The *Euclidean* Algorithm Generates Traditional Musical Rhythms

Godfried Toussaint[*]
School of Computer Science, McGill University
Montréal, Québec, Canada
godfried@cs.mcgill.ca

2005

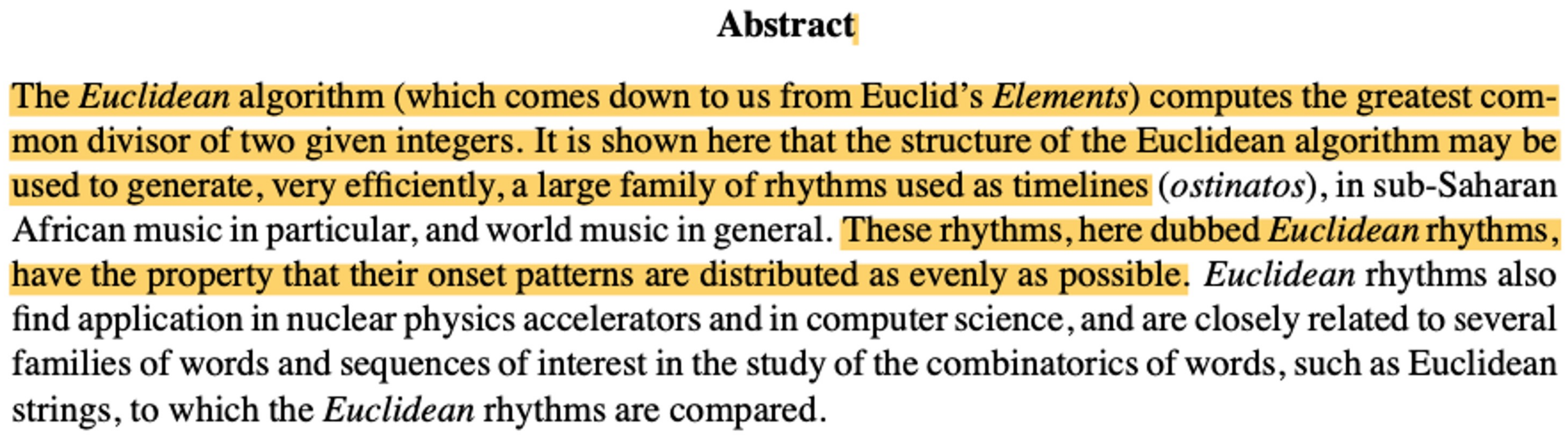
The Distance Geometry of Music

Erik D. Demaine[*] Francisco Gomez-Martin[†] Henk Meijer[‡] David Rappaport[§]
Perouz Taslakian[¶] Godfried T. Toussaint[§||] Terry Winograd[**] David R. Wood[††]

Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA

2009

Introducción



Abstract

The *Euclidean* algorithm (which comes down to us from Euclid's *Elements*) computes the greatest common divisor of two given integers. It is shown here that the structure of the Euclidean algorithm may be used to generate, very efficiently, a large family of rhythms used as timelines (*ostinatos*), in sub-Saharan African music in particular, and world music in general. These rhythms, here dubbed *Euclidean* rhythms, have the property that their onset patterns are distributed as evenly as possible. *Euclidean* rhythms also find application in nuclear physics accelerators and in computer science, and are closely related to several families of words and sequences of interest in the study of the combinatorics of words, such as Euclidean strings, to which the *Euclidean* rhythms are compared.

Hipótesis

Varios investigadores han observado que en los **ritmos de la música** tradicional del mundo hay una tendencia a encontrar **patrones distribuidos lo más regular** o uniformemente posible.

$\longrightarrow$

Patrones de regularidad máxima se pueden describir mediante el uso del **algoritmo de Euclides** sobre el máximo común divisor de dos números enteros

Patrones de regularidad máxima

x = nota

· = silencio

[×· ×· ×· ×·] → **[1 0 1 0 1 0 1 0]**

(8,4) = 4 notas distribuidas regularmente en los 8 pulsos

[×· ·×· ·×·] → **[1 0 0 1 0 0 1 0]**

(8,3) = 3 notas distribuidas regularmente en 8 pulsos

3. The Euclidean Algorithm

One of the oldest algorithms known, described in Euclid's *Elements* (*circa* 300 B.C.) in Proposition 2 of *Book VII*, today referred to as the Euclidean algorithm, computes the greatest common divisor of two given integers [12], [14]. The idea is very simple. The smaller number is repeatedly subtracted from the greater until the greater is zero or becomes smaller than the smaller, in which case it is called the remainder. This remainder is then repeatedly subtracted from the smaller number to obtain a new remainder. This process is continued until the remainder is zero. To be more precise, consider as an example the numbers 5 and 8 as before. First 5 divides into 8 once with a remainder of 3. Then 3 divides into 5 once with a remainder of 2. Then 2 divides into 3 once with a remainder of 1. Finally, 2 divides into 2 once with a remainder of 0. The greatest common divisor is therefore 1. Although Euclid's original algorithm used repeated subtraction in this manner, standard division will work just as well, and is even faster. The steps of this process can be summarized by the following sequence of equations:

