Universidad de los Andes
Modelos de Interacciones Sociales
Taller 1

Integrantes:
Juan Felipe Agudelo Rios
David Santiago Caraballo Candela
Rafael Andrés Santofimio Rivera
Juan Diego Valencia Romero

13 de septiembre de 2022

GitHub: Repositorio Taller 1

1 Eficiencia y Estabilidad

1. Calcule todas las posibles redes de tamaño cinco (5) y guárdelas en un listado. Asegúrese de guardar únicamente las redes que no son isomorfas entre sí, i.e. guarde únicamente un representante de cada conjunto de redes isomorfas entre sí.

Código

```
def matrix_creator(list, n):
   m=len(list)
   if 2*m==(n*(n-1)):
      counter=1
      A=[]
      AA=[0 for i in range(n)]
      while counter<n+1:
         limit=n-counter
         B=list[:limit]
         list=list[limit:]
         C=AA[:counter]+B
         A.append(C)
         counter+=1
      A=np.array(A)
      return A+np.transpose(A)
   else:
      print("Fill with right inputs")

def network_creator(n):
   A=[0]*n
   B=[1]*n
   saver=set()
   counter=1
   while counter<n:
      C=B[:counter]+A[counter:]
      counter=saver.union(set(it.permutations(C, n)))
      counter+=1
   saver=[list(i) for i in saver]
   saver=[A]+saver+[B]
   return(saver)

n=5
long=int(((n-1)*n)/2)
adjM=[matrix_creator(i, n) for i in network_creator(long)]
gs=[nx.from_numpy_array(i) for i in adjM]

revisor=i:True for i in range(len(gs))
   for i in revisor:
   for j in revisor:
      if j>i and revisor[i] and revisor[j]:
      if nx.is_isomorphic(gs[i], gs[j]):
         revisor[j]=False

gs_def=[gs[i] for i in revisor if revisor[i]]
adjM_def=[adjM[i] for i in revisor if revisor[i]]
```

2. Calcule la utilidad de cada nodo para cada una de las redes que creó y guarde esto como un atributo de las redes. Utilice la función de utilidad dada por $U_i(g) = deg(i)^{\frac{1}{2}}$.

Código

```
def utility_function(g):
   utility_0=dict(g.degree())
   utility=i:(utility_0[i])**0.5 for i in utility_0
   return utility

utilities=[utility_function(i) for i in gs_def]
for i in range(len(gs_def)):
   nx.set_node_attributes(gs_def[i], utilities[i], "Utility")
```

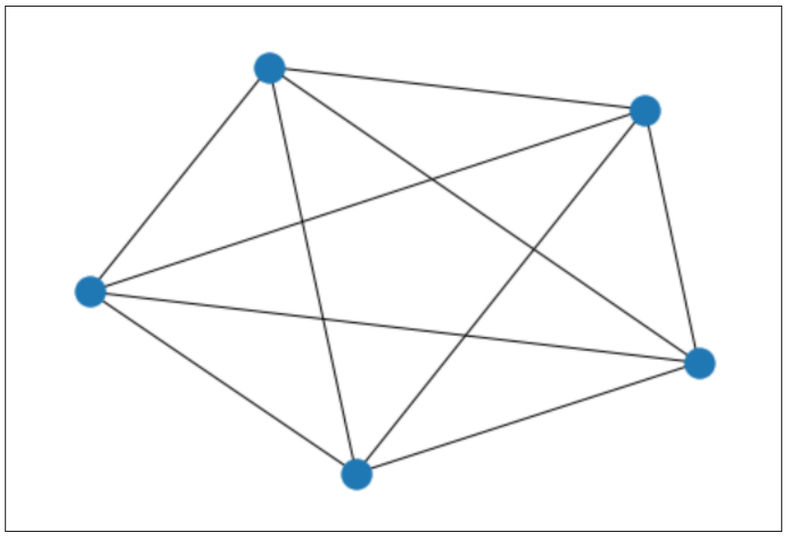
3. Calcule los grafos socialmente eficientes (SE). Reporte cada uno de estos.

Código

```
social_utility=([sum(list(nx.get_node_attributes(gs_def[i], "Utility")
.values())) for i in range(len(gs_def))])
max_social_utility=max(social_utility) gs_SE=[gs_def[i] for i in
range(len(social_utility)) if social_utility[i]==max_social_utility]

for i in gs_SE:
   print("_"*30)
   nx.draw(i)
```

Figura 1. Grafo social-eficiente.



