# 1 Introducción a Redes Complejas en Biología de Sistemas

## 1.1 Trabajo Computacional 1

### 1.1.1 Ejercicio 2

In [1]:
```
import pandas as pd
import numpy as np
import networkx as nx
import matplotlib.pylab as plt
import random as ran
import matplotlib
import scipy.stats
from tqdm import tqdm_notebook as tqdm
```

**Considere la red social de 62 delfines de Nueva Zelanda**

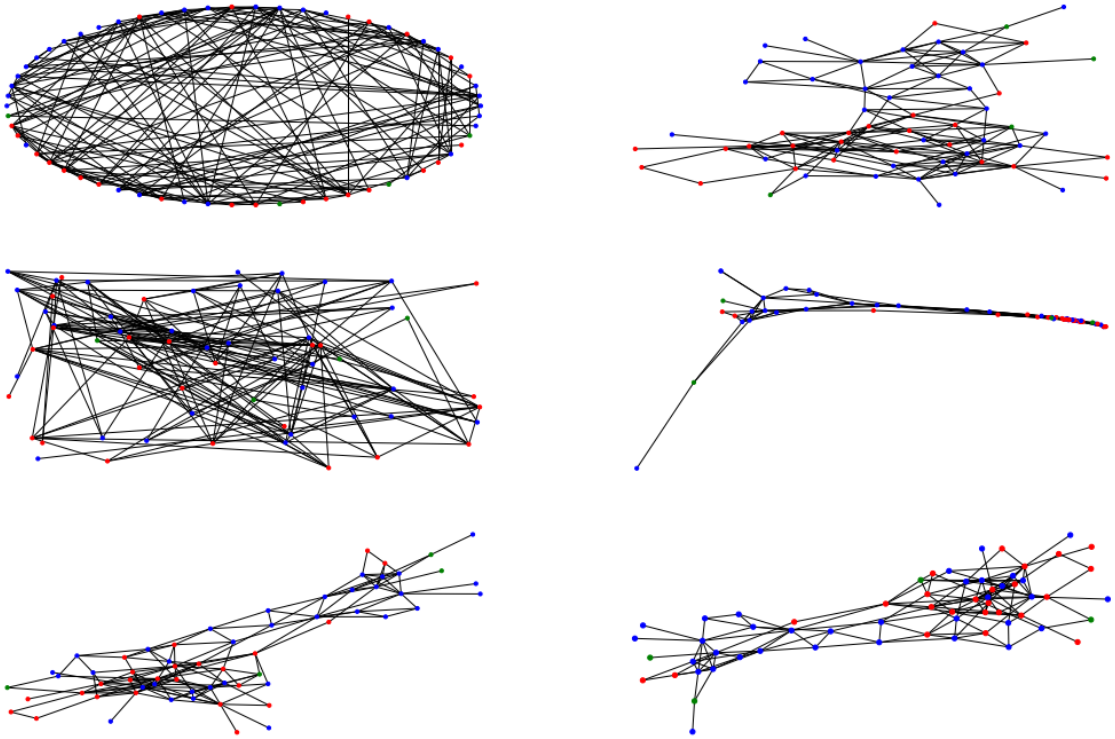
a. Examine diferentes opciones de layout para este grafo e identifique la que le resulte más informativa. Justifique su elección detallando las características estructurales de la red que su elección pone en evidencia. Incluya en la representación gráfica de la red información sobre el sexo de los delfines.

In [3]:
```
#cargando archivos:
file_d='.\\tc01_data\\dolphins.gml'
f_genero='..\\tc01_data\\dolphinsGender.txt'
g = nx.read_gml(file_d)
```

