

exposicion

July 1, 2024

0.1 Cargamos los imports.

```
[1]: import init
import warnings
warnings.filterwarnings('ignore')

[2]: import grafo as grafo
import experimento as exp
import modelo as modelo
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
import quantecon_book_networks
import quantecon_book_networks.input_output as qbn_io
import quantecon_book_networks.plotting as qbn_plt
import quantecon_book_networks.data as qbn_data
```

0.2 Estudio de la propagación y dinámica de precios a partir de la Matriz de Insumo-Producto (MIP)

Grupo: 6

Integrantes: Miguel De Lillo, Manuel Fernandez, Augusto Kielbowicz, Mariano Oca

[Repositorio](#)

1 Motivación

La idea de este modelo es estudiar, ante un shock de precios en un sector dado, cómo afectan las dinámicas propuestas entre los sectores productivos de la Argentina en la inflación global calculada a partir de las variaciones en los precios de los mismos. (Utilizando las relaciones dadas por la matriz de Insumo-Producto).

TODO: Agregar imagenes acá

- Introducción
 - Preguntas a responder
 - Suposiciones (y limitaciones) del modelo
- A

- B
- C
- Conclusiones
- Apéndices
 - Código

2 Introducción

2.1 Preguntas a Responder

- ¿Cómo se propaga el aumento de precios a través de la red definida por la MIP?
- ¿Cómo impacta el aumento de precio en un producto/sector sobre otros productos/sectores?
¿Se mantiene en la misma cadena productiva?
- ¿Cómo influyen la dinámica de comportamiento de los agentes en la variación de la inflación global?
- ¿Cuál es la sensibilidad del sistema respecto a variaciones de precio en nodos específicos?
¿Cuáles son los nodos que propagan de mayor forma la variación de precios?

2.2 Descripción del Modelo

- Cada sector productivo de la Argentina es representado por un nodo del digrafo. El mismo representaría a todos los productos (y productores) del sector.
- Cada arista del digrafo representa la relación “le vende a” donde el peso de la arista es el porcentaje de la producción total que es comparada.
- Tanto los sectores productivos como los pesos que le corresponden a cada arista son extraídos de la Matriz Inzumo Producto del 1997 publicada por el INDEC. Los valores de dicha matriz son normalizados para que representen porcentajes, afines a la experimentación que se quiere realizar sobre el modelo.

2.3 Suposiciones (y limitaciones) del Modelo

- Un cambio de precios >0 en los insumos provoca un cambio de precios saliente del agente (producción). Sólo vamos a estudiar variaciones positivas en los precios para atenernos a las preguntas a investigar con el modelo.
- La economía es cerrada. Esto es, no se traen productos de otros países (importaciones) ni se vende nada a ellos (exportaciones) durante la evolución del sistema.
- Rige la Ley de Say: la oferta es igual a la demanda.

3 Matriz Insumo Producto, Grafo y Análisis

Cargamos la matriz insumo producto normalizada por demanda saliente.

```
[ ]: mip = pd.read_csv('../resources/MIP_normalizada.csv', index_col=0)
mip.size()
```

```
[ ]: mip.head()
```

[]:	Cultivo de cereales,
oleaginosas y forrajeras \	
Cultivo de cereales, oleaginosas y forrajeras	
0.010319	
Cultivo de hortalizas, legumbres, flores y plan...	
0.000000	
Cultivo de frutas y nueces	
0.000000	
Cultivos industriales	
0.000000	
Producción de semillas	
0.783146	
	Cultivo de hortalizas,
legumbres, flores y plantas ornamentales \	
Cultivo de cereales, oleaginosas y forrajeras	
0.000000	
Cultivo de hortalizas, legumbres, flores y plan...	
0.167226	
Cultivo de frutas y nueces	
0.000000	
Cultivos industriales	
0.000000	
Producción de semillas	
0.016374	
	Cultivo de frutas y nueces
\	
Cultivo de cereales, oleaginosas y forrajeras	0.001542
Cultivo de hortalizas, legumbres, flores y plan...	0.009565
Cultivo de frutas y nueces	0.000000
Cultivos industriales	0.000000
Producción de semillas	0.029820
	Cultivos industriales \
Cultivo de cereales, oleaginosas y forrajeras	0.001601
Cultivo de hortalizas, legumbres, flores y plan...	0.000000
Cultivo de frutas y nueces	0.000000
Cultivos industriales	0.020212
Producción de semillas	0.017254
	Producción de semillas \
Cultivo de cereales, oleaginosas y forrajeras	0.000000
Cultivo de hortalizas, legumbres, flores y plan...	0.000000
Cultivo de frutas y nueces	0.000000
Cultivos industriales	0.000000
Producción de semillas	0.02174

Cría de ganado y producción

de leche, lana y pelos \
Cultivo de cereales, oleaginosas y forrajeras
0.181660
Cultivo de hortalizas, legumbres, flores y plan...
0.000000
Cultivo de frutas y nueces
0.000000
Cultivos industriales
0.000000
Producción de semillas
0.127875

Producción de granja \

Cultivo de cereales, oleaginosas y forrajeras 0.012263
Cultivo de hortalizas, legumbres, flores y plan... 0.000000
Cultivo de frutas y nueces 0.000000
Cultivos industriales 0.000000
Producción de semillas 0.002291

Servicios agropecuarios \

Cultivo de cereales, oleaginosas y forrajeras 0.0
Cultivo de hortalizas, legumbres, flores y plan... 0.0
Cultivo de frutas y nueces 0.0
Cultivos industriales 0.0
Producción de semillas 0.0

Caza \

Cultivo de cereales, oleaginosas y forrajeras 0.0
Cultivo de hortalizas, legumbres, flores y plan... 0.0
Cultivo de frutas y nueces 0.0
Cultivos industriales 0.0
Producción de semillas 0.0

Silvicultura y extracción de

madera \
Cultivo de cereales, oleaginosas y forrajeras 0.0
Cultivo de hortalizas, legumbres, flores y plan... 0.0
Cultivo de frutas y nueces 0.0
Cultivos industriales 0.0
Producción de semillas 0.0

	... Enseñanza pública \
Cultivo de cereales, oleaginosas y forrajeras	... 0.000038
Cultivo de hortalizas, legumbres, flores y plan...	... 0.005679
Cultivo de frutas y nueces	... 0.003827
Cultivos industriales	... 0.000000
Producción de semillas	... 0.000000
	Enseñanza privada \
Cultivo de cereales, oleaginosas y forrajeras	0.000080
Cultivo de hortalizas, legumbres, flores y plan...	0.012030
Cultivo de frutas y nueces	0.008099
Cultivos industriales	0.000000
Producción de semillas	0.000000
	Salud humana pública \
Cultivo de cereales, oleaginosas y forrajeras	0.000007
Cultivo de hortalizas, legumbres, flores y plan...	0.002192
Cultivo de frutas y nueces	0.011067
Cultivos industriales	0.000000
Producción de semillas	0.000000
	Salud humana privada \
Cultivo de cereales, oleaginosas y forrajeras	0.000042
Cultivo de hortalizas, legumbres, flores y plan...	0.008913
Cultivo de frutas y nueces	0.005904
Cultivos industriales	0.000000
Producción de semillas	0.000000
	Servicios veterinarios \
Cultivo de cereales, oleaginosas y forrajeras	0.000012
Cultivo de hortalizas, legumbres, flores y plan...	0.000000
Cultivo de frutas y nueces	0.000000
Cultivos industriales	0.000000
Producción de semillas	0.000000
	Servicios sociales \
Cultivo de cereales, oleaginosas y forrajeras	0.000000
Cultivo de hortalizas, legumbres, flores y plan...	0.001238
Cultivo de frutas y nueces	0.001638
Cultivos industriales	0.000000
Producción de semillas	0.000000
	Servicios de saneamiento \
Cultivo de cereales, oleaginosas y forrajeras	0.0
Cultivo de hortalizas, legumbres, flores y plan...	0.0
Cultivo de frutas y nueces	0.0

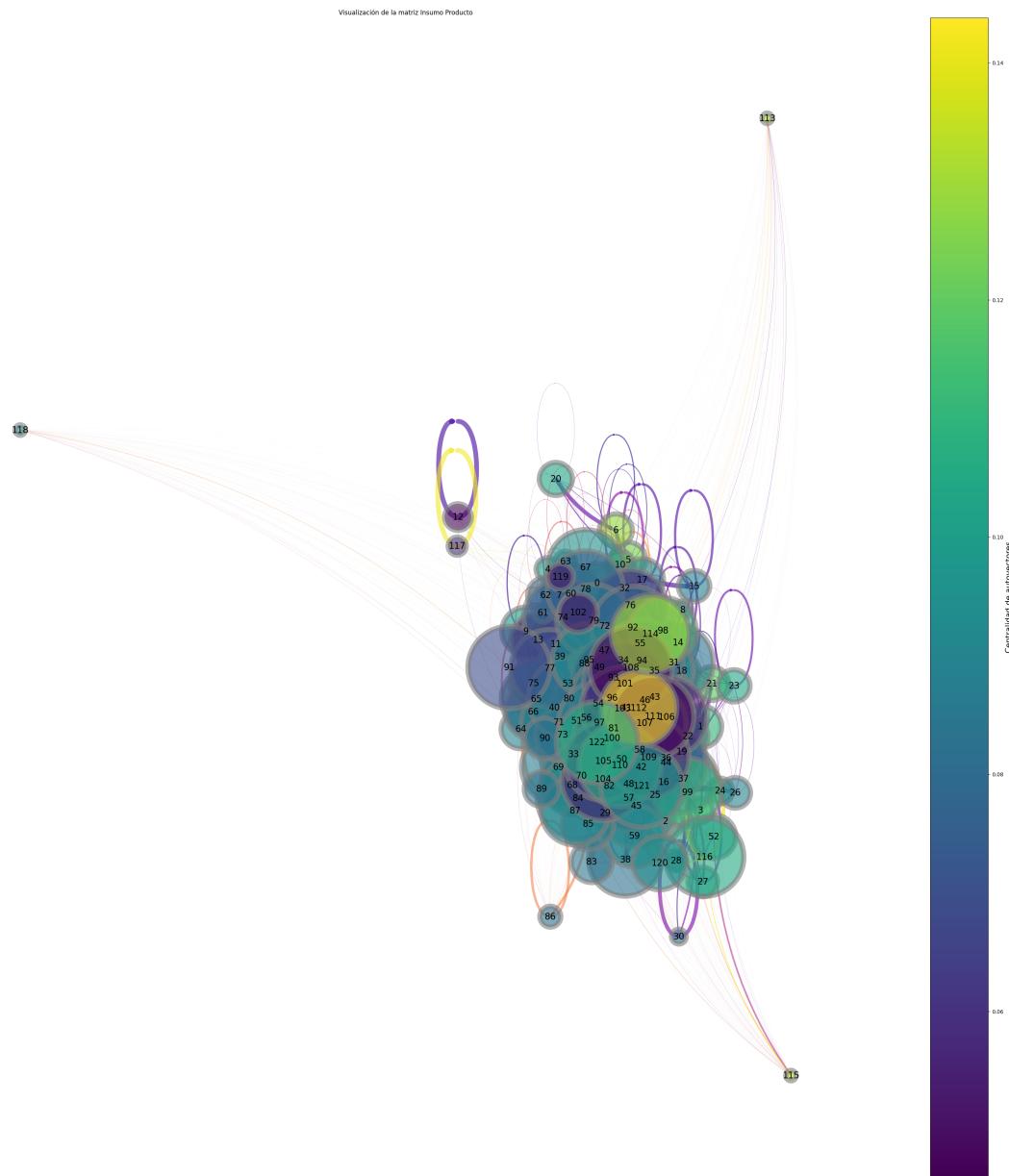
Cultivos industriales	0.0
Producción de semillas	0.0
	Actividad de asociaciones \
Cultivo de cereales, oleaginosas y forrajeras	0.000000
Cultivo de hortalizas, legumbres, flores y plan...	0.007999
Cultivo de frutas y nueces	0.000000
Cultivos industriales	0.000000
Producción de semillas	0.000000
	Servicios de cine, radio y televisión \
Cultivo de cereales, oleaginosas y forrajeras	
1.735054e-09	
Cultivo de hortalizas, legumbres, flores y plan...	
1.237709e-06	
Cultivo de frutas y nueces	
1.472092e-07	
Cultivos industriales	
0.000000e+00	
Producción de semillas	
0.000000e+00	
	Servicios personales, de reparación, actividades deportivas y de esparcimiento
Cultivo de cereales, oleaginosas y forrajeras	
0.003256	
Cultivo de hortalizas, legumbres, flores y plan...	
0.079292	
Cultivo de frutas y nueces	
0.000006	
Cultivos industriales	
0.000000	
Producción de semillas	
0.000000	

[5 rows x 123 columns]

3.1 Grafo

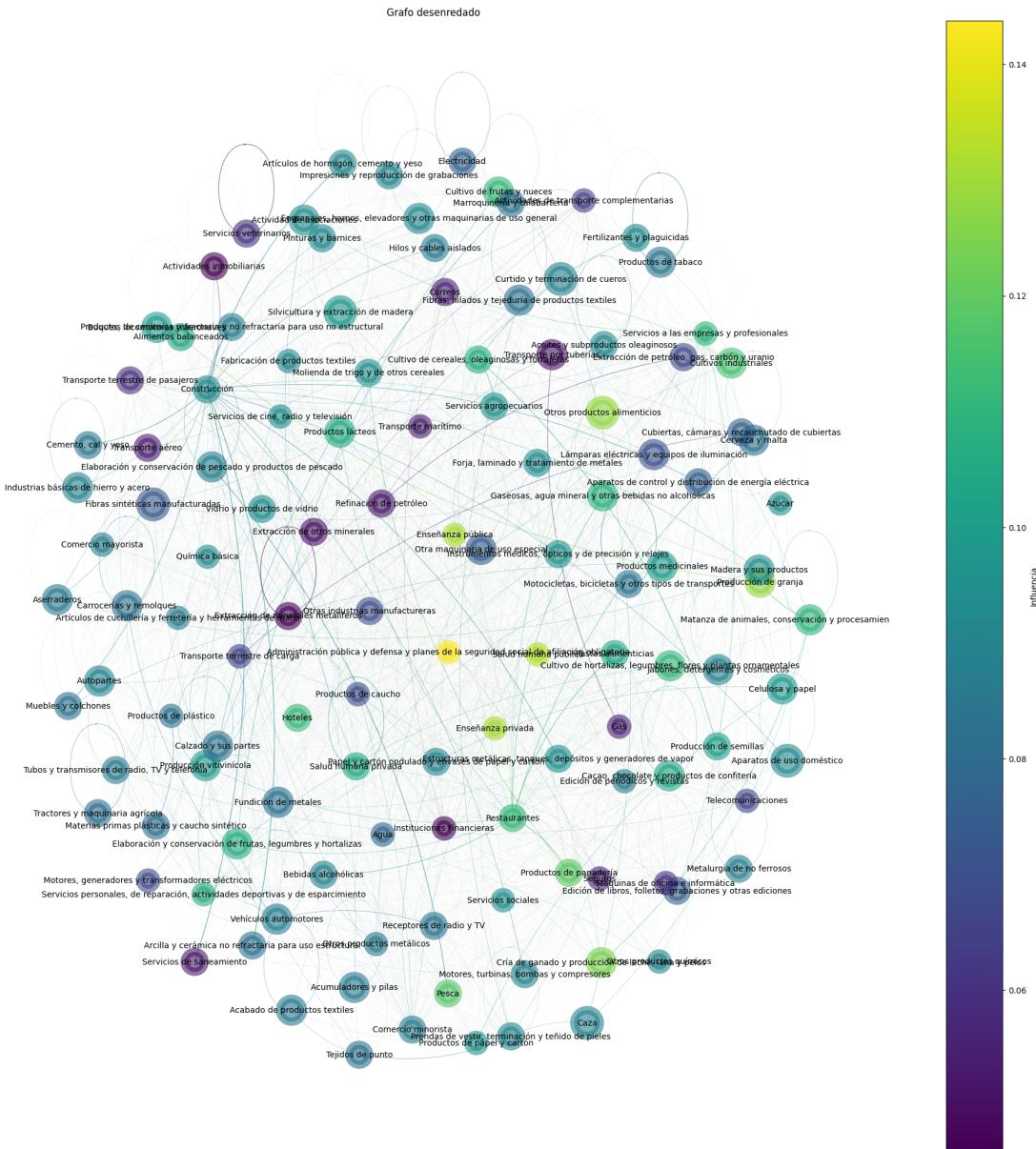
Veamos cómo se ve esta red.

```
[4]: fig_gr_centralizado = grafo.verGrafoCentralizado(mip.to_numpy())
```