4. Calcule los grafos Pareto-eficientes (PE). Reporte cada uno de estos.

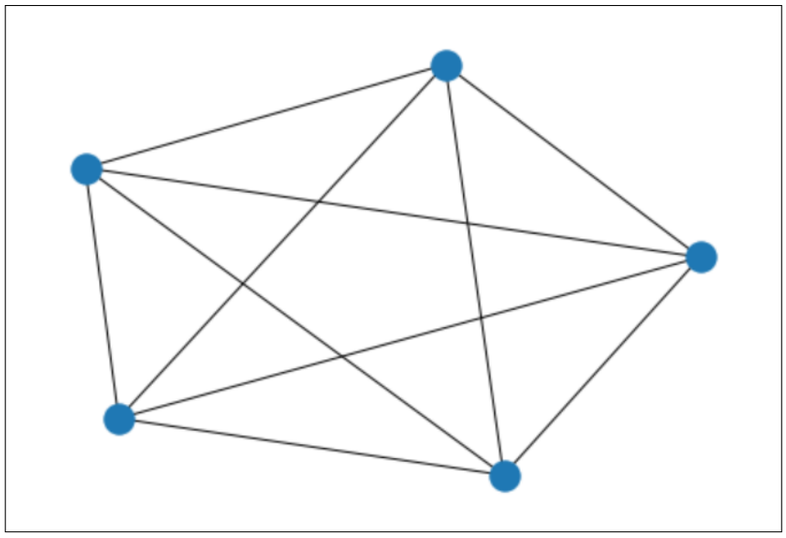
Código

```
utilities_i=([list(nx.get_node_attributes(gs_def[i], "Utility")
.values()) for i in range(len(gs_def))])
gs_PE=[]

for k in range(0, len(gs_def)):
   a=([sum(np.greater_equal(utilities_i[j], utilities_i[k]))
   for j in range(0,len(gs_def)) if j!=k and
   sum(np.equal(utilities_i[j], utilities_i[k]))!=5])
   if max(a)<5:
      print(k)
      gs_PE=gs_PE+[gs_def[k]]

for i in gs_PE:
   print("_"*30)
   nx.draw(i)
```

Figura 2. Grafo Pareto-eficiente.



5. Calcule los grafos estables (E). Reporte cada uno de estos.

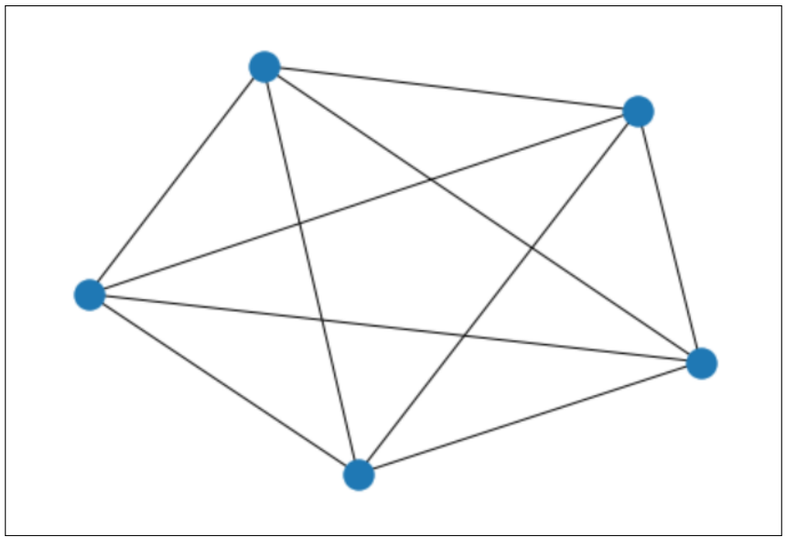
Código

```
degrees=([list(dict(gs_def[j].degree).values()) for j in
range(len(gs_def))])
gs_E=[]

for k in range(0,len(gs_def)):
   degrees_k=degrees[k]
   sumcu=[]
   for z in range(0,len(gs_def)):
      if z!=k:
         degrees_z=degrees[z]
         utilities_ik=utilities_i[k]
         utilities_iz=utilities_i[z]
         computilities=([np.greater(utilities_iz[h],utilities_ik[h])
         for h in range (0,5) if degrees_k[h]-1==degrees_z[h] or
         degrees_k[h]+1==degrees_z[h]])
         sumcu+=[sum(computilities)]
   if max(sumcu)==0:
      gs_E+=[gs_def[k]]

for i in gs_E:
   print("_"*30)
   nx.draw(i)
```

Figura 3. Grafo estable.



