Verificación de la existencia de un ciclo Hamiltoniano en un grafo aleatorio

Maquera Briguitte[1], Farut Jaafar[2] y Rivera Franklin[3]

***Abstract*—The project deals with the implementation of an algorithm that helps us to verify if a random graph is Hamiltonian with the help of the igraph package, libraries and functions that they offer in R language.**

***Abstract*—El proyecto trata sobre la implementación de un algoritmo que nos ayude a comprobar si un grafo aleatorio es Hamiltoniano con la ayuda del paquete *igraph*, librerias y funciones que ofrecen en lenguaje R.**

I. INTRODUCCIÓN

Objetivos:

1) El objetivo de nuestro proyecto es encontrar si existe o no un **Ciclo Hamiltoniano** en un grafo aleatorio dado sus números de aristas y vertices con ayuda de los paquetes que R ofrece.

Definiciones previas

• **Grafo**: Es un diagrama que representa mediante vertices y aristas las relaciones entre pares de elementos y que se usa para resolver problemas lógicos, topológicos y de cálculo combinatorio.

• **Grafo hamiltoniano**: Es aquel grafo que tiene un ciclo hamiltoniano el cual recorre una sola vez cada vertice y el vertice final sea adyacente al primero, de esa forma contiene un camino hamiltoniano o circuito hamiltoniano.

•¿Cómo identificar un grafo hamiltoniano?
Contrario al caso de los grafos eulerianos, para el caso de los grafos hamiltonianos no se conoce ninguna condición necesaria y suficiente que los caracterice. Esto es lamentable porque en muchas aplicaciones es fundamental poder determinar si un grafo es hamiltoniano.

•¿Qué es el Lenguaje de programación R?

Es un tipo de lenguaje de programación el cual es una implementación del lenguaje de programación S, creado en Auckland(New Zealand)

Características:
* R es un lenguaje pensado para la programación estadística y la creación de gráficos
* Posee mucho paquetes y librerias
* Es multi-paradigmático y Open Source ya que nos permite una facilidad en el uso de la escritura o implementación del código

Nota:RStudio es un entorno de desarrollo integrado (IDE) para el lenguaje de programación R, dedicado a la computación estadística y gráficos.

II. ESTADO DEL ARTE

1) El problema respecto a los grafos, específicamente los ciclos hamiltonianos, vienen siendo una adversidad histórica.
   En **1736** Leonhard Euler en uno de sus viajes en Koñigsberg en la costa del Mar Báltico, en la Prusia oriental (Rusia) habían siete puentes distribuidos donde planeó un paseo de manera que saliendo de casa cruce los siete puentes una sola vez cada uno antes de regresar a casa haciendo referencia a los caminos hamiltonianos.
   En **1805** Roman Hamilton se propuso a viajar a 20 ciudades del mundo, representadas como los vértices de un dodecaedro regular, siguiendo las aristas del dodecaedro.
   En **1824** Kirchoff se sirvió de la Teoría de Grafos para enunciar las leyes que permiten el cálculo de voltajes y circuitos eléctricos.
2) **Network Analysis and Visualization with R and igraph**
   This page gave us information about the various functions that we can use in Rstudio and also about the igraph package which will help us in the graph drawings in Rstudio

III. DISEÑO DEL EXPERIMENTO

A. ¿Es G hamiltoniano?

- No existe ningún método general válido aplicable a todos los grafos para determinar si es o no hamiltoniano.
- El método que trataremos a continuación es válido, en términos generales, si el grafo tiene vértices de grado dos y no tiene un gran número de aristas (aunque la aplicabilidad o no del método depende siempre del grafo en concreto).

[1]E.P. de Matemática-Facultad de Ciencias, Universidad Nacional de Ingeniería, briguittemaquera@gmail.com, código 20162254B
[2]E.P de Facultad de Ciencias, Universidad Nacional de Ingeniería, jaafarsahua@hotmail.com, código 20161395A
[3]E.P de Ciencias de la Computación, Facultad de Ciencias, Universidad Nacional de Ingeniería, friverag@uni.pe, código 20161331C

B. *¿Qué método utilizaremos?*

- El método será constructivista, buscando no sólo la existencia sino el ciclo hamiltoniano, caso de que exista.

C. *¿En qué se apoya?*

- El método se apoya en el hecho de que existe un ciclo hamiltoniano, éste debe contener exactamente dos de las aristas de cada uno de los vértices (por definición de ciclo).