$$\begin{aligned} 8 &= (1)(5) + 3 \\ 5 &= (1)(3) + 2 \\ 3 &= (1)(2) + 1 \\ 2 &= (1)(2) + 0 \end{aligned}$$

Algoritmo de Euclides

Consiste en hacer divisiones sucesivas para hallar el **máximo común divisor de dos números positivos** (*m.c.d.* de aquí en adelante).

- El **m.c.d.** de dos números **a** y **b**, suponiendo que **a** > **b**, primero dividimos **a** entre **b**, y obtenemos el resto **r** de la división.
- El m.c.d. de **a** y **b** es el mismo que el de **b** y **r**.
- Cuando dividimos **a** entre **b**, obtenemos un cociente **c** y un resto **r** de tal manera que se cumple que

$$\boldsymbol{a = c \cdot b + r}$$

Algoritmo de euclides - $a = c \cdot b + r$

m.c.d. (17,7) = 1

17	= 7 . 2 + 3
7	= 3 . 2 + **1**
3	= 1 . 3 + 0

m.c.d. (8,3) = 1

8	= 3 . 2 + 2
3	= 2 . 1 + **1**
2	= 1 . 2 + 0

calculemos el máximo común de 17 y 7. Como 17 = 7 · 2 + 3, entonces el m.c.d.(17, 7) es igual al m.c.d.(7, 3). De nuevo, como 7 = 3 · 2 + 1, entonces el m.c.d.(7, 3) es igual al m.c.d.(3, 1). Aquí es claro que el m.c.d. entre 3 y 1 es simplemente 1. Por tanto, el m.c.d entre 17 y 7 es 1 también.

¿Cómo se transforma el cálculo del **m.c.d.** en patrones distribuidos con **regularidad máxima**?

Representar una secuencia binaria de k unos y n−k ceros, donde cada bit [0] representa un intervalo de tiempo y los unos [1] se envía una señal.

El problema entonces se reduce a lo siguiente:

construir una secuencia binaria de n bits con k unos tal que los k unos **se distribuyan lo más uniformemente posible** entre los n − k ceros.

Un caso simple es cuando k divide uniformemente n (sin resto), en cuyo caso debemos colocar unos cada n/k bits.
Por ejemplo, si n = 16 y k = 4, entonces la solución es

[1000100010001000].

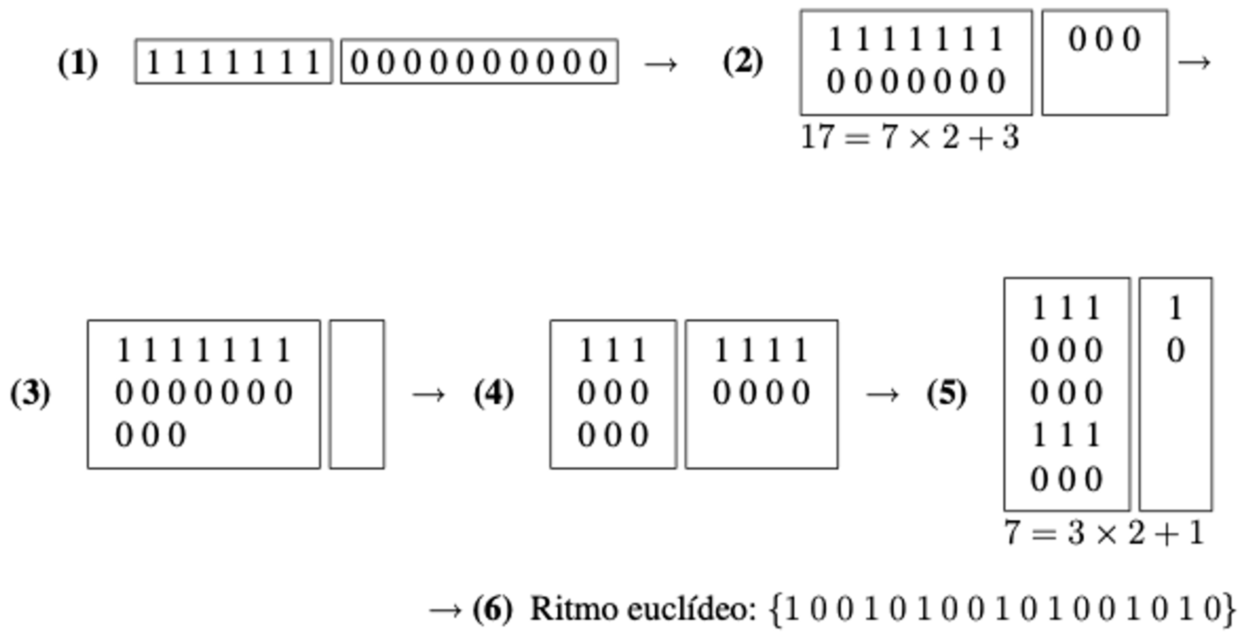
Este caso corresponde a que n y k tienen un divisor común de k (en este caso 4).

