(a) WIntComp(1, 0, 4, 3, 0, 4) with OARPs added to form a complete aggregate.

2	9	10	**10**	8	4	3	0	7	1	11	6	5
7	0	11	1	5	6	9	2	8	10	3	4	
3	4	9	11	5	10	1	2	6	8	7	0	
8	1	0	2	6	7	10	3	9	11	4	5	
6	5	0	10	4	11	8	7	3	1	2	9	
7	8	1	3	9	2	5	6	10	0	11	4	

(b) WIntComp(4, 1, 3, 3, 0, 1) with OARPs added to form a complete aggregate.

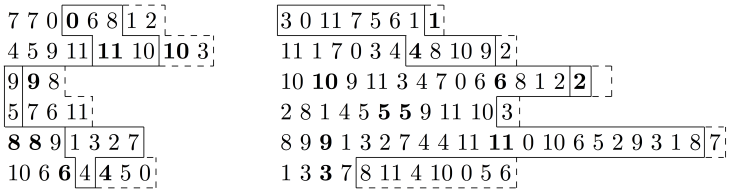
Fig. 2: Two possible sequences of two compositions. (a) Composition from Figure 1 with OARPs (in bold) added to complete its aggregate. (b) A second possible composition with its (single) OARP added.

As has been noted, Babbitt's method for completing aggregates only by repeating pitch classes was intended to preserve the order of pitch classes in a twelve-tone row [4]. One will note in Figure 2(a) that these pitch classes are 10 and 4 (shown in bold). However, Figure 2(b) shows that another composition in the second position is successful in forming a complete aggregate after the addition of its (single) OARP. There are, on average, ≈ 78 possible compositions at each position and the problem we seek to address next is determining which of these compositions are "better" and what it means for a composition to be better than another.

2 Self-contained sequences

We define a *self-contained sequence of compositions* as one in which each composition forms a region that either contains a complete aggregate or can be made to do so by using Babbitt's method of adding OARPs. We borrow the term "self-contained" from [4], used to describe a type of all-partition array, and extend it to describe such an array's corresponding sequence of compositions. Examples of a sequence of this type in Babbitt's works can be found in, for example,

Sheer Pluck and *Joy of More Sextets*. In a significant number of Babbitt's works, however, Babbitt used a non-self-contained sequence of compositions whose 58^{th} composition formed an incomplete aggregate that could not be made complete with OARPs. Such pieces include, for example, *About Time* and *Arie da Capo* [4]. For comparison, Figure 3 shows the final two compositions of a self-contained and a non-self-contained sequence.



(a) Self-contained sequence of compositions.

(b) Non-self-contained sequence of compositions.

Fig. 3: Final two compositions in, (a), a self-contained and, (b), a non-self-contained sequence of compositions. Note location of the missing pitch class at the end of the third row of the matrix corresponding to the 58^{th} composition.

Note in Figure 3(a) that each composition forms a complete aggregate made by adding OARPs; whereas in (b), the final composition is missing pitch class 9. Using Babbitt's method of adding OARPs, however, the only available pitch class is 2. Returning to the problem posed in Figure 2(b), we suggest then that the criterion by which we can judge a composition as better than another, is how likely that composition is to result in a self-contained sequence. We believe this is a valid criterion as Babbitt continued to use self-contained rather than non-self-contained sequences in many of his works.

3 Two heuristics for constructing a self-contained sequence

In constructing the self-contained sequence shown in Figure 3(a), there are $\approx 78^{58}$ possible sequences that make up the search space. Of these, very few will be non-self-contained and even fewer will be self-contained. Confronted with these odds, it is necessary to develop some heuristic or greedy approach for deciding which compositions to choose when constructing a sequence that satisfies the constraints of an all-partition array. By comparing the sequences of compositions from Figure 3(a) and (b) and their corresponding PCMatrixes we might be able to deduce some qualities that make a self-contained sequence. In doing so, one immediate difference is clear: the right edge of the PCMatrix belonging to the self-contained sequence is less ragged.

