Análisis de la relación entropía - robustez ante fallas en cascada para redes de Watts-Strogatz, Barabási-Albert y libres de escala con máxima entropía

J. C. Higuera Calderón* , K. N. Ramos G† , F. J. Ordóñez Araújo‡
Mecánica Estadística, Departamento de física, Universidad Nacional de Colombia, Bogotá.

30 de Julio de 2021

Resumen

A partir de los sesgos encontrados en el análisis analítico del algoritmo Preferential Attacthment se plantea ampliar los estudios de robustez ante fallas en cascada de la comunicación geodésica para redes con topologías representativas: pequeño mundo y libre de escala. Esto por medio de un algoritmo el cual validamos teórica y computacionalmente que permite muestrear el ensamble de redes con máxima entropía dada una distribución de grado promedio como ligadura. En este estudio de la robustez prestamos especial énfasis en la posibilidad de que la entropía de la distribución de algún parámetro nodal permita dar cuenta de las diferencias en robustez.

1. Introducción

Muchas redes complejas como el internet, la red de fluido eléctrico, las redes de transporte, etc, son parte esencial de nuestra sociedad moderna [1]. Esto motiva el estudio tanto de las redes como de sus propiedades de robustez.

Las redes complejas se modelan por medio de teoría de grafos y mecánica estadística [2], en el marco teórico que abordamos exponemos conceptos básicos de la teoría de redes tal como se presentan en [3]; un grafo consiste en un conjunto de nodos y de enlaces entre esos nodos [4], de tal forma que esencialmente contando enlaces y nodos de diversas maneras se accede a información sobre las propiedades de conectividad de la red, tales como el grado, intermediación, clusterización, etc.

Una de las propiedades principalmente estudiadas en las redes es la distribución de grado i.e. la probabilidad de que al sacar aleatoriamente un nodo de la red este tenga grado un K [5]. Está distribución de grado permite separar las posibles redes dentro de clases de equivalencia, las cuales nos permiten catalogar las distintas redes y sus topologías [2]. Dos de las propiedades topológicas más comúnmente encontradas en los sistemas naturales son una distribución de grado con ley de potencias y la propiedad de pequeño mundo [1], es decir que entre cualesquiera dos nodos puede haber comunicación con relativamente pocos intermediarios.

Atendiendo a la importancia del estudio de las clases de equivalencia desde la perspectiva de la distribución de grado en [2] se presenta un algoritmo que en base al principio de máxima entropía construye redes con una distribución de grado dada [6]. Este algoritmo permite construir de forma sencilla redes libres de escala, esto nos lleva naturalmente a dos cuestiones, primero la validación de que el algoritmo efectivamente es equivalente a un formalismo de máxima entropía, segundo reportar las diferencias en las propiedades de las redes libres de escala generadas por este algoritmo y las generadas por el algoritmo preferential attachment, esto último se desarrollará en el marco de la robustez ante fallas en cascada.

Este trabajo se divide en varías secciones, la introducción es la primera, en la segunda sección presentamos el marco teórico de la teoría de grafos, el principio de máxima entropía, herramientas conocidas desde la mecánica estadística y también se describen los algoritmos que fundamentan este trabajo. En la tercera sección se encuentra el estado del arte, en la cuarta parte el planteamiento del problema tentativo, en la quinta la motivación y justificación, en la sexta el objetivo general, en la séptima los objetivos específicos que guiarán el desarrollo del objetivo general, en la octava sección la metodología de cómo se desarrollaran los objetivos específicos, en la novena sección están los resultados esperados, en la décima el cronograma, en la onceava los recursos disponibles y finalmente las referencias.

*jchiguerac@unal.edu.co
†kramosg@unal.edu.co
‡fjordoneza@unal.edu.co

En el marco teórico a parte de abordar distintos modelos de red, como Preferential Attachtment para generar redes libres de escala [7] y el algoritmo de Watts-Strogatz [8], se exponen los conceptos que fundamentan los objetivos específicos, como el de robustez bajo ataques [9], la mecánica del equilibrio y no equilibrio, la entropía como herramienta para generar hipótesis nulas y los procesos estocásticos utilizados para procesos mecánico estadísticos que varían en el tiempo como la master equation 11. El marco teórico finaliza con la presentación de los algoritmos de falla en cascada[10], generación de redes con máxima entropía y una distribución de grado dada [2].

En el planteamiento del problema hablamos acerca de que en [2] se presenta un algoritmo para generar redes de máxima entropía según una distribución de grado, en [11] estudian la robustez ante fallos en cascada para redes libres de escala, estos dos artículos son muy importantes porque a partir de estas ideas expuestas planteamos tres problemáticas.

Primero, buscamos encontrar una equivalencia entre lo expuesto en [2] y lo expuesto en [12] y [6] está comparación entre la predicción analítica predicha por el ensamble canónico y el ensamble muestreado por el algoritmo se hará mediante la comparación de la matriz de adyacencia promedio muestreada con el algoritmo y la matriz de adyacencia promedio predicha por el ensamble canónico haciendo uso de las derivadas de la función de energía libre [13]. Segundo, ampliaremos lo expuesto en [9] considerando más mecanismos y objetivos de ataque [11]. Tercero, de acuerdo a [2] las redes libres de escala generadas por el algoritmo de Albert-Barabási no son representativas dentro del ensamble de todas las redes libres de escala, por esto nos proponemos evaluar esta afirmación mediante un desarrollo analítico utilizando la ecuación 11, es de esta forma que pretendemos descubrir los sesgos que se pueden obtener al solamente trabajar con la distribución Preferential Attacthment, también contrastaremos las propiedades de robustez para redes de Preferential Attacthment y redes libres de escala producidas mediante el algoritmo de máxima entropía.

En este trabajo se desarrollará analíticamente el proceso de evolución fuera del equilibrio Preferential Attacthment, con esto se hallarán los segos de las redes generadas por este proceso respecto al ensamble de redes libres de escala. Se desarrollará analíticamente el ensamble canónico imponiendo una secuencia de grado, a partir de este tratamiento analítico se calculará la matriz de adyacencia promedio para este ensamble. Se programaran los algoritmos necesarios para generar redes de Watts-Strogatz[8], barabási-albert[7] y máxima entropía con una secuencia de grado dada [2]. Usando el algoritmo para generar redes de máxima entropía, se muestreará el ensamble de redes con una secuencia de grado dada, tal que posea una distribución de grado libre de escala; a partir de estos muestreos se calculará la matriz de adyacencia promedio, la cual se comparará con la matriz de adyacencia promedio previamente calculada por medio del ensamble canónico. Finalmente se realizarán las comparaciones de robustez ante fallas en cascada, esto al mismo tiempo que se caracterizan las redes por sus entropías en base a distintos parámetros.

2. Marco Teórico

2.1. Teoría de redes complejas y conceptos básicos.

La teoría de redes complejas es una disciplina académica que hace uso de herramientas de la física estadística y la teoría de grafos con el fin de modelar sistemas complejos. Algunos ejemplos de sistemas susceptibles a ser modelados con la teoría de redes complejas son las redes sociales, por ejemplo propagación de epidemias , propagación de ideas o relaciones económicas; redes biológicas, por ejemplo redes metabólicas, de regulación de genes, ecológicas, de transducción de señales y neurales; redes artificiales como el internet o las redes del sistema eléctrico [14].

Un grafo es un conjunto de vértices y ejes, específicamente un grafo G queda totalmente especificado por una pareja ordenada $G = (V, E)$, en donde V representa el conjunto de vértices $V = \{v\}$ y $E = \{e\}$ el conjunto de ejes [4], además se dice que un grafo es simple si entre cualesquiera dos nodos hay como máximo un vértice y no existen vértices que conecten un nodo consigo mismo. En este trabajo nos enfocaremos en los grafos simples.

Una cantidad que en muchos casos puede distinguir unas redes de otras es el grado, podemos definir el grado de un grafo G denotado como $n(G)$ el cual es igual a la cardinalidad del conjunto $|V|$, es decir, el número de vértices, mientras que el tamaño de un grafo $e(G)$ viene definido por el número de ejes, la cardinalidad del conjunto $|E|$; también tenemos el grado de un nodo $v \in V$ denotado por $d(v)$ y representa la cantidad de conexiones entrantes que el vértice v tiene [4], este número cumple con ser un entero positivo y se puede conocer según el tipo de grafo, por ejemplo, para un grafo no dirigido el grado de un nodo $i-$ésimo se obtiene a través de sumar todos los elementos de la fila $i-$ésima de la matriz de adyacencia [3] que básicamente nos dice qué vértice está unido con qué otro vértice.

Una de las metodologías para acceder estadísticamente a la topología de las redes se basa en tomar una características de los nodos, por ejemplo el grado, y preguntarnos ¿cuál es la probabilidad de que al coger un

nodo aleatoriamente este tenga grado k? la respuesta a esta pregunta nos lleva a una distribución de probabilidad que se conoce como distribución de grado $P(K)$. [14].

