Parcialito 3

Teoría de Algoritmos 2

Guido Botta

Padrón 102103

Enunciado

Teoría de Algoritmos II (75.30)

3.er Parcialito Domiciliario – 13/06/2022 - Fecha de Entrega: 15/07/2022

Aclaraciones: Cada ejercicio dice al final del mismo la cantidad de puntos que otorga por hacerse completamente bien (en total, 10). Se deben obtener al menos 5 puntos para aprobar, y se deben aprobar al menos 3 de los parcialitos para aprobar/regularizar la cursada. Para la fecha de entregar, enviar un mail a mbuchwald@fi.uba.ar con un pdf con la resolución, con nombre P3 - PADRON.pdf. Pueden incluir todo el material adicional que les parezca relevante (desde código hasta gráficos).

1. Se quiere convocar a una elección a la que se presentan 4 candidatos (A, B, C y D). Hay 3 votantes del jurado que tienen sus siguientes rankings individuales:

- Jurado 1: $B \succ C \succ D \succ A$
- Jurado 2: $C \succ D \succ A \succ B$
- Jurado 3: $D \succ A \succ B \succ C$

1. A. ¿Quién ganaría por eliminación iterativa?
   B. ¿Quién ganaría por Borda rule?
   C. Suponé que estás a cargo de definir las reglas/formato de la votación, y sos un miembro corrupto que desea que si o si gane la alternativa A (te asegura favores si logra ganar la elección). Definir (si existe) un sistema de votación en el cual A resulte ganador de la elección. En caso de no existir, explicar por qué. ¿Cuál propiedad deseable de los sistemas de votación no se está cumpliendo si, efectivamente, ganara A?

[1 Punto]

1. Considerando el modelo de cascadas de información visto en clase, supongamos que hay una nueva tecnología que los individuos pueden optar por aceptar o rechazar. Supongamos que cada uno que acepta la tecnología recibe una ganancia positiva o negativa (sin conocerla a priori). Estos valores son aleatorios para cada nodo, y si la tecnología es *"Buena"*, entonces el promedio será positivo, y si la tecnología es *"Mala"* el promedio será negativo (esta información es conocida por los individuos). Quienes

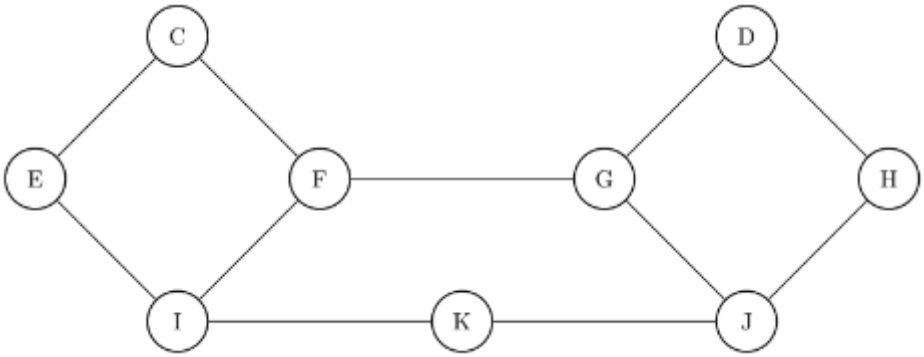
rechacen la tecnología reciben ganancia 0. En este modelo, cuando a un individuo le toca elegir si acepta o rechaza la nueva tecnología, recibe la información de las ganancias de todos los que vinieron antes.

A. Supongamos que esta nueva tecnología es, en realidad, *"Mala"*. ¿Cómo afecta esta nueva información (qué ganancia tuvo cada uno de los que vinieron antes) a la potencial formación de una cascada para que persista la nueva tecnología? (No es necesario dar una demostración, simplemente argumentar)

B. Supongamos que esta nueva tecnología es, en realidad, *"Buena"*. ¿Puede surgir una cascada de rechazo de esta nueva tecnología?

[2 Puntos]

1. Considerá la siguiente red, suponiendo que todos los nodos tienen inicialmente un comportamiento B. Cada nodo puede cambiar al comportamiento A si al menos la mitad de sus vecinos tiene dicho comportamiento.



A. Supongamos que los nodos E y F son *early adopters* del comportamiento A. Si los demás nodos siguen la regla del umbral (threshold) para adherir a este nuevo comportamiento, ¿qué nodos implementarían el comportamiento A?

B. Explicar a qué se debe que el comportamiento Ano se propaga a través de toda la red en el escenario del punto (a). ¿Qué característica de la red lo impide? (responder a esta pregunta no apuntando a nodos particulares sino a presencias de ciertas características) ¿Dónde más tendría que haber otro *early adopter* de Así o sí para que el comportamiento se propague a través de toda la red?

[2 Puntos]

1. Tenemos dos grafos no dirigidos $G1$ y $G2$, con la misma cantidad de vértices y aristas. $G1$ es un grafo aleatorio de Erdös-Rényi, mientras $G2$ es un grafo que cumple la ley de potencias en la distribución de los grados. Consideremos un virus que comienza en un único vértice aleatorio y se expande según el modelo **SIR**.
   A. ¿En cuál grafo es más probable que ocurra una epidemia (i.e. se infecte al menos un 30% de la red)? Justificar brevemente la respuesta.
   B. Supongamos que en vez de comenzar en un vértice aleatorio, la epidemia comenzara en el vértice de mayor grado de $G1$ y $G2$, respectivamente. ¿En cuál de los grafos es más probable que ocurra una epidemia? Justificar brevemente la respuesta.

C. ¿Cómo afecta la existencia (o no existencia) de comunidades en la expansión de la epidemia?

Para responder estas preguntas, se les recomienda realizar simulaciones. Pueden agregar todo tipo de resultados obtenidos para justificar sus respuestas.

[3 Puntos]

1. Aplicar el Algoritmo REV2 al siguiente set de datos de reviews de productos de Amazon, para detectar potenciales usuarios maliciosos y otros ciertamente honestos. Por simplificación (y unificación de criterios), considerar $\gamma 1 = \gamma 2 = 0.5$. Obtener aquellos usuarios cuya *justicia* (*fairness*) es menor o igual a 0.2 (son maliciosos) y tienen al menos 5 reviews, así como la proporción de nodos que son extremadamente justos: aquellos con *justicia* mayor o igual a 0.9, y con al menos 10 reviews (aristas de salida).

[2 Puntos]

Resolución

Ejercicio 1

1. Se quiere convocar a una elección a la que se presentan 4 candidatos (A, B, C y D). Hay 3 votantes del jurado que tienen sus siguientes rankings individuales:

- Jurado 1: $B \succ C \succ D \succ A$
- Jurado 2: $C \succ D \succ A \succ B$
- Jurado 3: $D \succ A \succ B \succ C$

1. A. ¿Quién ganaría por eliminación iterativa?
   B. ¿Quién ganaría por Borda rule?
   C. Suponé que estás a cargo de definir las reglas/formato de la votación, y sos un miembro corrupto que desea que si o si gane la alternativa A (te asegura favores si logra ganar la elección). Definir (si existe) un sistema de votación en el cual A resulte ganador de la elección. En caso de no existir, explicar por qué. ¿Cuál propiedad deseable de los sistemas de votación no se está cumpliendo si, efectivamente, ganara A?