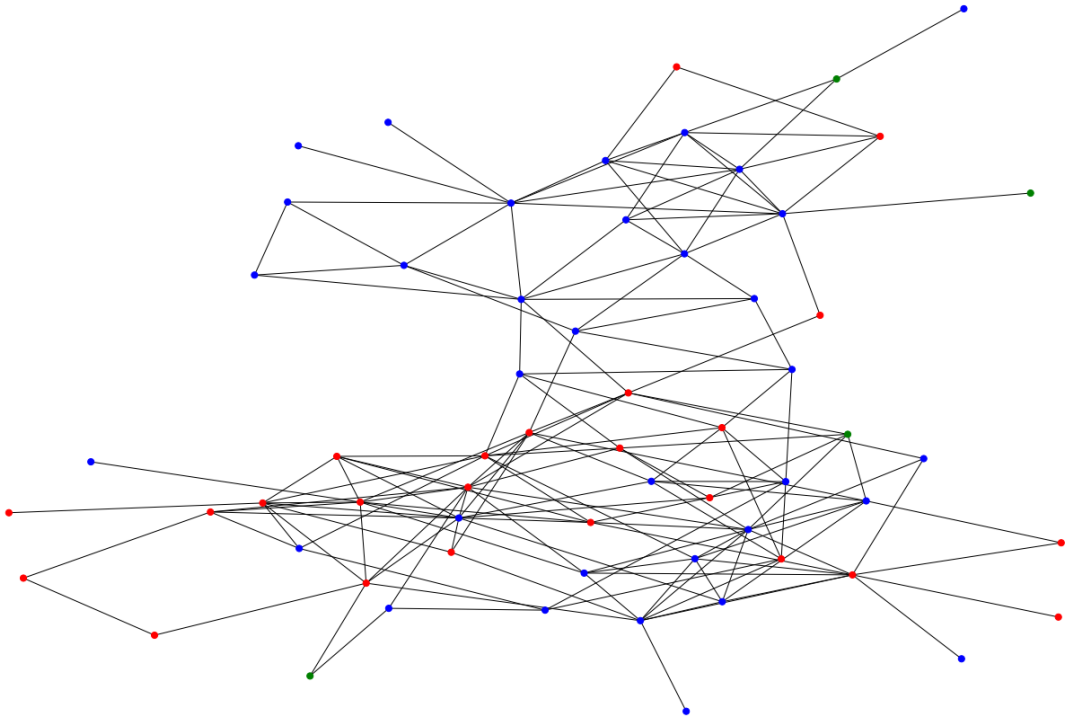
In [4]:
```
#Crenado un diccionario a partir del archivo donde cada nodo tiene referenciado su
def ldata(archive):
    f=open(archive)
    data={}
    for line in f:
        line=line.strip()
        col=line.split()
        data[col[0]]=col[1]
    return data

dic_genero=ldata(f_genero)
```

In [5]:
```
#Datos para los plots (color segun sexo):
def Colores_Generos(gr, dic):
    l_generos=[]
    for i in gr.nodes():
        if dic[i]=='m':
            l_generos.append('b')
        elif dic[i]=='f':
            l_generos.append('r')
        else:
            l_generos.append('g')
    return l_generos

ls_gen=Colores_Generos(g,dic_genero)
```

In [6]:
```
#plots con distintos layouts:
matplotlib.rcParams['figure.figsize'] = [18, 12]

plt.subplot(321)
g1=nx.draw_circular(g,node_color=ls_gen,arrows=True,node_size=10)


plt.subplot(322)
g2=nx.draw_kamada_kawai(g,node_color=ls_gen,arrows=True,node_size=10)


plt.subplot(323)
g3=nx.draw_random(g,node_color=ls_gen,arrows=True,node_size=10)


plt.subplot(324)
g4=nx.draw_spectral(g,node_color=ls_gen,arrows=True,node_size=10)


plt.subplot(325)
g5=nx.draw_spring(g,node_color=ls_gen,arrows=True,node_size=10)


plt.subplot(326)
nx.draw(g,node_color=ls_gen,arrows=True,node_size=18)

plt.show()
```



In [7]:
```
#layout seleccionado:
nx.draw_kamada_kawai(g,node_color=ls_gen,arrows=True,node_size=50)
plt.show()
```



Se selecciono el layout **kamada kawai** ya que es uno de los layouts que mejor distribuye los nodos dismuyendo el solapamiento de aristas y permite asi, que la representacion de la red y su visualizacion sean mas amigable y accesible. Tambien, en este layout se destacan nodos esenciales, para que la red mantenga una sola componente (nodos puentes entre los 2 clusters).

b. Se trata una red donde prevalece la homofilia en la variable género? Para responder

i. Considere la distribución nula para la fracción de enlaces que vinculan géneros diferentes, generada a partir de al menos 1000 asignaciones aleatorias de género.

ii. A partir de lo obtenido proponga una estimación para el valor y el error de dicha cantidad cuando no existe vínculo entre topolgía de la red medio y asignación de género. Compare su estimación con el valor medio esperado.

iii. Estime la significancia estadística (p-valor) del valor observado en el caso de la red real.