2.2. Medidas de centralidad

En la teoría de redes encontramos unas medidas aplicables sobre cada nodo de la red conocidas como medidas de centralidad, estas nos indican la influencia que tiene un determinado nodo y según sea la medida se verá a ciertos nodos u otros como más importantes. [3]

La medida de centralidad más conocida es la centralidad de grado, la cual se basa en clasificar la importancia de un nodo v en función del número de ejes que este tenga con otros nodos k_v. [3]

$$D(v) = k_v \tag{1}$$

La medida de centralidad de eigenvectores asigna puntajes a todos los nodos en función de sus conexiones con otros nodos de alto grado, de esta forma un nodo con pocas conexiones puede llegar a considerarse importante aún si tiene un número de conexiones pequeño si las conexiones que este tiene son con nodos considerados centrales en la red. Este valor se calcula para cada nodo x_v mediante el autovalor más grande λ de la matriz de adyacencia del grado y los valores de $a_{v,t}$ los cuales son 1 o 0 según se presente una conexión entre el nodo x_v y el nodo x_t: [3]

$$x_v = \frac{1}{\lambda} \sum_{t \in G} a_{v,t} x_t \tag{2}$$

En un grafo conexo la centralidad closeness es una medida que puede ser calculada como la suma reciproca de todos los caminos más cortos entre un nodo y los otros nodos de la red, por lo tanto, entre mas central se considere un nodo, más cerca se encuentra de todos los nodos de la red en general, para obtener esta medida sobre un nodo se suman las distancias geodésicas con cada nodo sobre la red. [3]

$$C(x) = \frac{1}{\sum_y d(x,y)} \tag{3}$$

La medida de centralidad conocida como betweenness califica la importancia de un nodo en función de su intermediación en la red cuando dos nodos cualesquiera se comunican, de esta forma, un nodo que intermedie la mayoría de rutas para conectar dos nodos en la red se consideraría más central, esta medida se puede obtener para un nodo v conociendo la cantidad de caminos que hay entre dos nodos a y b y la cantidad de caminos entre estos dos nodos en los que el nodo v puede intermediar: [3]

$$\sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{4}$$

Una medida de centralidad muy relacionada con la medida de betweenness es conocida como "load centrality", esta medida es una fracción de todos los caminos geodésicos que atraviesan ese nodo, esta medida se calcula de la siguiente forma: [3]

$$\sum_{s \neq v \neq t} \sigma_{st}(v) \tag{5}$$

2.3. Tipos de redes

Para clasificar los distintos tipos de redes existentes es posible hacer uso de la distinción que otorga la distribución de grado $P(k)$ [5] debido a que esta se comporta según la topología que tiene la red, es decir el patrón de conectividad que presentan los nodos en la red. A continuación se describen las dos topologías estudiadas en este trabajo.

2.3.1. Modelo de redes libres de escala

El modelo de red compleja libre de escala tiene como característica que la distribución de grado $P(k)$ viene dada por una función de ley de potencias inversa, lo que significa que hay muchos nodos con un grado relativamente bajo y pocos nodos que tienen un grado alto en comparación al resto de nodos [3]

Se le llama red libre de escala porque el patrón que presenta la distribución de grado sigue la misma relación inversa sin importar si el sistema está compuesto por 100, 1000 o 10.000 nodos, siempre encontraremos que estos forman conexiones de tal forma que muchos nodos tienen pocas conexiones y pocos nodos tienen muchas conexiones, la distribución de grado analítica es:

$$P(k) = \frac{C}{k^\gamma} \tag{6}$$

En donde C es una constante que se toma de tal manera que se logre satisfacer la condiciones de normalidad, γ es una constante mayor que cero ($\gamma > 0$) y depende de parámetros relacionados con el grado promedio de cada nodo [3]. También es importante mencionar que esta distribución de grado adquiere importancia cuando se considera en relación a la cola en donde se encuentran nodos de grados más altos [3].

A continuación una representación gráfica de una red compleja con modelo libre de escala propio:

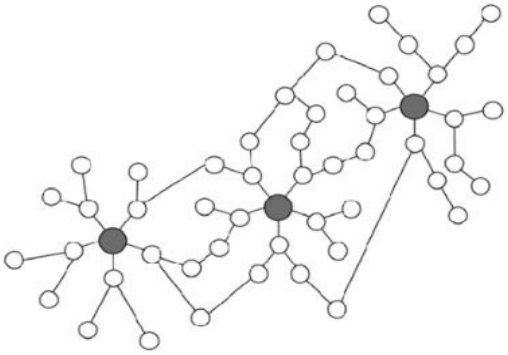

Figura 1: Representación de una red libre de escala en donde existen muchos nodos con un bajo y poco nodos con un grado alto, imagen tomada de [15].

2.3.2. Modelo preferential attachment

El modelo de preferential attachment es un modelo para generar redes libres de escala, razón por la cual presenta la misma distribución de grado (6), pero no todas las redes libres de escala siguen el patrón de preferential attachment, en [2] se menciona que existen distintos tipos de redes libres de escala, sin embargo no es claro qué propiedades son típicas de todas las redes libres de escala y cuáles son propias de un tipo restringido como lo son las generadas por el modelo de Preferential Attachment, el cual presenta una estructura denominada hub-centrica en donde muchos nodos tienen conexiones con nodos centrales y debido a esta estructura la red se hace vulnerable a ataques dirigidos [1] está propiedad de fragilidad ante ataques deliberados se ha reportado frecuentemente en la literatura, sin embargo en [2] se reporta que al muestrear el ensamble de las redes libres de escala usando un principio de máxima entropía no sé encuentra está propiedad de fragilidad ante ataques deliberados.

El modelo de Preferential Attachment comenzaría con m_o vértices unidos aleatoriamente entre ellos donde su grado $d(v_i)$ es mayor o igual a 1, a cada paso de tiempo un nuevo nodo $v \in V$ es introducido a la red de tal forma que se le da un número mínimo de enlaces m que puede tener con otros nodos, es así como obtendremos el coeficiente γ de la ley de potencias inversa presente en las redes libres de escala. [1]

$$\gamma = 2 + \frac{A}{m}$$

En donde A representa una cantidad llamada "$initial - attractiveness$" importante en la caracterización de una red con Preferential Attachment $PA(m, A)$ [16]. Cada vez que un nuevo nodo ingresa a la red se unirá a un determinado nodo dependiendo de la cantidad de enlaces que tenga cada nodo, entonces un nodo v se unirá a un nodo k_i existente en la red con la siguiente probabilidad [17]:

$$\pi(k_i) = \frac{k_i}{\sum_j k_j} \tag{7}$$

El denominador de la ecuación (7) suma todos los grados de cada nodo presente en la red en donde $k_j \in V$.

Este tipo de redes pueden ser generadas por el algoritmo de Barabási-Albert [7] donde también se menciona que diversos sistemas que tienen elementos vistos como nodos y que interactúan entre si pueden ser bien representados por este tipo de red, como por ejemplo las redes genéticas o las relacionadas con la conectividad en el mundo del internet.

Según lo observado el modelo de Barabási-Albert genera redes libres de escala las cuales tienden a formar nodos centrales que tienen grados altos en comparación al promedio; particularmente estas redes se caracterizan porque el valor del factor γ de la ecuación (6) se encuentra en el intervalo $2 \leq \gamma \leq 3$. Sin embargo, hay que recalcar que como se estudia en [2] esta no es una característica típica de este tipo de redes ya que solo se presenta para redes libres de escala generadas por el algoritmo de Preferential Attachment visto en [7], pero a un nivel general se presentaría que $\gamma > 0$.

2.3.3. Modelo de redes pequeño mundo

Las redes de mundo pequeño cumplen con que no todos los nodos son vecinos, es decir, la mayoría de los nodos no tienen un grado $d(k_i)$ muy alejado respecto a la media, sin embargo, todos los nodos en este modelo de red compleja se encuentran conectados mediante algún camino de ejes y vértices. En este modelo se define la distancia media entre dos nodos L la cual nos habla acerca de el numero de saltos intermedios que se tienen que hacer para que en promedio dos nodos cualesquiera de la red se alcancen, este valor es proporcional al número de nodos que existen en la red $|V| = N$ [8].

$$L \propto Ln(N)$$

Una propiedad presente en las redes de pequeño mundo nos dice que el grado para cada vértice en el grafo es el mismo si no se considera la contabilidad de atajos, es decir, $d(v) = c \ \forall v \in V$. Tengamos en cuenta que en el modelo de arreglo circular de las redes de pequeño mundo también pueden existir otro tipo de ejes que conectan a los vértices llamados atajos los cuales conectan dos nodos que originalmente se podrían comunicar mediante un promedio de L interacciones, entonces

si tenemos en cuenta estos ejes tendremos que el grado total de un nodo está compuesto por el número de ejes unidos pertenecientes a ejes más cercanos c y el número de atajos que tiene el nodo s, en donde $d(v) = c + s$. [3]

La cantidad de ejes que no son atajos que posee un grafo G con modelo pequeño mundo, es igual a $1/2nc$, ya que originalmente si cada nodo tiene c ejes y tenemos n nodos pensaríamos que el número de ejes en el grafo es de nc, sin embargo, existen ejes que estamos contando dos veces ya que el grafo no es dirigido, razón por la cual adoptamos el $1/2$ en la expresión. Si añadimos atajos con probabilidad p, el número promedio de atajos sería $1/2ncp$ en donde cp sería el número de atajos en promedio que finalizan sobre un vértice en particular; aquí el número S de atajos a cada vértice sigue una distribución de Poisson con media cp, entonces la distribución de grado s es la siguiente: [3]

$$P_s = e^{-cp}\frac{(cp)^s}{s!}$$

Recordemos que el grado total de cada vértice se considera compuesto por los ejes que no son atajos y los atajos, $k = d(v) = c + s$, luego $s = k - c$, entonces la distribución de grado para el modelo de red pequeño mundo es la siguiente: [3]

$$P_s = e^{-cp}\frac{(cp)^{k-c}}{(k-c)!}$$

A continuación una representación gráfica de una red compleja con modelo pequeño mundo:

Figura 2: Representación de una red modelo pequeño mundo en donde todos los nodos están conectados con un mínimo de C ejes, imagen tomada de [8].

2.4. Medidas topológicas nodales

2.5. Robustez en redes

Muchas redes complejas como el internet, la red de fluido eléctrico, las redes de transporte, etc son parte esencial de nuestra sociedad moderna [1]. Estas redes poseen una estructura de conectividad (topología) que influencia el flujo sobre la red, por ejemplo flujo eléctrico. Este hecho hace relevante las investigaciones que indaguen sobre las propiedades de robustez en las redes complejas; es común que este tipo de estudios se centren en dos modalidades de ataque, ataques aleatorios y ataques premeditados a los hubs de la red.

Existen diversas maneras de cuantificar la robustez de una red, hay medidas que se basan en cuantificar la cantidad de nodos que tienen que ser removidos para desconectar la componente principal de la red, otras medidas como la propuesta en [10] se basan en el fenómeno de falla en cascada de las redes, fenómeno por el cual la falla o ausencia de un nodo implica la falla o ausencia de otros nodos, de tal forma que existe una falla en cascada de la estructura de flujos sobre la red.