6. Calcule el precio de estabilidad (P_E) y el precio de anarquía (P_A). Muestre la relación entre ambos precios.

Código

```
utilities_E=([sum(list(nx.get_node_attributes(gs_E[i],"Utility")
.values())) for i in range(len(gs_E))])
UEmax=max(utilities_E)
UEmin=min(utilities_E)
PE=max_social_utility/UEmax
PA=max_social_utility/UEmin
print(PE, PA)
```

Debido que el precio de estabilidad (PE) y el precio de anarquía (PE) son funciones de la utilidad del social-eficiente y que, a su vez, hay un (1) único grafo estable, estos precios son iguales pues la utilidad máxima y mínima que toma la red son iguales. Así las cosas, el valor calculado de PE y PA equivale a uno (1), lo que cumple la condición de $PE \leq PA$. Lo anterior implica que para alcanzar la estabilidad de la red, se deben generar los mismos enlaces requeridos que en el máximo social (i.e. las máximas posibles) y, dado que solo hay una (1), el comportamiento egoísta de los nodos (agentes) debe ceñirse también al eficiente.

2 Medidas sobre Redes

Usando el archivo de relaciones en Twitter y lo aprendido en clase, realice los siguientes puntos:

1. Cree una función que permita calcular la medida de importancia Bonanich.

Código

```
def Bonanich(Network):
   M=nx.to_numpy_matrix(Network)
   auva, auve=sp.sparse.linalg.eigs(M, k=1)
   auve_G=auve.flatten().real
   norm=np.sign(auve_G.sum())*np.linalg.norm(auve_G)
   vectores=(auve_G/norm).astype(float)
   centrality=dict(zip(g_twi, vectores))
   return centrality
```

2. Cree una función que permita calcular la medida de importancia Hubs.

Código

```
def Hubs(Network, percen):
   df=pd.DataFrame()
   Hubs=[i[0] for i in Network.edges()]
   df["Hubs"]=Hubs
   Hubs=df["Hubs"].value_counts(normalize=percen)
   return Hubs
```

3. Cree una función que permita calcular la medida de importancia Authorities.

Código

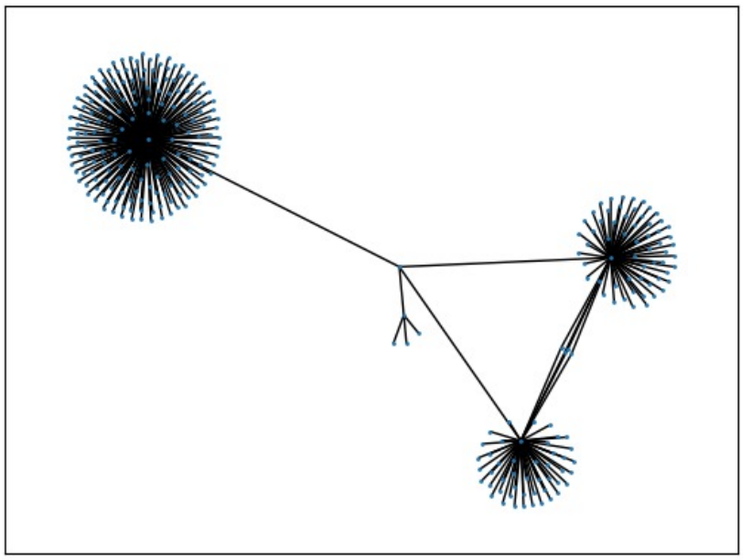
```
def Author(Network, percen):
   df=pd.DataFrame()
   Author=[i[1] for i in Network.edges()]
   df["Author"]=Author
   Author=df["Author"].value_counts(normalize=percen)
   return Author
```

4. Cargue los primeros 300 enlaces del archivo de Twitter y cree la red correspondiente.

Código

```
twi=pd.read_csv("2020_20_MIS_T1_Twitter.csv")
twi.pop("Unnamed:  0")
g_twi=nx.Graph()
g_twi=nx.from_pandas_edgelist(twi, source="From", target="To")
M_twi=nx.to_numpy_matrix(g_twi)
nx.draw_networkx(g_twi,with_labels=False, node_size=1)
plt.savefig("net.jpg")
```

Figura 4. Red de Twitter.



5. Calcule la medida de importancia de Bonanich, Hubs y Authorities para la red creada usando las funciones que ya propuso.