De manera más general, si el máximo común divisor entre n y k es g, esperaríamos que la solución se descomponga en g repeticiones de una secuencia de n/g bits.

Esta conexión con los máximos comunes divisores sugiere que se podría calcular un ritmo de máxima uniformidad usando un algoritmo como el de Euclides.

Esta conexión con los máximos comunes divisores sugiere que se podría calcular un ritmo de máxima uniformidad usando un algoritmo como el de Euclides.

(17,7) →



1. Alineamos el número de notas y el número de silencios (7 unos y 10 ceros)

2, 3 y 4. Formamos grupos de 7, los cual corresponde a efectuar la división de 17 entre 7; obtenemos 7 grupos de formados por [1 0] y sobran 3 ceros, lo cual indica que en el paso siguiente formaremos grupos de 3 hasta que queden uno o cero grupos.

5. De nuevo, esto es equivalente a efectuar la división de 7 entre 3. En nuestro caso, queda un solo grupo y hemos terminado.

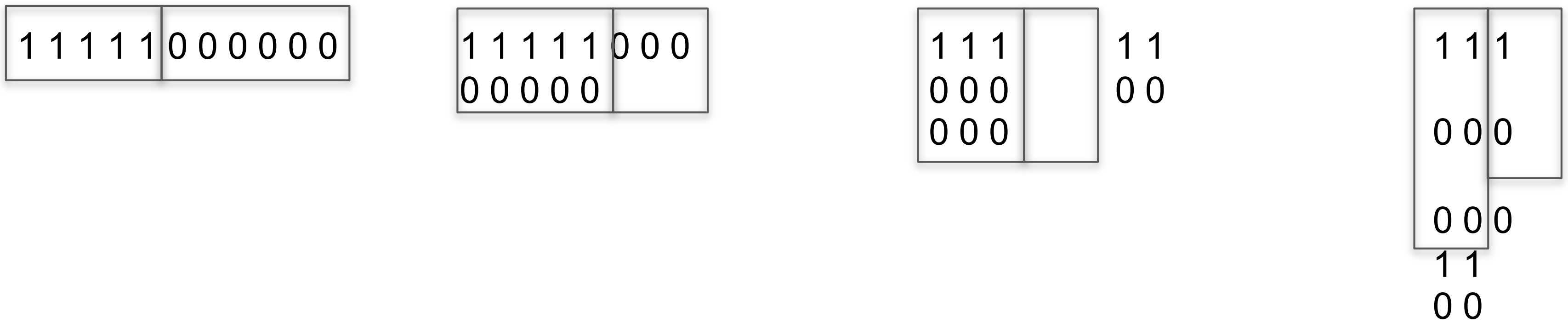
4. Finalmente, el ritmo se obtiene leyendo por columnas y de izquierda a derecha la agrupación obtenida, paso