Intentemos separar los sectores para poder tener una mirada amplia.

```
[5]: fig_gr_abierto = grafo.verGrafoAbierto(mip)
```



3.2 Distribución de grado

Veremos cómo se distribuye el grado en la red. Para tener una noción de qué está pasando.

```
[6]: G = nx.DiGraph(mip)
grados_in = G.in_degree
grados_out = G.out_degree
nodos= []
grado_salida = []
grado_entrada = []
j=0
```

```

for sector in mip.columns:
    nodos.append(j)
    j+=1
    grado_salida.append(grados_out[sector])
    grado_entrada.append(grados_in[sector])

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(15, 6))

axes[0].bar(nodos, grado_salida, color ='blue')
axes[0].set_xlabel('Sectores Productivos (Índice)')
axes[0].set_ylabel('Grado de salida')
axes[0].set_title('Distribución de grado de salida')
print(f'Media del Grado de salida:' + str(np.mean(grado_salida)))
print(f'Desvio del Grado de salida:' + str(np.var(grado_salida)))

axes[1].bar(nodos, grado_entrada, color ='red')
axes[1].set_xlabel('Sectores Productivos (Índice)')
axes[1].set_ylabel('Grado de entrada')
axes[1].set_title('Distribución de grado de entrada')

print(f'Media del Grado de entrada:' + str(np.mean(grado_entrada)))
print(f'Desvio del Grado de entrada:' + str(np.var(grado_entrada)))
plt.show()
plt.close()

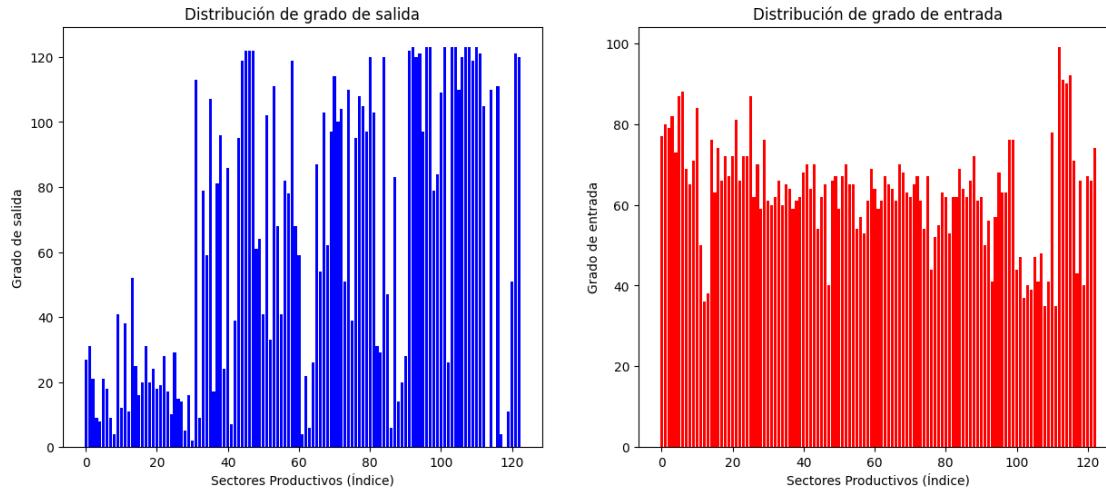
```

Media del Grado de salida:63.31707317073171

Desvío del Grado de salida:1961.1108467182228

Media del Grado de entrada:63.31707317073171

Desvío del Grado de entrada:162.3791394011501



3.3 Centralidad de autovectores

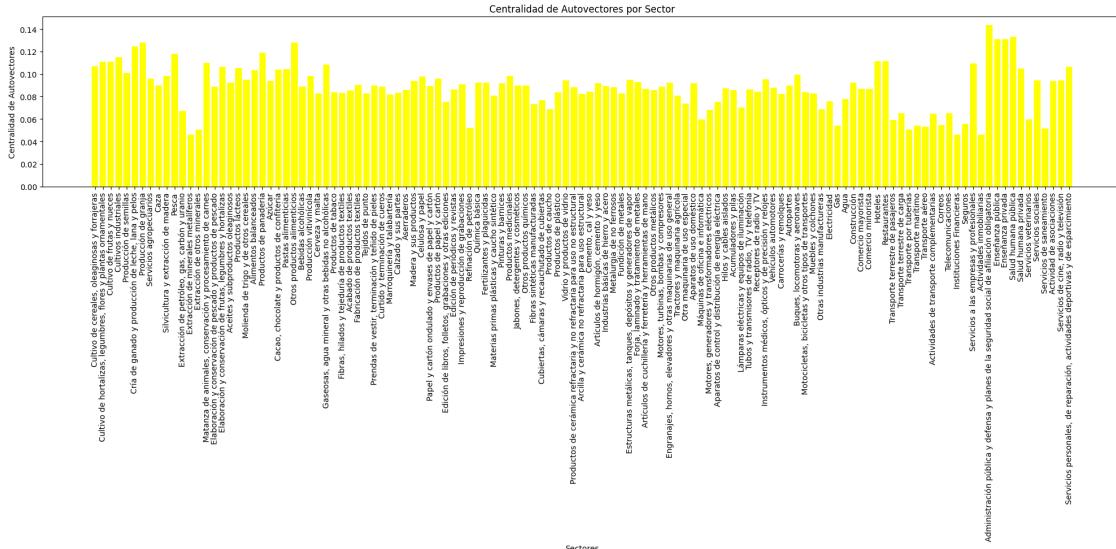
La centralidad de autovalores mide la influencia de un nodo sobre la red basado en la influencia de sus vecinos. Elegimos esta métrica para analizar el grafo pues los nodos que poseen un valor alto de esta medida de centralidad están conectados a otros nodos que a su vez son muy relevantes, en el sentido de la misma medida, o bien a muchos otros nodos, quizás menos relevantes. Por el contrario, los nodos conectados a otros nodos periféricos o poco relevantes, tendrán una baja centralidad de autovectores.

```
[7]: centrality = nx.eigenvector_centrality(G, max_iter=1000)
values = list(centrality.values())

centralidad = []
nodos_nombre = []
j = 0
for sector in mip.columns:
    nodos_nombre.append(sector)
    centralidad.append(values[j])
    j +=1

plt.figure(figsize=(20, 10)) # Ancho 20, Alto 10
plt.bar(nodos_nombre, centralidad, color='yellow')
plt.xticks(rotation=90)
plt.title('Centralidad de Autovectores por Sector')
plt.xlabel('Sectores')
plt.ylabel('Centralidad de Autovectores')
plt.tight_layout()

plt.show()
```



4 Los experimentos

En esta etapa, haremos un análisis de la divergencia y convergencia de la inflación. Para 4 valores de alfa distintos, vamos a hacer un shock inflacionario en cada uno de los 123 sectores productivos y computaremos la media móvil de la inflación entre estas 123 simulaciones para cada alfa. Además, mostraremos el grafo resultante luego de la evolución del sistema para dos sectores representativos de niveles de centralidad alto y bajo, respectivamente.

```
[8]: def plot_inflaciones(inflaciones, aumento, umbral_label=0.5):
    fig, (ax,ax2) = plt.subplots(1,2,figsize=(22, 10))
    ax.set_title(f"Valor de la inflación en el tiempo, a partir de un shock de {aumento}% \n para todos los sectores.")
    ax.set_xlabel("Tiempo")
    ax.set_ylabel("Porcentaje de inflación")
    ax2.set_title(f"Variación de la inflación en el tiempo, a partir de un shock de {aumento}%, \n para todos los sectores.")
    ax2.set_xlabel("Tiempo")
    ax2.set_ylabel("Variación en el porcentaje de inflación")
    for i, inflacion in enumerate(inflaciones):
        tiempo = range(0,len(inflacion))
        if (max(inflacion) > umbral_label):
            ax.plot(tiempo, inflacion, label=sectores[i])
            ax2.plot(tiempo[1:],np.diff(inflacion), label=sectores[i])
        else:
            ax.plot(tiempo, inflacion)
            ax2.plot(tiempo[1:],np.diff(inflacion))
    ax.legend()
    ax2.legend()
    plt.show()
```

```
[9]: def verEvolucion(df, alfa):
    df = df.T
    df_diff = df.diff().dropna()
    window_size = 5 # Tamaño de la ventana para la media móvil
    df['mean'] = df.mean(axis=1).rolling(window=window_size).mean()
    df['max'] = df.max(axis=1)
    df['min'] = df.min(axis=1)
    df_diff['mean_diff'] = df_diff.mean(axis=1).rolling(window=window_size).mean()
    df_diff['max_diff'] = df_diff.max(axis=1)
    df_diff['min_diff'] = df_diff.min(axis=1)

    fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(15, 6))
    a = len(df)//10

    # Gráfico para datos absolutos
```

```

        axes[0].fill_between(df.index, df['min'], df['max'], color='red', alpha=0.3, label='Rango (Mín-Máx)')
        axes[0].plot(df.index, df['mean'], color='red', linewidth=2, label='Media Móvil')
        axes[0].set_title(f'Inflación Absoluta en 123 simulaciones para {alfa}')
        axes[0].set_xlabel('Tiempo en cantidad de iteraciones')
        axes[0].set_ylabel('Inflación absoluta')
        axes[0].set_xticks(df.index[::a])
        axes[0].legend()

# Gráfico para diferencias de inflación
        axes[1].fill_between(df_diff.index, df_diff['min_diff'], df_diff['max_diff'], color='red', alpha=0.3, label='Rango (Mín-Máx)')
        axes[1].plot(df_diff.index, df_diff['mean_diff'], color='red', linewidth=2, label='Media Móvil')
        axes[1].set_title(f'Variación de inflación en 123 simulaciones para {alfa}')
        axes[1].set_xlabel('Tiempo en cantidad de iteraciones')
        axes[1].set_ylabel('Variación de Inflación')
        axes[1].set_xticks(df_diff.index[::a])
        axes[1].legend()

# Ajustar el layout
plt.tight_layout()

# Mostrar los gráficos
plt.show()

```

[10]:

```

alfa0= pd.read_csv('../resources/alpha00.csv', index_col= 0)
alfa02= pd.read_csv('../resources/alpha02.csv', index_col= 0)
alfa03 = pd.read_csv('../resources/alpha03.csv', index_col= 0)
alfa04= pd.read_csv('../resources/alpha04.csv', index_col= 0)

```

4.1 Experimentación con la dinámica 1.

Veamos qué sucede con un sector muy central y otro poco central.

[11]:

```

sector = 'Administración pública y defensa y planes de la seguridad social de afiliación obligatoria'
mip_grafo = grafo.armar_grafo(mip, precios_random=False)
experimento = exp.Experimento(mip_grafo,
                               dinamica=modelo.dinamica_1,
                               calcular_inflacion=modelo.calcular_inflacion,
                               duracion_periodo=20)
experimento.shock(sector,20)
experimento.step(600)

```

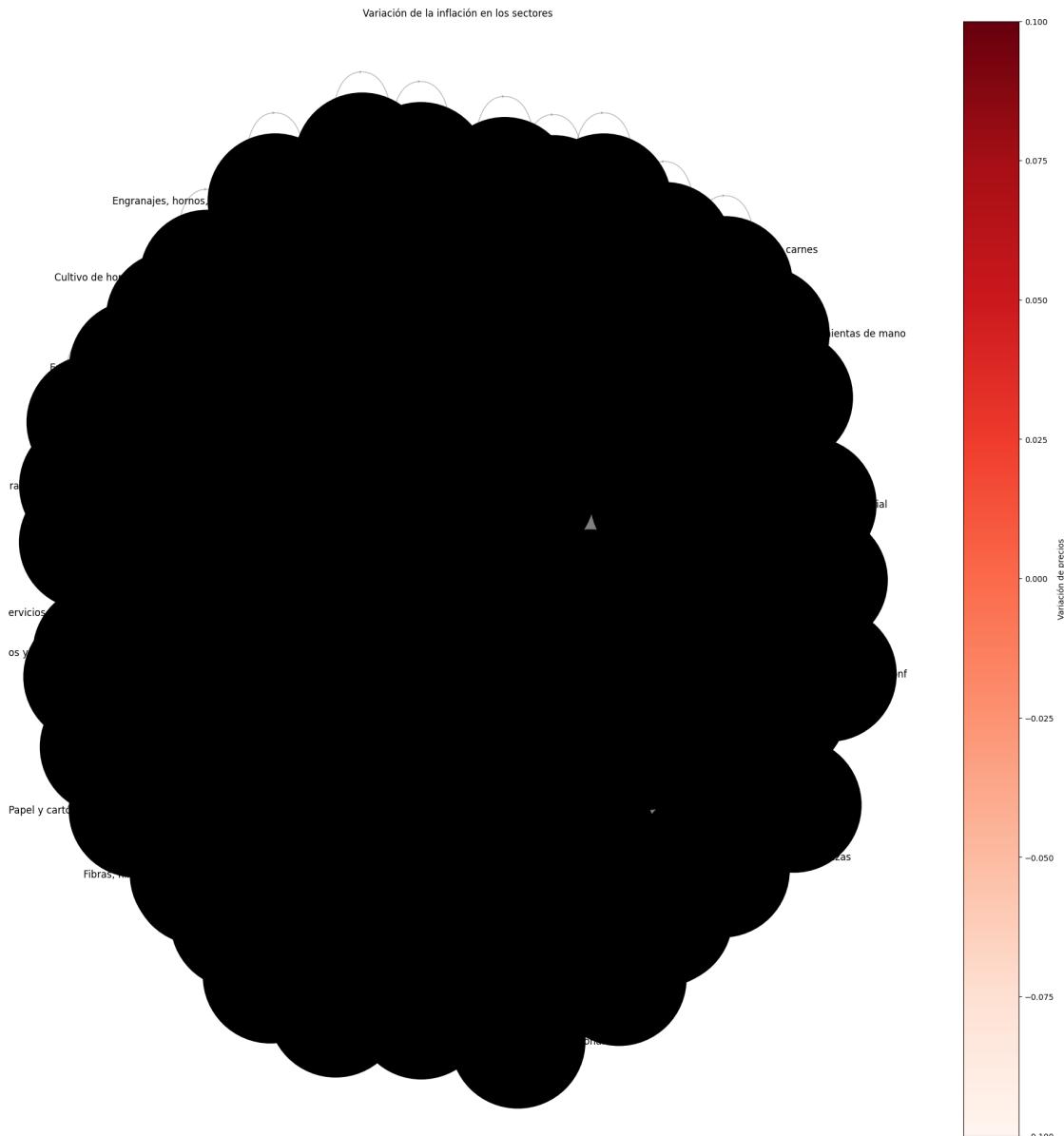
```

precios_finales = []
for sector in mip_grafo.nodes():
    precio = mip_grafo.nodes()[sector]['precio']
    precios_finales.append(precio)

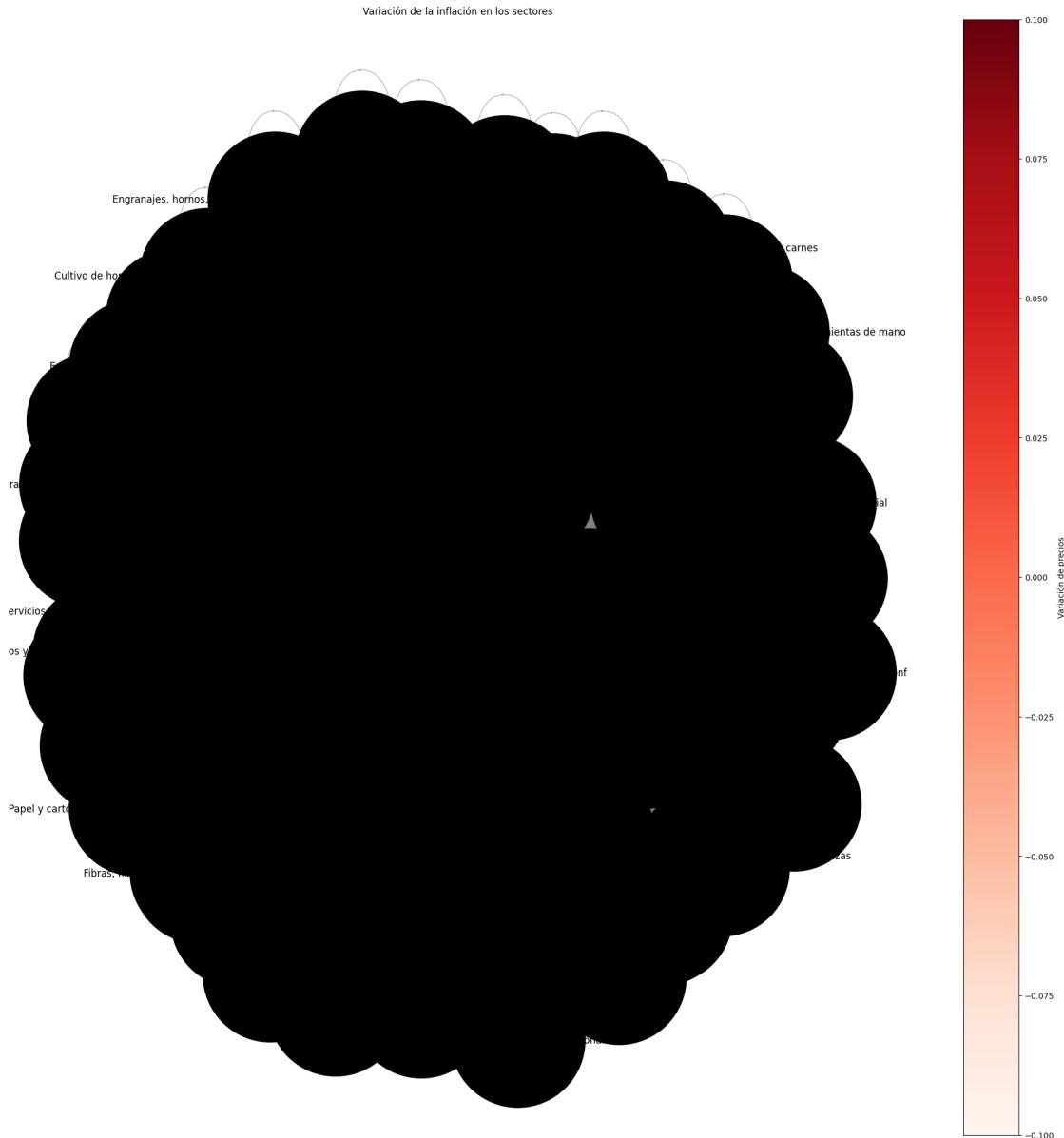
precios_iniciales = [100 for _ in precios_finales]