Código

```
HUBS=Hubs(g_twi, percen=True)
s=pd.Series(HUBS, name="HUBS")
s=s.to_frame().reset_index()
s.columns=["ID (nodo)", "Hubs"]
s=s.sort_values(by="Hubs", ascending=False)
HUBS=s
HUBS

AUTHOR=Author(g_twi, percen=True)
s=pd.Series(AUTHOR, name="AUTHOR")
s=s.to_frame().reset_index()
s.columns=["ID (nodo)", "Authorities"]
s=s.sort_values(by="Authorities", ascending=False)
AUTHOR=s
AUTHOR

BONA=Bonanich(g_twi)
s=pd.DataFrame.from_dict(BONA, orient="index", columns=["Bonanich"])
s=s.reset_index()
s.columns=["ID (nodo)","Bonanich"]
s=s.sort_values(by="Bonanich", ascending=False)
BONA=s
BONA
```

Tabla 1. Valores destacados de las medidas de importancia.

ID (nodo)	Authorities	ID (nodo)	Hubs	ID (nodo)	Bonanich
20	0,016	41	0,600	41	0.71
38253	0,003	20	0,196	8762941	0.05
8762942	0,003	3	0,166	293	0.05
3	0,003	1	0,013	1067925	0,05
41	0,003	8762941	0,010	8762993	0,05

Nota: Solo se consideran los cinco (5) nodos con el valor más alto.

6. Comente los resultados que obtuvo en los incisos anteriores. Use estadísticas descriptivas. ¿Tienen sentido?, ¿fueron bien aplicadas las medidas?

Tabla 2. Estadísticas descriptivas de las medidas de importancia.

Estadística	Authorities	Hubs	Bonanich
Conteo	296	9	297
Media	1,01	33,33	0,03
Desviación	0,23	59,59	0,046
Mínimo	1	1	0,0003
P25	1	1	0,0005
Mediana	1	3	0,05
P75	1	50	0,05
Máximo	5	180	0,71

Nota: Para construir las estadísticas descriptivas se utilizaron las funciones de Authorities y Hubs sin normalizar.

La Figura 4, de 297 nodos y 300 conexiones, identifica que, en promedio, todos los nodos exceptuando uno (1) tienen por lo menos una (1) conexión entrante. El nodo con mayor conexiones dispone de cinco (5) mientras que el de menor es de uno (1), desplegando una heterogeneidad entre nodos pues la mayoría recibía uno (1) e incluso solo una (1) observación recibió un número más alto que es el nodo con el ID 20 según nuestras medidas. Mientras que sólo eran nueve (9) aquellos nodos que originaban todas las conexiones. En promedio estos nueve (9) nodos eran el origen de 33,3 conexiones dentro de la red el que menos originó conexiones solo generó una (1) conexión mientras que el que más originó conexiones en la red generó 180. Con respecto a la medida de centralidad de Bonanich se observa que en promedio todos los nodos tienen una importancia de 0,03 mientras que el que mas tiene importancia tiene 0,05 por ultimo el nodo menos importante tiene una centralidad de 0,0003.

3 Producción

Usando la matriz insumo-producto por actividad para 2017 del Departamento Administrativo Nacional de Estadística (DANE) (cuadro 4, actividades A - R+S):

1. Construya la matriz A, genere un archivo .csv que adjunte en el .zip y que se llame A.csv.

Código

```
A=pd.read_excel(r"Matriz insumo-producto 2017.xlsx", "MIP",
engine="openpyxl")
A.rename(columns="Unnamed:  0":"Sectores", inplace=True)
A.set_index("Sectores", inplace=True)
num=A.iloc[:, 0:24]
B=A.iloc[:, 0:25]
d=A.iloc[:, 24:25]
den=B.sum(axis=1)
den=pd.DataFrame(den)
den=den[den.columns.repeat(24)]
den=den.T
cols=list(den.columns.values)
den["Sectores"]=cols
den.set_index("Sectores", inplace=True)
A=num/den
A=pd.DataFrame(A)
A.to_csv("A.csv", decimal=",", index=False)
```

2. Construya la matriz inversa de Leontief.

Código

```
L=np.linalg.inv(np.identity(24)-A)
```

3. Construya el vector de demanda dado por la variable `Gasto de Consumo Final`. Debe generar un archivo .csv que adjunte en el .zip y que se llame d.csv.