3.1 Equal-row-length heuristic

Our *equal-row-length* heuristic works from the hypothesis that self-contained sequences are more likely to contain compositions that collectively result in a PCMatrix with rows of approximately equal length. We suggest that a straightforward way to achieve this is to ensure that compositions *progress* at approximately equal rates in each row. Let's suppose we have a list of possible compositions, $C_k = \langle c_1, c_2, ...c_n \rangle$, at position k, where $1 \leq k \leq \ell$. Ideally, if all rows of the PCMatrix have progressed at the same rate, then after choosing a composition for position k, the lengths of each row up to position k would be $12k/r$, where r is the number of rows in the PCMatrix. Let's suppose that the actual length of a row j would be $l_{i,j}$ after choosing c_i from C_k. To measure the "raggedness" or degree of inequality of line length that results from choosing c_i at position k, we use the following formula, based on city-block distance:

$$D_i = \sum_{j=1}^{r} \left| l_{i,j} - \frac{12k}{r} \right| \tag{1}$$

where $|x|$ denotes the absolute value of x and where, for each k, we choose the c_i that minimizes the total difference, D_i. Returning now to the two possible compositions shown in Figure 2, the distance measure after choosing (a) is $D = 6$ while the distance measure after choosing (b) is $D = 10$. According to our heuristic, the composition in (a) is thus better than the composition in (b). That is, this composition contributes to producing rows of more similar length at this position than the other composition. Moreover, it is also the composition Babbitt chose when constructing his own sequence.

3.2 Zero-gain segments heuristic

Using city block distance in the manner described, it is possible that two compositions will result in the same distance, D_i, defined in Eq. 1. This does not mean, however, that each of their rows will have the same length. We thus propose a second heuristic for further discriminating compositions, which we call *zero-gain segments*. This heuristic judges a composition better than another based on the qualities it shares with the composition that immediately precedes it in the sequence. Let's suppose we have two adjacent compositions, A and B where $A = \langle s_1, s_2, ..., s_k \rangle$ and $B = \langle t_1, t_2, ..., t_k \rangle$. The weight of B, as defined by $w = \sum_{i=1}^{k} r_i$ is the sum of reward, r_i, where

$$r_i = \begin{cases} 1, \text{if } (s_i = 0 \veebar t_i = 0); \text{ and} \\ 0, \text{otherwise.} \end{cases} \tag{2}$$

The zero-gain segments heuristic assigns an integer weight equal to the number of instances where there is a zero summand in the current composition and a non-zero summand at the corresponding position in the previous composition (or vice versa). For example, we return again to our two possible compositions,

WIntComp$(1, 0, 4, 3, 0, 4)$ and WIntComp$(4, 1, 3, 3, 0, 1)$ shown in Figure 2. According to the zero-gain segments heuristic, the former composition has a $w = 3$ while the latter composition has a $w = 2$, with respect to the composition, WIntComp$(3, 3, 2, 2, 2, 0)$, that immediately precedes them in the sequence. Unlike the equal-row-length heuristic, the higher the weight found by zero-gain segments heuristic, the better. We may combine both of these heuristics to produce a new weight, $W = D - w$, which we attempt to minimize at each step in the process of choosing compositions.

4 Evaluating the heuristics

By applying our heuristics for each position, k, from 1 to ℓ and subsequently sorting all c_i in each C_k in ascending order of either D_i or W, we propose that a self-contained sequence will contain compositions that lie nearer the top of each C_k. In other words, the lower the index, i, for the chosen c_i, the better. For comparison, Table 1 shows the index into C_k at each position, k, after applying the equal-row-length heuristic, for each composition found in the non-self-contained sequence and the self-contained sequence shown in Figure 3.