2.6. Mecánica estadística

Para poder estudiar todos los aspectos anteriormente mencionados de las redes de forma analítica son esenciales las herramientas de la termodinámica y la mecánica estadística.

La termodinámica y la mecánica estadística son herramientas fundamentales en el estudio de la física de la materia. Mecánica estadística junto a mecánica cuántica proveen los fundamentos de la física moderna, cuyo principal avance ha sido el entendimiento de los fenómenos físicos a escala atómica. Los conceptos y técnicas de la mecánica estadística han resultado importantes no sólo para la física de la materia, si no también para la física nuclear o incluso la astrofísica. Afuera del campo de la física su importancia ha rápidamente penetrado la química, biología y varías áreas de la tecnología [18].

El primero en hacer un contacto real entre la termodinámica y la mecánica estadística fue Boltzmann, el cual en 1872 desarrollo el teorema H estableciendo una conexión directa entre la entropía y la dinámica molecular. Simultáneamente la teoría cinética clásica daba lugar a su más sofisticado sucesor, la teoría de ensambles. La potencia de las técnicas que emergieron lograron reducir finalmente la termodinámica al estatus de una consecuencia de la estadística y la mecánica de las moléculas que constituyen el sistema físico. Por lo cual es natural que este formalismo se conozca como Mecánica Estadística [19].

La esencia de las consideraciones de la teoría de en-

sambles es que para un macroestado (N,V,E) de un sistema estadístico en un tiempo t, son igualmente probables un gran número de microestados distintos. A medida que transcurre el tiempo el sistema cambia entre un microestado y otro, como resultado de esto en un tiempo razonable lo que se observa es un comportamiento promediado sobre la cantidad de microestados compatibles con ese macroestado. Un ensamble es entonces una colección de copias del sistema en un mismo instante de tiempo pero en distinto microestado, de tal forma que las propiedades promedio de esta colección correspondan a las propiedades promediadas en el tiempo del sistema físico[19].

Esta teoría se puede formular para la sistemas clásicos a partir del espacio de fase y para sistemas cuánticos a partir del espacio de Hilbert.

2.6.1. Mecánica estadística del equilibrio

En la mecánica estadística del equilibrio las variables macroscópicas se consideran independientes del tiempo y estas son descritas a través de promedios de variables microscópicas, es en esta relación entre lo micro y lo macro que la mecánica estadística en el equilibrio sea ampliamente aplicable no solo en la física, sino también en otros campos de la ciencias, incluso las ciencias económicas.

Según sea la dinámica del sistema al que se le quiere someter a un estudio mecánico-estadístico podremos decidir representarlo con el uso de alguna de las tres colectividades que existen en la mecánica estadística, estas colectividades son también conocidas como ensambles y se conocen tres, estos son: 1. Ensamble micro canónico, 2. Ensamble macro-canónico, 3. Ensamble grancanónico. [20]

El ensamble microcanónico se emplea para sistemas que estén en equilibrio y que estén aislados de su medio, en donde no se permite el intercambio de energía. Para este ensamble todos los posibles estados tienen la misma probabilidad, además de que el número de microestados que el sistema tiene puede conocerse conociendo la función de partición, por ejemplo, para el caso del gas ideal, esta es la siguiente función de partición: [20]

$$\Omega = \frac{V^N}{N!}\frac{2\pi^{3N/2}}{\Gamma(3n/2)}(2mE)^{(3N-1)/2} \tag{8}$$

Esta función de partición resulta muy importante para conocer propiedades macroscópicas del sistema, por ejemplo, Boltzmann propuso que la entropía de un sistema podría llegar a ser conocida si se conocía el numero total de microestados de la siguiente forma [20]:

$$S = k_B Ln(\Omega) \tag{9}$$

Como ejemplo de que con una función de partición es posible conocer información del mundo macroscópico, veamos que utilizando las ecuaciones 8 y 9 y las relaciones de Maxwell es posible conocer la ecuación de estado de los gases ideales:

$$\frac{P}{T} = \frac{\partial S}{\partial V} \longrightarrow PV = Nk_BT$$

El ensamble canónico a diferencia del microcanónico permite extraer información relacionada con el intercambio de energía térmica del sistema con sus alrededores, pero no existe un flujo de partículas, aquí el volumen y la temperatura son constantes, en general su función de partición tiene la siguiente forma: [20]

$$Z = \sum_j e^{-\beta E_j}$$

Por ultimo, el ensamble macrocanónico también da la misma información que se puede conocer del ensamble canónico ya que en este también se permite el intercambio de energía, sin embargo, la ventaja que tiene el ensamble macrocanónico es que de todos los demás ensambles permite el intercambio de partículas, por lo que es el más completo y el que más se aproxima a los sistemas que aparecen en la naturaleza, aunque en varios casos resulta ser más complicado trabajar con este ensamble; esta es su función de partición: [20]

$$Z = \sum_j \sum_i e^{-\beta(E_j - \mu N_i)}$$

2.6.2. Mecánica estadística del no equilibrio

La mayoría de los procesos naturales ocurren fuera del equilibrio y es de gran importancia que se tenga en cuenta la variable temporal, ya que es respecto a esta variable que se explica la continuidad en el cambio de los procesos y magnitudes que después se observan a un nivel microscópico, en general cualquier proceso irreversible tiene una naturaleza lejana al equilibrio, como por ejemplo, las reacciones químicas dirigías por un decrecimiento en la energía libre, la fricción, los sistemas disipativos, la decoherencia cuántica o simplemente el transporte de calor debido a movimientos internos en los materiales debido al desequilibrio de temperaturas. [21]

Para sistemas en el no equilibrio se esperaría que las probabilidades para los microestados dependan del tiempo, lo que evocaría en la aparición de ensambles que tienen en cuenta este aspecto mediante el uso de funciones de partición no estáticas, entonces aquí conceptos relacionados con el tiempo como el de la ecuación de Liouville serían importantes e indispensables al tenerlos en cuenta ya que este tipo de resultados habla de la dinámica en un sistema en donde sus estados cambian en el tiempo y cómo estos conservan algunas propiedades a niveles micro y macro. [21]

Otras formas de estudiar lo que pasa en el no equilibrio es precisamente estudiando los puntos aledaños al equilibrio sin necesidad de considerar que el sistema se encuentra en equilibrio total y de esta manera permitiéndonos utilizar varias herramientas que se utilizan en el equilibrio, ejemplos de estos escenarios son los obtenidos cuando se consideran pequeñas perturbaciones en un sistema en donde se utilizan teorías para estudiar fenómenos con respuestas lineales [22]. Cuando se consideran escenarios cercanos al equilibrio podríamos pensar en herramientas como el teorema de fluctuación-disipación formulado originalmente por Harry Nyquist en 1928 [23], este teorema es basado en la suposición de que la respuesta de un sistema en equilibrio termodinámico a la aplicación de una fuerza muy pequeña es igual a su respuesta frente a un fluctuación espontánea, entonces el teorema relaciona la relajación de respuestas lineales de un sistema perturbado en un estado del no equilibrio [24].

2.6.3. Procesos estocásticos en mecánica estadística

Los procesos estocásticos al introducir variables aleatorias pueden resultar muy útiles para trabajar en la mecánica estadística considerando sistemas fuera del equilibrio, el carácter aleatorio se agrega para reflejar que la información de interés se convierte con el tiempo en correlaciones sutiles dentro del sistema. Estas correlaciones toman un carácter caótico sobre las variables de interés. [25] Reemplazando estas correlaciones con la aleatoriedad propiamente dicha, los cálculos se pueden hacer mucho más fáciles. Un ejemplo de esto resulta ser la ecuación de transporte de Boltzmann en donde James Clerk Maxwell había demostrado que las colisiones moleculares conducirían a un movimiento aparentemente caótico dentro de un gas. Posteriormente, Ludwig Boltzmann demostró que, al dar por sentado este caos molecular como una aleatorización completa, los movimientos de las partículas en un gas seguirían una ecuación de transporte de Boltzmann simple que restauraría rápidamente un gas a un estado de equilibrio. [25] Como otros ejemplos de esto podemos encontrar la Jerarquía BBGKY en donde un conjunto de ecuaciones describen la dinámica de un sistema con un gran número de partículas interaccionantes, o también el formalismo de Keldysh el cual es un marco de referencia para describir la evolución de un sistema mecánico-cuántico en un estado de no equilibrio en donde este se encuentra sometido a la variación de campos externos [26].