Código

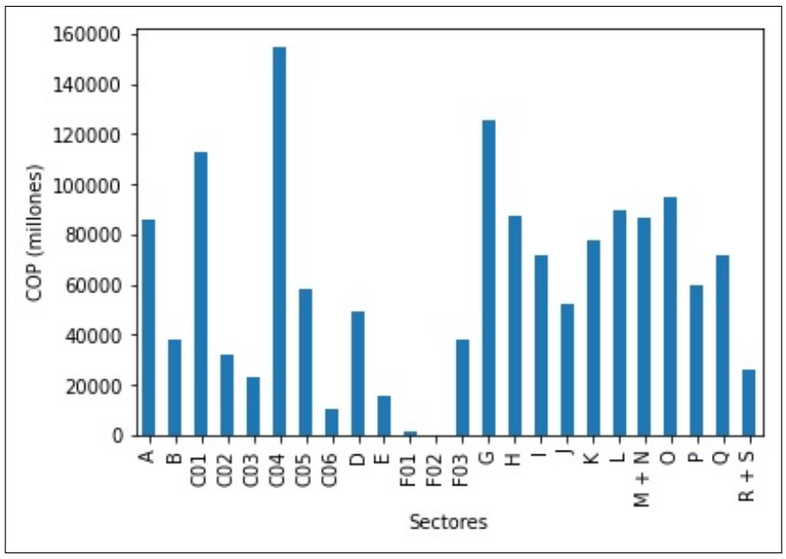
```
d=pd.DataFrame(d)
d.to_csv("d.csv", decimal=",", index=False)
```

4. Reporte el vector de equilibrio x. Use estadísticas descriptivas.

Código

```
x=np.matmul(L, d)
x["Sectores"]=cols
x.set_index("Sectores", inplace=True)
x.rename(columns="Gasto de Consumo Final":"x", inplace=True)
x.plot.bar(y="x", legend=None)
plt.xlabel("Sectores")
plt.ylabel("COP (millones)")
plt.savefig("x.jpg", bbox_inches = "tight")
x.describe()
```

Figura 5. Vector de equilibrio.



La Figura 5 muestra el vector de equilibrio que soluciona el sistema de ecuaciones del modelo de Leontief el cual consiste en una igualdad matricial con el fin de determinar las interrelaciones entre los diferentes sectores de una región concreta. Este supone que la economía está formada por diversos sectores de producción y servicios, entre los cuales existe una demanda interna a satisfacer, a la vez que una demanda externa (la cual se desprecia en este ejercicio). La ecuación matricial inicial es $x = Ax + d$ donde A es la matríz de coeficientes técnicos construida en el literal 3.1. y d el vector de demanda dado por la variable `Gasto de Consumo Final` que proviene de la base de datos. Luego, se aísla el vector de producción (no negativo, bajo el supuesto de que cada sector genera valor agregado) para poder resolver el sistema de manera que: $Ix - Ax = d$; $(I - A)x = d$; $x = (I - A)^{-1}d$ con $(I - A)^{-1}$ siendo la matriz inversa de Leontief derivada del literal 3.2.

Tabla 3. Estadísticas descriptivas del vector de equilibrio.

VARIABLE	Media	Desviación	Mínimo	Mediana	Máximo
x	60964,54	39940,16	201,76	59078,09	154659,66

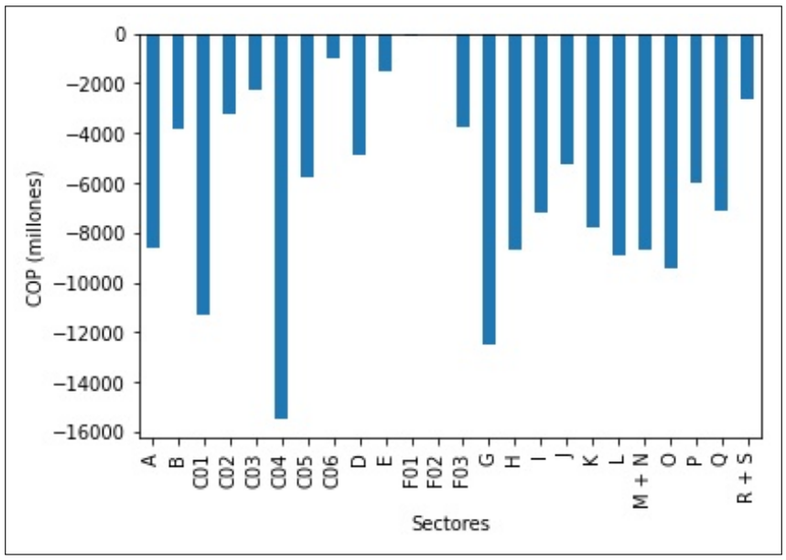
La Tabla 1 reporta las estadísticas descriptivas del vector de equilibrio (u oferta) el cual indica las cantidades necesarias que debe producir cada sector de la economía para cubrir la demanda interna. En promedio, del sector A - R+S es necesario aumentar en COP $60.964 millones la producción; el sector F02[1] constituye el mínimo con COP $201,7 millones mientras que el sector C04[2] es quien más oferta pendiente posee con COP $154.659 millones.