Por Eliminación Iterativa

Idea:

Comparamos A con B, el ganador va contra C, el ganador contra D, y así sucesivamente.

Suponiendo que se realiza en orden A,B,C,D, funcionaría de la siguiente manera:

- Paso 1, A vs B:
  - Jurado 1: B
  - Jurado 2: A

    - Jurado 3: A

Gana A

- Paso 2: A vs C:
    - Jurado 1: C
    - Jurado 2: C
    - Jurado 3: A

Gana C

- Paso 3: C vs D:
    - Jurado 1: C
    - Jurado 2: C
    - Jurado 3: D

Gana C

Por lo tanto, el ganador sería C.

Cabe aclarar que en eliminación iterativa importa el orden en el que los candidatos compiten, esto se verá en el sistema corrupto.

Por Borda Rule

Idea:

Cada votante elige su orden de preferencia y asigna n-1 a su favorito, n-2 al segundo, ..., y 0 al último. El que tenga más puntos gana.

En este caso son 4 candidatos, por lo que los puntajes irán de 3 a 0.

- Para el jurado 1:
    - A: 0
    - B: 3
    - C: 2
    - D: 1
- Para el jurado 2:
    - A: 1
    - B: 0
    - C: 3
    - D: 2
- Para el jurado 3:
    - A: 2
    - B: 1
    - C: 0
    - D: 3

Finalmente, quedaría:

- A: 0 + 1 + 2 = 3
- B: 3 + 0 + 1 = 4
- C: 2 + 3 + 0 = 5
- D: 1 + 2 + 3 = 6

Por lo tanto, ganaría D.

Sistema Corrupto

Para favorecer a A se pronpondrá el mismo sistema utilizado en el primer punto, eliminación iterativa, pero alterando el orden elegido para la competencia.

La idea es encontrar a un candidato al cual A le pueda ganar, y dejar a A para lo último. Como A le gana a B, este entrará a competir en la anterior elección contra C, al cual B le gana. Por lo tanto, el orden podría ser (C, D, B, A) o (D, C, B, A). En el resto de casos A perdería. Esto funciona ya que C le gana a D.

Realizando el paso a paso, suponiendo un orden de C, D, B, A, quedaría:

Paso 1, C vs D:

- Jurado 1: C
- Jurado 2: C
- Jurado 3: D

Gana C

Paso 2: C vs B:

- Jurado 1: B
- Jurado 2: C
- Jurado 3: B

Gana B

Paso 3: B vs A:

- Jurado 1: B
- Jurado 2: A
- Jurado 3: A

Gana A

Por lo tanto, el ganador sería A.

¿Cuál propiedad deseable de los sistemas de votación no se está cumpliendo si, efectivamente, ganara A?

Este sistema de votación no está cumpliendo la propiedad de ser pareto-eficiente. Todos prefieren a D antes que a A, sin embargo gana A.

Ejercicio 2

1. Considerando el modelo de cascadas de información visto en clase, supongamos que hay una nueva tecnología que los individuos pueden optar por aceptar o rechazar. Supongamos que cada uno que acepta la tecnología recibe una ganancia positiva o negativa (sin conocerla a priori). Estos valores son aleatorios para cada nodo, y si la tecnología es *"Buena"*, entonces el promedio será positivo, y si la tecnología es *"Mala"* el promedio será negativo (esta información es conocida por los individuos). Quienes rechacen la tecnología reciben ganancia 0. En este modelo, cuando a un individuo le toca elegir si acepta o rechaza la nueva tecnología, recibe la información de las ganancias de todos los que vinieron antes.

    A. Supongamos que esta nueva tecnología es, en realidad, *"Mala"*. ¿Cómo afecta esta nueva información (qué ganancia tuvo cada uno de los que vinieron antes) a la potencial formación de una cascada para que persista la nueva tecnología? (No es necesario dar una demostración, simplemente argumentar)

    B. Supongamos que esta nueva tecnología es, en realidad, *"Buena"*. ¿Puede surgir una cascada de rechazo de esta nueva tecnología?

Cascadas

Se llama **contagio** a algo que se extiende por la red.

Una **cascada de información** es un fenómeno en el que un número de personas toman la misma decisión de una manera secuencial.

En este caso hay una tecnología que se puede aceptar o rechazar. Cada individuo tiene una ganancia aleatoria por aceptar la tecnología o 0 por rechazarla. La media de la ganancia es positiva o negativa dependiendo si la teconología es "buena" o "mala".

Cuando a un individuo le toca elegir si aceptar o rechazar, recibe la información de las ganancias de los que vinieron antes.

Caso Tecnología "Mala"

¿Cómo afecta esta nueva información (qué ganancia tuvo cada uno de los que vinieron antes) a la potencial formación de una cascada para que persista la nueva tecnología?

Como cada individuo acepta o rechaza según la suma de ganancias que tuvieron los anteriores, una vez que un individuo rechace, se formará una cascada de rechazos, ya que la suma de ganancias de los anteriores será siempre negativa.

Como la media de la ganancia en este caso es negativa, entonces siempre habrá un momento en el tiempo en el que la suma de las ganancias será negativa. Por lo tanto, se puede afirmar que, en este caso, siempre ocurrirá una cascada de rechazos en algún momento del tiempo.

Caso Tecnología "Buena"

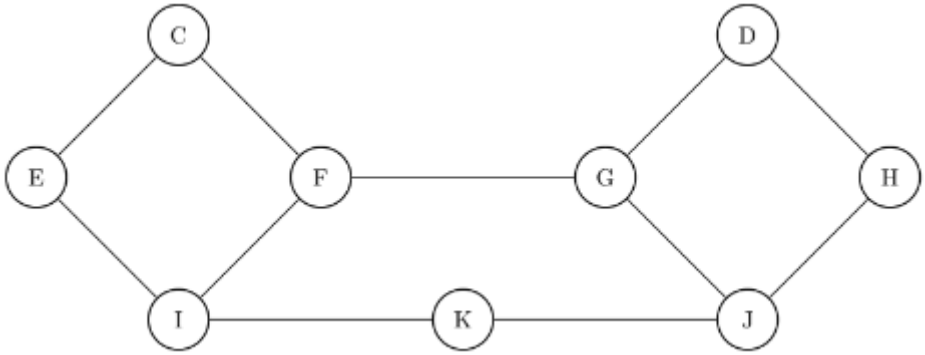
¿Puede surgir una cascada de rechazo de esta nueva tecnología?

Así como en el caso anterior podía haber una suma de ganancias positivas en algún momento del tiempo antes de encontrarse con una suma de ganancias negativas y comenzar la cascada de rechazo, en este caso puede surgir en algún momento del tiempo una suma de ganancias que de un valor negativo. Si bien la media de las ganancias es positivas, esto no impide que haya un momento en el tiempo en el que el acumulado sea negativo, causando así una cascada de rechazo.