```

[12]: `grafo.verInflacion(mip, precios_finales, precios_iniciales)`



[12] :

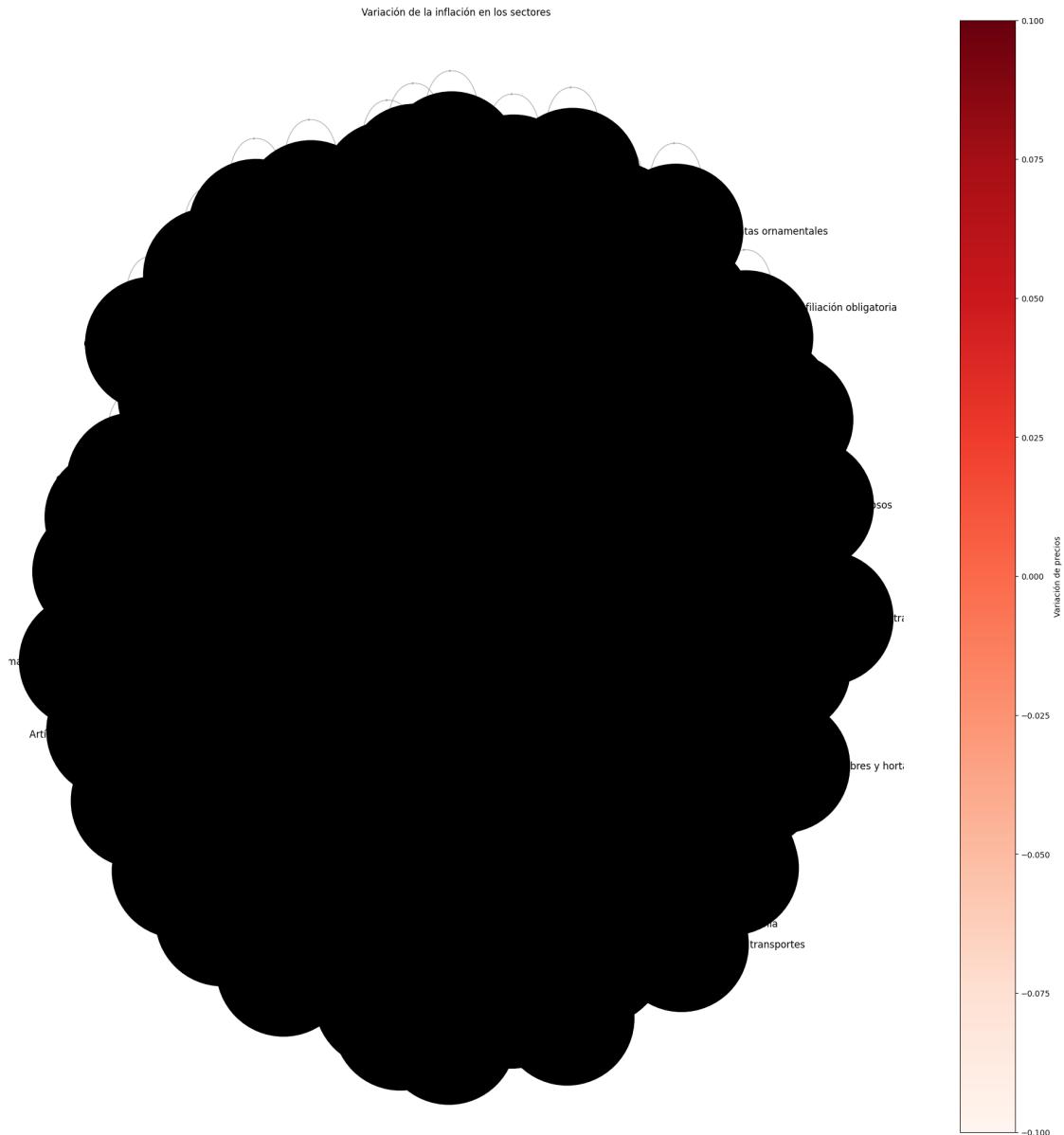


```
[13]: sector = ('Instituciones Financieras')
mip_grafo = grafo.armar_grafo(mip, precios_random=False)
experimento = exp.Experimento(mip_grafo,
                               dinamica=modelo.dinamica_1,
                               calcular_inflacion=modelo.calcular_inflacion,
                               duracion_periodo=20)
experimento.shock(sector,20)
experimento.step(600)
precios_finales = []
for sector in mip_grafo.nodes():
```

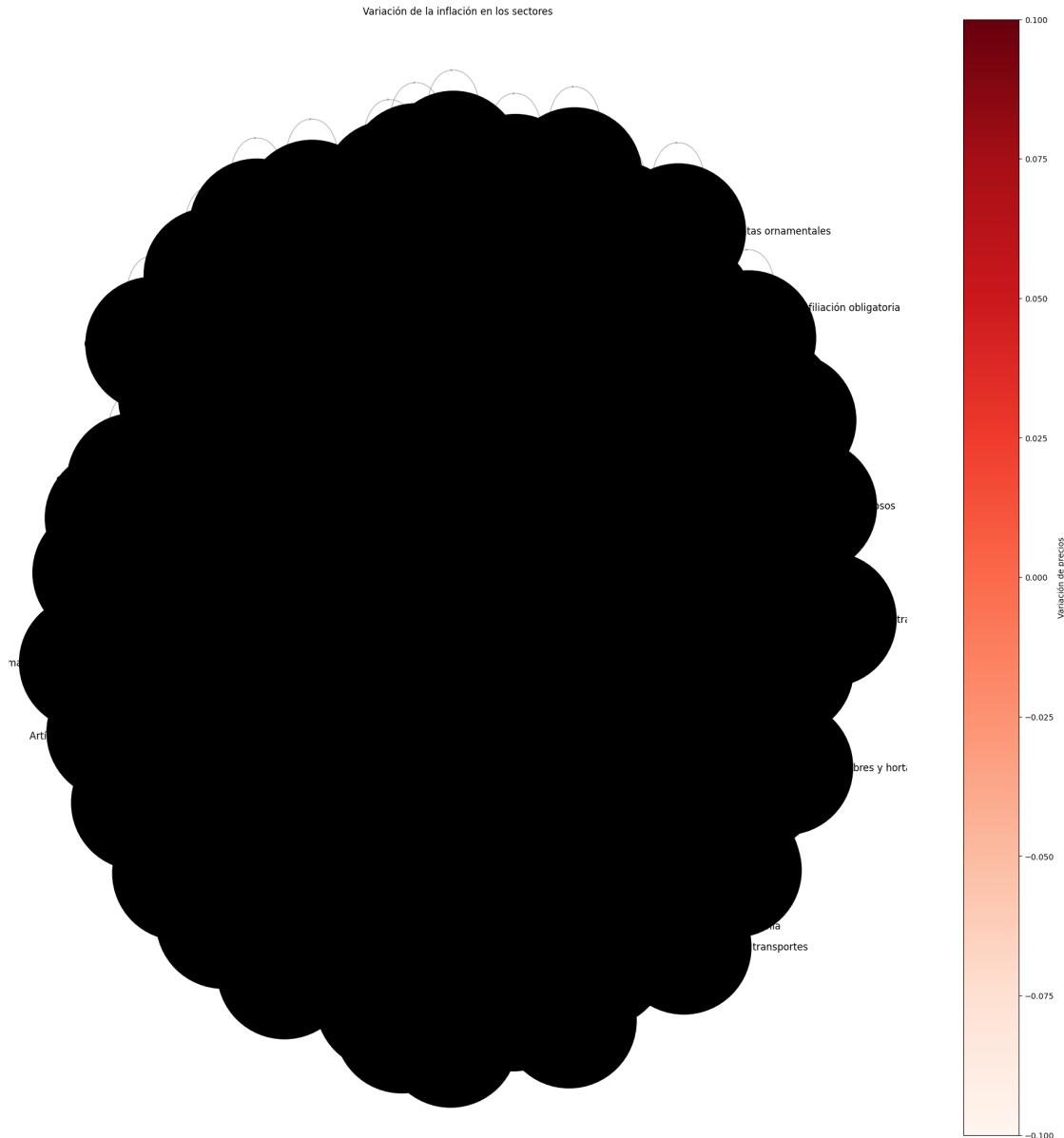
```
    precio = mip_grafo.nodes()[sector]['precio']
    precios_finales.append(precio)

precios_iniciales = [100 for _ in precios_finales]
```

```
[14]: grafo.verInflacion(mip, precios_finales, precios_iniciales)
```



[14] :

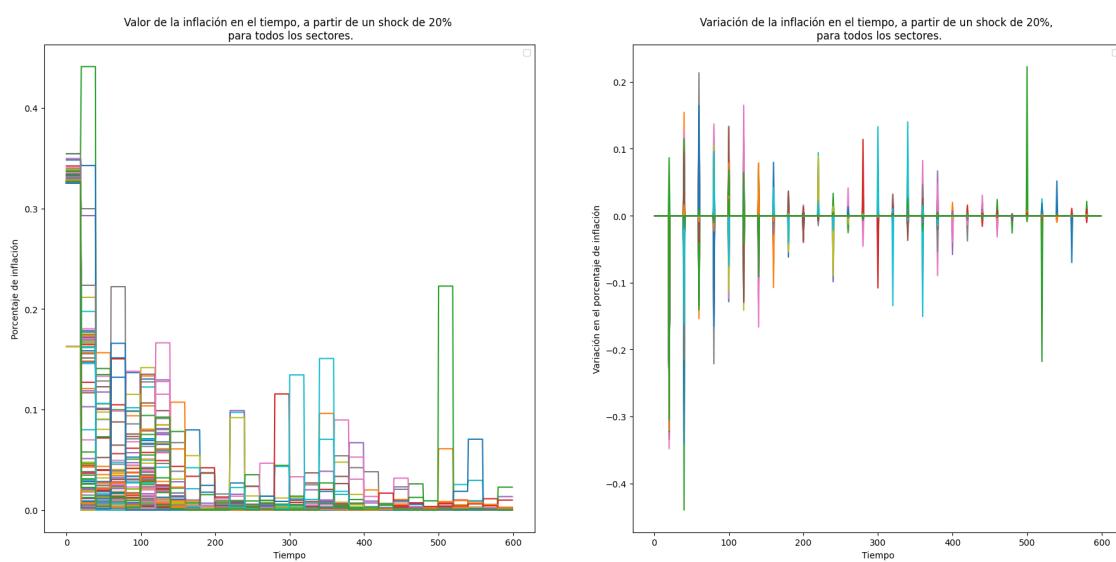
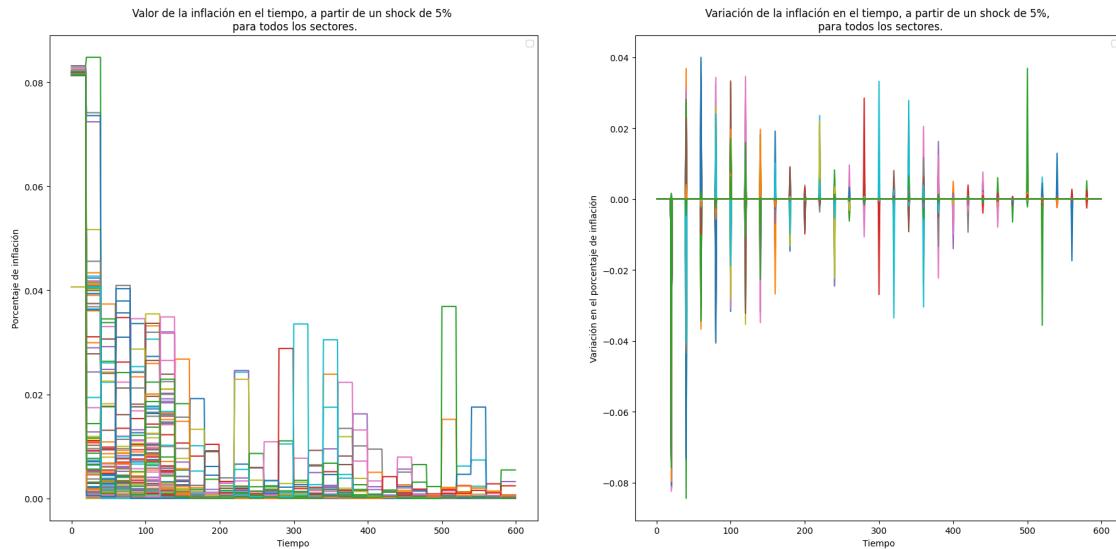