En la teoría de procesos estocásticos es normal aprender acerca de una herramienta utilizada para estudiar mediante la probabilidad eventos de ocurrencia futura basándose en datos anteriores, esta herramienta es conocida como cadenas de Márkov, en esta teoría podríamos presentar cualquier evento como un nodo en un red en donde estos distintos eventos se unen mediante un eje de carácter directo al que se le otorga un peso entre 0 y 1 el cual representa la probabilidad de que después de cierto evento ocurra ese otro evento al que se encuentra dirigido, debido a esto podemos utilizar una matriz que caracterice que nodos (filas) se unen a otros (columnas) con ciertos ejes con peso. [27] Para predecir la probabilidad que tendrían cada uno de los eventos considerados después de n pasos de carácter discreto necesitaríamos un vector de probabilidad X_o (vector en donde la suma de todas sus componentes da como resultado 1 y que habla de las probabilidades obtenidas en el tiempo $t = 0$, en el caso de que se parta de un nodo fijo j-ésimo, la fila j-ésima de este vector es unitaria y todas las demás cero), esta teoría dice que después de k pasos de tiempo el nuevo vector de probabilidades puede ser obtenido con la matriz de transición de la siguiente forma: [27]

$$X_f = P^k X_o \tag{10}$$

Según lo hablado en la subsección perteneciente a la mecánica estadística del no equilibrio, un sistema que demuestra comportarse de manera caótica puede ser tratado con métodos de naturaleza estocástica, un método estocástico útil es la máster equation la cual se usa para describir la evolución temporal de un sistema que de alguna forma pudo ser modelado bajo un tratamiento de combinaciones y estados probabilísticos en cualquier punto del tiempo en donde la transición entre estados es determinada por una matriz de ratio de transición de la misma forma que se hace con las cadenas de Márkov en la ecuación 10. [27]

La master equation en su totalidad nos representa un conjunto de ecuaciones diferenciales de primer orden que describen de que forma cambian las probabilidades para ciertos eventos punto a punto de manera continua, teniendo en cuenta esta información en una matriz de transición $A(t)$ dependiente del tiempo, su forma es la siguiente: [28]

$$\frac{d\vec{P}}{dt} = A(t)\vec{P} \tag{11}$$

Las master equation es una ecuación que representa el limite continuo de un proceso de Márkov, de hecho, es posible escribir el vector de estado final de un proceso de Márkov en un tiempo $t = n$ como una diferencia de vectores a lo largo del tiempo, de manera tal que:

$$X_f = \frac{X_1 - X_o}{t_1 - t_o} \longrightarrow Xf = \frac{dX_o}{dt}$$

En la teoría de probabilidad la master equation identifica la evolución de un proceso continuo de Márkov como dejar ver la similitud entre 10 y 11, esta ecuación puede ser simplificada de tal forma que los terminos en donde $l = k$ de la matriz no aparezcan en la sumatoria, esto permite el calculo de variables en el problema incluso si la diagonal de la matriz de transición A no se ha definido o si tiene un valor arbitrario, entonces una forma de simplificar la master equation 11: [28]

$$\frac{dP_k}{dt} = \sum_l (A_{kl}P_l) = \sum_{l=k} (A_{kl}P_l) + A_{kk}P_k$$
$$\frac{dP_k}{dt} = \sum_{l\neq k} (A_{kl}P_l - A_{lk}P_k)$$

Volviendo a la ecuación 11, sabemos que si sumamos todas las componentes de $\vec{P}$ al ser un vector de probabilidad nos dará como resultado 1, entonces:

$$\sum_l P_l = 1 \longrightarrow \frac{d}{dt}\sum_l P_l = 0$$

De aquí también tenemos que:

$$\sum_l \sum_k A_{lk}P_k = 0 \tag{12}$$

Si asumimos que nuestro vector de probabilidad es un vector que nos dice que solo un estado (columna) está definido con una probabilidad del 100 % y que por lo tanto en las demás componentes tenemos un cero, tendremos que $P_l = \delta_{lk}$ que remplazaremos en 12 para obtener que para todo estado k: [28]

$$\sum_l A_{lk} = 0$$

Usando la anterior ecuación es posible expresar las diagonales de la siguiente forma:

$$A_{kk}P_k = -\sum_{l\neq k} A_{lk}P_k$$

Según la ecuación anterior debido a que si cada termino desaparece en el equilibrio el sistema exhibe comportamientos cercanos al equilibrio, entonces si para todo termino k y l si estos tienen unas probabilidades en el equilibrio π_k y π_l, tenemos la siguiente relación que se relaciona con las relaciones reciprocas de Onsager con simetria temporal rota [29].

$$A_{kl}\pi_l =_{lk} \pi_k$$

2.7. Principio de máxima entropía (teoría de la información)

La teoría de la información nos provee de un formalismo para construir distribuciones de probabilidad en presencia de conocimiento parcial del sistema. Esto nos lleva a un tipo de inferencia estadística que es conocida como estimado de máxima entropía [30].

La base de este método de inferencia estadística recae en dos objetos matemáticos, en primera instancia la entropía de Shannon, que tiene como fin cuantificar la información/incertidumbre promedio, esta entropía de Shannon puede probarse que es la única cantidad positiva, que incrementa con la información y que es aditiva para fuentes independientes de incertidumbre [31]

$$\boldsymbol{H}(p_1, p_2, ..., p_n) = -K\sum_i p_i ln(p_i) \tag{13}$$

En segunda instancia el método de los multiplicadores de Lagrange, el cual se utiliza para resolver problemas de optimización.

La metodología a seguir para aplicar este principio de inferencia estadística sera delineada a continuación:

1. Se establecen las ligaduras que se impondrán, estas ligaduras son extraídas de los datos a los cuales se les quiera inferir la distribución de probabilidad, un ejemplo común de ligadura es el valor esperado de X cantidad, por ejemplo la energía en un sistema termodinámico o el grado en un grafo.
2. Se utiliza el método de los multiplicadores de Lagrange para hallar la distribución de probabilidad que satisfaga las ligaduras junto a las condiciones de maximizar la entropía de Shannon y de normalizar la distribución.

La efectividad de la inferencia de este método recae en la interpretación de la entropía de Shannon, de tal forma que buscar la distribución que haga máxima esta entropía es equivalente a buscar la distribución más uniforme que cumpla las ligaduras, en otras palabras buscar la distribución que minimice las supocisiones sobre los datos [32].

Es la aplicación de este principio el que nos permitirá usar las herramientas de la mecánica estadística del equilibrio en teoría de redes.

2.7.1. Principio de máxima entropía para generar ensambles de redes aleatorias

Las redes aleatorias son los modelos de hipótesis nula con los que se contrastan supocisiones hechas sobre redes reales. Por esto es importante tener un mecanismo sistemático para producir redes aleatorias con las características deseadas. El principio de máxima entropía cumple los requisitos para ser este mecanismo [5].

Supóngase que tenemos una serie de medidas de propiedades de una red, las cuales impondremos sobre nuestro ensamble de grafos aleatorios simples. Ejemplo de estás medidas puede ser el grado, la distribución de grado, el número de vértices, etc. Denotemos estas medidas por $x_1, x_2, ..x_n$. [13]

Ahora consideremos el conjunto de todos los grafos simples $\wp$ con n vértices y definamos el ensamble dándole una probabilidad a cada grafo tal que: [13]

$$\sum_{G\in\wp} P(G) = 1$$

El valor esperado $\langle x_i\rangle$ de la medida de red x_i al interior de este ensamble está dado por: [13]

$$\langle x_i\rangle = \sum_{G\in\wp} P(G)x_i(G)$$

Ahora impongamos la condición de que los esperados del ensamble correspondan al valor esperado de las medidas tomadas de la red real i.e. [13]

$$\langle x_i\rangle = \sum_{G\in\wp} P(G)x_i(G)$$

donde $\langle x_i\rangle$ corresponde al promedio del valor esperado de la i-ésima medida tomada de la red real.[13]

El siguiente paso es usar el método de multiplicadores de Lagrange para encontrar la función P(G) de probabilidad que cumpla las siguientes condiciones: [13]

$$S_{max} = -\sum_{G\in\wp} P(G)lnP(G)$$
$$\sum_{G\in\wp} P(G) = 1$$
$$\langle x_i\rangle = \sum_{G\in\wp} P(G)x_i(G)$$

Esto nos lleva a que la distribución P(G) buscada es tal que maximiza la cantidad. [13]

$$-\sum_{G\in\wp} P(G)lnP(G) + \alpha\left[1 - \sum_{G\in\wp} P(G)\right] +$$
$$+\sum_i \beta_i\left[\langle x_i\rangle - \sum_{G\in\wp} P(G)x_i(G)\right]$$

Donde α y β_i corresponden a los multiplicadores de lagrange. Derivando respecto a la probabilidad P(G) e igualando a 0: [13]

$$lnP(G) + 1 + \alpha + \sum_i \beta_i x_i(G) = 0$$

de donde obtenemos: [13]

$$P(G) = exp\left[-(\alpha+1) - \sum_i \beta_i x_i(G)\right]$$

que podemos escribir como: [13]

$$P(G) = \frac{exp\left[-\sum_i \beta_i x_i(G)\right]}{Z}$$

Donde Z es la función de partición y toma la siguiente forma: [13]

$$Z = e^{\alpha+1} = \sum_{g\in G} exp\left[-\sum_i \beta_i x_i(G)\right] \qquad (14)$$

La energía libre se define como [13]

$$F = -ln(Z) \qquad (15)$$

2.8. Algoritmos

A continuación se presentan los dos algoritmos principales que son base de este trabajo

2.8.1. Algoritmo para generar redes con máxima entropía y una distribución de grado dada

Este algoritmo es presentado en [2] la idea de este algoritmo consiste en construir una secuencia de grado por medio de un muestreo multinomial de una distribución de probabilidad P, posteriormente se construye un grafo a partir de está secuencia de grado, grafo que finalmente se somete ante un proceso de aleatorización manteniendo su distribución de grado constante.

El algoritmo puede ejecutarse siguiendo los siguientes pasos

1. Seleccionar distribución **P** objetivo para la distribución de grado
2. Generar una muestra uniforme **n** de la distribución multinomial de **P**
3. Generar una secuencia de grado **d** muestreando de forma uniforme a **n**
4. Testear si la secuencia de grado generada por los pasos anteriores puede corresponder a la secuencia de grado de un grafo simple, para esto se puede usar el algoritmo de Havel-Hakimi[2], si no lo es, regresar al tercer paso
5. Testear si el grafo simple generado es además un grafo conectado, es decir evaluar si existe un camino finito para llegar de cualquier nodo i a cualquier nodo j, si el grafo no es conectado regresar al tercer paso
6. Finalmente aleatorizar la red manteniendo la distribución de grado i.e. el grado de cada nodo, para esto se buscan cuatro nodos distintos i,j,k,l de forma aleatoria tal que siendo A_{ij} las componentes de la matriz de adyacencia se cumpla $A_{ij} = A_{kl} = 1$ y $A_{il} = A_{kj} = 0$, entonces se cambian los vértices de tal manera que se cumple $A_{ij} = A_{kl} = 0$ y $A_{il} = A_{kj} = 1$, este proceso se repite hasta que se muestree el ensamble de forma uniforme.