Por lo tanto, sí, es posible que surja una cascada de rechazo aún siendo una tecnología "buena".

Ejercicio 3

1. Considerá la siguiente red, suponiendo que todos los nodos tienen inicialmente un comportamiento B. Cada nodo puede cambiar al comportamiento A si al menos la mitad de sus vecinos tiene dicho comportamiento.

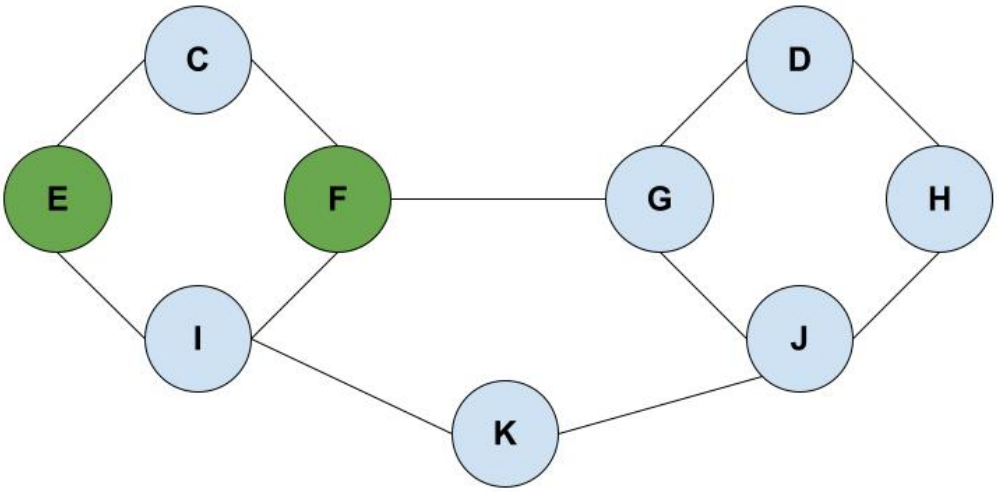


   A. Supongamos que los nodos E y F son *early adopters* del comportamiento A. Si los demás nodos siguen la regla del umbral (threshold) para adherir a este nuevo comportamiento, ¿qué nodos implementarían el comportamiento A?
   B. Explicar a qué se debe que el comportamiento A no se propaga a través de toda la red en el escenario del punto (a). ¿Qué característica de la red lo impide? (responder a esta pregunta no apuntando a nodos particulares sino a presencias de ciertas características) ¿Dónde más tendría que haber otro *early adopter* de A sí o sí para que el comportamiento se propague a través de toda la red?

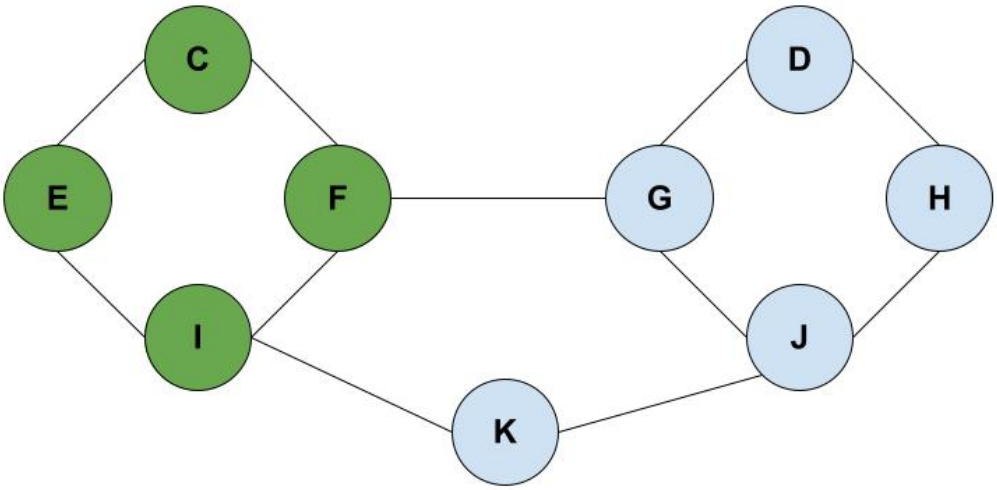
Punto A

Toda la red tiene un comportamiento B. Cada nodo cambia a A si al menos la mitad de sus vecinos tiene dicho comportamiento.

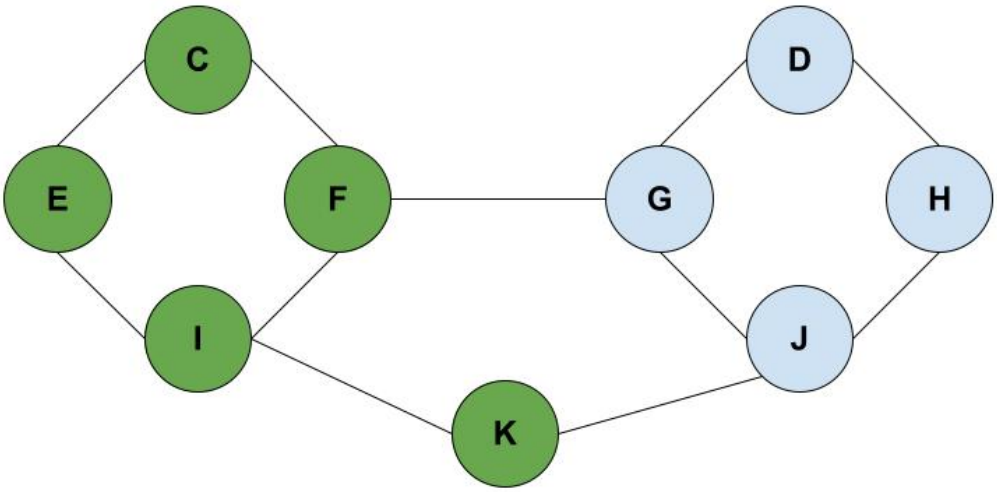
E y F son early adopters del comportamiento A. Por lo tanto, siendo los nodos azules aquellos con comportamiento A y los verdes con comportamiento B, la red queda de la siguiente manera:



Se puede ver que C tiene ambos vecinos con comportamiento B, por lo tanto pasa a tener comportamiento B. Lo mismo ocurre con el nodo I, que tiene 2/3 de los vecinos con comportamiento B. G sigue con comportamiento A ya que tiene solo 1/3 de los vecinos con comportamiento B.



Ahora se puede observar que K tiene 1/2 de los vecinos con comportamiento B. Por lo tanto, cambiará a comportamiento B. G y el resto de los nodos continuarán con comportamiento A.



Este es el estado final de la red, ya que no hay ningún otro nodo con al menos la mitad de los nodos con comportamiento B.

Punto B

A qué se debe que el comportamiento A no se propaga a través de toda la red en el escenario del punto (a). ¿Qué característica de la red lo impide? ¿Dónde más tendría que haber otro early adopter de A sí o sí para que el comportamiento se propague a través de toda la red?