D. *Estrategia*

- Dado un grafo no dirigido G, con $|v| > 2$, suponemos que tiene ciclo hamiltoniano e intentaremos construirlo a partir de cuatro reglas.
- Si las reglas 1 o 4 no se cumplen, el grafo no será hamiltoniano.
- En caso contrario habremos obtenido, tras un número determinado de pasos, un ciclo hamiltoniano en G.

E. *Reglas*

Suponemos que existe un ciclo hamiltoniano C en G.

Regla 1: Si existe ciclo hamiltoniano en G entonces todos los vértices tienen grado mayor o igual que dos.

Regla 2: Sea v un vértice de grado 2. Entonces las dos aristas incidentes en v deben pertenecer al ciclo C.

Regla 3: Si v es un vértice de grado mayor que 2, y ya hemos incorporado al ciclo C que estamos reconstruyendo dos de sus aristas, el resto de aristas incidentes en v deben ser desechadas.

Regla 4: Si el grafo es realmente hamiltoniano, con la construcción obligada que estamos realizando no podemos encontrar un ciclo que contenga un número de vértices menor que $|v|$.

IV. IMPLEMENTACIÓN EN R

A. *Funciones y técnicas a usar*

1) **Plot:**
La función plot es una función genérica para la representación gráfica de objetos en R. Los gráficos más sencillos que permite generar esta función son nubes de puntos (x, y).
2) **Grafos con igraph:**
El paquete para Igraph, necesita que se le presente los datos de la matriz de adyacencia por parejas. Es decir, una matriz de doble entrada convencional (también llamada sociomatriz, tabla de confundido o tabla de concordancia) ha de pasarse al formato de igraph.
3) **Archivos CSV:**
Archivo de texto que contiene una serie de valores separados por comas. Los valores pueden ser cualquier cosa, desde números de un presupuesto de una hoja de cálculo, hasta nombres y descripciones de una lista de clientes de un negocio.

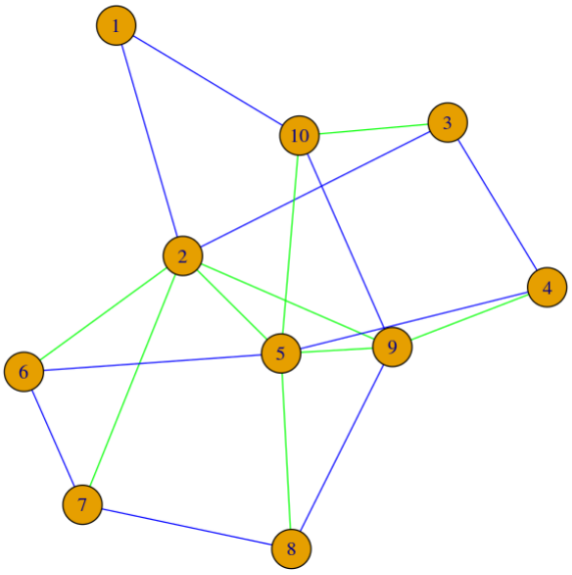
```
library(igraph)


## https://www.r-bloggers.com/lexicographic-permutations-euler-problem-24/

  nextPerm <- function(a) {
  #Encuentra el sufijo no creciente más largo.
  i <- length(a)

  while (i > 1 && a[i - 1] >= a[i])
    i <- i - 1
  #i es el índice de cabeza del sufijo.
  #¿Estamos en la última permutación?
  if (i <= 1) return(NULL)
  #a[i-1] es el pivote.
  #Encuentra el elemento más a la derecha que excede el pivote.
  j <- length(a)
  while (a[j] <= a[i - 1])
    j <- j - 1
  #intercambiamos el pivote con a[j].
  aux <- a[i - 1]
  a[i - 1] <- a[j]
  a[j] <- aux
  #Invierte el sufijo.
  a[i:length(a)] <- rev(a[i:length(a)])
  return(a)
}
```

```
esHamiltoniano <- function(g, cycle) {

  n <- vcount(g)

  p <- 1:n

  cont = 1
  for (i in 1:n)
    cont <= cont *  i

  hamiltoniano <- T
  while(cont > 0) {
    cont <- cont - 1
    for (i in 1:(n-1)) {
      if (!are_adjacent(g, p[i], p[i+1])) {
        hamiltoniano <- F
      }
    }

    if (!are_adjacent(g, p[n], p[1])) {
      hamiltoniano <- F
    }

    if (hamiltoniano == T) break
    p <- nextPerm(p)
  }

  if (hamiltoniano == F) return (F)
  eval.parent(substitute(cycle <- c(p, p[1])))
  return(T)
}
```

```
buildGraph <- function(n) {
  adjm <- matrix(sample(0:1, n*n, replace=T, prob=c(0.45, 0.55)), nr=n) ## [1] ##

  # hacemos simétrica (no dirigida) la matriz random.
  adjm <- (adjm + t(adjm))/2
  return(graph.adjacency(adjm, mode="undirected", diag=F))
}

n <- as.integer(readline(prompt = "Ingrese el orden del grafo: "))
while (TRUE) {
  # construimos el grafo hasta que sea hamiltoniano.
  g <- buildGraph(n)
  cycle <- c()

  if (esHamiltoniano(g, cycle)) {
    # visualiza los nodos tan grandes como su grado.
    # http://www.shizukalab.com/toolkits/sna/plotting-networks-pt-2
    #V(g)$size=degree(g)*5

    #dado el ciclo formamos las aristas.
    edgeCycle <- c()
    for (i in 1:(length(cycle)-1)) {
      edgeCycle <- c(edgeCycle, c(cycle[i], cycle[i+1]))
    }

    #cambiando de color al ciclo.
    E(g)$color <- 'green'    #color que no forma parte del ciclo hamiltoniano.
    E(g)$color[get.edge.ids(g, edgeCycle)] <- 'blue'                    #color del ciclo hamil

    plot(g)
    break
  }
  print("...")
}
```