2.8.2. Algoritmo de falla en cascada

Este algoritmo presentado en [10] se basa en la medida para nodos load, medida que dado un nodo K cuantifica de todas las geodésicas entre cuales quiera dos nodos de la red, cuantas son intermediadas por el nodo K, está medida puede considerarse una primera aproximación a la cuantificación de la intermediación del nodo K en la comunicación al interior de la red. Este mismo concepto puede ser aplicado a los enlaces.

A partir del valor inicial del load de cada nodo/enlace se construye una capacidad

$$C_j = (1 + \alpha)L_j(0)$$

Donde α es un parámetro del modelo llamado constante de tolerancia y $L_j(0)$ es el load inicial del nodo/enlace. Esta capacidad cuantifica que tanto load puede soportar un nodo/enlace antes de fallar y desconectarse de la red.

La falla en cascada ocurre cuando al atacar una red/enlace/subgrafo se redistribuye la estructura de geodésicas de la red, de tal forma que nodos/enlaces que antes tenían un bajo load pueden tener de repente uno muy alto, de tal manera que si este load supera su capacidad el nodo falla y se repite este proceso. En este trabajo se modificará ligeramente el algoritmo, de tal forma que mientras en este se ataca sólo a los nodos con mayor load, aquí atacaremos también de forma aleatoria.

El aporte del algoritmo planteado en este artículo es establecer una resiliencia para los nodos/enlaces, de tal manera que una vez el flujo supera la capacidad del nodo este tiene una probabilidad $P(L_j)$ de fallar igual a:

$$P(L_j) = \frac{1}{\rho - 1}\left[\frac{L_j}{C_j} - 1\right] \tag{16}$$

probabilidad que aumenta hasta 1 cuando $L_j = \rho C_j$.

El daño causado por la falla en cascada es cuantificado en términos del tamaño relativo de la componente más grande conectada

$$G = \frac{N'}{N} \tag{17}$$

Donde N' corresponde al número de nodos de la componente más grande conectada y N al número de nodos iniciales.

El algoritmo pude ejecutarse siguiendo los siguientes pasos

1. Calcular el load y las capacidades para todos los nodos/enlaces en la red
2. remover vértices o nodos bajo alguna estrategia de ataque
3. Recalcular la load de los nodos/enlaces y determinar cuales son los nodos fallidos al comparar un parámetro aleatorio $\beta \in (0, 1)$ con el resultado de la probabilidad mostrada en la ecuación (16)

4. remover los nodos fallidos de la red y repetir el tercer paso hasta que no haya más nodos/vértices con sobrecargados
5. calculamos el tamaño de la componente más grande conectada para así calcular el daño tal como se muestra en la ecuación (17)

3. Estado del arte

Desde un punto de vista histórico, el estudio de las redes ha derivado del estudio de una rama de las matemáticas conocida como teoría de grafos, la cual tiene su origen en 1736 cuando el matemático Leonhard Euler resolvió el problema de los puentes de Konigsber, publicando el primer trabajo en teoría de grafos.

Sin embargo, esta rama no se desarrollo de manera prolífica justo después de la publicación de Leonhard Euler, sino que tardo unos siglos más en desarrollarse tanto en la parte estructural ligada a las matemáticas, como en la gran cantidad de escenarios en la que se hacia aplicable; por ejemplo, en 1878 Sylvester amplió en parte la teoría de grafos con el fin de hacerla aplicable a problemas en química que se estudiaban en la época, en 1920 se comenzó a aplicar el estudio de redes en las ciencias sociales, con el fin de estudiar relaciones de organización tanto en las personas como otras entidades sociales y así estudiar procesos en donde estuviese implicada la comunicación, interacción, transacción, entre otras formas de relación presentes que resultan muy habituales, en 1936 el matemático húngaro Dénes Kónig formalizó aún más este estudio publicando el primer libro en teoría de grafos llamado "Theory of finite and infinite graphs" [33].

A partir de los años 70 el estudio de redes fue desarrollándose y haciéndose aplicable a muchos otros campos de la ciencia, en los años 90 comienzan a publicarse artículos que hoy en día se consideran centrales en la teoría de redes, los cuales nos hablan de distribuciones de grado y cómo el conocimiento de estas pueden dar bastante información relacionada con los fenómenos a los que es posible aplicar esta teoría, a su vez que se relacionan con trabajos de aleatoriedad, robustez y entropía.

Las distribuciones de grado han sido ampliamente estudiadas en la teoría de redes, la clasificación de las redes en base a esta propiedad tiene una gran utilidad en problemas prácticos, las redes de tipo pequeño-mundo se introdujeron por primera vez en 1997 por Frank Ball, Denis Mollison y Gianpaolo Scalia-Tomba en el artículo conocido como "Epidemics with two levels of mixing" bajo el nombre de "modelo epidémico del gran circulo" [34], pero posteriormente popularizado por Watts y Strogatz en 1998 con su reconocido artículo [8] "Collective dynamics of small-world networks", por ultimo en 1999 algunas modificaciones le fueron realizadas a este modelo en el artículo [35] publicado por Watts y Newman.

Otra distribución de grado estudiada en el proyecto es la correspondiente a las redes libres de escala, las cuales se encuentran íntimamente relacionadas con el modelo "preferential attachment", sin embargo, los origines de este comportamiento y distribución se remontan al año 1925 en donde Yule Udny utilizó este modelo de redes para explicar el número de especies por genero en flores en su artículo "A Mathematical Theory of Evolution, based on the Conclusions of Dr. J. C. Willis, F.R" [36], razón por la cual al proceso se le denomina también "proceso Yule" en su honor.

Aunque procedimientos más recientes son vistos en el artículo publicado por Herbert Simon en 1955 conocido como "On a class of skew distribution functions" [37] en donde estudió el tamaño y crecimiento de algunas sociedades, entre otros fenómenos pero sin hacer mucho énfasis en el uso de la teoría de redes, o más reciente aún es el trabajo publicado por Derek Price en 1976 "A general theory of bibliometric and other cumulative advantage processes" [38] en donde se refirió a este fenómeno como "cumulative advantage process", este trabajo se considera la primera aplicación tal y como la conocemos hoy en día en la teoría de redes. Cabe mencionar que formalizaciones más recientes en el método y aplicabilidad junto a la designación "preferential attachment" para referirse al proceso se le atribuye a Barabási-Albert.

La ley de potencias en el modelo libre de escala indica que el crecimiento y el comportamiento conocido como "preferential attachment" juegan un papel importante en el desarrollo de las redes, pero explorando los casos límites es importante cuestionarse si ambos son necesarios para el surgimiento de una ley de potencias como distribución de grado, para abordar esta cuestión en el año 2000 Barabási-Albert, Réka Albert y Hawoong Jeong trataron este planteamiento en el artículo publicado en la revista nature conocido como "Error and attack tolerance of complex networks " [11] en donde se explora el caso límite en donde se considera solo uno de estos mecanismos para encontrarnos con esta distribución de grado al demostrar que mediante ataques en la red es posible concluir que no todas las redes libres de escala siguen el comportamiento de "preferential attachment".

Esto ha sido apoyado últimamente en 2015, publicaciones como "Exactly scale-free scale-free networks." [2] en donde sugieren también que las redes en la que se encuentra el proceso de "preferential attachment"no resultan adecuadas para exponer la generalidad de una red libre de escala ya que las redes más generales que siguen esta distribución de escala no exponen comportamientos típicos que sí se pueden encontrar en una red

con el proceso de "preferential attachment".

Entre los años 1870 y 1900 el físico austriaco Ludwig Boltzmann junto a otros como Willard Gibbs influenciados por el concepto estadístico que se le podría dar a la entropía desarrollaron la física estadística, de tal manera que tuviera concordancia con conceptos ya postulados desde la física térmica. Más tarde en analogía con la entropía térmica y estadística, en 1948 Claude Shannon mientras trabajaba en una medida de naturaleza estadística para cuantificar la perdida de información en las señales de las lineas telefónicas, elaboró un concepto teórico que resulto muy general conocido como entropía de la información, el cual en 1957 se relacionaría junto al método de máxima entropía y las definiciones ya existentes de la entropía estadística en un trabajo conocido como "Information theory and statistical mechanics" publicado por E. T Jaynes [39] en donde argumentaba que en realidad la entropía estadística y la conocida en la teoría de la información resultaban ser la misma, dando una correspondencia natural entre la mecánica estadística y la teoría de la información.

Las primeras aplicaciones en donde se utilizó mecánica estadística y teoría de redes llegaron a ser publicadas entre los años 90, estas publicaciones se basaban en el aprendizaje de una red neural en donde se vieron fuertemente implicadas las interacciones físicas de un sistema; en 1991 MacKay, D. J aplicó por primera vez el método de máxima entropía a las redes neurales en "Maximum entropy connections: neural networks. In Maximum entropy and Bayesian methods" [40] pero sin relacionar el tema directamente con la mecánica estadística, sin embargo, los primeros trabajos más reconocidos en hacer uso de todos estos conceptos tanto estadísticos como pertenecientes a la teoría de redes fueron publicados en 1992 por HS Seung, H Sompolinsky, N Tishby en un trabajo conocido como "Statistical mechanics of learning: Generalization." [41] y L. Ingber con la publicación "Generic mesoscopic neural networks based on statistical mechanics of neocortical interactions" [42] .

Mas tarde en el año 2000 Watts dió a conocer la importancia que tenía el estudio de los ataques a las redes que producían fallos en cascada, dando varios ejemplos acerca de este fenómeno en "A simple model of fads and cascading failures" [43]. En este mismo año esta cuestión de los fallos que surgían ante ataques intencionales en las redes también fue expuesta en trabajos reconocidos en la teoría de redes como el publicado por R Albert, H Jeong, Barabási-Albért "Error and attack tolerance of complex networks" [11] y también el expuesto por Callaway, Newman, Strogatz y Watts en "Network robustness and fragility: Percolation on random graphs" [44] en donde se relaciona el concepto de percolación, robustez y fragilidad para grafos producidos por algoritmos aleatorios con el fin de observar su resiliencia bajo ataques dirigidos y ver si esta tiene relación con su distribución de grado. Más reciente aún el artículo [9] publicado en 2009 sobre el cual se basa este proyecto realiza una comparación en los fallos de cascada que sufren las redes de pequeño-mundo y libres de escala.