Este comportamiento se debe a que hay distintas comunidades en la red. En este caso hay dos comunidades: (E, C, F, I) y (G, H, J, D). C, E, D y H tienen dos aristas, mientras que F, I, G y J tienen tres aristas. Por lo tanto, para que en una comunidad se propague el comportamiento, deben tener dos early adopters y, al menos uno de esos dos, debe ser un nodo con tres aristas. Por ejemplo, en la primera comunidad:

- E y F ya fue analizado anteriormente.
- I y F, en el primer paso contagiarían a E y C.
- F y C, en el primer paso contagiarían a E y en el segundo a I.
- I y E, en el primer paso contagiarían a C y en el segundo a F.

Pero si se eligiesen a E y C, no lograrían contagiar a I y F, ya que ambos quedan con 1/3 de aristas de comportamiento B.

Es suficiente con que una comunidad cumpla con tener un early adopter de tres aristas y uno de dos y que la otra comunidad tenga al menos un early adopter. Este early adopter de la otra comunidad puede ser cualquiera de los cuatro. Por ejemplo, suponiendo el caso anteriormente analizado, donde quedaron E, C, I, F y K con el comportamiento B, si el early adopter fuera:

- G, en el primer paso se contagiaría D y J y en el segundo se contagiaría H.
- D, en el primer paso se contagiaría G y H y en el segundo se contagiaría J.
- H, en el primer paso se contagiaría D y J y en el segundo se contagiaría G.
- J, en el primer paso se contagiaría G y H y en el segundo se contagiaría D.

Ejercicio 4

1. Tenemos dos grafos no dirigidos $G1$ y $G2$, con la misma cantidad de vértices y aristas. $G1$ es un grafo aleatorio de Erdös-Rényi, mientras $G2$ es un grafo que cumple la ley de potencias en la distribución de los grados. Consideremos un virus que comienza en un único vértice aleatorio y se expande según el modelo **SIR**.
   A. ¿En cuál grafo es más probable que ocurra una epidemia (i.e. se infecte al menos un 30% de la red)? Justificar brevemente la respuesta.
   B. Supongamos que en vez de comenzar en un vértice aleatorio, la epidemia comenzara en el vértice de mayor grado de $G1$ y $G2$, respectivamente. ¿En cuál de los grafos es más probable que ocurra una epidemia? Justificar brevemente la respuesta.
   C. ¿Cómo afecta la existencia (o no existencia) de comunidades en la expansión de la epidemia?

Para responder estas preguntas, se les recomienda realizar simulaciones. Pueden agregar todo tipo de resultados obtenidos para justificar sus respuestas.

Modelo SIR

Hay dos parámetros que definen a la propagación:

- La tasa de nacimiento (β) → probabilidad de infectar a un vecino
- La tasa de mortalidad del virus ($\boldsymbol{\delta}$) → probabilidad que un nodo se recupere

En particular, en el modelo SIR existen tres estados:

- Susceptible: puede ser infectado.
- Infectado: está infectado. Puede contagiar con una probabilidad β y recuperarse con una probabilidad $\boldsymbol{\delta}$.
- Recuperado: está recuperado de la infección y no se volverá a infectar.

In [1]:
```
# IMPORTS
import networkx as nx
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import socialnetworksutils.metricas as snu_metrics
import socialnetworksutils.modelos as snu_models
import socialnetworksutils.homofilia as snu_homofilia
import socialnetworksutils.embeddings as snu_embeddings
```

In [2]:
```
from random import randint, random

def infectRandom(graph, susceptibles, infected):
    toBeInfected = randint(0, len(graph)-1)
    susceptibles.remove(toBeInfected)
    infected.append(toBeInfected)

def infectHighDegree(graph, susceptibles, infected):
    toBeInfected = sorted(graph.degree, key=lambda x: x[1], reverse=True)[0][0]
    susceptibles.remove(toBeInfected)
    infected.append(toBeInfected)

def simulateInfection(infectedNode, graph, susceptibles, infected, beta):
    for neighbor in graph.neighbors(infectedNode):
```

```
        if neighbor in susceptibles:
            infect = random() <= beta
            if infect:
                susceptibles.remove(neighbor)
                infected.append(neighbor)

def simulateRecover(infectedNode, infected, recovered, gamma):
    recover = random() <= gamma
    if recover:
        infected.remove(infectedNode)
        recovered.append(infectedNode)

def simulate(graph, beta, gamma, highDegree):
    susceptibles = list(graph.nodes)[::]
    infected = []
    recovered = []
    if highDegree: infectHighDegree(graph, susceptibles, infected)
    else: infectRandom(graph, susceptibles, infected)

    while len(infected):
        for infectedNode in infected[::]: # Hago copia para recorrer solo infectado
            simulateInfection(infectedNode, graph, susceptibles, infected, beta)
            simulateRecover(infectedNode, infected, recovered, gamma)

    return susceptibles, infected, recovered

def getSimulationResultFor(graph, iters, beta, gamma, highDegree=False):
    epidemics = 0
    infectedTotal = 0
    for i in range(iters):
        sus, inf, rec = simulate(graph, beta, gamma, highDegree)
        infectedTotal += len(rec)
        if (len(rec) / len(graph)) > 0.3:
            epidemics += 1
    return epidemics, (infectedTotal / (iters * len(graph)))
```

Simulación con distintos Grafos

In [3]:
```
# Es dificil estimar buenos valores de K, pero elijo estos números dado que números
# daban porcentajes de epidemias demasiado altos y es dificil de analizar.
N = [300, 1000, 5000]
K = [10, 20, 30]
p = [K[i] / N[i] for i in range(3)]
seed = 17971

smallErdosRenyi = nx.erdos_renyi_graph(N[0], p[0], seed=seed)
mediumErdosRenyi = nx.erdos_renyi_graph(N[1], p[1], seed=seed)
bigErdosRenyi = nx.erdos_renyi_graph(N[2], p[2], seed=seed)

smallPrefAtt = snu_models.preferential_attachment(False, 2, N[0], K[0]/2)
mediumPrefAtt = snu_models.preferential_attachment(False, 2, N[1], K[1]/2)
bigPrefAtt = snu_models.preferential_attachment(False, 2, N[2], K[2]/2)
```

Análisis tomando un nodo aleatorio

Se analizarán las simulaciones en los modelos comenzando con un nodo aleatorio.