In [8]:
```
#homofilia:

#devuelve listas con todos los nodos del genero que se le pase como parametro.
def nodos_del_genero(s,gr, dic):
    nodes=list(gr.nodes())
    nodes_g=[]
    for n in nodes:
        if dic[n]==s:
            nodes_g.append(n)
    return nodes_g

#listas de males, females y sin genero:
males=nodos_del_genero('m',g,dic_genero)
females=nodos_del_genero('f',g,dic_genero)
sin_gen=nodos_del_genero('NA',g,dic_genero)
```

In [9]:
```
#Calculando el valor esperado de edges entre generos distintos (para esto se decid

n_nodos_con_genero=len(list(g.nodes()))-len(sin_gen)

p=float(len(males))/float(n_nodos_con_genero)
q=float(len(females))/float(n_nodos_con_genero)

edges_esperados_intergenero=round(2*p*q,2)
#edges_esperados_intergenero

print('El valor da 0,49; es decir que se espera que el 49% de los edges de la red
```

```
El valor da 0,49; es decir que se espera que el 49% de los edges de la red sea
n entre nodos de distintos sexos.
```

In [10]:
```
#Calculando el valor observado de edges entre generos distintos (para esto se deci

edges=list(g.edges())

#se seleccionan los edges que tiene en los extremos sexos opuestos:
def edges_generos(edg,dic_g):
    f=0
    for e in edg:
        if dic_g[e[0]]!='NA' and dic_g[e[1]]!='NA':
            if len(np.unique((dic_g[e[0]],dic_g[e[1]])))==2:
                f+=1
    return f

edg_intergenero=edges_generos(edges,dic_genero)

#se genera una lista de los edges que tienen en algun extremo un nodos sin sexo as
#posteriormente no tenrlos en cuenta a la hora de calcular frecuencias.
def edges_sin_genero(gr,dic):
    edges_sin_genero=[]
    nodes=list(gr.nodes())
    for n in nodes:
        if dic[n]=='NA':
            nb=list(g.neighbors(n))
            edges_sin_genero=edges_sin_genero+nb
    return edges_sin_genero


n_edges_con_generos=float(len(edges)-len(edges_sin_genero(g,dic_genero)))

edges_obs_intergenero=round(float(edg_intergenero)/n_edges_con_generos,2)
(n_edges_con_generos,edges_obs_intergenero)

print('El valor da 0,35; es decir que el 35% de los edges de la red son entre nodo
```