Posteriormente en el 2005 se realiza una publicación en donde por primera vez se propone la utilización del método de máxima entropía con la intención de hacer las redes más tolerantes a fallos aleatorios de tal manera que la entropía de la distribución de grado resulta ser una medida efectiva para la tolerancia de la red cuando se mide a través del uso del concepto de robustez, este artículo es conocido como "Entropy optimization of scale-free networks' robustness to random failure" [45], publicado por Wang, B, Tang, H, Guo, C, y Xiu, este método de máxima entropía también resultó muy útil para que Z.J. Bao, Y.J. Cao,, L.J. Ding , G.Z. Wang en [2] en el 2015 concluyeran que el modelo de "preferential attachment" no resulta del todo adecuado para representar la generalidad de las redes libres de escala, publicación la cual resulta central para el presente proyecto.

4. Planteamiento del Problema Tentativo

En [2] se presenta un algoritmo para generar redes con máxima entropía que posean una topología dada, impuesta a través de la distribución de grado. Respecto a este algoritmo se hacen tres afirmaciones: el algoritmo presentado es equivalente al formalismo de máxima entropía para ensambles canónicos mostrado en [12],[6], este algoritmo es más adecuado para muestrear el ensamble de redes libres de escala que el de preferential attachment y por último que la propiedad robustez pero fragilidad de las redes de Barabási-Albert no está presente en general al muestrear el ensamble de redes libres de escala usando el algoritmo de máxima entropía.

Nuestro primer objetivo consiste en el uso de herramientas de la mecánica estadística fuera del equilibrio, como lo son las cadenas de Markov [27] y la master equation [28], con el fin de realizar el desarrollo analítico del modelo Preferential Attachment [7] en donde a través del uso de la ecuación 11 y la ligadura de grado 18 se imponen las condiciones para plantear una ecuación que de cuenta del cambio de los nodos y sus conexiones en la red para posteriormente lograr mediante aproximaciones y el uso de la función beta de Euler la forma analítica de la distribución de grado para el modelo libre de escala de preferential attachment [3], en donde se contrastan sus resultados con desarrollos similares en la literatura de redes. [16]

Como otras consideraciones, vemos que en [1] se comparó en términos de robustez ante ataques deliberados hacia nodos y enlaces las redes libres de escala generadas por el algoritmo de Albert-Barabási [7] y las

redes de pequeño mundo generadas por el algoritmo de Watts y strogatz [8], en este articulo se reporta que las redes de Albert-Barabási son más robustas ante ataques deliberados hacia los nodos que las redes de Watts y Strogatz, mientras para ataques a enlaces la situación es contraria.

En [9] estudian la robustez ante fallos en cascada para redes libres de escala, haciendo énfasis en la ausencia de estudios que consideren este tipo de análisis no sólo en fallos de nodos, sino también en fallos de relaciones y parejas de nodos.

Con estas consideraciones en mente nos pusimos cuatro objetivos adicionales para dar 5 en total. Dos están relacionados con la afirmación respecto a la equivalencia del algoritmo presentado en [2] y el desarrollo analítico expuesto en [12] y [6], esta afirmación no se valida en este artículo, por lo cual está validación se desarrollará en este trabajo por medio de la comparación entre las matrices de adyacencia promedio producto del muestreo in silico del ensamble de redes libres de escala y la matriz de adyacencia promedio obtenida por medio de la predicción analítica del ensamble canónico a través de las derivadas de la energía libre. En los últimos dos objetivos nos decidimos ampliar el trabajo hecho en [1] primero ampliando los mecanismos de ataque, puesto que consideramos ataques deliberados pero también aleatorios, luego ampliando los objetivos de ataque tal como sugiere [9], de tal manera que se atacarán nodos y enlaces. Finalmente dado que en [2] reportan que las propiedades de robustez de las redes de Albert-Barabási no son representativas del ensamble de redes libres de escala, nos disponemos a evaluar esta afirmación por medio del desarrollo analítico del algoritmo Preferential Attacthment a través de la master equation, hallando así los sesgos de las redes generadas por este modelo de crecimiento y contrastando las propiedades de robustez al confrontar las conclusiones de la comparación hecha en [1] pero con redes libres de escala generadas por el algoritmo de máxima entropía, evaluando la posibilidad de que la entropía de algún parámetro de la red pueda dar cuenta de las diferencias en robustez.

5. Motivación y Justificación

El funcionamiento de la expresión génica, la comunicación al interior de la célula, el cerebro, el internet, la economía, el metabolismo, las relaciones sociales, el sistema eléctrico, la actividad solar, etc. Son sistemas susceptibles a ser parcialmente modelados por la teoría de redes complejas y dado que muchos de estos sistemas se relacionan directamente con los humanos, es deseable una mayor comprensión de las redes y sus propiedades; de estas propiedades es particularmente importante la robustez, dado que las redes que sustentan nuestra vida y la sociedad son susceptibles a errores y ataques deliberados es fundamental comprender que tipos de redes son frágiles bajo que tipos de ataques y que propiedades de la red pueden permitirnos construir redes más robustas.

En este contexto nos decidimos a ampliar los estudios de robustez para las dos topologías más comunes en las redes complejas, la de pequeño mundo y la libre de escala; en particular prestamos atención en validar analítica y numéricamente que las redes libres de escala analizadas sean representativas del ensamble, esto con el fin de que nuestros resultados tengan un mayor rango de validez.

6. Objetivo General

Comparar in silico la robustez ante fallas en cascada para redes de pequeño mundo generadas por el algoritmo de Watts-Strogatz y libre de escala generadas por medio del algoritmo Preferential attacthment y un algoritmo para generar redes con máxima entropía dada una distribución de grado en teoría equivalente al ensamble canónico; esto evaluando la posibilidad de que la entropía respecto a algún parámetro de la red pueda dar cuenta de las diferencias en robustez encontradas.

7. Objetivos Específicos

1. Desarrollar analíticamente por medio de herramientas de la mecánica estadística del no equilibrio el proceso de evolución de redes conocido como preferential attacthment, esto con el fin de hallar condiciones para las redes generadas por este modelo que limiten su muestreo del ensamble de redes libres de escala.
2. Desarrollar analíticamente el ensamble canónico imponiendo como ligadura la secuencia de grado promedio, a partir de la función de partición calcular la energía libre y finalmente haciendo uso de la energía libre calcular las componentes de la matriz de adyacencia promedio para este ensamble.
3. Establecer en el lenguaje Python los algoritmos que se usarán para el estudio de robustez y entropía, es decir los algoritmos para generar redes de Watts-Strogatz, Barabási-Albert y Máxima entropía; los algoritmos de falla en cascada y los necesarios para calcular la entropía de Shannon para distintos parámetros nodales de la red.
4. Muestrear el ensamble de redes libres de escala usando el algoritmo de máxima entropía programado previamente, a partir de este muestreo calcular la matriz de adyacencia promedio y comparar con la matriz de adyacencia promedio calculada a partir del ensamble canónico.

5. Realizar las comparaciones de robustez ante fallas en cascada para redes generadas por los algoritmos de Watts-Strogatz, Barabási-Albert y el de Máxima Entropía. Evaluando la posibilidad de que la entropía respecto a algún parámetro de la red de cuenta de las diferencias en robustez.

8. Metodología

1. A partir del planteamiento de la master equation y siguiendo el desarrollo de [5] y [16] se desarrollará el tratamiento analítico del algoritmo preferential Attachtment, haciendo enfásis en los resultados que relacionen la distribución de grado con los parámetros del modelo.

2. Para el desarrollo analítico del ensamble canónico bajo la imposición de una secuencia de grado promedio se utilizará el método de multiplicadores de Lagrange, el sistema de ecuaciones no lineales producto de este método se resolverá numéricamente y posteriormente se calculará la matriz de adyacencia promedio a partir del desarrollo mostrado en [13].

3. Los algoritmos de Watts-Strogatz y Barabási-Albert vienen incorporados en la librería Networkx, con su correspondiente validación. Para el algoritmo de máxima entropía se seguirá lo delineado en [2] y para los algoritmos de falla en cascada lo delineado en [10]. Finalmente para calcular la entropía se usará la librería Numpy Stats.

4. Generar un número N de grafos, construir sus matrices de adyacencia y promediarlas. Luego comparar está matriz de adyacencia insilico promedio con la matriz de adyacencia calculada numéricamente.

5. Usando los algoritmos previamente programados se generaran redes de pequeño mundo y libre de escala, posteriormente usando los algoritmos de falla en cascada se evaluará la robustez de estas redes bajo ataques aleatorios y dirigidos. Para cada una de estas redes se calculará la entropía respecto a algún parámetro nodal y se compararán la relación entropía-robustez para estas redes.

9. Resultados esperados

Los resultados esperados se dividieron en dos, productos del trabajo y objetivos de aprendizaje.

Productos del trabajo

1. Validación de la equivalencia del algoritmo presentado en [2] y el desarrollo analítico del ensamble canónico para redes presentado en [12] y [6].

2. Dejar una base de funciones programadas en un repositorio de Github para teoría de redes que puedan servir para futuros trabajos.

3. Estudiar la posible relación entropía-robustez.

4. Reproducir los resultados de robustez presentados en [1] y [2].

5. Ampliación de la caracterización de la robustez ante fallas en cascada causados por ataques aleatorios y deliberados para dos redes con topologías representativas; pequeño mundo y libre de escala.

Objetivos de aprendizaje

1. Conocer como se pueden aplicar herramientas de mecánica estadística en otras ramas del conocimiento, en este caso la teoría de grafos.

2. Introducirnos a las herramientas de la mecánica estadística del no equilibrio, en particular procesos estocásticos y master equation.