In [4]:
```
nodos = len(smallErdosRenyi)
aristas = len(smallErdosRenyi.edges)
grado = (aristas * 2) / nodos
```

```
print(f"Erdos-Renyi con {nodos} nodos, {aristas} aristas y grado promedio de {grad

iters = 100
betas = [0.02, 0.2] # 2% y 20%
gammas = [0.05, 0.3] # 5% y 30%

print("Beta | Gamma | Epidemias | Infectados")
for beta in betas:
    for gamma in gammas:
        epidemics, infected_average = getSimulationResultFor(smallErdosRenyi, iter
        print(f"{beta:.2f} | {gamma:.2f}  |  {epidemics}/{iters}   | {infected_aver
```

```
Erdos-Renyi con 300 nodos, 1558 aristas y grado promedio de 10.386666666666667.
Beta | Gamma | Epidemias | Infectados
0.02 | 0.05  |  82/100   | 77.52333333333333%
0.02 | 0.30  |  73/100   | 68.89999999999999%
0.20 | 0.05  |  79/100   | 74.41666666666666%
0.20 | 0.30  |  71/100   | 67.17999999999999%
```

In [5]:
```
nodos = len(mediumErdosRenyi)
aristas = len(mediumErdosRenyi.edges)
grado = (aristas * 2) / nodos

print(f"Erdos-Renyi con {nodos} nodos, {aristas} aristas y grado promedio de {grad

iters = 50
betas = [0.02, 0.2] # 2% y 20%
gammas = [0.05, 0.3] # 5% y 30%

print("Beta | Gamma | Epidemias | Infectados")
for beta in betas:
    for gamma in gammas:
        epidemics, infected_average = getSimulationResultFor(mediumErdosRenyi, iter
        print(f"{beta:.2f} | {gamma:.2f}  |   {epidemics}/{iters}   | {infected_av
```

```
Erdos-Renyi con 1000 nodos, 10091 aristas y grado promedio de 20.182.
Beta | Gamma | Epidemias | Infectados
0.02 | 0.05  |   45/50   | 89.78399999999999%
0.02 | 0.30  |   49/50   | 97.706%
0.20 | 0.05  |   44/50   | 87.746%
0.20 | 0.30  |   47/50   | 93.72200000000001%
```

In [6]:
```
nodos = len(bigErdosRenyi)
aristas = len(bigErdosRenyi.edges)
grado = (aristas * 2) / nodos

print(f"Erdos-Renyi con {nodos} nodos, {aristas} aristas y grado promedio de {grad

iters = 10
betas = [0.02, 0.2] # 2% y 20%
gammas = [0.05, 0.3] # 5% y 30%

print("Beta | Gamma | Epidemias | Infectados")
for beta in betas:
    for gamma in gammas:
        epidemics, infected_average = getSimulationResultFor(bigErdosRenyi, iters,
        print(f"{beta:.2f} | {gamma:.2f}  |   {epidemics}/{iters}   | {infected_av
```

```
Erdos-Renyi con 5000 nodos, 75082 aristas y grado promedio de 30.0328.
Beta | Gamma | Epidemias | Infectados
0.02 | 0.05  |   9/10    | 89.984%
0.02 | 0.30  |   10/10    | 99.978%
0.20 | 0.05  |   10/10    | 99.99%
0.20 | 0.30  |   9/10    | 89.986%
```

In [7]:
```
nodos = len(smallPrefAtt)
aristas = len(smallPrefAtt.edges)
grado = (aristas * 2) / nodos

print(f"Preferential Attachment con {nodos} nodos, {aristas} aristas y grado promed

iters = 100
betas = [0.02, 0.2] # 2% y 20%
gammas = [0.05, 0.3] # 5% y 30%

print("Beta | Gamma | Epidemias | Infectados")
for beta in betas:
    for gamma in gammas:
        epidemics, infected_average = getSimulationResultFor(smallPrefAtt, iters, 
        print(f"{beta:.2f} | {gamma:.2f}  |  {epidemics}/{iters}   | {infected_aver
```

```
Preferential Attachment con 300 nodos, 1484 aristas y grado promedio de 9.89333333
3333333.
Beta | Gamma | Epidemias | Infectados
0.02 | 0.05  |  76/100   | 72.04333333333334%
0.02 | 0.30  |  70/100   | 66.21000000000001%
0.20 | 0.05  |  75/100   | 71.07333333333334%
0.20 | 0.30  |  78/100   | 73.9%
```

In [8]:
```
nodos = len(mediumPrefAtt)
aristas = len(mediumPrefAtt.edges)
grado = (aristas * 2) / nodos

print(f"Preferential Attachment con {nodos} nodos, {aristas} aristas y grado promed

iters = 50
betas = [0.02, 0.2] # 2% y 20%
gammas = [0.05, 0.3] # 5% y 30%

print("Beta | Gamma | Epidemias | Infectados")
for beta in betas:
    for gamma in gammas:
        epidemics, infected_average = getSimulationResultFor(mediumPrefAtt, iters,
        print(f"{beta:.2f} | {gamma:.2f}  |  {epidemics}/{iters}   | {infected_aver
```

```
Preferential Attachment con 1000 nodos, 9908 aristas y grado promedio de 19.816.
Beta | Gamma | Epidemias | Infectados
0.02 | 0.05  |  48/50   | 95.826%
0.02 | 0.30  |  46/50   | 91.83%
0.20 | 0.05  |  45/50   | 89.86%
0.20 | 0.30  |  44/50   | 87.85%
```

In [9]:
```
nodos = len(bigPrefAtt)
aristas = len(bigPrefAtt.edges)
grado = (aristas * 2) / nodos

print(f"Preferential Attachment con {nodos} nodos, {aristas} aristas y grado promed

iters = 10
betas = [0.02, 0.2] # 2% y 20%
gammas = [0.05, 0.3] # 5% y 30%

print("Beta | Gamma | Epidemias | Infectados")
for beta in betas:
    for gamma in gammas:
        epidemics, infected_average = getSimulationResultFor(bigPrefAtt, iters, be
        print(f"{beta:.2f} | {gamma:.2f}  |  {epidemics}/{iters}   | {infected_aver
```

```
Preferential Attachment con 5000 nodos, 74789 aristas y grado promedio de 29.9156.
Beta | Gamma | Epidemias | Infectados
0.02 | 0.05  |  10/10   | 99.99%
0.02 | 0.30  |  10/10   | 99.988%
0.20 | 0.05  |  9/10   | 89.992%
0.20 | 0.30  |  10/10   | 99.992%
```

¿En cuál grafo es más probable que ocurra una epidemia (i.e. se infecte al menos un 30% de la red)? Justificar brevemente la respuesta.

Sería razonable que, al elegir un nodo al azar y realizar el experimento en múltiples ocaciones, haya una mayor probabilidad de epidemias en el grafo generado por erdos renyi. Ya que, si tiene un grado promedio alto, la mayor parte de los nodos de este grafo tendrán un grado promedio alto. En cambio, con un modelo generado por preferential attachment, uno esperaría que haya nodos con muy alto grado, y otros con menor grado, siguiendo una distribución exponencial. Por lo que la probabilidad de tomar un nodo con bajo grado es mayor.

Se corrieron 3 escenarios para cada modelo, donde se utilizaron grafos de distintos tamaños y con una cantidad distinta de iteraciones (ya que en grafos grandes demora mucho la simulación).

- Erdos-Renyi con 300 nodos, 1558 aristas y grado promedio de 10.386666666666667.

Beta	Gamma	Epidemias	Infectados
0.02	0.05	82/100	77.53%
0.02	0.30	73/100	68.90%
0.20	0.05	79/100	74.42%
0.20	0.30	71/100	67.18%

- Erdos-Renyi con 1000 nodos, 10091 aristas y grado promedio de 20.182.

