Grafy i Złożoność Kołmogorowa

Mikolaj Morzy

September 19, 2015

Pakiet `acss` pozwala na wyznaczenie Złożoności Kołmogorowa (ZK), ale jedynie dla bardzo krótkich łańcuchów znaków (do 10-12 znaków w zależności od rozmiaru alfabetu). Dla przypomnienia, ZK(S) dla łańcucha znaków S to długość najkrótszego programu maszyny Turinga który posługując się alfabetem o zadanym rozmiarze potrafi napisać łańcuch S. Generalnie rzecz biorąc, ZS jest bardzo dobrą teoretyczną miarą stopnia losowości łańcucha znaków.

Istnieje duża różnica między entropią H(S) łańcucha znaków S i jego złożnością ZK(S). Entropia bierze pod uwagę jedynie rozkład częstości znaków alfabetu, a nie ich wzajemne ułożenie.

```
library(acss)
```

```
## Loading required package: acss.data
```

```
S1 <- '0010111001'
S2 <- '0000011111'

entropy(S1)
```

```
## 0010111001
##          1
```

```
entropy(S2)
```

```
## 0000011111
##          1
```

```
acss(S1, alphabet = 2)
```

```
##                 K.2          D.2
## 0010111001 29.23765 1.579757e-09
```

```
acss(S2, alphabet = 2)
```

```
##                 K.2          D.2
## 0000011111 27.28202 6.127647e-09
```

Jak widać, oba łańcuchy znaków mają identyczną entropię i inną Złożoność Kołmogorowa.

Pomysł wykorzystania Złożoności Kołmogorowa do porównywania grafów jest następujący:

1. dla zadanego grafu wyznaczyć jego macierz sąsiedztwa wierzchołków i posortować ją wierszami
2. posortowaną macierz sąsiedztwa zamienić na jeden długi łańcuch znaków
3. podzielić łańcuch znaków na fragmenty długości 10 znaków (dla których można policzyć ZK)
4. dla każdego fragmentu wyznaczyć jego ZK

5. uśrednić uzyskane ZK wszystkich fragmentów w celu znalezienia Złożoności Kołmogorowa całego łańcucha znaków

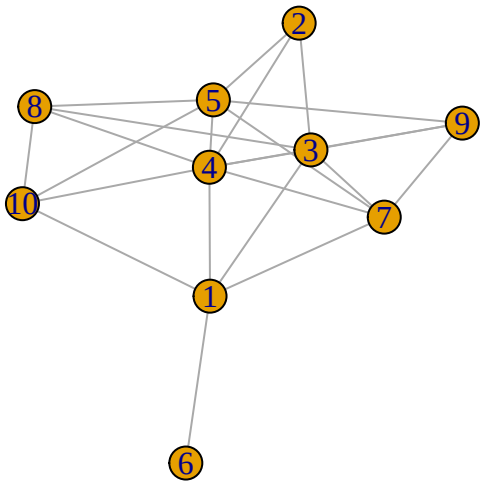
Wartość wyznaczoną w punkcie 6 można uważać za oszacowanie złożoności całego grafu. Poniżej kod realizujący ten plan.

```
library(igraph)
```

```
##
## Attaching package: 'igraph'
##
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
##
## The following object is masked from 'package:base':
##
##     union
```

```
# create a random graph according to the Erdos-Renyi model

graph <- sample_gnp(10, 0.5)
plot(graph)
```