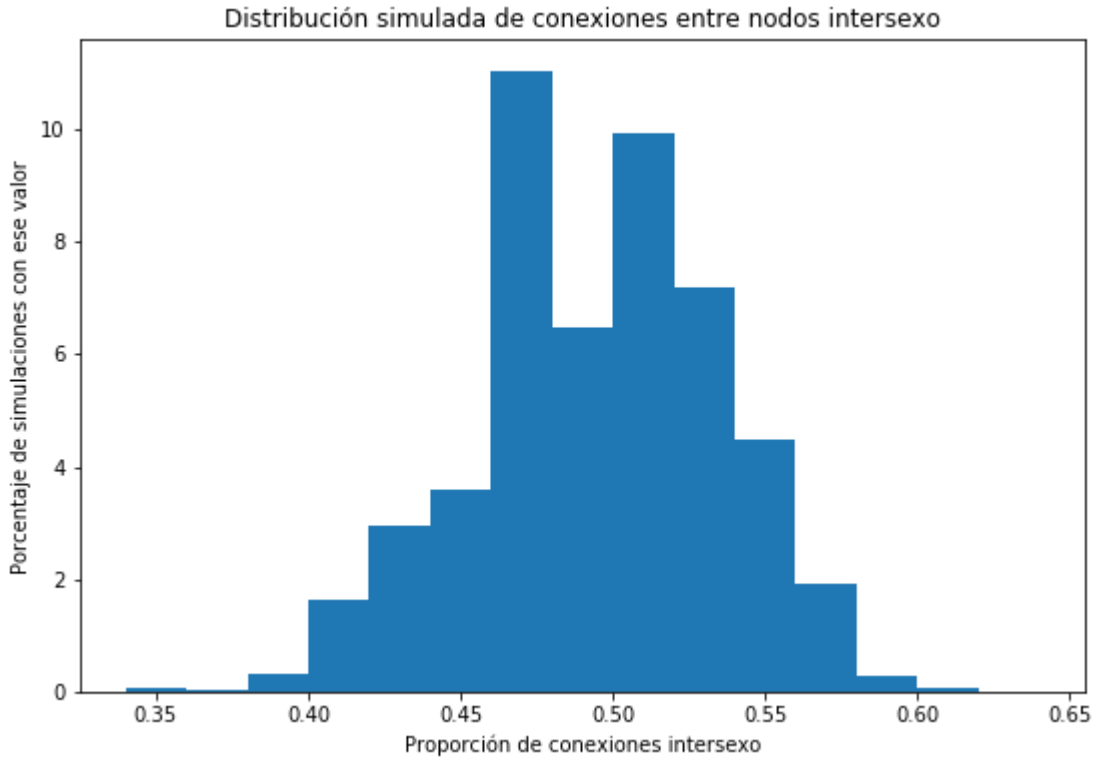
```
El valor da 0,35; es decir que el 35% de los edges de la red son entre nodos d
e distintos sexos.
```

Hasta ahora tenemos que el valor esperado de aristas intersexos para la red es de 0,49 y el observado es de 0,35. Para saber si esta diferencia es significativa tendriamos que simular asignaciones de sexo al azar para dicha red (respetando las proporciones de los sexos). A continuacion se simularon 10000 asignaciones al azar, y se analizo la distribucion generada a partir de dichas asignaciones.

In [11]:
```
##random generos:

#se asigna al azar generos a los nodos de la red (respetando las proporciones),
#los nodos sin genero no son tenidos en cuenta, devuelve un dic con los nodos de l
def random_dic_generos(m,f,sin):
    nodos_con_genero=m+f
    dic_sin_genero=dict((x,'NA') for x in sin)
    random_males=ran.sample(nodos_con_genero, len(m))
    dic_random_male=dict((i,'m')for i in random_males)
    dic_random_female=dict(dict((x,'f') for x in nodos_con_genero if x not in rand
    dic_random=dict(dic_random_male, **dic_random_female)
    return dic_random

#se genera una distribucion a partir de 10000 iteraciones de asiganar generos al a
ls_estimados=[]
for i in tqdm(range(10000)):
    dic_ran_generos=random_dic_generos(males,females,sin_gen)
    n_edges_intergenero=edges_generos(edges,dic_ran_generos)
    estimado=round(float(n_edges_intergenero)/n_edges_con_generos,2)
    ls_estimados.append(estimado)
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In [12]:
```
matplotlib.rcParams['figure.figsize'] = [9, 6]

#plot del histograma de la distribucion random:
plt.hist(ls_estimados,bins=15, density=True)
plt.title('Distribución simulada de conexiones entre nodos intersexo')
plt.xlabel('Proporción de conexiones intersexo')
plt.ylabel('Porcentaje de simulaciones con ese valor')
plt.show()
```




In [13]:
```
#Por ultimo se calcula el p-valor para el valor observado en nuestra red, para pod

menores = len([x for x in ls_estimados if x < edges_obs_intergenero])
( menores , 10000 - menores,float(menores)/10000 )
```

Out[13]: (4, 9996, 0.0004)

El p-valor es de 0.0005 (menor al valor critico de 0.05), es decir que hay diferencia significativa entre nuestra red y una donde los extremos de las aristas se encuentran distribuidos de forma aleatoria. Con lo cual podriamos decir que se trata de una red homofilica para el sexo, donde sexos iguales tienden a relacionarse entre ellos.

# 2 Punto C

Identifique alguna metodología basada en observables topológicos para eliminar nodos secuencialmente de la red de manera de dividirla en dos componentes de tamaños comparables en el menor número de pasos. Explique y muestre los resultados obtenidos. Intente cuantificar su estrategia comparándola con lo que se obtendría al eliminar nodos de manera aleatoria.

In [14]:
```
inter_sexo = {} # clave nodo, cantidad aristas intersexo
for node in g.nodes():
    inter_sexo[node] = 0
    for n1,n2 in g.edges(node):
        if (n1 in males and n2 in females ) or (n2 in males and n1 in females ):
                inter_sexo[node] += 1
nodos_ord_intersex = sorted( inter_sexo.items(),key=lambda x:x[1],reverse=True )
#nodos_ord_intersex
```

In [15]:
```
#desarmando la red por nodos con mayor cantidad de aristas intersexos:

g_a_desarmar = g.copy()
tamanios =[] #tamanios de los componentes
for nodo,cant in nodos_ord_intersex :
    ccs = sorted(nx.connected_components( g_a_desarmar ), key=lambda x:len(x),reve
#     tamanios.append([len(ccs[0]),len(ccs[1]) if len(ccs) > 1 else 0 ])
    tamanios.append([len(cc) for cc in  ccs])
    g_a_desarmar.remove_node(nodo)


#suponemos que dos redes son comparables en tamanio si la diferencia es como maxim
s=0
flag=True
for x in tamanios:
    if len(x)>1 and flag and x[0]-x[1]<=10:
        flag=False
        pasos_para_comp_comparables=s
    s+=1


print(pasos_para_comp_comparables)

#for x in tamanios:
#    print x
```

```
36
```

In [16]:
```
##ahora hacmeos exactamente lo mismo pero desarmando por grado:

nodos_por_grado={}
for node in g.nodes():
    nodos_por_grado[node]=len(list(g.neighbors(node)))

nodos_por_grado_ord=sorted( nodos_por_grado.items(),key=lambda x:x[1],reverse=True



g_a_desarmar_grdo = g.copy()
tamanios =[]

for nodo,cant in nodos_por_grado_ord :
    ccs = sorted(nx.connected_components( g_a_desarmar_grdo ), key=lambda x:len(x)
#    tamanios.append([len(ccs[0]),len(ccs[1]) if len(ccs) > 1 else 0 ])
    tamanios.append([len(cc) for cc in  ccs])
    g_a_desarmar_grdo.remove_node(nodo)

s=0
flag=True
for x in tamanios:
    if len(x)>1 and flag and x[0]-x[1]<=10:
        flag=False
        pasos_para_comp_comparables_grdo=s
    s+=1


print(pasos_para_comp_comparables_grdo)

#for x in tamanios:
#    print x
```

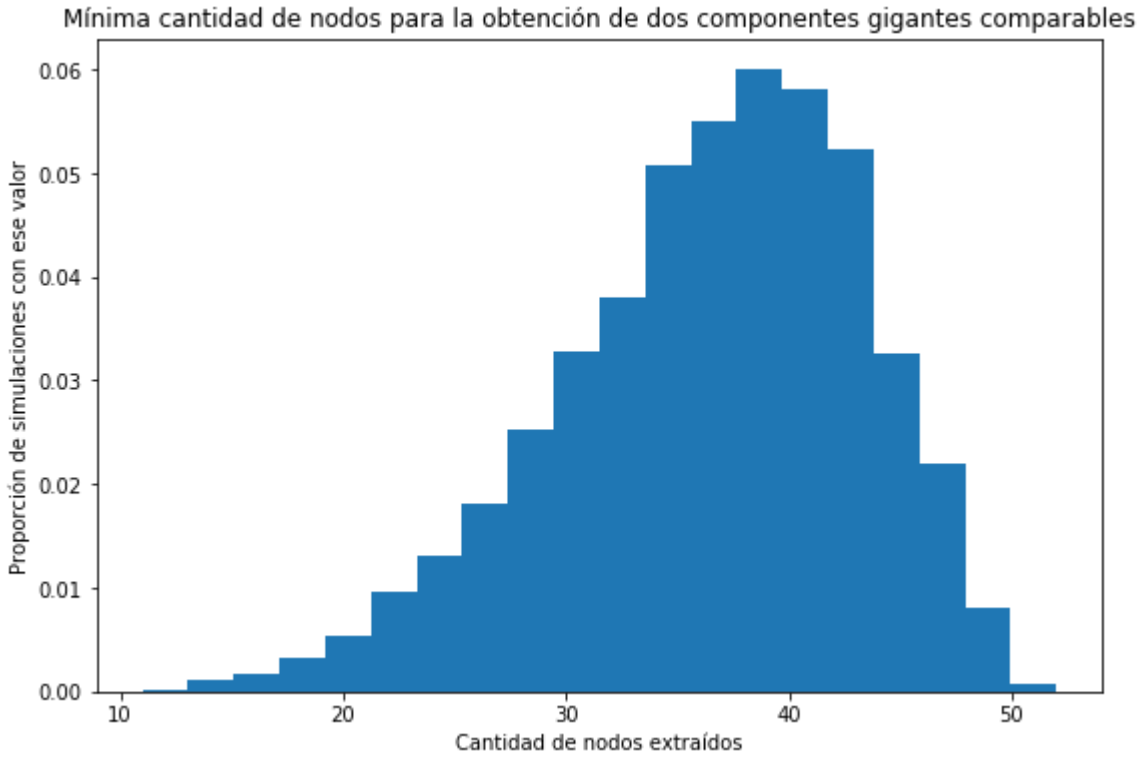
```
28
```

Entonces, tenemos dos formas distintas de ir desarmando las redes: una es por el grado y la otra es por la cantidad de aristas intersexos que tienen los nodos. Lo que queremos ver es que la segunda forma es más eficiente a la hora de desarmar la red en componentes comparables que la primera. Para esto desarmamos la red 10000 veces sacando nodos al azar y veremos si alguna de las formas difiere significativamente, con un p-valor menor a 0,05.

In [17]:
```
#desarmando 10000 veces la red quitando nodos al azar:

n_pasos_componentes_comparables_random=[]#cada elemento es cuantos pasos se tuvo q
for i in tqdm(range(10000)):
    g_a_desarmar_random = g.copy()
    tamanios =[]
    random_nodes=list(g.nodes())
    ran.shuffle(random_nodes)
    for nodo in random_nodes :
        ccs = sorted(nx.connected_components( g_a_desarmar_random ), key=lambda x:
        tamanios.append([len(ccs[0]),len(ccs[1]) if len(ccs) > 1 else 0 ])
#         tamanios.append([len(cc) for cc in  ccs])
        g_a_desarmar_random.remove_node(nodo)
    s=0
    flag=True
    for x in tamanios:
        if len(x)>1 and flag and x[0]-x[1]<=10:
            flag=False
            r=s
        s+=1
    n_pasos_componentes_comparables_random.append(r)

#n_pasos_componentes_comparables_random
```

A Jupyter widget could not be displayed because the widget state could not be found. This could happen if the kernel storing the widget is no longer available, or if the widget state was not saved in the notebook. You may be able to create the widget by running the appropriate cells.

In [18]:
```
matplotlib.rcParams['figure.figsize'] = [9, 6]

#plot del histograma de la distribucion random:
plt.hist(n_pasos_componentes_comparables_random,bins=20, density=True)
plt.title('Mínima cantidad de nodos para la obtención de dos componentes gigantes
plt.xlabel('Cantidad de nodos extraídos')
plt.ylabel('Proporción de simulaciones con ese valor')
plt.show()
```




In [19]:
```
#calculando p-valor para desarmar por cantidad de aristas intersexo:

menores = len([x for x in n_pasos_componentes_comparables_random if x < pasos_para
( menores , 10000 - menores,float(menores)/10000 )
```

Out[19]: (4078, 5922, 0.4078)

In [20]:
```
#calculando p-valor para desarmar por grado:

menores = len([x for x in n_pasos_componentes_comparables_random if x < pasos_para
( menores , 10000 - menores,float(menores)/10000 )
```

Out[20]: (1070, 8930, 0.107)

Para intersexo obtenemos un p-valor de 0,02, lo cual indica que es significativamente distinto que desarmar la red tomando nodos al azar. En cambio al desarmar por grado obtenemos un p-valor de 0.06, lo cual indica que no hay diferencia significativa con desarmar la red tomando nodos al azar. Por lo tanto desarmar la red por el numero de aristas intersexo parece ser una mejor opción a la hora de partir la red en componentes comparables.