Beta	Gamma	Epidemias	Infectados
0.02	0.05	45/50	89.78%
0.02	0.30	49/50	97.71%
0.20	0.05	44/50	87.75%
0.20	0.30	47/50	93.72%

- Erdos-Renyi con 5000 nodos, 75082 aristas y grado promedio de 30.0328.

Beta	Gamma	Epidemias	Infectados
0.02	0.05	9/10	89.98%
0.02	0.30	10/10	99.98%
0.20	0.05	10/10	99.99%
0.20	0.30	9/10	89.99%

- Preferential Attachment con 300 nodos, 1484 aristas y grado promedio de 9.893333333333333.

Beta	Gamma	Epidemias	Infectados
0.02	0.05	76/100	72.04%
0.02	0.30	70/100	66.21%
0.20	0.05	75/100	71.07%
0.20	0.30	78/100	73.90%

- Preferential Attachment con 1000 nodos, 9908 aristas y grado promedio de 19.816.

Beta	Gamma	Epidemias	Infectados
0.02	0.05	48/50	95.83%
0.02	0.30	46/50	91.83%
0.20	0.05	45/50	89.86%
0.20	0.30	44/50	87.85%

- Preferential Attachment con 5000 nodos, 74789 aristas y grado promedio de 29.9156.

Beta	Gamma	Epidemias	Infectados
0.02	0.05	10/10	99.99%
0.02	0.30	10/10	99.99%
0.20	0.05	9/10	89.99%
0.20	0.30	10/10	99.99%

En general se pueden observar resultados muy similares para ambos modelos, con un gran de porcentaje de epimedias e infectados totales tanto para el grafo de 1000 nodos como para el de 5000. También se observa un menor porcentaje de epidemias e infectados totales en los grafos más pequeños, lo que probablemente se deba a que estos grafos cuentan con una cantidad de aristas por nodo menor, lo que aumenta la probabilidad de que un nodo no contagie a sus vecinos y, por lo tanto, no se propague.

También se nota una variación en los resultados al modificar los valores de beta y gamma, pero sin ninguna diferencia notable entre los modelos.

Por lo tanto, con esta configuración de grafos y los valores elegidos de beta y gamma, no se puede concluir con certeza qué modelo es más probable en el que ocurra una epidemia.

Análisis tomando vértice de mayor grado

Se analizarán las simulaciones en los modelos comenzando con el nodo de mayor grado.

In [10]:
```
nodos = len(smallErdosRenyi)
aristas = len(smallErdosRenyi.edges)
grado = (aristas * 2) / nodos

print(f"Erdos-Renyi con {nodos} nodos, {aristas} aristas y grado promedio de {grad

iters = 100
betas = [0.02, 0.2] # 2% y 20%
gammas = [0.05, 0.3] # 5% y 30%
```

```
print("Beta | Gamma | Epidemias | Infectados")
for beta in betas:
    for gamma in gammas:
        epidemics, infected_average = getSimulationResultFor(smallErdosRenyi, iters
        print(f"{beta:.2f} | {gamma:.2f}  |  {epidemics}/{iters}   | {infected_aver
```

```
Erdos-Renyi con 300 nodos, 1558 aristas y grado promedio de 10.386666666666667.
Beta | Gamma | Epidemias | Infectados
0.02 | 0.05  |  90/100   | 84.63333333333334%
0.02 | 0.30  |  89/100   | 84.0%
0.20 | 0.05  |  89/100   | 83.86333333333333%
0.20 | 0.30  |  93/100   | 87.31%
```

In [11]:
```
nodos = len(mediumErdosRenyi)
aristas = len(mediumErdosRenyi.edges)
grado = (aristas * 2) / nodos

print(f"Erdos-Renyi con {nodos} nodos, {aristas} aristas y grado promedio de {grado

iters = 50
betas = [0.02, 0.2] # 2% y 20%
gammas = [0.05, 0.3] # 5% y 30%

print("Beta | Gamma | Epidemias | Infectados")
for beta in betas:
    for gamma in gammas:
        epidemics, infected_average = getSimulationResultFor(mediumErdosRenyi, iter
        print(f"{beta:.2f} | {gamma:.2f}  |   {epidemics}/{iters}   | {infected_ave
```

```
Erdos-Renyi con 1000 nodos, 10091 aristas y grado promedio de 20.182.
Beta | Gamma | Epidemias | Infectados
0.02 | 0.05  |   48/50   | 95.746%
0.02 | 0.30  |   47/50   | 93.73%
0.20 | 0.05  |   44/50   | 87.764%
0.20 | 0.30  |   45/50   | 89.762%
```

In [12]:
```
nodos = len(bigErdosRenyi)
aristas = len(bigErdosRenyi.edges)
grado = (aristas * 2) / nodos

print(f"Erdos-Renyi con {nodos} nodos, {aristas} aristas y grado promedio de {grado

iters = 10
betas = [0.02, 0.2] # 2% y 20%
gammas = [0.05, 0.3] # 5% y 30%

print("Beta | Gamma | Epidemias | Infectados")
for beta in betas:
    for gamma in gammas:
        epidemics, infected_average = getSimulationResultFor(bigErdosRenyi, iters,
        print(f"{beta:.2f} | {gamma:.2f}  |   {epidemics}/{iters}   | {infected_ave
```

```
Erdos-Renyi con 5000 nodos, 75082 aristas y grado promedio de 30.0328.
Beta | Gamma | Epidemias | Infectados
0.02 | 0.05  |   10/10   | 99.984%
0.02 | 0.30  |   10/10   | 99.992%
0.20 | 0.05  |   9/10    | 89.988%
0.20 | 0.30  |   10/10   | 99.982%
```

In [13]:
```
nodos = len(smallPrefAtt)
aristas = len(smallPrefAtt.edges)
grado = (aristas * 2) / nodos

print(f"Preferential Attachment con {nodos} nodos, {aristas} aristas y grado promed
```

```
iters = 100
betas = [0.02, 0.2] # 2% y 20%
gammas = [0.05, 0.3] # 5% y 30%

print("Beta | Gamma | Epidemias | Infectados")
for beta in betas:
    for gamma in gammas:
        epidemics, infected_average = getSimulationResultFor(smallPrefAtt, iters, 
        print(f"{beta:.2f} | {gamma:.2f}  |  {epidemics}/{iters}   | {infected_aver
```

```
Preferential Attachment con 300 nodos, 1484 aristas y grado promedio de 9.89333333
3333333.
Beta | Gamma | Epidemias | Infectados
0.02 | 0.05  |  86/100   | 81.34333333333333%
0.02 | 0.30  |  86/100   | 81.14666666666666%
0.20 | 0.05  |  79/100   | 75.07000000000001%
0.20 | 0.30  |  89/100   | 84.35666666666667%
```