Equal-row-length heuristic	Arie da Capo – non-self-contained sequence
Index into C for each k	$\langle 1, 6, 10, 2, 13, 55, 14, 8, 5, 36, 7, 27, 17, 1, 6, 28, 7, 3, 20, 21, 58, 4, 8, 24, 40, 3, 10, 13, 13, 3, 2, 29, 58, 8, 4, 35, 2, 23, 51, 2, 5, 50, 6, 8, 3, 32, 21, 6, 5, 2, 1, 3, 6, 6, 1, 1, 1, 1\rangle$
Number of compositions in C for each k	$\langle 44, 94, 204, 89, 116, 141, 252, 42, 123, 72, 51, 141, 99, 61, 49, 129, 53, 46, 43, 53, 124, 112, 40, 61, 60, 28, 29, 28, 105, 84, 110, 69, 69, 59, 61, 84, 113, 83, 87, 49, 62, 61, 36, 46, 47, 53, 31, 37, 24, 13, 21, 17, 17, 6, 1, 1, 1, 1\rangle$
	Sheer Pluck – self-contained sequence
Index into C for each k	$\langle 1, 51, 73, 5, 11, 14, 15, 6, 51, 222, 57, 12, 1, 9, 5, 18, 17, 36, 3, 83, 23, 23, 22, 28, 53, 9, 21, 3, 5, 42, 56, 45, 66, 11, 5, 2, 105, 25, 66, 9, 3, 3, 10, 5, 35, 4, 1, 3, 31, 1, 6, 6, 26, 3, 21, 4, 7, 1\rangle$
Number of compositions n C for each k	$\langle 57, 73, 198, 53, 65, 179, 123, 81, 163, 295, 102, 31, 23, 188, 13, 58, 148, 130, 88, 117, 160, 136, 71, 72, 76, 125, 104, 208, 109, 44, 70, 147, 69, 66, 61, 32, 134, 98, 143, 34, 17, 20, 21, 61, 39, 8, 2, 9, 40, 17, 18, 13, 42, 12, 23, 4, 8, 1\rangle$

Table 1: Equal-row-length indices, i, of the composition, c_i, in C_k for each k for the non-self-contained sequence in *Arie da Capo* (top) and the self-contained sequence in *Sheer Pluck* (bottom), shown in Figure 3.

One will note that many of the indices for the non-self-contained sequence shown in Table 1 (top) are relatively low, given the total number of compositions found in C for each corresponding position appearing immediately below. For example, in C_3 there are 204 possible compositions yet the composition used by Babbitt is the 10^{th} best according to our heuristic. There are a number of exceptions, however. For example, while in C_{33} there are 69 possible compositions, the one chosen by Babbitt is the 58^{th}. By comparison, the indices of the self-contained sequence (bottom) appear higher.

We may evaluate our heuristic here by simply finding the mean value of all indices for a particular sequence. As with city block distance, the lower the value the better. The non-self-contained sequence shown in Table 1 has a mean index of 14.4. In contrast, the self-contained sequence shown in Table 1 has a mean index of 24.5. The difference between the two is in favor not of the self-contained sequence (from which our heuristic was modeled), but of the non-self-contained sequence. For comparison, we can combine the two heuristics and apply this to the same sequences in the manner described above. Table 2 shows the index into C_k at every position, k, after applying both equal-row-length and zero-gain segments, for each composition found in both the non-self-contained and self-contained sequences shown in Figure 3.