3. The Euclidean Algorithm

One of the oldest algorithms known, described in Euclid's *Elements* (*circa* 300 B.C.) in Proposition 2 of *Book VII*, today referred to as the Euclidean algorithm, computes the greatest common divisor of two given integers [12], [14]. The idea is very simple. The smaller number is repeatedly subtracted from the greater until the greater is zero or becomes smaller than the smaller, in which case it is called the remainder. This remainder is then repeatedly subtracted from the smaller number to obtain a new remainder. This process is continued until the remainder is zero. To be more precise, consider as an example the numbers 5 and 8 as before. First 5 divides into 8 once with a remainder of 3. Then 3 divides into 5 once with a remainder of 2. Then 2 divides into 3 once with a remainder of 1. Finally, 2 divides into 2 once with a remainder of 0. The greatest common divisor is therefore 1. Although Euclid's original algorithm used repeated subtraction in this manner, standard division will work just as well, and is even faster. The steps of this process can be summarized by the following sequence of equations:

$$\boxed{\begin{array}{l} 8 = (1)(5) + 3 \\ 5 = (1)(3) + 2 \\ 3 = (1)(2) + 1 \\ 2 = (1)(2) + 0 \end{array}}$$

m.c.d. (13,5) **13 = 5 . 2 + 3** **5 = 3 . 1 + 2** **3 = 2 . 1 + 1**



Euclid serie = [1 0 0 1 0 1 0 0 1 0 1 0 0]

Bjorklund's algorithm will be described simply by using one of his examples. Consider a sequence with $n = 13$ and $k = 5$. Since $13 - 5 = 8$, we start by considering a sequence consisting of 5 one's followed by 8 zero's which should be thought of as 13 sequences of one bit each:

[1 1 1 1 1 0 0 0 0 0 0 0 0]

We begin moving zero's by placing a zero after each one, to produce five sequences of two bits each, with three zero's remaining:

[10] [10] [10] [10] [10] [0] [0] [0]

Next we distribute the three remaining zeros in a similar manner, by placing a [0] sequence after each [10] sequence to obtain:

[100] [100] [100] [10] [10]

Now we have three sequences of three bits each, and a remainder of two sequences of two bits each. Therefore we continue in the same manner, by placing a [10] sequence after each [100] sequence to obtain:

[10010] [10010] [100]

The process stops when the remainder consists of only one sequence (in this case the sequence [100]), or we run out of zero's. The final sequence is thus the concatenation of [10010], [10010], and [100]:

[1 0 0 1 0 1 0 0 1 0 1 0 0]

Implementación en PD - Euclidean sequencer

$$\begin{tikzcd}\textbf{(index * hits) \% steps} \arrow[d] \\ \textbf{[< notas]}\end{tikzcd}$$

índice = índice de la euclid serie (array)

hits = notas

steps = pulsos

Rotación del ritmo euclídeo

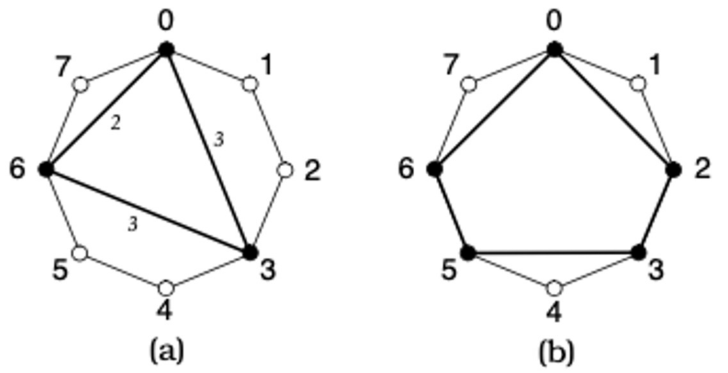


Figure 1: *(a) The Euclidean rhythm $E(3, 8)$ is the Cuban* tresillo*, (b) The Euclidean rhythm $E(5, 8)$ is the Cuban* cinquillo.

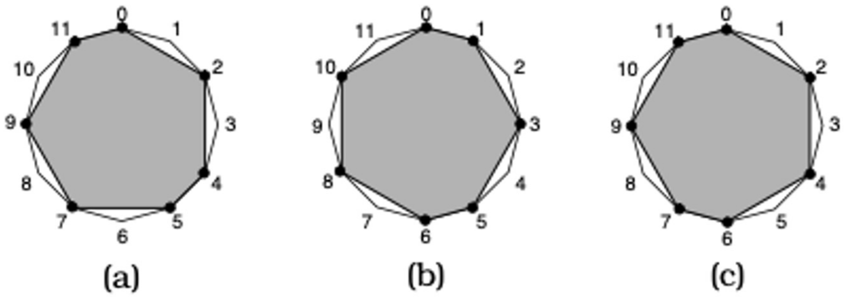


Figure 3: *Two right-rotations of the* Bemb´e *string: (a) the* Bemb´e*, (b) rotation by one unit, (c) rotation by seven units*.