3. Obtener un conocimiento sobre herramientas de programación útiles en mecánica estadística, tales como el método de monte-carlo o resolver numéricamente las ecuaciones producto del método de multiplicadores de Lagrange.

4. Ganar conocimiento en la practica de la magnitud entropía y su significado en distintos contextos.

5. Introducirnos en la investigación en teoría de redes complejas.

10. Cronograma

CRONOGRAMA SUSPENDIDO POR MOTIVO DE PARO

Fechas importantes	Objetivos realizados
10 abril:	Presentación preproyecto
24 abril:	El 24 de abril se habrán desarrollado 2 objetivos, estos son los relacionados con la programación de los tres tipos de redes que se estudiarán incluyendo el algoritmo de máxima entropía y el análisis entropía-robustez de las redes bajo ataques aleatorios y deliberados a nodos, vértices y parejas de nodos; a su vez que se empezará a trabajar en el desarrollo analítico.
25 de mayo:	El 25 de mayo se habrá realizado el análisis analítico, tanto la validación como la comparación entre las redes generadas por preferential attachment y el algoritmo de máxima entropía.
10 de junio:	El 10 de junio se espera haber desarrollado todo el trabajo computacional necesario para las comparaciones que se harán en los penúltimos 4 objetivos.
20 de junio:	Para esta fecha se espera haber terminado los desarrollos computacionales y analíticos propuestos de tal manera que se compararán también con los resultados expuestos en los artículos base para el desarrollo del proyecto.
25 junio:	Entrega del proyecto

Cuadro 1: Cronograma.

11. Recursos Disponibles

Software Este trabajo se desarrollará en el lenguaje de programación Python, asistido por las librerías Scipy, matplotlib, powerlaw, math Numpy y Networkx.

Hardware Se cuenta con una computadora ACER nitro 5, con 12 GB de ram y un procesador intel core i5-8300H de 2.30GHz.

Bases de datos Scopus.

12. Desarrollo de los objetivos específicos

A continuación se presentan los resultados de cada uno de los objetivos específicos, los desarrollos de cada objetivo se encuentran en los anexos.

12.1. Resultados primer objetivo específico

Cuando se empezó con los procedimientos de obtener de forma analítica el ensamble de redes se supuso que dada una red con una cantidad inicial de nodos y ciertas conexiones una vez un nuevo nodo entra a la red este tiene una probabilidad de formar conexiones con otros nodos dada de la siguiente forma, en donde el parámetro q_i representa el grado del nodo número i en la red y el parámetro a es una constante positiva que se considera debido a que cada nodo tiene una posibilidad de aumentar su grado y por esto se debe sumar para dar cuenta de las posibles nuevas conexiones: [3]

$$p_j = \frac{q_j + a}{\sum_i (q_i + a)} \tag{18}$$

La anterior expresión 18 se obtuvo mediante un desarrollo mostrado en la sección de anexos 15.1. La forma que presenta esta probabilidad tiene como consecuencia a largo plazo la aparición de nodos clusterizados puesto que cada vez es más frecuente que nodos con un grado mayor sean la preferencia para hacer conexiones una vez entra un nuevo nodo a la red, una vez conocida esta probabilidad mediante la aplicación de la master equation 11 fue posible obtener una ecuación que nos permitiese de forma iterativa hallar la distribución de grado, la master equation resultó muy útil debido a que esta puede modelar la evolución de un sistema en el tiempo mediante el conocimiento de una matriz de transición y ciertos rasgos previos del sistema. Debido a las propiedades de la master equations, fue posible obtener la ecuación 32 que permite relacionar el grado p_q con los parámetros de la red, esto fue posible ya que la master equation se puede escribir de la forma 29, resultando en un patrón que permite relacionar las ecuaciones 26, 15.1 y 28 que hablan de la evolución del grafo una vez se le suministran nuevos nodos, como se muestra en la ecuación 30. Obtenida la ecuación 32 se procedió a utilizar más tratamientos analíticos como despejes y el uso de las funciones gamma y beta Euler para obtener la distribución de grado, el desarrollo en mayor detalle se presenta en la sección de anexos 15.1.

Una vez se obtuvo la distribución de grado se utilizaron aproximaciones para reescribir todo en términos de la función beta de Euler la cual depende de funciones gamma [46], en donde obtuvimos que la forma aproximada de esta distribución de potencias es la siguiente:

$$p_q \approx \frac{e^{-1}(2 + \frac{a}{c})^{\frac{3}{2}+\frac{a}{c}}}{(1 + \frac{a}{c})^{\frac{1}{2}+\frac{a}{c}}} (q + a)^{-(2+\frac{a}{c})} \tag{19}$$

La deducción de la ecuación anterior 19, la definición

del parámetro c, y la ecuación obtenida mediante una aproximación 20 se presentan en la sección de anexos 15.1. Debido a que esta distribución de grado depende únicamente del grado q pudimos considerar el termino del lado derecho que acompaña a $(q+a)^{-(2+\frac{a}{c})}$ como si este fuese una constante, además asumimos que el grado q después de tiempos largos es mucho mayor en comparación a la constante positiva a, de tal forma que fue posible despreciarla, finalmente se obtuvo de forma analítica la forma de la distribución de grado para preferential attachment: [3]

$$p_q \approx kq^{-(2+\frac{a}{c})} = kq^{-\alpha} \tag{20}$$

12.2. Resultados segundo objetivo específico

Partiendo de la imposición por medio del método de los multiplicadores de Lagrange de las siguientes restricciones:

$$\begin{aligned} S_{max} &= -\sum_{G\in\wp} P(G) ln P(G) \\ \sum_{G\in\wp} P(G) &= 1 \\ \boldsymbol{k_i} &= \sum_{G\in\wp} P(G) k_i(G) \end{aligned} \tag{21}$$

Donde $\boldsymbol{k_i}$ corresponde al grado impuesto como ligadura, k_i al grado del nodo i-ésimo y P(G) la probabilidad de encontrar un grafo G en este ensamble.

Se obtuvo para la probabilidad y la función de partición.

$$\begin{aligned} P(G) &= \frac{exp[-H(G)]}{Z} \\ Z &= \sum_{g\in G} e^{-H(G)} \end{aligned}$$

Donde $H(G)$ correspondé al hamiltoniano del grafo.

Considerando el hamiltoniano 15.2:

$$H = \sum_i \theta_i k_i = \sum_{i<j} (\theta_i + \theta_j)\sigma_{ij}$$

Donde θ_i corresponde al multiplicador de Lagrange asociado a imponer un grado $\boldsymbol{k_i}$ en promedio sobre el ensamble al nodo i-ésimo.

Y Siguiendo los desarrollos mostrados en 15.2 se encontró que la función de partición, la energía libre y la probabilidad de que dos nodos estuvieran conectados en el ensamble canónico bajo la imposición de una secuencia de grado promedio tienen la siguiente forma:

$$\begin{aligned} Z &= \prod_{i<j} (1 + e^{(\theta_i+\theta_j)}) \\ F &= -\sum_{i<j} ln(1 + e^{-\Theta_{ij}}) \\ p_{ij} &= \langle \sigma_{ij} \rangle = \frac{\partial F}{\partial \Theta_{ij}} = \frac{1}{e^{\Theta_{ij}} + 1} \end{aligned} \tag{22}$$

Donde $\langle \sigma_{ij} \rangle$ son la componentes de la matriz de adyacencia promedio, $\Theta_{ij} = \theta_i + \theta_j$ y θ_i los multiplicadores de Lagrange asociados con la ligadura de que el nodo i-esimo tenga grado $\boldsymbol{k_i}$.

A partir de las ecuaciones 22 y 21 se puede llegar a 49 tal como se muestra en 15.2. Para resolver este sistema de ecuaciones se estableció la función **mult-lagrange-canonico** la cual recibe como input una secuencia de grado y da como output los multiplicadores de Lagrange resultado de resolver la ecuación 49. Una descripción del algoritmo y el código comentado se puede encontrar en 15.2.

Una vez hallados los multiplicadores de Lagrange se pueden reemplazar en la ecuación 22 y así hallar las componentes de la matriz de adyacencia promedio. Para esto se estableció la función **canonical-matrix** la cual recibe como input un vector con los multiplicadores de Lagrange y entrega como output una lista de listas conteniendo las componentes de la matriz de adyacencia promedio predicha por el ensamble canónico. Una descripción del algoritmo y el código comentado se puede encontrar en 15.2.

12.3. Resultados tercer objetivo específico

Producto del objetivo de esta sección se establecieron 12 funciones, las cuales se describirán brevemente a continuación, una descripción más amplia y el código comentado se encuentra en 15.3

Las funciones presentadas a continuación permiten generar grafos aleatorios con una distribución de grado dada, medir su entropía para las distribuciones de medidas nodales Load, Degree, Eigenvector y Betweenness; y atacarlos generando así fallas en cascadas, de las cuales evaluaremos la población de nodos sobrevivientes ante los ataques y consideraremos esto como una medida de robustez de los grafos. En particular para comparar la robustez ante fallas en cascada se sumaran los % de población de nodos sobrevivientes para cada ataque y el número resultante se tomará como representante de la robustez en la red.

- **pareto-cumm-probabilities**: Esta función genera un vector con las probabilidades acumuladas de una distribución de Pareto con un exponente dado. Una descripción más detallada se encuentra en 15.3.

- **Degree-Sec-Generator**: Esta función muestrea secuencias de grado con un grado mínimo dado de una distribución de ley de potencias con exponente entre 2 y 3. El muestreo se hace imponiendo un cutoff para el grado máximo.Una descripción más detallada se encuentra en 15.3.

- **maxent-generator**: Función para generar grafos aleatorios con una distribución de grado promedio dada como input en forma de un vector con sus probabilidades acumuladas. Hace uso de la función **Degree-Sec-Generator**. Una descripción más detallada se encuentra en 15.3.

- **graph-entropys**: Esta función calcula las medidas nodales Load, Degree, Betweenness, Closeness y Load para un grafo dado como input, luego normaliza la distribución de valores para cada medida de tal forma que su suma sea 1 y luego calcula la entropía de Shannon 13 de la distribución. Una descripción más detallada se encuentra en 15.3.