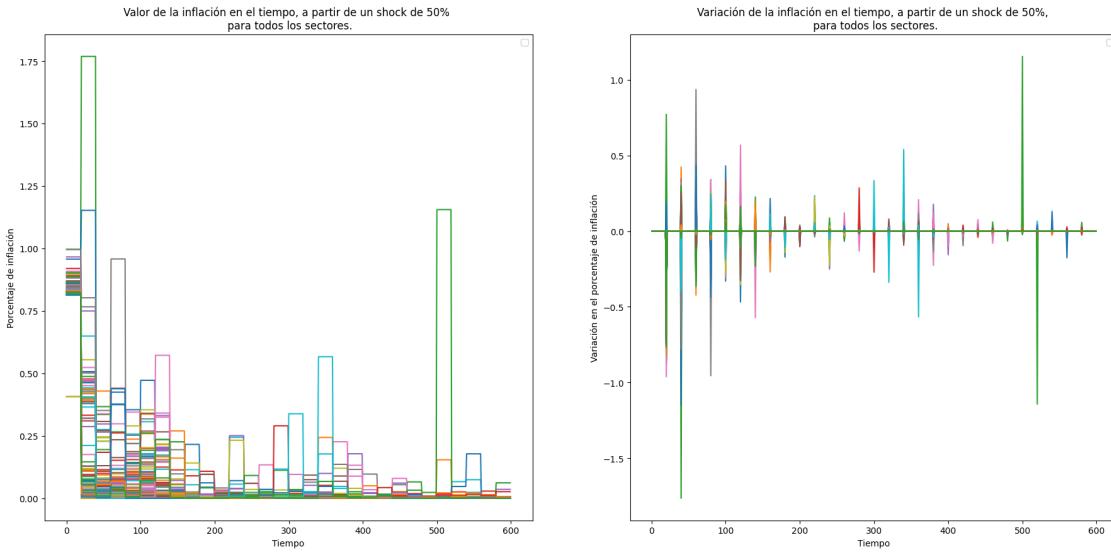
```
[15]: aumentos = [5,20,50]
for aumento in aumentos:
    mip_grafo = grafo.armar_grafo(mip,precios_random=False)
    sectores = list(mip_grafo.nodes)
    inflaciones = []
    for sector in sectores:
        experimento = exp.Experimento(mip_grafo,
                                         dinamica=modelo.dinamica_1,
                                         calcular_inflacion=modelo.calcular_inflacion,
                                         duracion_periodo=20)
```

```

experimento.shock(sector,aumento)
experimento.step(600)
inflacion = experimento.metricas_evaluadas['inflacion']
inflaciones.append(inflacion)
plot_inflaciones(inflaciones,aumento,3)

```



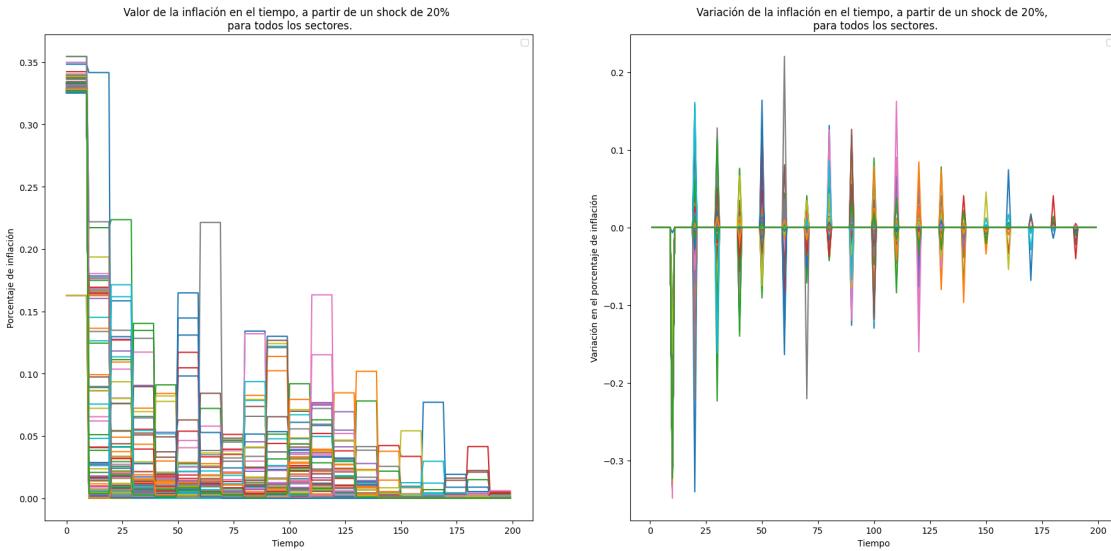


4.2 Experimentos con las dinámicas.

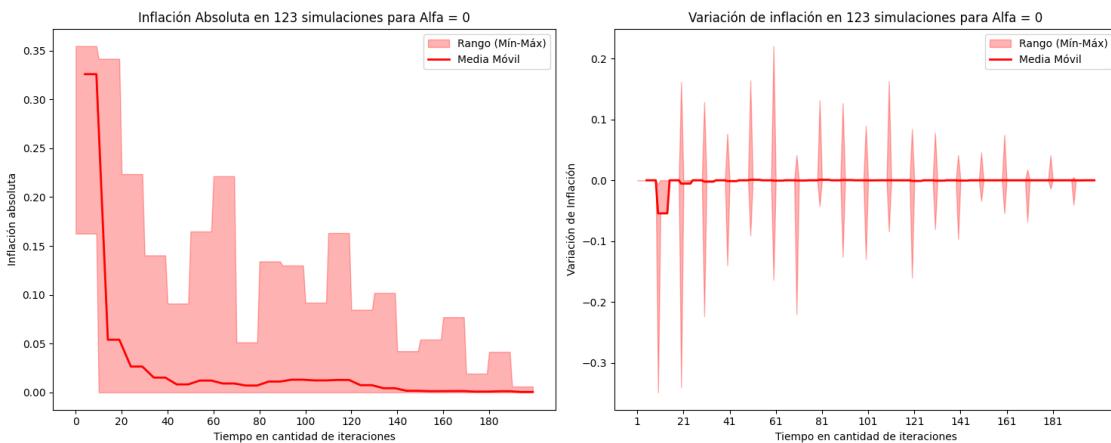
4.2.1 $\alpha = 0$

```
[16]: # shock en dos sectores representativos
```

```
[17]: aumentos = [20]
alpha = 0
mip_grafo = grafo.armar_grafo(mip, precios_random=False)
sectores = list(mip_grafo.nodes)
for j, aumento in enumerate(aumentos):
    inflaciones = []
    for sector in sectores:
        experimento = exp.Experimento(mip_grafo,
                                         dinamica=modelo.dinamica_3,
                                         calcular_inflacion=modelo.calcular_inflacion,
                                         alpha=alpha)
        experimento.shock(sector,aumento)
        experimento.step(200)
        inflacion = experimento.metricas_evaluadas['inflacion']
        inflaciones.append(inflacion)
plot_inflaciones(inflaciones, aumento)
```



```
[18]: verEvolucion(alfa0, 'Alfa = 0')
```



4.2.2 Alpha = 0.2

```
[19]: sector = ('Administración pública y defensa y planes de la seguridad social de la afiliación obligatoria')

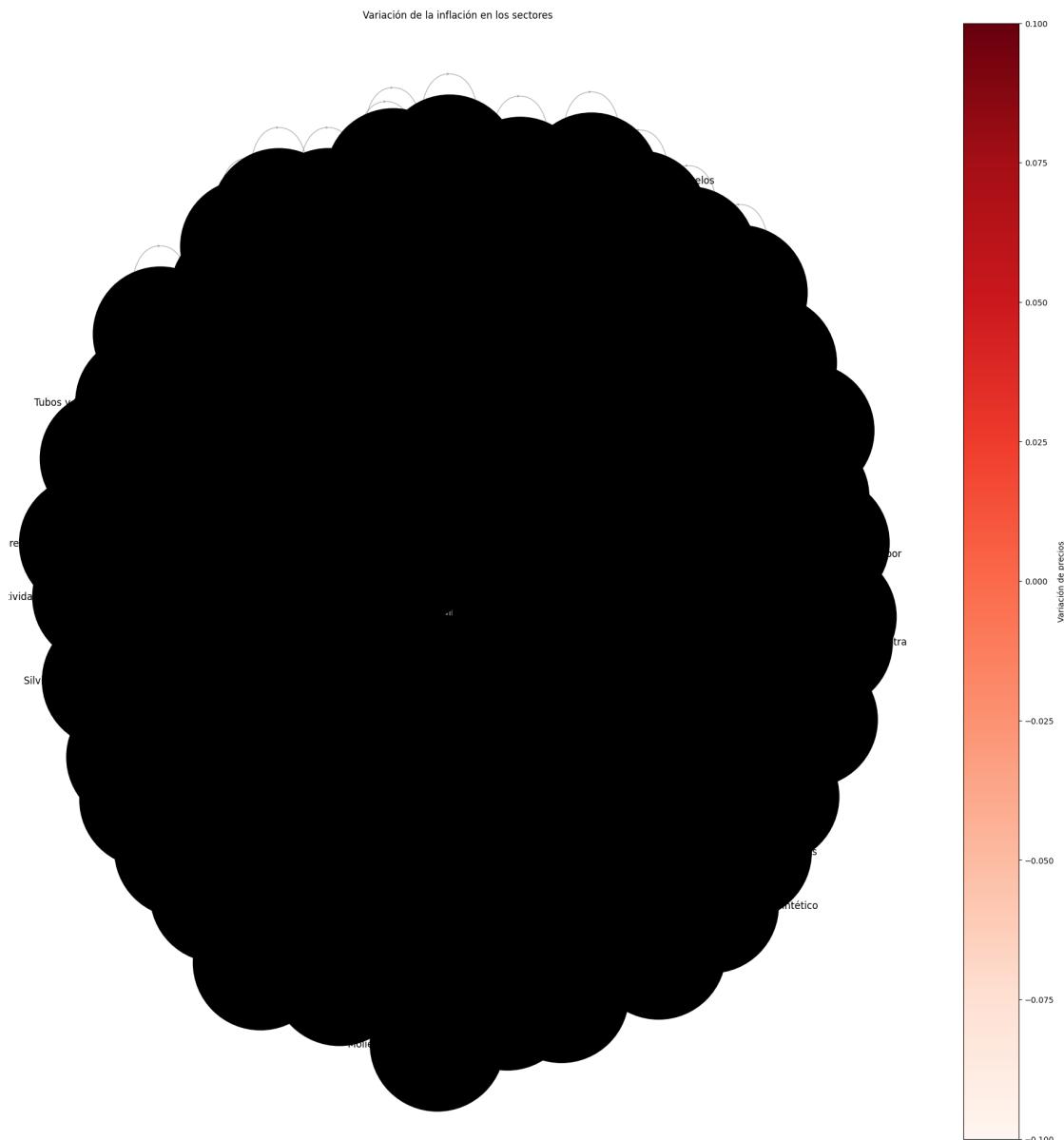
mip_grafo = grafo.armar_grafo(mip, precios_random=False)
experimento = exp.Experimento(mip_grafo,
                               dinamica=modelo.dinamica_3,
                               calcular_inflacion=modelo.calcular_inflacion,
                               duracion_periodo=20,
                               alpha=0.2)

experimento.shock(sector, 20)
```

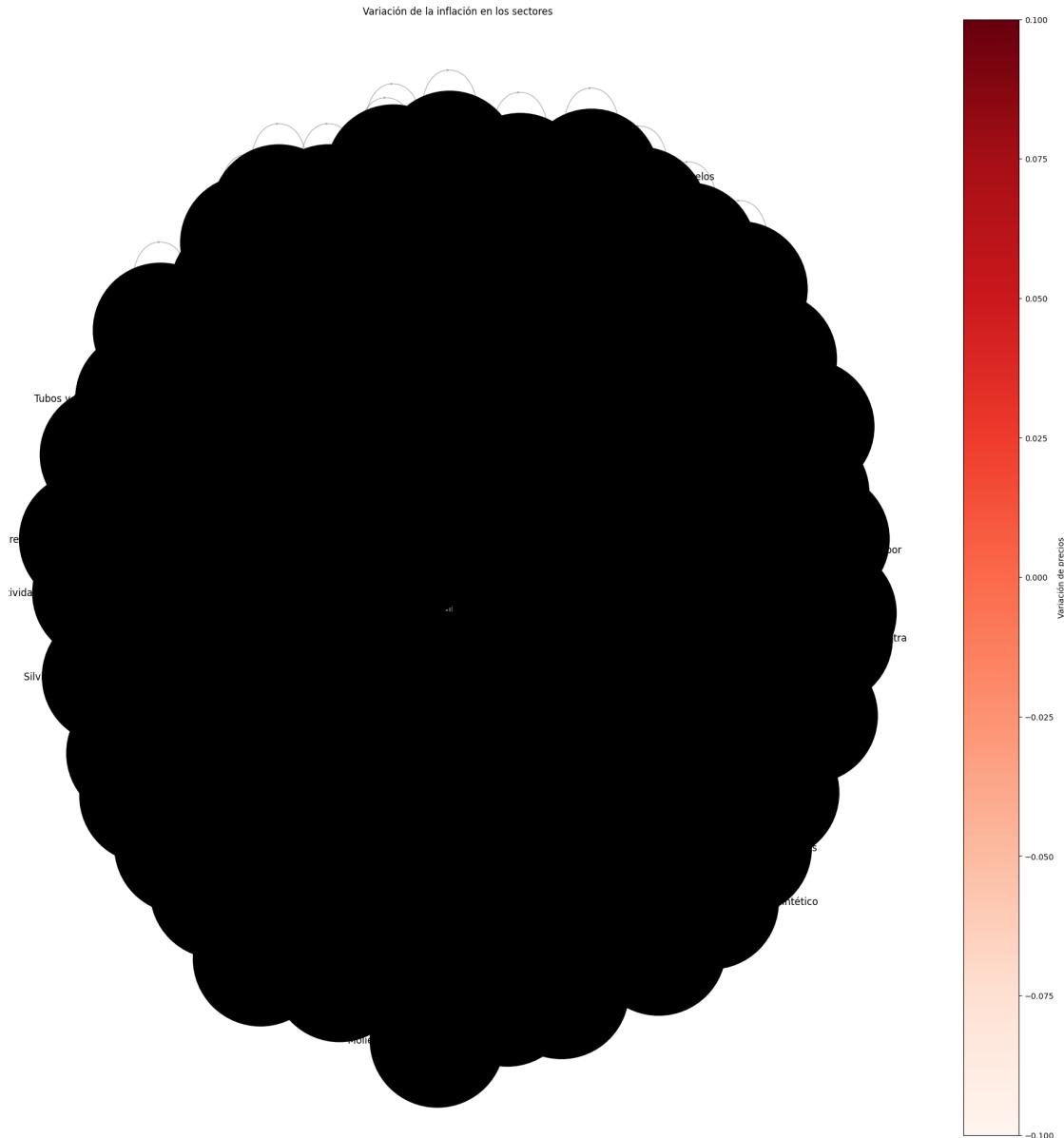
```
experimento.step(600)
precios_finales = []
for sector in mip_grafo.nodes():
    precio = mip_grafo.nodes()[sector]['precio']
    precios_finales.append(precio)

precios_iniciales = [100 for _ in precios_finales]
```

```
[20]: grafo.verInflacion(mip, precios_finales, precios_iniciales)
```



[20] :