In [14]:
```
nodos = len(mediumPrefAtt)
aristas = len(mediumPrefAtt.edges)
grado = (aristas * 2) / nodos

print(f"Preferential Attachment con {nodos} nodos, {aristas} aristas y grado promed

iters = 50
betas = [0.02, 0.2] # 2% y 20%
gammas = [0.05, 0.3] # 5% y 30%

print("Beta | Gamma | Epidemias | Infectados")
for beta in betas:
    for gamma in gammas:
        epidemics, infected_average = getSimulationResultFor(mediumPrefAtt, iters,
        print(f"{beta:.2f} | {gamma:.2f}  |  {epidemics}/{iters}   | {infected_aver
```

```
Preferential Attachment con 1000 nodos, 9908 aristas y grado promedio de 19.816.
Beta | Gamma | Epidemias | Infectados
0.02 | 0.05  |  47/50   | 93.84599999999999%
0.02 | 0.30  |  47/50   | 93.83399999999999%
0.20 | 0.05  |  46/50   | 91.80199999999999%
0.20 | 0.30  |  45/50   | 89.83%
```

In [15]:
```
nodos = len(bigPrefAtt)
aristas = len(bigPrefAtt.edges)
grado = (aristas * 2) / nodos

print(f"Preferential Attachment con {nodos} nodos, {aristas} aristas y grado promed

iters = 10
betas = [0.02, 0.2] # 2% y 20%
gammas = [0.05, 0.3] # 5% y 30%

print("Beta | Gamma | Epidemias | Infectados")
for beta in betas:
    for gamma in gammas:
        epidemics, infected_average = getSimulationResultFor(bigPrefAtt, iters, be
        print(f"{beta:.2f} | {gamma:.2f}  |  {epidemics}/{iters}   | {infected_aver
```

```
Preferential Attachment con 5000 nodos, 74789 aristas y grado promedio de 29.9156.
Beta | Gamma | Epidemias | Infectados
0.02 | 0.05  |  10/10   | 99.984%
0.02 | 0.30  |  9/10   | 89.988%
0.20 | 0.05  |  10/10   | 99.99%
0.20 | 0.30  |  9/10   | 89.996%
```

Supongamos que en vez de comenzar en un vértice aleatorio, la epidemia comenzara en el vértice de mayor grado de G1 y G2, respectivamente. ¿En cuál de los grafos es más probable que ocurra una epidemia? Justificar brevemente la respuesta.

En este caso, a diferencia del anterior, lo esperable sería que la pandemia tenga mayor probabilidad de ocurrir en un grafo generado con un modelo de potencias, ya que el grado máximo probablemente sea mucho mayor. Igualmente, los casos en los que ocurre una pandemia también deberían aumentar para el modelo de erdos renyi con respecto al anterior.

- Erdos-Renyi con 300 nodos, 1558 aristas y grado promedio de 10.386666666666667.

Beta	Gamma	Epidemias	Infectados
0.02	0.05	90/100	84.63%
0.02	0.30	89/100	84.00%
0.20	0.05	89/100	83.86%
0.20	0.30	93/100	87.31%

- Erdos-Renyi con 1000 nodos, 10091 aristas y grado promedio de 20.182.

Beta	Gamma	Epidemias	Infectados
0.02	0.05	48/50	95.75%
0.02	0.30	47/50	93.73%
0.20	0.05	44/50	87.76%
0.20	0.30	45/50	89.76%

- Erdos-Renyi con 5000 nodos, 75082 aristas y grado promedio de 30.0328.

Beta	Gamma	Epidemias	Infectados
0.02	0.05	10/10	99.98%
0.02	0.30	10/10	99.99%
0.20	0.05	9/10	89.99%
0.20	0.30	10/10	99.98%

- Preferential Attachment con 300 nodos, 1484 aristas y grado promedio de 9.893333333333333.

Beta	Gamma	Epidemias	Infectados
0.02	0.05	86/100	81.34%
0.02	0.30	86/100	81.15%
0.20	0.05	79/100	75.07%
0.20	0.30	89/100	84.36%

- Preferential Attachment con 1000 nodos, 9908 aristas y grado promedio de 19.816.

Beta	Gamma	Epidemias	Infectados
0.02	0.05	47/50	93.85%
0.02	0.30	47/50	93.83%
0.20	0.05	46/50	91.80%
0.20	0.30	45/50	89.83%

- Preferential Attachment con 5000 nodos, 74789 aristas y grado promedio de 29.9156.

Beta	Gamma	Epidemias	Infectados
0.02	0.05	10/10	99.98%
0.02	0.30	9/10	89.99%
0.20	0.05	10/10	99.99%
0.20	0.30	9/10	90.00%

Podemos ver que hay una gran diferencia en la cantidad de pandemias que ocurrieron para el caso del grafo de 300 nodos. Al comenzar por el nodo de mayor grado, ocurrieron más pandemias que en el caso anterior. Para los grafos mediano y grande, la diferencia no es notoria. Probablemente haya que analizar con distintos valores para beta, gamma, el grado promedio y la cantidad de nodos para llegar a una conclusión.

Si bien hay una diferencia con la ejecución de la simulación tomando un nodo aleatorio, no se ve una clara diferencia entre modelos, por lo que no se puede llegar a una conclusión certera.

Análisis de existencia de comunidades

¿Cómo afecta la existencia (o no existencia) de comunidades en la expansión de la epidemia?

Dentro de comunidades, la probabilidad de que se propague una infección es mayor. Ya que dentro de una comunidad los nodos se encuentran conectados con mayor densidad. Sin embargo, es mucho menor la probabilidad de infección hacia fuera de la comunidad. Esto va a depender de características como las tasas de nacimiento y mortalidad del virus, y cuán conectada está una comunidad con el exterior de esta.

Ejercicio 5

1. Aplicar el Algoritmo REV2 al siguiente set de datos de reviews de productos de Amazon, para detectar potenciales usuarios maliciosos y otros ciertamente honestos. Por simplificación (y unificación de criterios), considerar $\gamma 1 = \gamma 2 = 0.5$. Obtener aquellos usuarios cuya *justicia* (*fairness*) es menor o igual a 0.2 (son maliciosos) y tienen al menos 5 reviews, así como la proporción de nodos que son extremadamente justos: aquellos con *justicia* mayor o igual a 0.9, y con al menos 10 reviews (aristas de salida).

In [16]:
```
import pandas as pd

df = pd.read_csv("ratings_Electronics.csv", names=["User ID", "Product ID", "Rating
df = df[["User ID", "Product ID", "Rating"]]
```

```
df['Rating'] = (df['Rating'] - 3) / 2
df
```

Out[16]:

	User ID	Product ID	Rating
0	AKM1MP6P0OYPR	0132793040	1.0
1	A2CX7LUOHB2NDG	0321732944	1.0
2	A2NWSAGRHCP8N5	0439886341	-1.0
3	A2WNBOD3WNDNKT	0439886341	0.0
4	A1GI0U4ZRJA8WN	0439886341	-1.0
...	...	...	...
7824477	A2YZI3C9MOHC0L	BT008UKTMW	1.0
7824478	A322MDK0M89RHN	BT008UKTMW	1.0
7824479	A1MH90R0ADMIK0	BT008UKTMW	0.5
7824480	A10M2KEFPEQDHN	BT008UKTMW	0.5
7824481	A2G81TMIOIDEQQ	BT008V9J9U	1.0

7824482 rows × 3 columns

In [17]:
```
print(f"Hay {len(pd.unique(df['User ID']))} User ID únicos")
print(f"Hay {len(pd.unique(df['Product ID']))} Product ID únicos")
```

```
Hay 4201696 User ID únicos
Hay 476002 Product ID únicos
```

Descripción del dataset

Reviews de productos de Amazon.