V. EXPERIMENTOS Y RESULTADOS

```
Ingrese el orden del grafo: 10
[1] "..."
[1] "..."
[1] "..."
[1] "..."
[1] "..."
[1] "..."
[1] "..."
[1] "..."
[1] "..."
[1] "..."
[1] "..."
[1] "..."
[1] "..."
[1] "..."
[1] "..."
[1] "..."
[1] "..."
[1] "..."
[1] "..."
[1] "..."
[1] "..."
```



VI. DISCUSIONES

1) Al ejecutar el código nos solicitará el número vértices n.
2) luego se procede a construir el grafo aleatorio: Usando la función **sample()** generamos $n*n$ números aleatorios entre 0 y 1 , luego con la función **matrix()** creamos una matriz(**adjm**) a partir del conjunto de valores obtenidos el cual será nuestra matriz de adyacencia. luego hacemos simétrica la matriz usando la propiedad **(adjm + t(adjm)/2**.
3) Luego pasamos a verificar si el grafo obtenido es hamiltoniano:
   Entraremos a un bucle en el cuál su peor caso será n!, donde en el primer **for** usando la función $are\_adjacent()$ con el cual verificaremos si los vértices de las posiciones 1 hasta n-1 son adyacentes es decir si están conectados por un camino y por último verificamos si los vértices de la posición 1 y n son adyacentes. En caso no se cumpla lo anterior nos apoyamos de la función **nextPerm()** para permutar los vértices y seguir con el paso anterior hasta conseguir el grafo deseado.
4) ¿Cómo se puede mejorar sus resultados?

   La complejidad del ódigo es n! por lo cual es muy ineficiente y una manera de mejorarlo es optimizandolo e incluso se podría bajar hasta O(nlogn).

VII. CONCLUSIONES

Se pudó implementar un código en R y utilizando el paquete igraph(principalmente la función plot) que R nos ofrece se pudo visualizar dicho grafo aleatorio y verificar que era hamiltoniano.

REFERENCES

[1] **Libro**: *Invitación a Matemática Discreta*
Autores: "Jiri Matousek y Jaroslav Nesetril", (2008) *Universidad de Charles-Praga*, **páginas**: 105-142
[2] **Pagina Web-Video**: https://www.youtube.com/watch?v=LcL-tO2TMIY
[3] **Pagina Web-Artículo PDF**: https://www.ugr.es/ batanero/pages/ARTICULOS/libroR.pd
[4] **Libro**: *El arte de programar en R (Un lenguaje para la estadística)*
Autores: "Julio Sergio Santana y Efraín Mateos Farfán ", (2014) *Instituto Mexicano de Tecnología del agua*
[5] **Referencias del código:**

1) https://www.r-bloggers.com/lexicographic-permutations-euler-problem-24/
2) http://igraph.org/r/doc/
3) https://stat.ethz.ch/R-manual/R-devel/library/base/html/bitwise.html
4) https://stackoverflow.com/questions/4678333/nn-1-what-does-this-expression-do
5) https://www.r-graph-gallery.com/248-igraph-plotting-parameters/
6) http://www.shizukalab.com/toolkits/sna/plotting-networks-pt-2