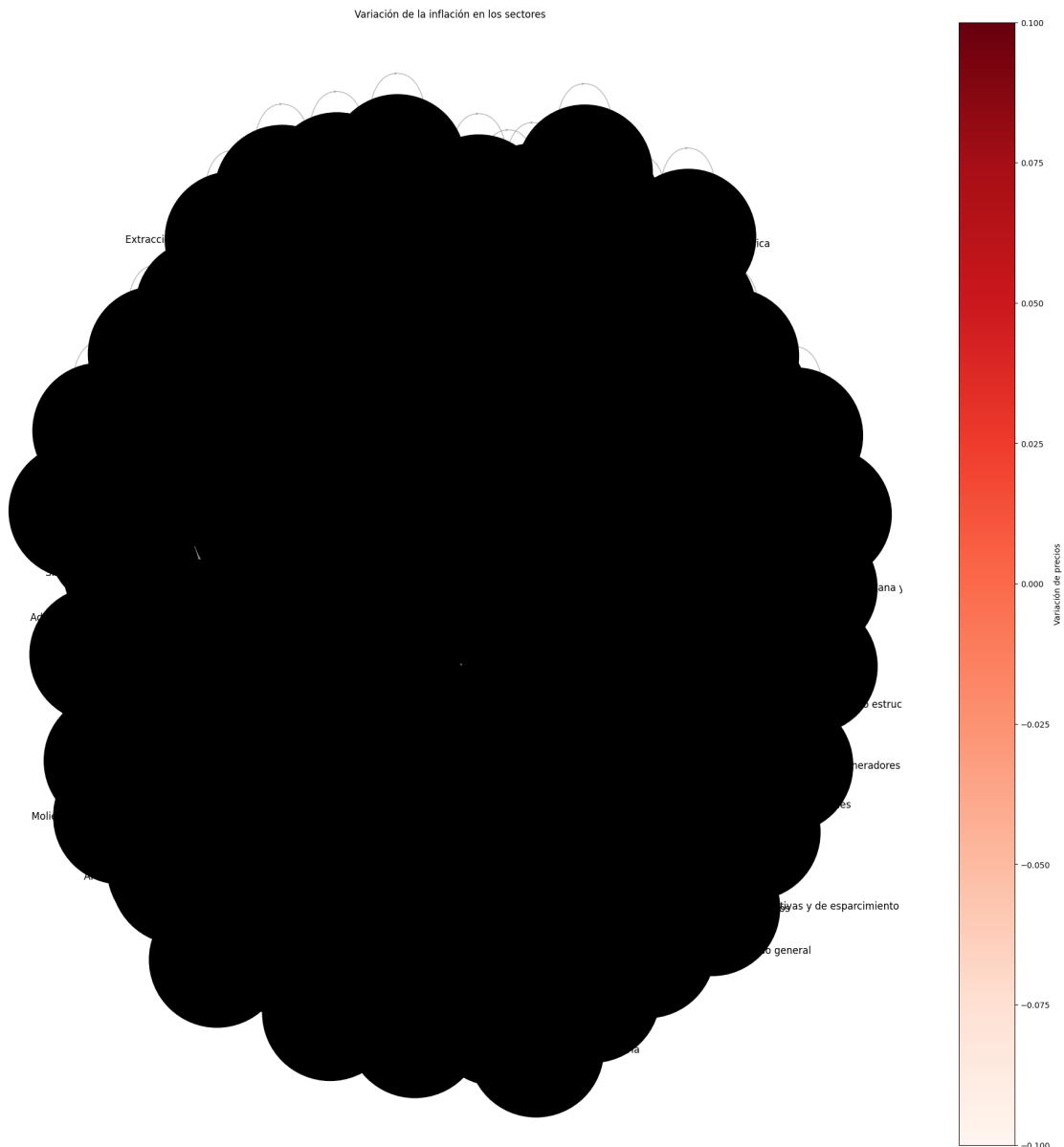


```
[21]: sector = ('Instituciones Financieras')
mip_grafo = grafo.armar_grafo(mip, precios_random=False)
experimento = exp.Experimento(mip_grafo,
                               dinamica=modelo.dinamica_3,
                               calcular_inflacion=modelo.calcular_inflacion,
                               duracion_periodo=20,
                               alpha=0.2)
experimento.shock(sector,20)
experimento.step(600)
precios_finales =[]
```

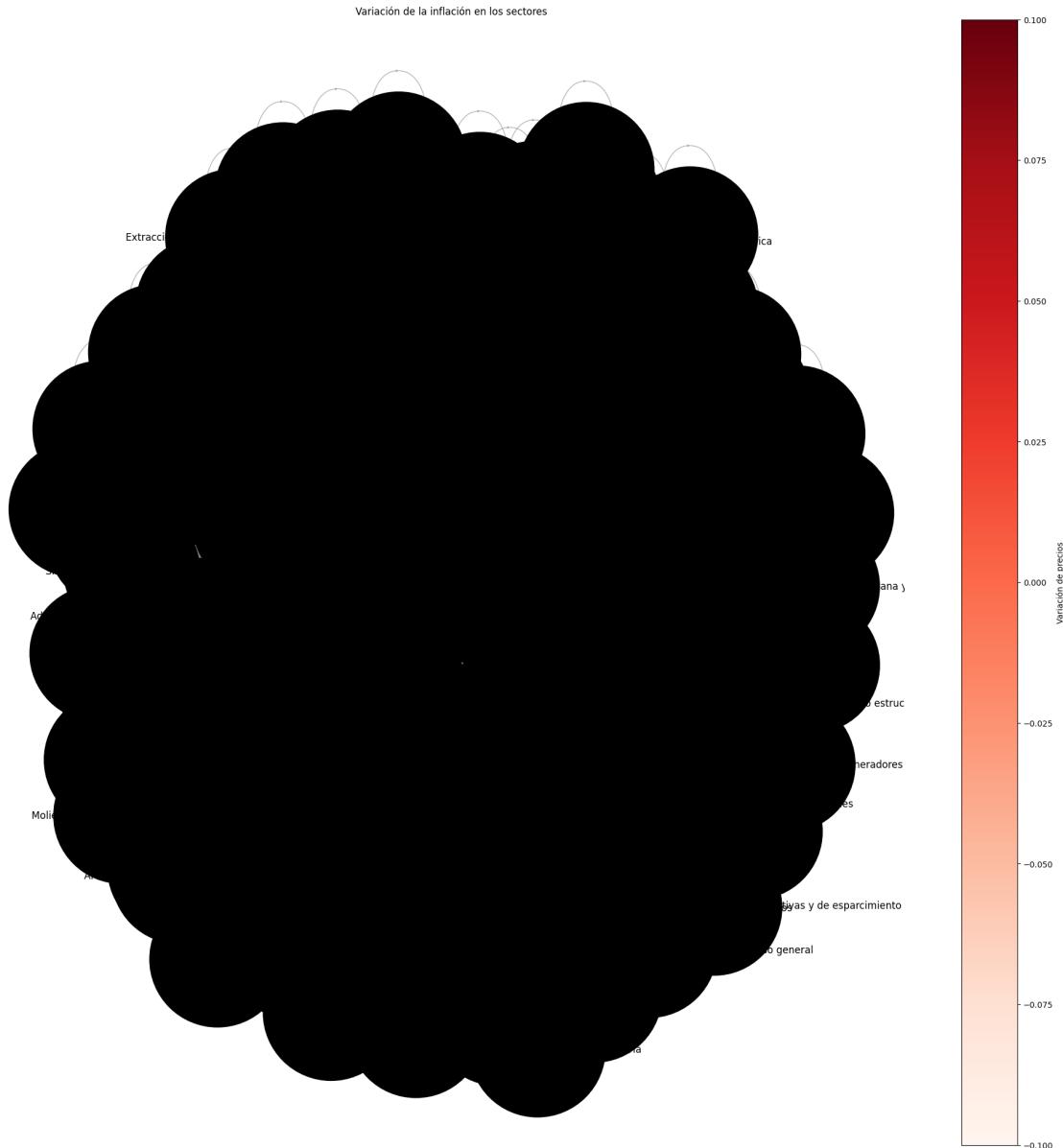
```
for sector in mip_grafo.nodes():
    precio = mip_grafo.nodes()[sector]['precio']
    precios_finales.append(precio)

precios_iniciales = [100 for _ in precios_finales]
```

```
[22]: grafo.verInflacion(mip, precios_finales, precios_iniciales)
```



[22] :



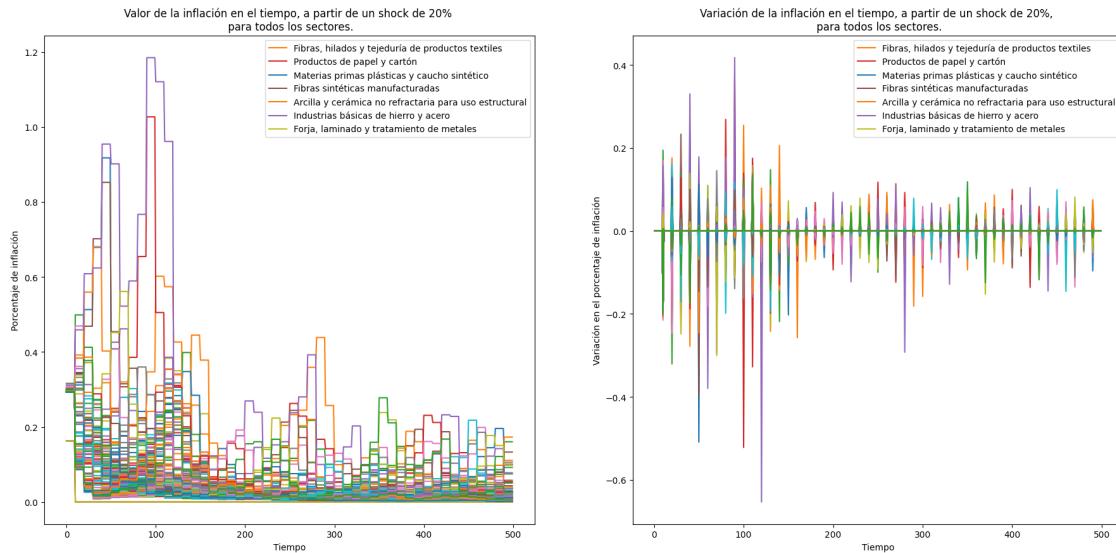
```
[23]: aumentos = [20]
umbral_visualizacion = [1000,2000,2000]
alpha = 0.2
mip_grafo = grafo.armar_grafo(mip,precios_random=False)
sectores = list(mip_grafo.nodes)
for j, aumento in enumerate(aumentos):
    inflaciones = []
    for sector in sectores:
        experimento = exp.Experimento(mip_grafo,
                                         dinamica=modelo.dinamica_3,
```

```

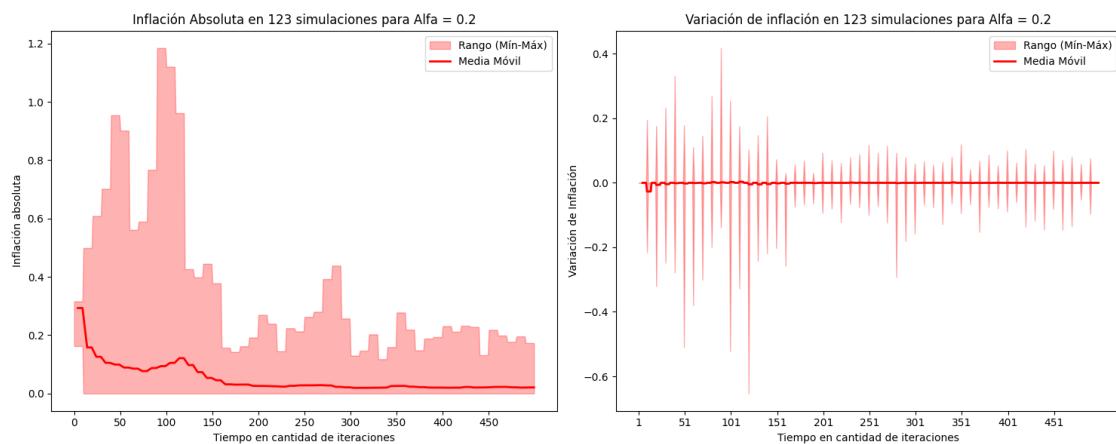
    calcular_inflacion=modelo.calcular_inflacion,
    alpha=alpha)
experimento.shock(sector,aumento)
experimento.step(500)
inflacion = experimento.metricas_evaluadas['inflacion']
inflaciones.append(inflacion)

plot_inflaciones(inflaciones,aumento,0.5)

```



[24]: verEvolucion(alfa02, 'Alfa = 0.2')