5. Para un choque de demanda dado por $\Delta d = -0,1 \times d$, encuentre Δx. Use estadísticas descriptivas.

Código

```
delta_d=-0.1*d delta_x=np.matmul(L, delta_d)
delta_x["Sectores"]=cols
delta_x.set_index("Sectores", inplace=True)
delta_x.rename(columns="Gasto de Consumo Final":"delta_x",
inplace=True)
delta_x.plot.bar(y="delta_x", legend=None)
plt.xlabel("Sectores")
plt.ylabel("COP (millones)")
plt.savefig("delta_x.jpg", bbox_inches = "tight")
delta_x.describe()
```

Figura 6. Cambio en el vector de equilibrio.



La Figura 6 muestra el cambio en el vector de equilibrio que soluciona el sistema de ecuaciones luego de un choque exógeno negativo sobre el vector de demanda.

Tabla 4. Estadísticas descriptivas del cambio en el vector de equilibrio.

VARIABLE	Media	Desviación	Mínimo	Mediana	Máximo
Δx	-6096,45	3994,01	-15465,96	-5907,81	-20,17

La Tabla 2 muestra las estadísticas descriptivas del cambio en el vector de equilibrio luego de un choque exógeno negativo sobre el vector de demanda. Se define a Δx como el cambio en magnitud simple del periodo t_1 (*ex-post* al choque) respecto a t_0 (*ex-ante* al choque); es decir, $\Delta x = x_1 - x_0$. Esto es matemáticamente equivalente a calcular $\Delta x = L\Delta d$

6. Comente el resultado anterior.

Una vez la economía percibe que un 10% menos de insumos se están demandando, ajustará la producción para que se equilibre con el nuevo vector de demanda. En particular, la Figura 6 y la Tabla 2 reportan el comportamiento numérico de este cambio. Note que del sector A - R+S es necesario disminuir, en promedio, en COP $6.096 millones la producción. Tal cifra equivale a una pérdida de ≈ COP $146.315 millones de valor agregado en conjunto.

[1] Construcción de carreteras y vías de ferrocarril, de proyectos de servicio público y de otras obras de ingeniería civil.

[2] Coquización, fabricación de productos de la refinación del petróleo y actividad de mezcla de combustibles; fabricación de sustancias y productos químicos; fabricación de productos farmacéuticos, sustancias químicas medicinales y productos botánicos de uso farmacéutico; fabricación de productos de caucho y de plástico; fabricación de otros productos minerales no metálicos.

En la misma línea del literal 3.5., tanto el sector F02 y C04 son quienes menos y más deben dejar de producir, respectivamente. El primero con una caída de tan solo COP $\$20,17$ millones, pero el segundo con COP -$15.465 millones. Fíjese que estos resultados son simétricos respecto a la Tabla 1 debido que el choque impacta de manera homogénea a todos los sectores y no hay efectos diferenciales (la matriz A es invariante).

A partir del gráfico es posible inferir que en respuesta a choques de demanda, los sectores que más se verían afectados son: i) el sector petrolero e industria de hidrocarburos; ii) sector comercial (mayorista y al detal), y; iii) sector de alimentos y bebidas. Análogamente, el sector F02, F01[3] y D[4] verán reducidas sus facturas en la menor cuantía de toda la economía. Estos hallazgos son consistentes con la economía colombiana ya que es un país que depende fuertemente (12-15% de los ingresos) de la exportación de crudo, frutas, verduras, textiles y plásticos de un (1) solo uso, los cuales son productos de los sectores con mayor pérdida en cuestión. Sin embargo, parece contraintuitivo que la vivienda, los servicios públicos y las carreteras no presenten mayor afectación, la razón subyacente de esto es que gran parte de las transacciones que se gestan dentro de estos sectores son respaldadas por el Estado (e.g. vivienda de interés social, subsidio de energía y luz, vías 4G y malla vial, entre otras).

[3]Construcción de edificaciones residenciales y no residenciales.
[4]Suministro de electricidad, gas, vapor y aire acondicionado.