Wyznaczenie macierzy sąsiedztwa, jej sortowanie i zamiana na łańcuch, dzielenie łańcucha na fragmenty.

```
# transform the graph into the adjacency matrix

adjacency.matrix <- as_adjacency_matrix(graph, type = 'lower')
adjacency.matrix
```

```
## 10 x 10 sparse Matrix of class "dgCMatrix"
##
##  [1,] . . . . . . . . . .
##  [2,] . . . . . . . . . .
##  [3,] 1 1 . . . . . . . .
##  [4,] 1 1 1 . . . . . . .
##  [5,] . 1 . 1 . . . . . .
##  [6,] 1 . . . . . . . . .
##  [7,] 1 . 1 1 1 . . . . .
##  [8,] . . 1 1 1 . . . . .
##  [9,] . . 1 1 1 . 1 . . .
## [10,] 1 . . 1 1 . . 1 . .
```

```
# sort the matrix row-wise

sorted.adjacency.matrix <- adjacency.matrix[ do.call(order, lapply(1:ncol(adjacency.matrix), function(x)
sorted.adjacency.matrix
```

```
## 10 x 10 sparse Matrix of class "dgCMatrix"
##
##  [1,] . . . . . . . . . .
##  [2,] . . . . . . . . . .
##  [3,] . . 1 1 1 . . . . .
##  [4,] . . 1 1 1 . 1 . . .
##  [5,] . 1 . 1 . . . . . .
##  [6,] 1 . . . . . . . . .
##  [7,] 1 . . 1 1 . . 1 . .
##  [8,] 1 . 1 1 1 . . . . .
##  [9,] 1 1 . . . . . . . .
## [10,] 1 1 1 . . . . . . .
```

```
# change the matrix into a string

adjacency.string <- paste(sorted.adjacency.matrix, collapse = '')
adjacency.string
```

```
## [1] "0000011111000010001100110001010011101100001100110000000000000000100000000000010000000000000000000
```

```
# chop string into equal length chunks
# acss can compute Kolmogorov's Complexity only for short strings

adjacency.string.chunks <- substring(adjacency.string, seq(1, nchar(adjacency.string), 10), seq(10, ncha
adjacency.string.chunks
```

```
##  [1] "0000011111" "0000100011" "0011000101" "0011101100" "0011001100"
##  [6] "0000000000" "0001000000" "0000001000" "0000000000" "0000000000"
```

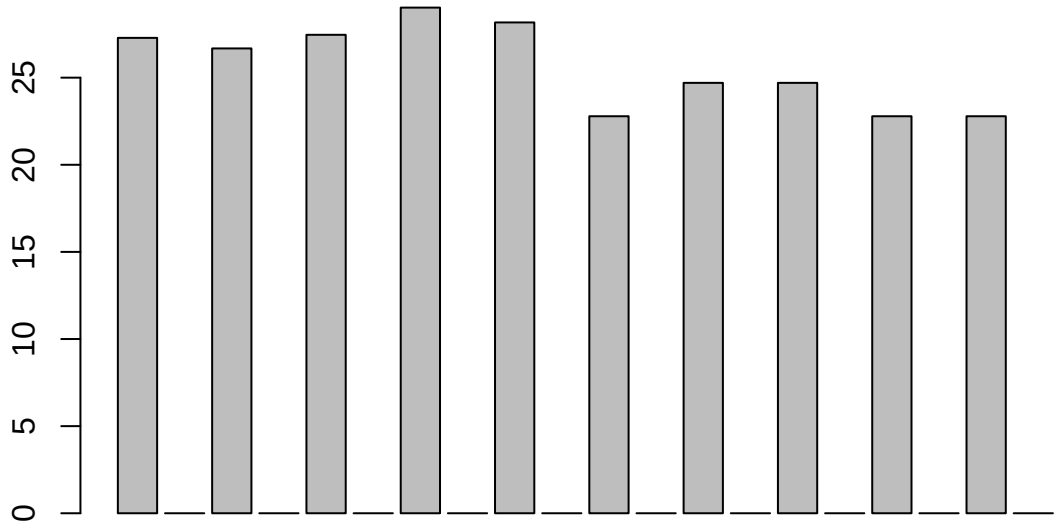
Wyznaczenie ZK dla poszczególnych fragmentów

```
# compute Kolmogorov's Complexity for all the chunks from the adjacency matrix

kolmogorov.complexities <- lapply(adjacency.string.chunks, acss, alphabet = 2)

# plot Kolmogorov's Complexities of all the chunks

barplot(unlist(kolmogorov.complexities))
```




```
# compute the mean Kolmogorov's Complexity over all chunks

mean.kolmogorov.complexity <- lapply(unlist(kolmogorov.complexities)[1], mean)
mean.kolmogorov.complexity
```

```
## [[1]]
## [1] 27.28202
```

Zatem, dla zadanego grafu `graph` uzyskujemy jego złożoność w wysokości 27.2820196.