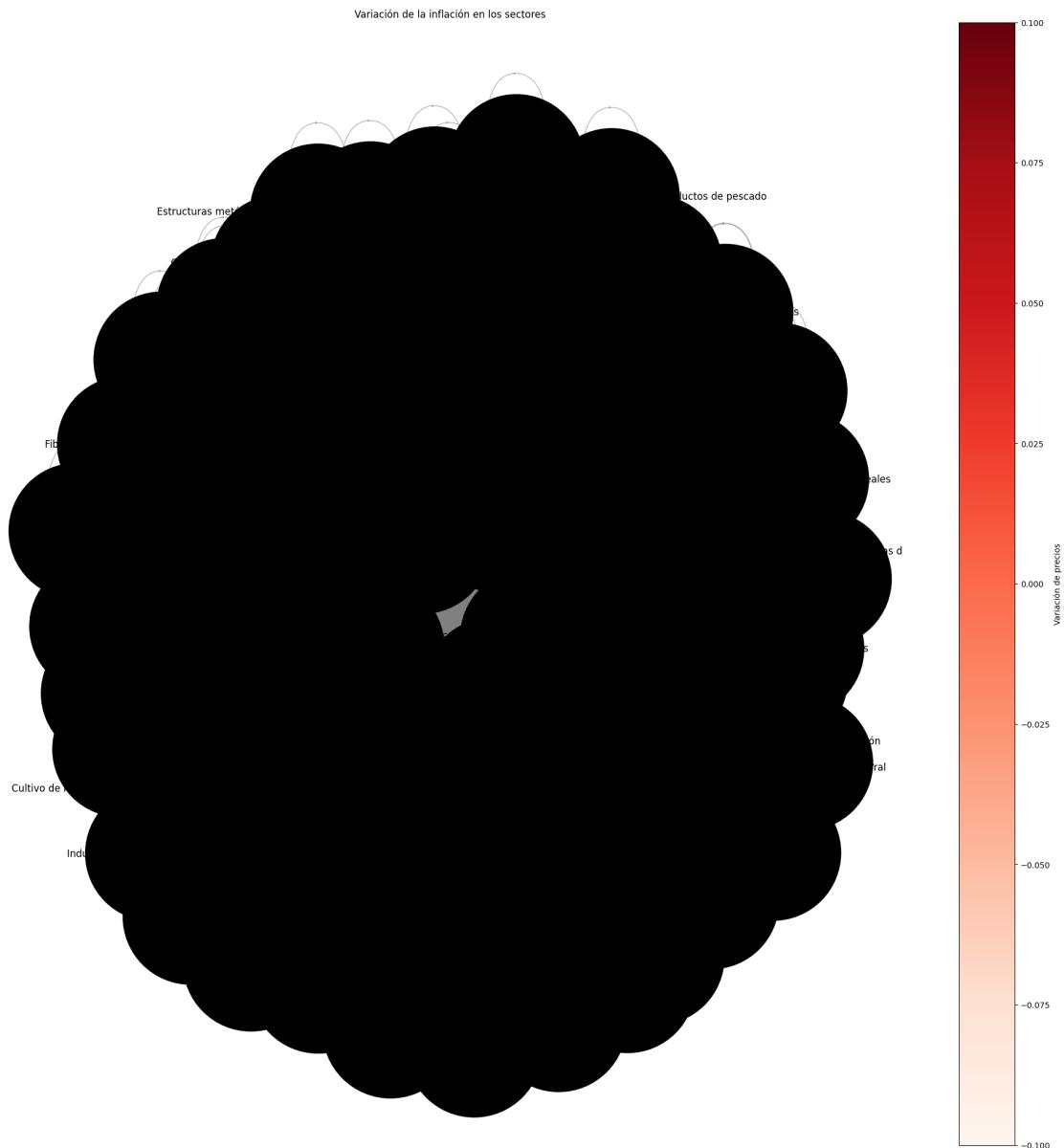


4.2.3 Alpha 0.3

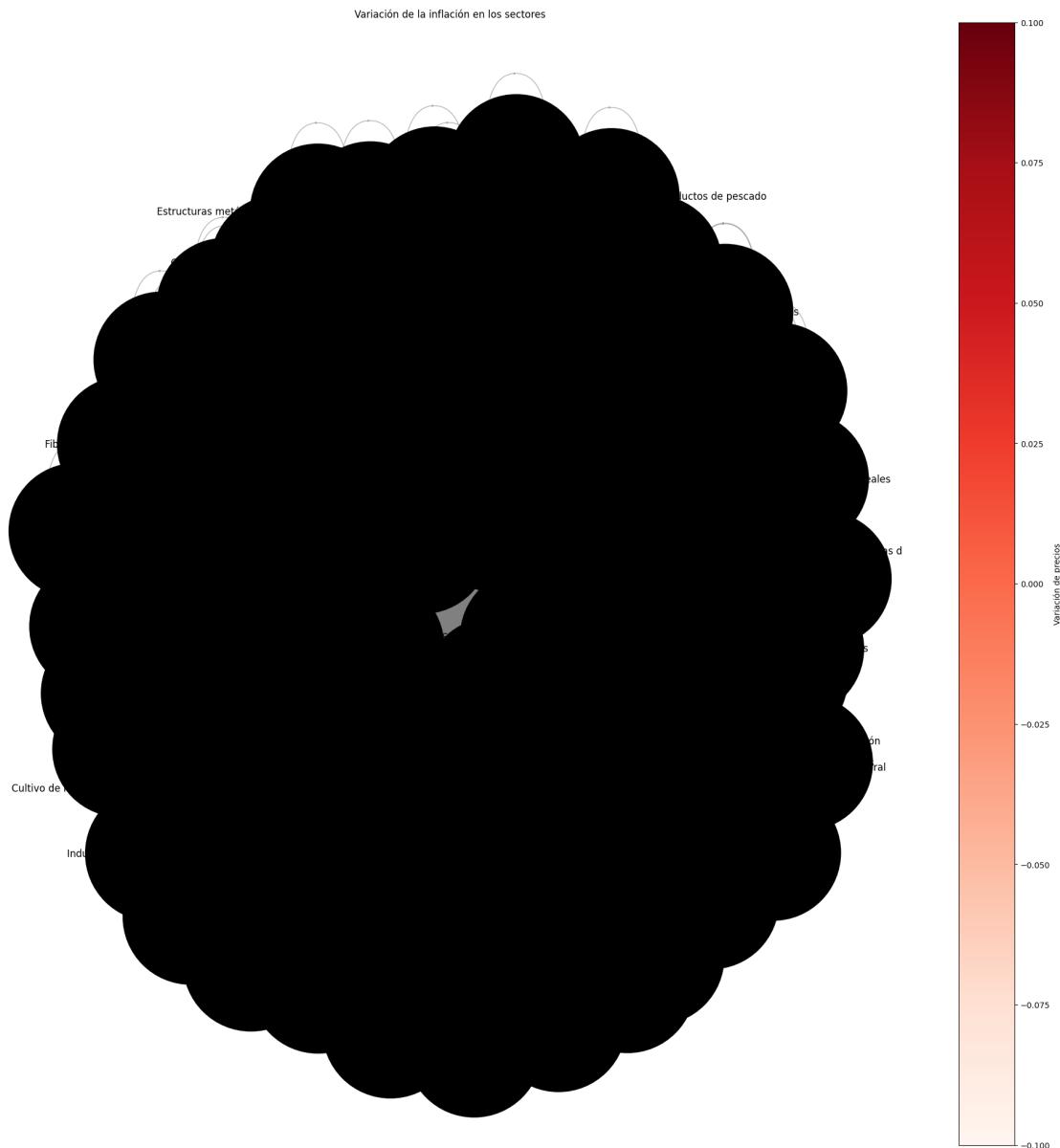
```
[25]: sector = ('Administración pública y defensa y planes de la seguridad social de la afiliación obligatoria')
mip_grafo = grafo.armar_grafo(mip, precios_random=False)
experimento = exp.Experimento(mip_grafo,
                               dinamica=modelo.dinamica_3,
                               calcular_inflacion=modelo.calcular_inflacion,
                               duracion_periodo=20,
                               alpha=0.2)
experimento.shock(sector, 20)
experimento.step(600)
precios_finales = []
for sector in mip_grafo.nodes():
    precio = mip_grafo.nodes()[sector]['precio']
    precios_finales.append(precio)

precios_iniciales = [100 for _ in precios_finales]
```

```
[26]: grafo.verInflacion(mip, precios_finales, precios_iniciales)
```



[26] :



```
[27]: sector = ('Instituciones Financieras')
mip_grafo = grafo.armar_grafo(mip, precios_random=False)
experimento = exp.Experimento(mip_grafo,
                               dinamica=modelo.dinamica_3,
                               calcular_inflacion=modelo.calcular_inflacion,
                               duracion_periodo=20,
                               alpha=0.3)
experimento.shock(sector, 20)
experimento.step(600)
precios_finales = []
```

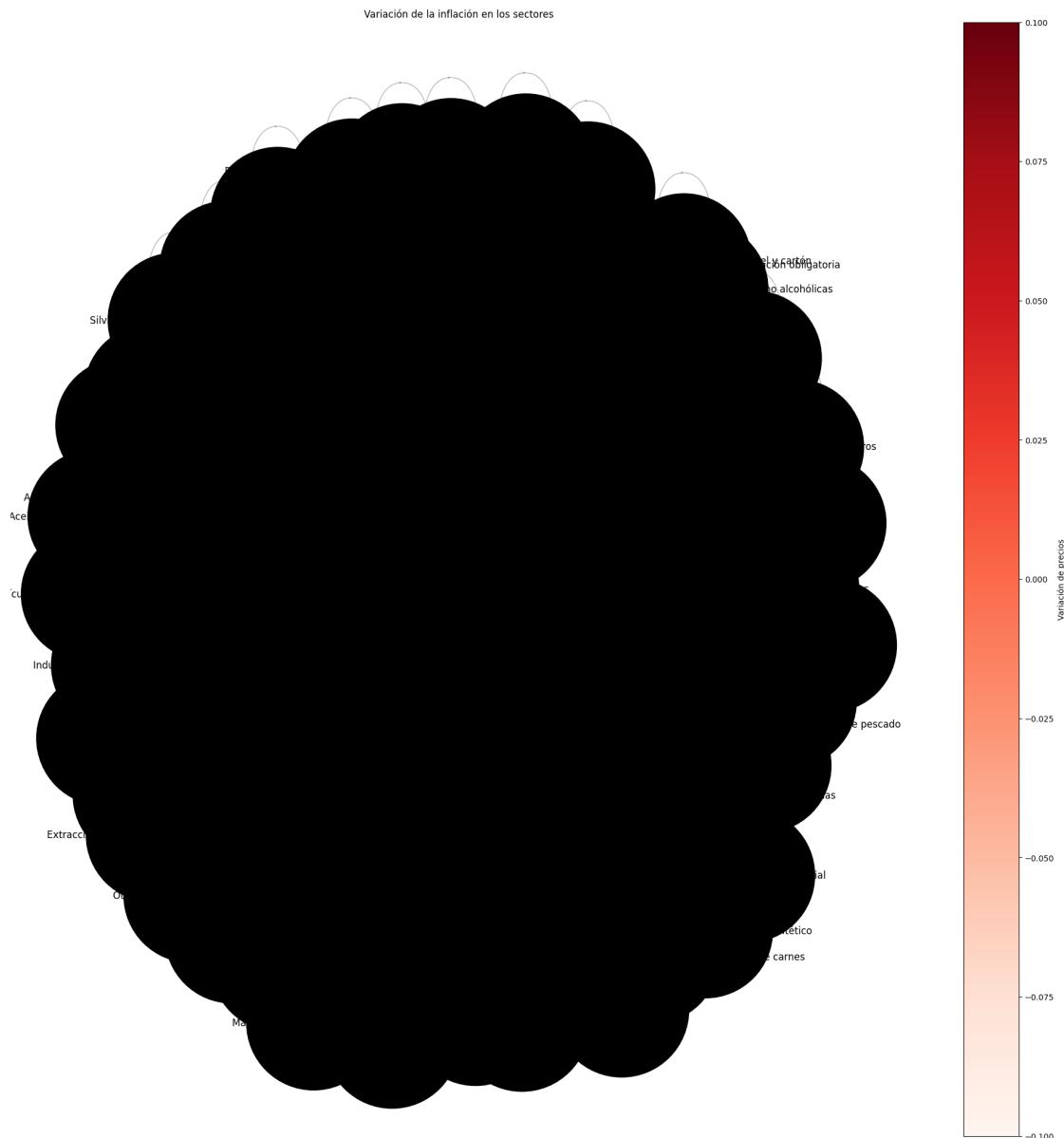
```

for sector in mip_grafo.nodes():
    precio = mip_grafo.nodes()[sector]['precio']
    precios_finales.append(precio)

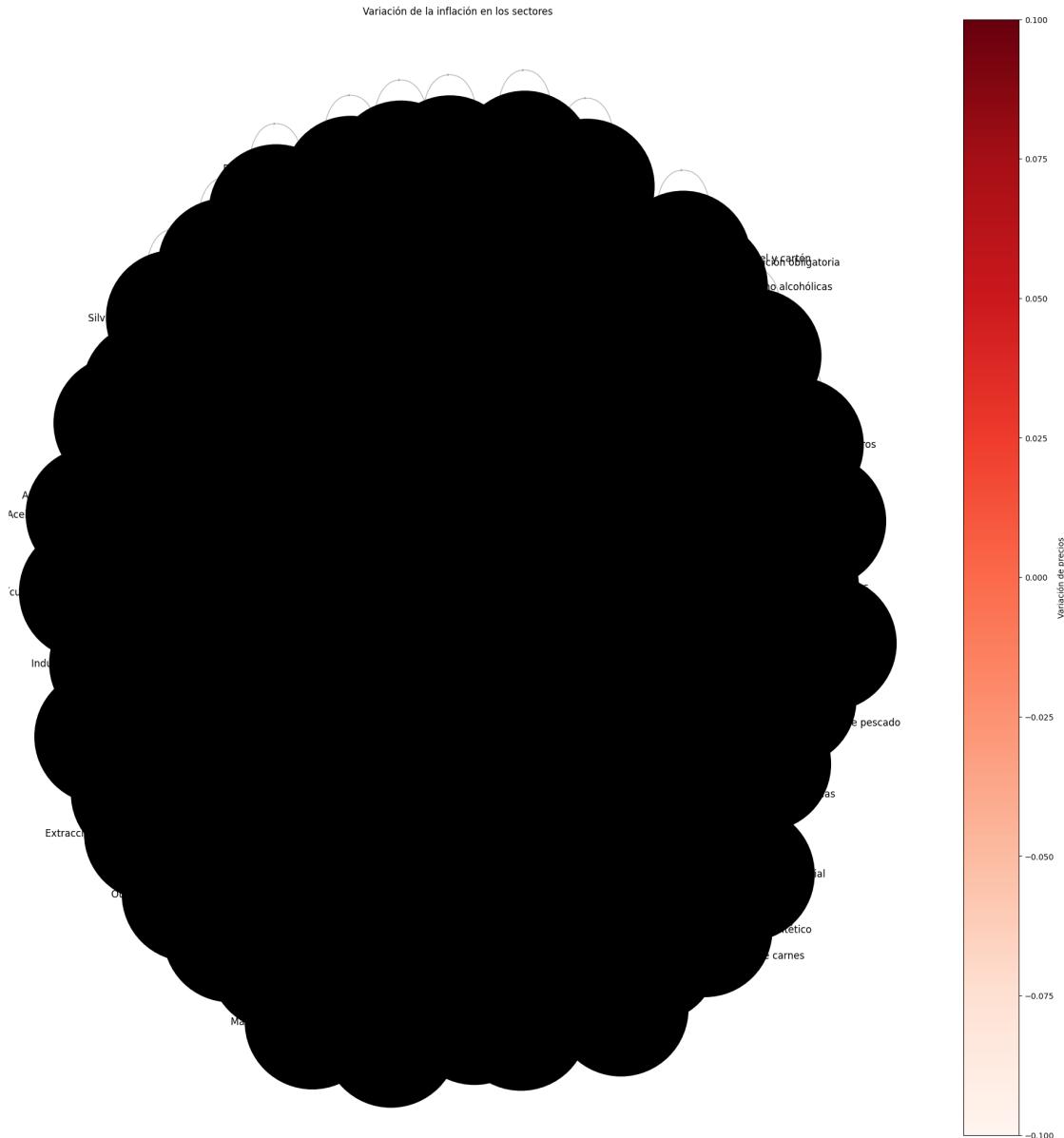
precios_iniciales = [100 for _ in precios_finales]

```

[28]: `grafo.verInflacion(mip, precios_finales, precios_iniciales)`



[28]:

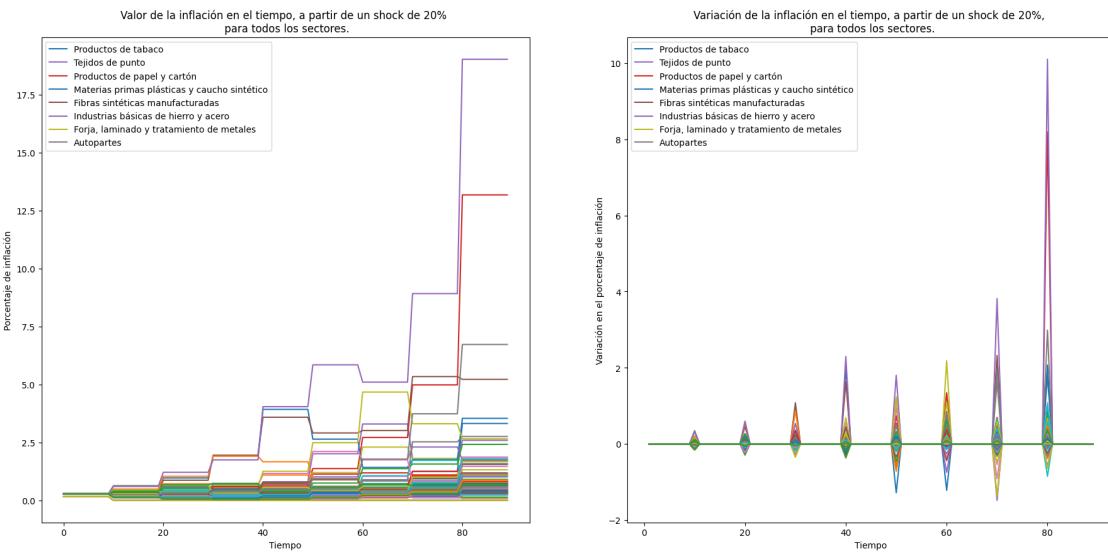


```
[29]: aumento = 20
# umbral_visualizacion = [1000]
alpha = 0.3
mip_grafo = grafo.armar_grafo(mip,precios_random=False)
sectores = list(mip_grafo.nodes)
inflaciones = []
for sector in sectores:
    experimento = exp.Experimento(mip_grafo,
                                    dinamica=modelo.dinamica_3,
                                    calcular_inflacion=modelo.calcular_inflacion,
```

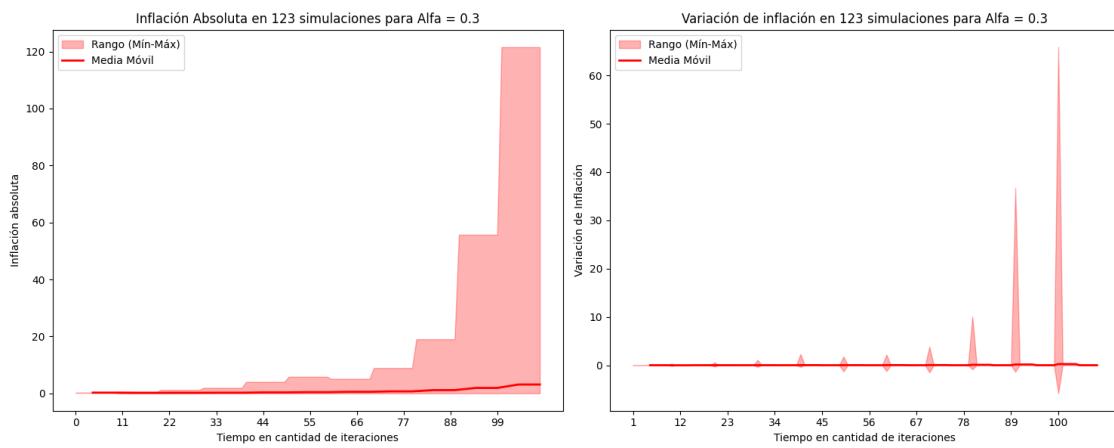
```

alpha=alpha)
experimento.shock(sector,aumento)
experimento.step(90)
inflacion = experimento.metricas_evaluadas['inflacion']
inflaciones.append(inflacion)
plot_inflaciones(inflaciones, aumento,3)

