# A novel algorithm for finding the prime form of a pitch class set

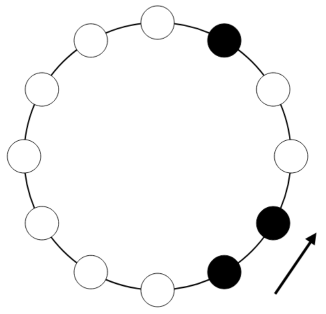
Joni Pääkkö

March 5, 2023

## 1 Introduction

A very basic operation in the analysis of atonal music is finding the prime form of a pitch class set. This is achieved easily by first drawing a clock, drawing pitch classes on their corresponding places on the hour arm and then visually finding the most compact form – a procedure which I believe is familiar for anyone who has analyzed atonal music.

An example of the this procedure is shown in the figure below. Given that 12 o'clock corresponds with C, and the pitch class set to be analyzed has pitches C# E and F, the prime form can be found by labeling F as 0 and moving counter-clockwise:



The prime form of the pitch class set $\{C\#, E, F\}$ or $\{1, 4, 5\}$ is then $\{0, 1, 4\}$.

But what if one has to the find prime form programmatically? That is a surprisingly complicated problem. Michiel Schuijer[1] proposes an algorithm which I will be treating here as a kind of ground truth, for the purpose of introducing a new algorithm.

[1]Schuijer, Michiel (2008): Analyzing Atonal Music Pitch – Pitch-Class Set Theory and Its Contexts. University of Rochester Press.

## 2 Schuijer's algorithm

Schuijer's algorithm consists of the following steps. Given a pitch class set $A$:

- Find the inversion of $A$: $A_{inv}$
- Calculate all of the transpositions of $A$ and $A_{inv}$. Call the set that contains all of these transpositions $B$.
- Order every set in $B$ in ascending order.
- Select sets for which the first element is 0 and last element is the smallest of all the last elements. Call these sets $C$.
- From the set $C$, select one with the smallest non-common integer.

(Note: my summary of the algorithm comprises of more steps than the summary given by Schuijer in his book. My intention is not to make this algorithm seem more complicated than it is, quite the contrary, I'm trying to summarize it as cleary as possible, and I feel this is better achieved by including intermediary steps).

I find this algorithm unsatisfactory for two reasons: 1) first, it does not offer any benefits over the visual method discussed earlier and 2) in terms of computation, it requires many non-trivial steps. Since I am conserned with computation, I will further elaborate the second point.

Whether or not the implementation of the algorithm is cumbersome or not obviously depends on one's language of choice. For example in R, implementing the algorithm is relatively straightforward since base R contains many functions pertaining to the manipulation of sets. However in languages such as C or Java one would need quite a bit of additional code to achieve the desired operations – if one does not use additional libraries.

Regardless, a computational issue with the aforementioned algorithm is that for some sets there does not exist a unique set (of sets) for which one can find the smallest non-common integer. Take for example the set $A = [5, 0, 7]$.

When we perform the steps of Schuijer's algorithm up to the point in which we have to find the set with the smallest non-common integer, we are left with the following sets

$$\begin{array}{l} \{0, \quad 5, \quad 7\} \\ \{0, \quad 2, \quad 7\} \\ \{0, \quad 5, \quad 7\} \\ \{0, \quad 2, \quad 7\} \end{array}$$

We can't identify a unique set with the lowest non-common integer. This is not a fatal flaw, of course: there are ways around this. But in any case this is something to take into account when implementing the algorithm: it does add an additional layer of complexity; of arbitrary choices to make.

## 3 The proposed algorithm

The algorithm I propose is simpler with regards to computation. On a side note, it is fairly simple also to implement on pen and paper, but does not – again – fare better than the classic visual method, but might serve as a reasonable more quantitative alternative.

Consider a pitch class set $A$. Create a matrix $B$ such that

$$B_{ij} = (A_i - A_j + 12) \bmod 12 \tag{1}$$

The indices $i$ and $j$ both span the extent of $A$.

After this, find the row and column sums of the matrix $B$ and select the row or column with the smallest sum. Note that the result is not necessarily in an ascending order, however, since we are conserned about sets, this is not strictly required – but one obviously can sort the result. (That is what I do in the R implementation that's introduced later on.)

Sometimes multiple minima exist, but this is not a problem – from a computational perspective – since one only needs select one of the minima because they are all equal. Selecting a single minimum is more stable and easier than finding a set which hasn't got any common integers with the other sets.

An important limitation of the algorithm is that its time complexity increases as a square of the elements in the pitch class set. However, the operations themselves – additions and modulos – are not computationally expensive.

## 4 An example

I'll walk you through an example of how one might implement this algorithm on pen and paper.

Consider the pitch class set $A = \{0, 3, 4\}$. We'll create a matrix which has these pitch classes in its margins:

$$\begin{array}{c|ccc|c} & 0 & 3 & 4 & \\ \hline 0 & & & & \\ 3 & & & & \\ 4 & & & & \\ \hline & & & & \end{array}$$

Then, for each entry, we will calculate the row margin minus the column margin plus twelve modulo twelve (as per the equation I introduced earlier). For the first column:

$$\begin{aligned} 0 - 0 + 12 \bmod 12 &= 0 \\ 3 - 0 + 12 \bmod 12 &= 3 \\ 4 - 0 + 12 \bmod 12 &= 4 \end{aligned}$$

for the 2nd column:

$$\begin{aligned}0 - 3 + 12 \bmod 12 &= 9\\ 3 - 3 + 12 \bmod 12 &= 0\\ 4 - 3 + 12 \bmod 12 &= 1\end{aligned}$$

and finally for the third column:

$$\begin{aligned}0 - 4 + 12 \bmod 12 &= 8\\ 3 - 4 + 12 \bmod 12 &= 11\\ 4 - 4 + 12 \bmod 12 &= 0\end{aligned}$$

I have collected these results, and the row and column sums, into the following matrix:

$$\begin{array}{c|ccc|c} & 0 & 3 & 4 & \\ \hline 0 & 0 & 9 & 8 & 17\\ 3 & 3 & 0 & 11 & 14\\ 4 & 4 & 1 & 0 & 5\\ \hline & 7 & 10 & 19 & \end{array}$$

The smallest sum is 5 which means that we select the third row and that the prime form is $4, 1, 0$ or $0, 1, 4$ in an ascending order.

## 5 An implementation in R

```
# Function for finding the prime form of a pitch class set.
# INPUT:
# pitchClassSet :  a vector of pitch classes (b/w 0 and 11).
# OUTPUT:
# Vector, the prime form of the given pitch class set.
findPrimeForm = function(pitchClassSet){
  if(any(pitchClassSet > 11) || any(pitchClassSet < 0)) {
    stop("Error: pitch classes must be between 0 and 11")
  }

  if(length(pitchClassSet) == 1){
    return(0)
  }

  intervalMatrix = matrix(NaN,
                            ncol = length(pitchClassSet),
                            nrow = length(pitchClassSet))

  for(i in 1:length(pitchClassSet)){
    for(j in 1:length(pitchClassSet)){
      intervalMatrix[i, j] = (pitchClassSet[i] - pitchClassSet[j] + 12) %% 12
    }
  }

  rowS = rowSums(intervalMatrix)
  colS = colSums(intervalMatrix)

  if(min(rowS) < min(colS)){
    return(sort(intervalMatrix[which.min(rowS),]))
  } else {
    return(sort(intervalMatrix[,which.min(colS)]))
  }
}
```