- **User ID**: Identificador único de cada usuario.
- **Product ID** : Identificador único de cada producto.
- **Rating**: Rating del correspondiente producto por el correspondiente usuario.
- **Timestamp**: Fecha del rating.

Hay 7.824.482 reviews, 4.201.696 usuarios y 476.002 productos.

Algoritmo REV2

- Cada usuario tiene una cierta noción de justicia (fairness) → $F(u) \in [0, 1]$
- Cada producto tiene un "valor" → $G(p) \in [-1, 1]$
- Los ratings tienen una fiabilidad → $R(u, p) \in [0, 1]$

Inicializamos con:

$$F(u) = 1; G(p) = 1; R(u, p) = 1$$

Los valores se calculan con clasificación iterativa de la siguiente manera:

$$F(u) = \frac{\sum R(u,p)}{d_{out}(u)}$$

$$G(p) = \frac{\sum R(u,p) * score(u,p)}{d_{in}(p)}$$

$$R(u,p) = \frac{1}{\gamma_1 + \gamma_2}\left(\gamma_1 F(u) + \gamma_2\left(1 - \frac{|score(u,p) - G(p)|}{2}\right)\right)$$

En este caso $\gamma_1 = \gamma_2 = 0.5$

Para la convergencia, el error cuadrático medio se define como:

$$EMC = \frac{1}{n}\sum_{i}^{n}(\hat{Y}_i - Y_i)^2$$

In [18]:

```
from math import inf
import time

def getError(F, G, R, F_old, G_old, R_old):
    # Suma de errores cuadraticos medios
    errorF = 0
    errorG = 0
    errorR = 0

    for (u, p) in R:
            errorF += (F[u] - F_old[u]) ** 2
            errorG += (G[p] - G_old[p]) ** 2
            errorR += (R[(u,p)] - R_old[(u,p)]) ** 2

    errorF /= len(F)
    errorG /= len(G)
    errorR /= len(R)

    return errorF + errorG + errorR

def REV2():
    start = time.time()
    print("Inicializando score...")
    score = {(row["User ID"], row["Product ID"]): row["Rating"] for index, row in d
    print(f"{time.time() - start} segs\nInicializando F...")
    F = {u: 1 for u in list(df["User ID"])}
    print(f"{time.time() - start} segs\nInicializando G...")
    G = {p: 1 for p in list(df["Product ID"])}
    print(f"{time.time() - start} segs\nInicializando R...")
    R = {up: 1 for up in score}

    print(f"{time.time() - start} segs\nInicializando reviews...")
    reviews_u = {}
    reviews_p = {}
    for (u, p) in score:
        reviews_u[u] = reviews_u.get(u, 0) + 1
        reviews_p[p] = reviews_p.get(p, 0) + 1

    gamma1 = gamma2 = 0.5
    delta = 0.003
    error = inf
    it = 0

    while error > delta:
```

```
        it += 1
        print(f"{time.time() - start} segs\nIteración: {it}")
        # Creo diccionarios nuevos para poder calcular error
        F_it = {}
        G_it = {}
        R_it = {}

        first_term = 1 / (gamma1 + gamma2)
        for (u, p) in R:
            F_it[u] = F_it.get(u, 0) + R[(u, p)] # Suma acumulada de r para cada u
            G_it[p] = G_it.get(p, 0) + (R[(u, p)] * score[(u, p)]) # Suma acumulad
            # Calculo R
            second_term_1 = gamma1 * F[u]
            second_term_2 = gamma2 * (1 - (abs(score[u,p] - G[p]) / 2))
            second_term = second_term_1 + second_term_2
            R_it[(u, p)] = first_term * second_term

        for u in F_it:
            F_it[u] /= reviews_u[u] # Divido por salidas de u

        for p in G_it:
            G_it[p] /= reviews_p[p] # Divido por entradas de p

        error = getError(F_it, G_it, R_it, F, G, R)
        print(f"Error: {error}")

        F = F_it
        G = G_it
        R = R_it

    return F, G, R, reviews_u

F, G, R, reviews_u = REV2()
```

```
Inicializando score...
395.6974346637726 segs
Inicializando F...
398.77371525764465 segs
Inicializando G...
399.37759804725647 segs
Inicializando R...
401.6986231803894 segs
Inicializando reviews...
406.0593013763428 segs
Iteración: 1
Error: 5.590453362600987
440.4119882583618 segs
Iteración: 2
Error: 0.21589795618508825
477.7658324241638 segs
Iteración: 3
Error: 0.18458451660554961
516.5791988372803 segs
Iteración: 4
Error: 0.06586644575617759
555.5628082752228 segs
Iteración: 5
Error: 0.056115184432919976
594.7918992042542 segs
Iteración: 6
Error: 0.025518820566316783
633.750007390976 segs
Iteración: 7
Error: 0.023169809469757635
672.6525921821594 segs
Iteración: 8
Error: 0.010487131604067447
711.3873455524445 segs
Iteración: 9
Error: 0.010055835259295494
750.2416920661926 segs
Iteración: 10
Error: 0.0044963320009614284
789.1140599250793 segs
Iteración: 11
Error: 0.004471576572113778
828.0554280281067 segs
Iteración: 12
Error: 0.001994602468661844
```

Proporción de:

- Usuarios cuyo fairness es menor o igual a 0.2 y tienen al menos 5 reviews
- Usuarios cuyo fairness mayor o igual a 0.9, y con al menos 10 reviews (aristas de salida).

In [19]:

```python
susp = 0
fair = 0

for u in F:
    if reviews_u[u] >= 5 and F[u] <= 0.2:
        susp += 1
    if reviews_u[u] >= 10 and F[u] >= 0.9:
        fair += 1

print(f"- Hay {susp}/{len(F)} usuarios maliciosos ({susp * 100 / len(F)}%)")
print(f"- Hay {fair}/{len(F)} usuarios extremadamente justos ({fair * 100 / len(F)}
print(f"\nLa proporción contando únicamente justos y maliciosos es de:")
```

```python
print(f"- {susp}/{(fair + susp)} - {susp / (fair + susp)}% maliciosos")
print(f"- {fair}/{(fair + susp)} - {fair / (fair + susp)}% extremadamente justos")
```

```
- Hay 2/4201696 usuarios maliciosos (4.7599826355833455e-05%)
- Hay 447/4201696 usuarios extremadamente justos (0.010638561190528776%)

La proporción contando únicamente justos y maliciosos es de:
- 2/449 - 0.004454342984409799% maliciosos
- 447/449 - 0.9955456570155902% extremadamente justos
```

En total hay 2 usuarios maliciosos y 447 extremadamente justos.