```



[30]: `verEvolucion(alfa03, 'Alfa = 0.3')`



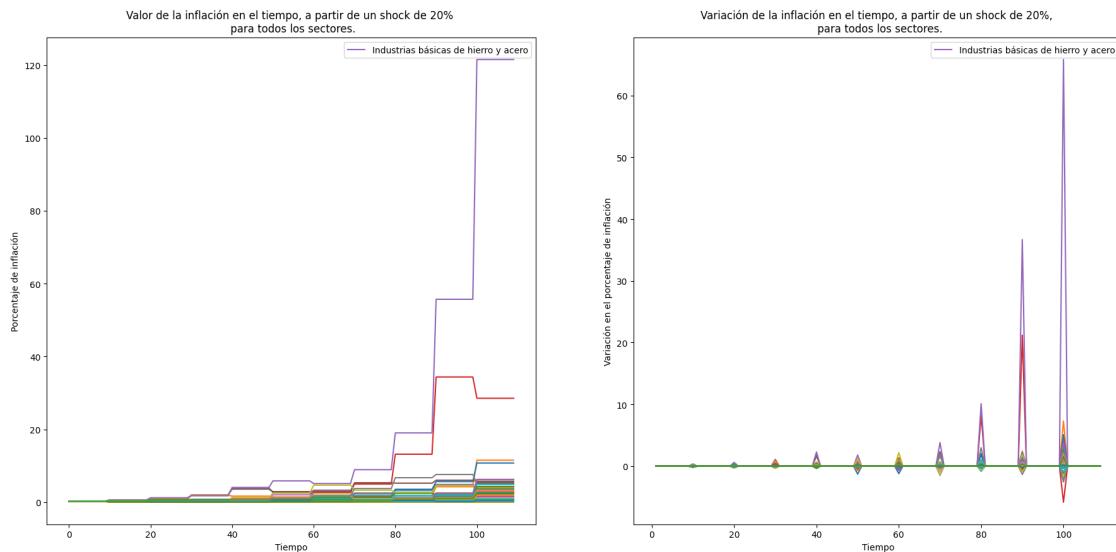
El mismo experimento pero con más pasos.

[31]: `aumento = 20
alpha = 0.3`

```

mip_grafo = grafo.armar_grafo(mip, precios_random=False)
sectores = list(mip_grafo.nodes)
inflaciones = []
for sector in sectores:
    experimento = exp.Experimento(mip_grafo,
                                    dinamica=modelo.dinamica_3,
                                    calcular_inflacion=modelo.calcular_inflacion,
                                    alpha=alpha)
    experimento.shock(sector,aumento)
    experimento.step(110)
    inflacion = experimento.metricas_evaluadas['inflacion']
    inflaciones.append(inflacion)
plot_inflaciones(inflaciones,aumento,50)

```



4.2.4 Alpha 0.4

```

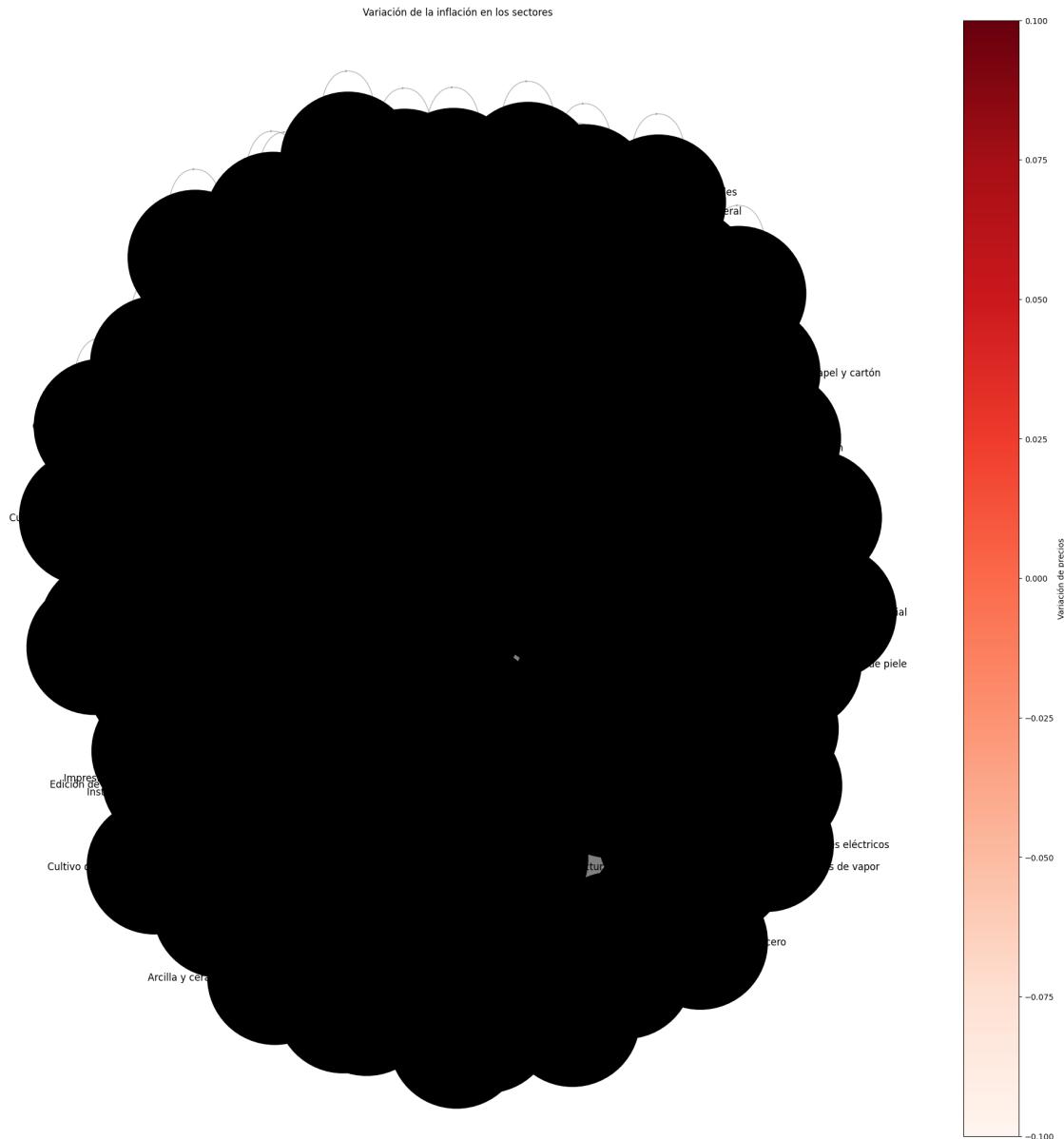
[32]: sector = ('Administración pública y defensa y planes de la seguridad social de la afiliación obligatoria')
mip_grafo = grafo.armar_grafo(mip, precios_random=False)
experimento = exp.Experimento(mip_grafo,
                               dinamica=modelo.dinamica_3,
                               calcular_inflacion=modelo.calcular_inflacion,
                               duracion_periodo=20,
                               alpha=0.2)
experimento.shock(sector, 20)
experimento.step(600)
precios_finales = []
for sector in mip_grafo.nodes():

```

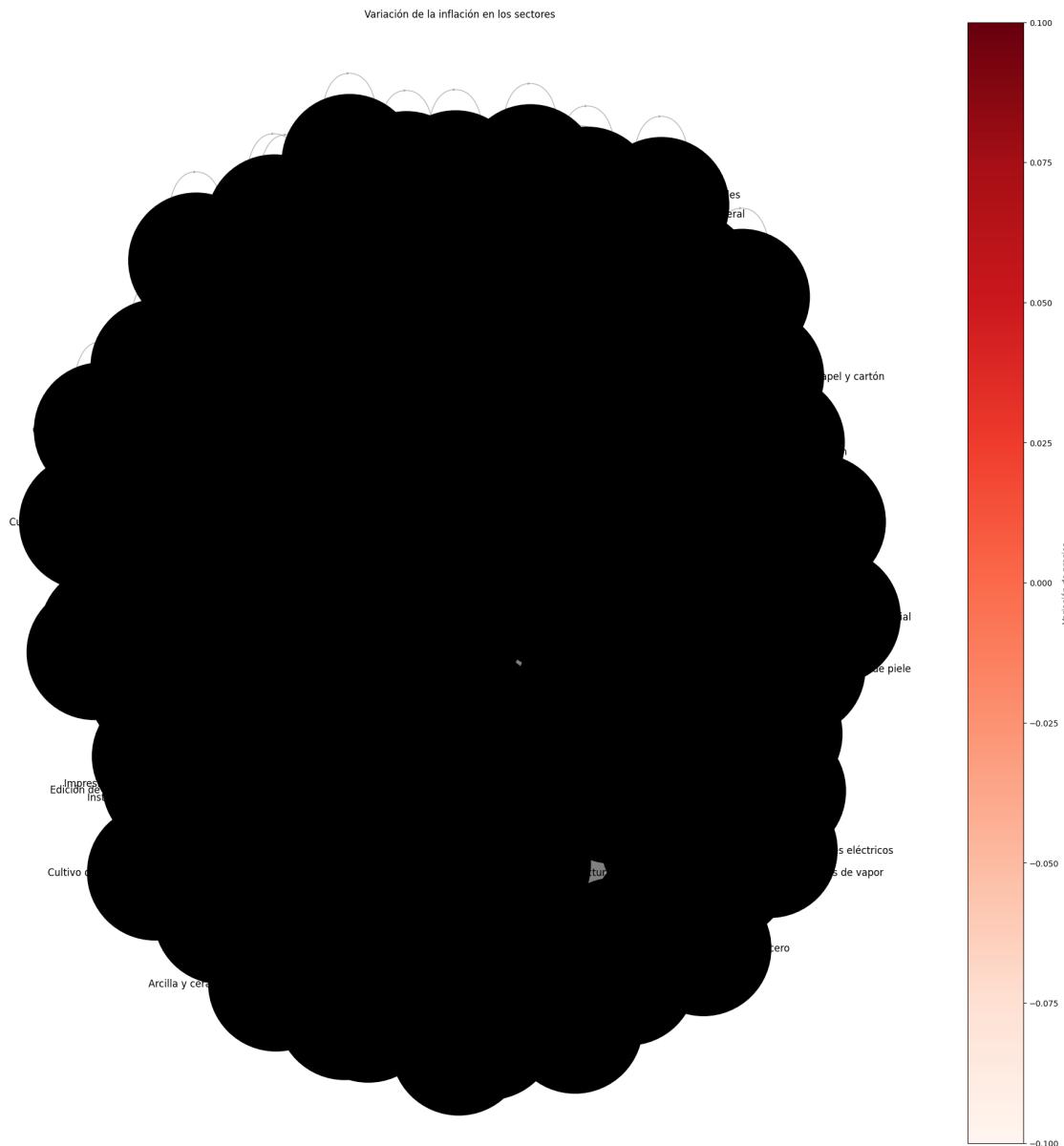
```
precio = mip_grafo.nodes()[sector]['precio']
precios_finales.append(precio)

precios_iniciales = [100 for _ in precios_finales]
```

```
[33]: grafo.verInflacion(mip, precios_finales, precios_iniciales)
```



```
[33]:
```



```
[34]: sector = ('Instituciones Financieras')
mip_grafo = grafo.armar_grafo(mip, precios_random=False)
experimento = exp.Experimento(mip_grafo,
                               dinamica=modelo.dinamica_3,
                               calcular_inflacion=modelo.calcular_inflacion,
                               duracion_periodo=20,
                               alpha=0.3)
experimento.shock(sector, 20)
experimento.step(600)
precios_finales = []
```

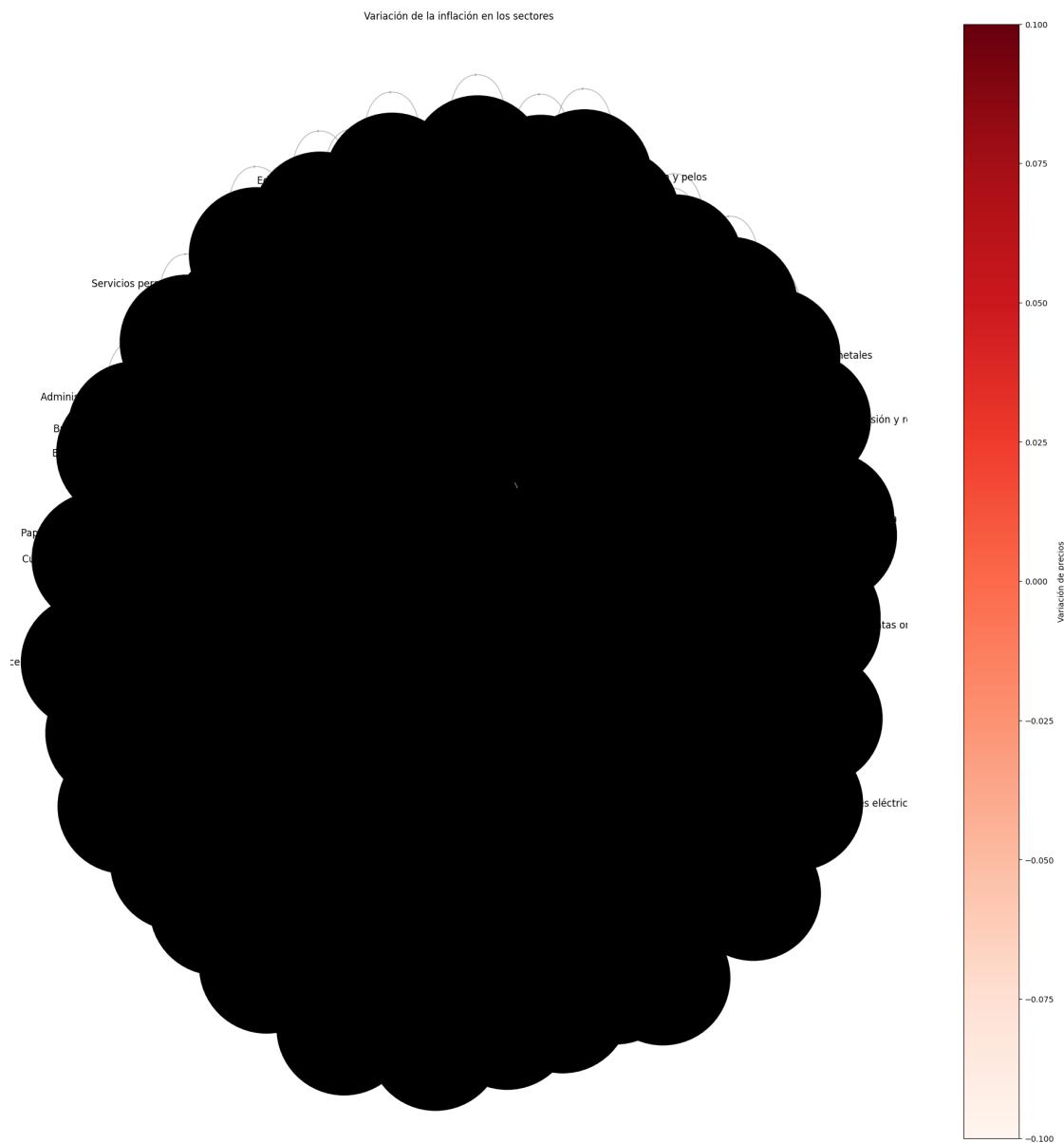
```

for sector in mip_grafo.nodes():
    precio = mip_grafo.nodes()[sector]['precio']
    precios_finales.append(precio)

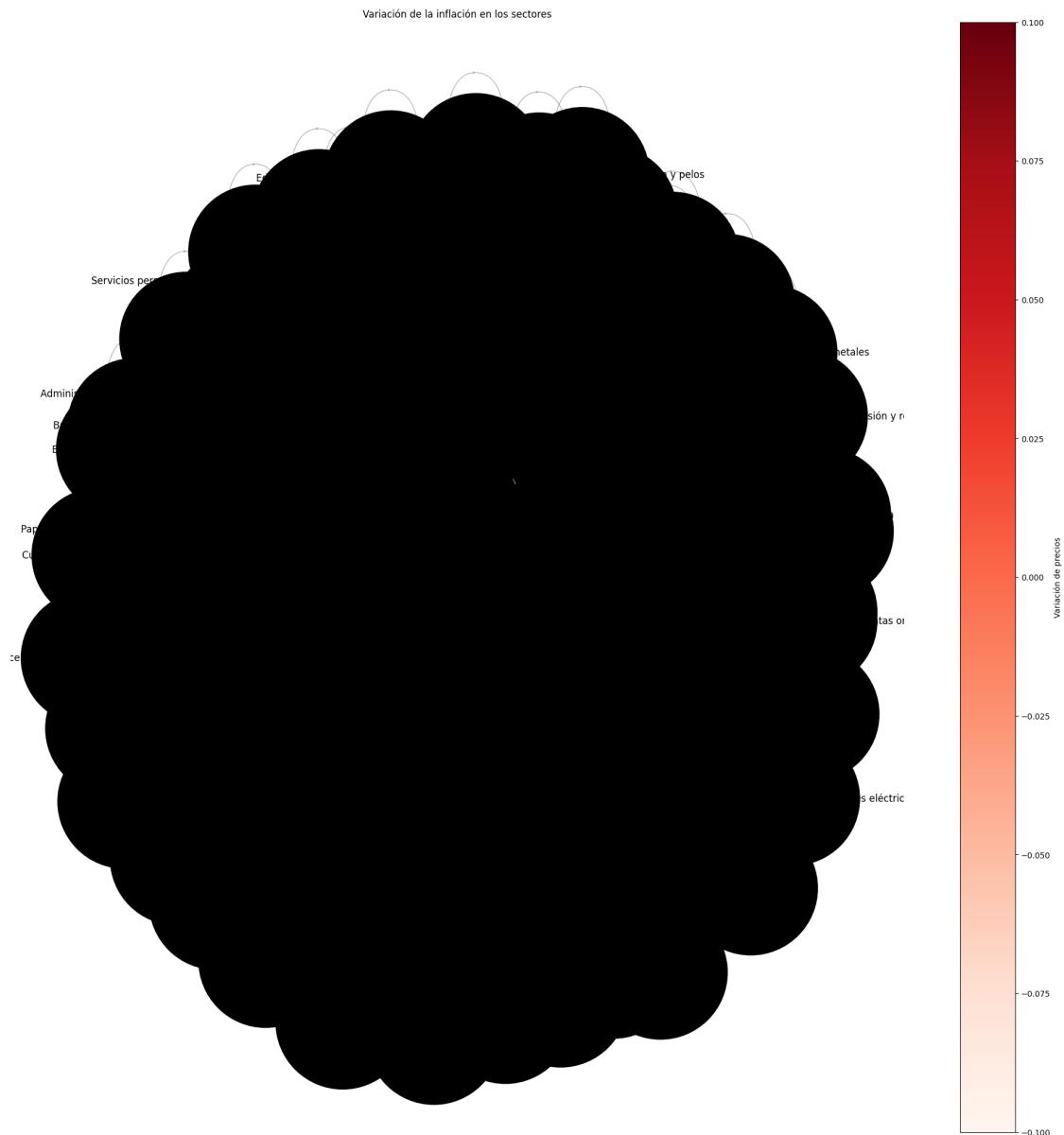
precios_iniciales = [100 for _ in precios_finales]