Equal-row-length and zero-gain heuristics	Arie da Capo – non-self-contained sequence
Index into C for each k	$\langle 1, 5, 34, 28, 20, 78, 28, 4, 3, 13, 7, 27, 4, 5, 6, 34, 19, 5, 18, 14,$ $65, 2, 9, 20, 38, 3, 12, 10, 13, 13, 6, 51, 58, 10, 5, 31, 2, 11, 50, 2$ $3, 43, 4, 7, 2, 23, 23, 11, 4, 4, 1, 4, 6, 5, 1, 1, 1, 1\rangle$
Number of compositions in C for each k	$\langle 44, 94, 204, 89, 116, 141, 252, 42, 123, 72, 51, 141, 99, 61, 49, 129,$ $53, 46, 43, 53, 124, 112, 40, 61, 60, 28, 29, 28, 105, 84, 110, 69, 69,$ $59, 61, 84, 113, 83, 87, 49, 62, 61, 36, 46, 47, 53, 31, 37, 24, 13, 21,$ $17, 17, 6, 1, 1, 1, 1\rangle$
	Sheer Pluck – self-contained sequence
Index into C for each k	$\langle 1, 60, 29, 5, 4, 2, 3, 3, 38, 175, 56, 10, 1, 10, 3, 16, 18, 60, 1, 70,$ $18, 29, 10, 36, 65, 17, 32, 10, 10, 44, 56, 63, 58, 12, 2, 2, 107, 12, 38, 8$ $6, 7, 8, 2, 34, 5, 1, 2, 33, 1, 5, 6, 21, 2, 21, 4, 7, 1\rangle$
Number of compositions n C for each k	$\langle 57, 73, 198, 53, 65, 179, 123, 81, 163, 295, 102, 31, 23, 188, 13, 58,$ $148, 130, 88, 117, 160, 136, 71, 72, 76, 125, 104, 208, 109, 44, 70,$ $147, 69, 66, 61, 32, 134, 98, 143, 34, 17, 20, 21, 61, 39, 8, 2, 9, 40,$ $17, 18, 13, 42, 12, 23, 4, 8, 1\rangle$

Table 2: Equal-row-length and zero-gain indices, i, of the composition, c_i, in C_k for each k for the non-self-contained sequence in *Arie da Capo* (top) and the self-contained sequence in *Sheer Pluck* (bottom), shown in Figure 3.

Using both heuristics this time yields a marginally worse result for the non-self-contained sequence shown in the top of Table 2 with a mean index of 15.7. However, there is a marginally better result for the self-contained sequence (bottom) with a mean index of 23.4.

5 Conclusion

Constructing any sequence of compositions that satisfies the constraints of an all-partition array is a difficult task because, while the number of possible sequences is very large, the number of these that will be self-contained sequences is extremely small in comparison. The number of non-self-contained sequences on the other hand, is higher, as the constraint that only OARPs can be used is abandoned. Our findings suggest the possibility that, when constructing the non-self-contained sequence of compositions used in *Arie da Capo*, Babbitt preferred compositions at each step that resulted in the rows progressing at rates that were as equal as possible. On the other hand, when constructing the self-contained all-partition array for *Sheer Pluck*, it seems that, if he used the equal-row-length heuristic at all, he did not assign it as high a priority in determining the sequence of compositions. By incorporating the zero-gain segments heuristic, however, we were able to better account for the sequence of compositions that Babbitt selected in Sheer Pluck. Unfortunately, to our knowledge, only one self-contained all-partition array sequence has been discovered to date, so it is currently difficult to evaluate our heuristics more rigorously. In future research we hope to use these heuristics as part of a greedy algorithm that will efficiently construct a new self-contained sequence of compositions in the style of Milton Babbitt.

6 Acknowledgements

The work reported in this paper was carried out as part of the EC-funded collaborative project, "Learning to Create" (Lrn2Cre8). The Lrn2Cre8 project acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET grant number 610859.

References

1. Babbitt, M.: Set structure as a Compositional Determinant. Journal of Music Theory 5, 72–94 (1987)
2. Bazelow, A.R., Brickle, F.: A Partition Problem Posed by Milton Babbitt. Perspectives of New Music 14, 2, 280–293 (1976)
3. Knuth, Donald, Dancing Links.: `http://www-cs-faculty.stanford.edu/~uno/musings.html`, Feb. 22, 2000
4. Mead, A.: An Introduction to the Music of Milton Babbitt. Princeton University Press, Princeton, NJ (1994)