```

[35] : grafo.verInflacion(mip, precios_finales, precios_iniciales)



[35] :

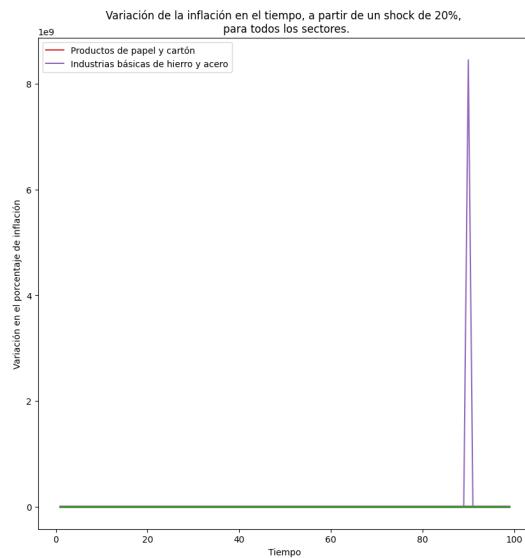
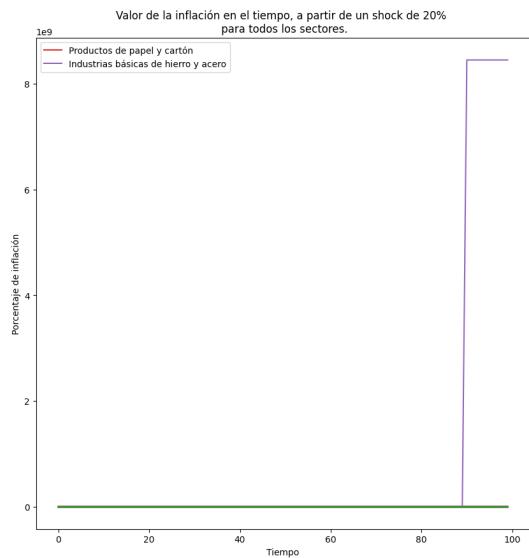


```
[36]: aumento = 20
# umbral_visualizacion = [1000]
alpha = 0.4
mip_grafo = grafo.armar_grafo(mip,precios_random=False)
sectores = list(mip_grafo.nodes)
inflaciones = []
for sector in sectores:
    experimento = exp.Experimento(mip_grafo,
                                    dinamica=modelo.dinamica_3,
                                    calcular_inflacion=modelo.calcular_inflacion,
```

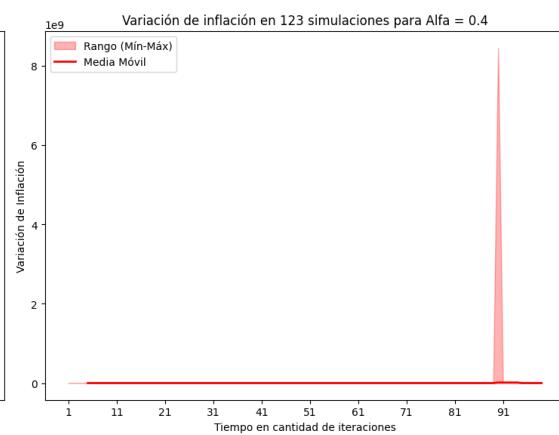
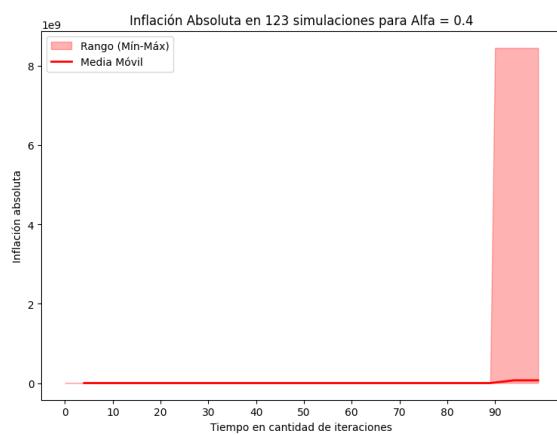
```

        alpha=alpha)
experimento.shock(sector,aumento)
experimento.step(100)
inflacion = experimento.metricas_evaluadas['inflacion']
inflaciones.append(inflacion)
plot_inflaciones(inflaciones, aumento,3000)

```



[37]: verEvolucion(alfa04, 'Alfa = 0.4')



5 Conclusiones

5.1 Julio Olivera

Inflación estructural: La inflación estructural se refiere a un tipo de inflación causada por desajustes sectoriales que afectan a productos específicos, lo que resulta en aumentos de precios individuales que luego se generalizan. Estos desajustes pueden ser el resultado de la rigidez de la estructura productiva y la imperfección de los mercados, y son ajenos a las decisiones de las autoridades monetarias. La inflación estructural se origina en desequilibrios sectoriales que afectan a productos específicos y no en un desajuste global entre la oferta y la demanda monetaria.

Olivera, J. H. G. (1965), "Inflación estructural y política financiera".

Podemos notar que este modelo apoya la noción de que la inflación tiene un origen estructural. El hecho de que exista un alfa tal que si el aumento local de precios en un sector productivo estuviese dado por $\alpha * \text{inflación_global}$, podría ser interpretado como una descoordinación entre los agentes. Estos hitos en la red estarían dando lugar a una espiralización de la inflación del modelo, que podría ser interpretada como la generalización de aumentos de precios individuales.

5.2 Sobre los experimentos y las preguntas respondidas

- ¿Cómo se propaga el aumento de precios a través de la red definida por la MIP?

La propagación de un shock en el sistema productivo varía dependiendo tanto del sector a cual se realiza el shock como del alpha del experimento. En nodos que estan fuertemente conectados con otros, que propagan más un aumento, los shocks tienen más impacto que en aquellos con grados o pesos menores en sus aristas.

- ¿Cómo impacta el aumento de precio en un producto/sector sobre otros productos/sectores? ¿Se mantiene en la misma cadena productiva? ¿Cómo influyen la dinámica de comportamiento de los agentes en la variación de la inflación global?

Si el alpha es 0 el shock a un sector sólo genera cambios de precios en las aristas alcanzables por ese nodo. Además, como todos los pesos en las aristas son menores a 1, existe un decaimiento en los aumentos que se van pasando de sector a sector. ““““Por eso, ese caso, sí se puede decir que el aumento de precio se mantiene en la misma cadena productiva.”““““ Por otro lado, cuando el alpha aumenta y los agentes se ven cada vez más influenciados por la inflación global a la hora de aumentar sus precios, la respuesta ya no es tan clara. Para que un agente tome la decisión de aumentar su precio, le tiene que llegar un aumento, es decir, no tienen la capacidad de actualizarse si no les “avisan”. Por eso, es técnicamente cierto que el aumento se propaga sólo en la cadena productiva del sector que es shockeado. Lo interesante del modelo es que, al no ser nosotros los que definimos esa cadena sino que generamos la topografía de la red a partir del MIP, estas cadenas productivas son muy complejas y logran capturar todas las dependencias que existen entre sectores. A partir de eso, cuando el alpha aumenta, sectores a los que le hubiese llegado un aumento casi residual en un primer momento, toman la decisión de aumentar sus precios viendo la variable macro y así comienzan a generar aumentos generalizados en toda la red, es decir, inflación.

- ¿Cuál es la sensibilidad del sistema respecto a variaciones de precio en nodos específicos?
¿Cuáles son los nodos que propagan de mayor forma la variación de precios?

Los sectores ‘Administración pública y defensa y planes de la seguridad social de afiliación obligatoria’, ‘Enseñanza Pública’, ‘Enseñanza privada’ y ‘Salud humana pública’ son los más influyentes en la red según la centralidad de autovalores.

5.3 Próximos pasos

- El modelo en su estado actual calcula la inflación a partir de un IPC basado en una canasta básica sin ponderar. Como primer paso sería interesante agregar ponderaciones a los sectores para lograr un mejor modelo.
- La MIP también tiene la información de cuánto de lo producido por un sector se vende al consumidor final. Con eso se puede generar un nuevo agente que represente ese sector, el del consumo final, y que tenga ciertos sectores más prioritarios que otros a la hora de consumirlo. Con eso, se podría eliminar el supuesto de la Ley de Say, agregando el consumo a las dinámicas. Eso podría dar lugar a deflación.
- Clusterizar ciertos nodos del grado en rubros, y estudiar los shocks dentro de esos microsectores.
- A partir de identificar los nodos en los cuales el grafo es mas vulnerable a un shock (Un shock en ellos generan significativamente más inflación que en el resto), plantear una reestructuración del grafo para hacerlo mas estable y seguro, teniendo en cuenta las restricciones pertinentes para que el grafo resultante siga siendo un esquema productivo.
- Si se logra eliminar la Ley de Say, sería interesante reemplazar los sectores por agentes individuales de aquel sector, para poder modelar dinámicas de competencia en los precios.

6 Apéndices

6.1 Código

- Experimento
- Bla

6.1.1 Experimento

[38] : exp.Experimento??

```
Init signature:  
exp.Experimento(  
    grafo,  
    dinamica,  
    duracion_periodo=10,  
    metricas=None,  
    calcular_inflacion=None,  
    alpha=0.5,  
)
```

Docstring: <no docstring>

Source:

```

class Experimento:

    def __init__(self, grafo, dinamica, duracion_periodo = 10, metricas=None, u
    -calcular_inflacion=None, alpha = 0.5):
        """
        Inicializa una instancia de la clase Experimento.

    Parámetros
    -----
    grafo : DiGraph
        Grafo que representa el MIP
    dinamica : Callable
        Función con la siguiente signatura : (aumento, peso_arista, inflacion,u
        -alpha)
    duracion_periodo : int, opcional
        La duración de cada período en pasos para actualizar la inflacion, poru
        -defecto 10 pasos.
    metricas : Dict(string,Callable), opcional
        Diccionario de con metricas a computar sobre el grafo, ej, u
        -{"cantidad_nodos", lambda grafo: len(grafo)}
    calcular_inflacion : Callable, opcional
        Función para calcular la inflación en un período, recibe dos arrays deu
        -precios. ej, calc(precios_actual, precios_pasado)
    alpha : float, opcional
        Parametro de ponderacion
        El alpha es CUANTO ve la inflación.
        Alpha == 1 es lo mismo que la dinamica 2.
        Alpha == 0 es lo mismo que la dinamica 1.

    """
    self.grafo = grafo.copy()
    self.dinamica = dinamica
    self.metricas = metricas or {}
    self._curr_step = 0
    self.metricas_evaluadas = {nombre:[] for nombre in self.metricas}
    self.metricas_evaluadas['inflacion'] = []
    self.queue = queue.Queue() #Fila de tuplas (Nodo, aumento)
    self.duracion_periodo = duracion_periodo
    self.precios_periodo_pasado = {}
    for nodo in self.grafo.nodes:

```

```
File:          c:\users\augus\dev\gh\akielbowicz\tp-msscae-2024\src\experimento.  
~py  
Type:         type  
Subclasses:
```

6.1.2 Dinámicas

```
[39]: modelo.dinamica_1??
```

```
Signature: modelo.dinamica_1(aumento, peso_arista, inflacion=0.0, alpha=1.0)  
Docstring: <no docstring>  
Source:  
def dinamica_1(aumento, peso_arista, inflacion=0.0, alpha=1.0):  
    aumento_vecino = aumento * peso_arista  
  
    return aumento_vecino  
File:          c:\users\augus\dev\gh\akielbowicz\tp-msscae-2024\src\modelo.py  
Type:         function
```

```
[40]: modelo.dinamica_2??
```

```
Signature: modelo.dinamica_2(aumento, peso_arista, inflacion, alpha=1.0)  
Docstring: <no docstring>  
Source:  
def dinamica_2(aumento, peso_arista, inflacion, alpha=1.0):  
    aumento_vecino = inflacion  
  
    return aumento_vecino  
File:          c:\users\augus\dev\gh\akielbowicz\tp-msscae-2024\src\modelo.py  
Type:         function
```

```
[41]: modelo.dinamica_3??
```

```
Signature: modelo.dinamica_3(aumento, peso_arista, inflacion, alpha)  
Docstring: <no docstring>  
Source:  
def dinamica_3(aumento, peso_arista, inflacion, alpha):  
    aumento_vecino = (alpha * inflacion) + ((1 - alpha) * ((aumento) *  
    ~peso_arista))  
  
    return aumento_vecino  
File:          c:\users\augus\dev\gh\akielbowicz\tp-msscae-2024\src\modelo.py  
Type:         function
```