- **remove-hubs-load**: Función para remover los nodos con mayor load de un grafo dado como input.

- **edge-remove-hub-load**: Función para remover los enlaces con mayor load de un grafo dado como input.

- **remove-aleatory**: Este algoritmo remueve nodos aleatoriamente de un grafo dado como input

- **edge-remove-aleatory**: Con esta función se pueden remover aleatoriamente enlaces de un grafo dado como input. El código comentado de estas cuatro funciones se encuentra en 15.3.

- **hub-cascade-failure**: Este algoritmo ejecuta la falla en cascada tal como se describe en 2.8.2, haciendo uso de la función **remove-hub-load** para remover los nodos con mayor load y cuantificar la falla en cascada causada por los mismos.

- **edge-hub-cascade-failure**: Este algoritmo es análogo al anterior, la diferencia radica en que este evalúa fallas en cascada 2.8.2 ante ataques dirigidos a los enlaces, por lo cual hace uso de la función **edge-remove-hub-load**. Una descripción más detallada y el código comentado de estas dos funciones se puede encontrar en 15.3.

- **aleatory-cascade-failure**: Esta función es análoga a las dos anteriores, la diferencia radica en que esta ataca aleatoriamente a los nodos y luego evalúa la falla en cascada, por lo cual hace uso de la función **remove-aleatory**.

- **edge-aleatory-cascade-failure**: Esta función es análoga a las tres anteriores, pero está ataca aleatoriamente los enlaces y luego evalúa la falla en cascada. El codigo comentado de estas dos funciones se encuentra en 15.3.

En el siguiente enlace de Github se encuentran todas las funciones desarrolladas en este objetivo y en todo el trabajo en general. `https://github.com/juancahica/Networks-Statistical-Mechanics-NetStPY`

12.4. Resultados cuarto objetivo específico

Dado que nos interesa comparar el algoritmo para generar grafos con máxima entropía dada una distribución de grado 2.8.1 con el formalismo aquí presentado en el segundo objetivo 12.2, 15.2. Es necesario traducir la distribución de grado a una secuencia de grado promedio, esto es posible ya que la distribución de ley de potencias con exponente $\gamma = -3.4$ considerada para este objetivo tiene promedio definido[47]. Esta traducción se hizo generando un vector con probabilidades acumuladas acorde a la distribución requerida usando la función **pareto-cum-probabilities**, a partir de este vector se usaron las siguientes dos funciones **Degree-Sec-Generator** y **Deg-Sec-Prom** donde la segunda hace uso de la primera para muestrear secuencias de grado a partir del vector de probabilidades acumuladas y promediarlas. Se puede encontrar una descripción de las dos primeras funciones en 15.3 y de la última en 15.4.

Una vez obtenida la secuencia de grado promedio a partir del procedimiento anterior, el sistema de ecuaciones 49 se solucionó usando la función **mult-lagrange-canonico** 15.2. Luego de obtener los multiplicadores de Lagrange se puede reemplazar en la ecuación 22 y obtener las componentes de la matriz de adyacencia promedio, lo cual se hizo con la función **canonical-matrix**, los resultados se presentan a continuación. El codigo que sustenta este resultado se encuentra en 15.4.

La matriz de adyacencia promedio en el ensamble canónico de grafos simples, de 150 nodos, sin autoloops y con una distribución de grado con forma funcional de ley de potencias y exponente -3.4 es:

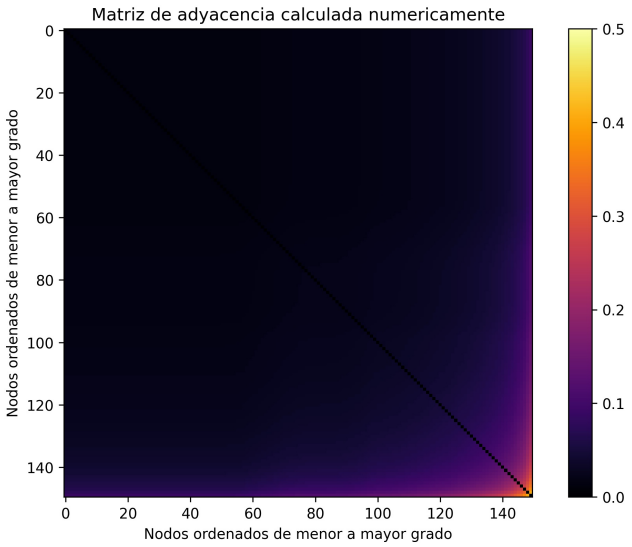

Figura 3: Matriz de adyacencia canónica calculada a partir de la expresión 48

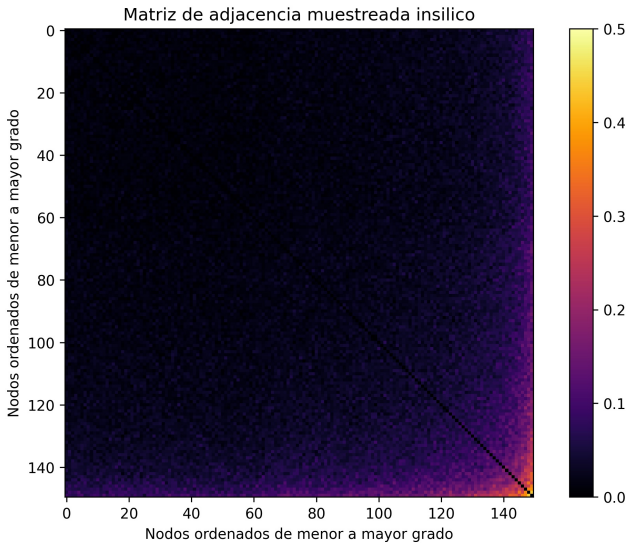

Figura 4: Matriz de adyacencia muestreada in-silico usando el algoritmo para generar redes de máxima entropía descrito en 2.8.1

Donde los nodos se ordenaron de menor a mayor grado y el colorbar se dispuso de 0 a 0.5 para una mejor visualización.

A continuación se presenta la matriz de adyacencia muestreada in-silico por medio del algoritmo para generar redes de máxima entropía 2.8.1, esto se hizo por medio de la función **muestra-ensamble-maxent**, una descripción de la función y el código comentado se puede encontrar en 15.4. Para contrastar el parecido entre estas dos matrices se presenta una matriz diferencia producto de la diferencia entre las dos matrices y una gráfica de la probabilidad de conexión entre el nodo 90 con más grado y los otros nodos, es decir una gráfica de la fila 90 de la matriz de adyacencia. En 15.4 se encuentra el código que produce las figuras 4, 5 y 6.

Al restar esta matriz con la obtenida numéricamente en el segundo objetivo 12.2, 3 se obtuvo:

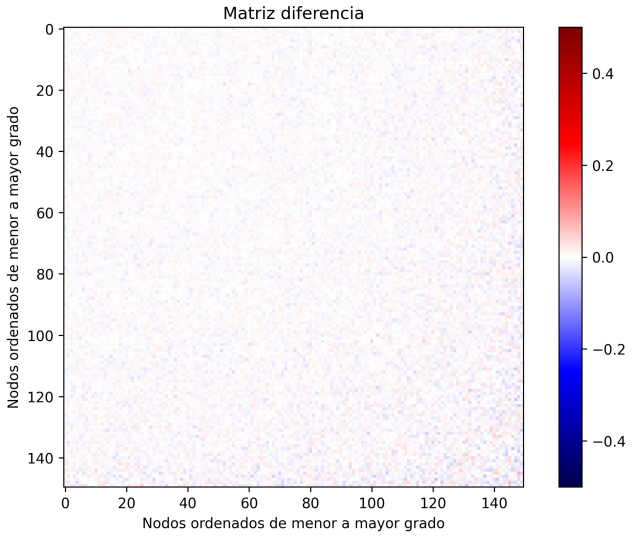

Figura 5: Matriz diferencia

En estas dos matrices se ordenaron los nodos de menor a mayor grado y el color bar se limito al rango 0 a 0.5 para una mejor visualización.

La gráfica de la probabilidad de conexión entre el nodo 90 y los demás nodos i.e. la fila 90 de las matrices de adyacencia promedio:

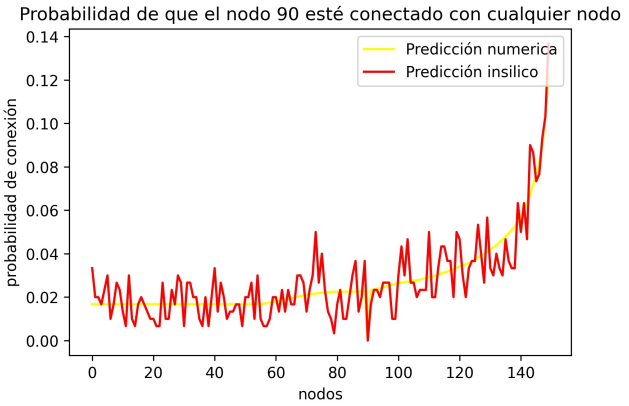

Figura 6: Probabilidad de conexión del nodo 90

En 5 se evidencia que las diferencias en las matrices halladas insilico y numéricamente son mínimas. Está concordancia se puede observar de forma más clara en la predicción de la probabilidad de estar conectado un nodo con otros nodos, tal como se muestra en 6 donde claramente se observa que el resultado insilico oscila alrededor del resultado numérico.

12.5. Resultados quinto objetivo específico

A continuación se presentan las entropías para varías medidas nodales y los resultados de las fallas en cascada causadas por el ataque deliberado y aleatorio a nodos y enlaces para las redes de Watts-Strogatz, Barabási-albert y libre de escala con máxima entropía.

12.5.1. Entropías de distribuciones de propiedades topológicas nodales

A continuación se presentan las entropías para las distribuciones de las medidas nodales expuestas en 2.4, para su calculo se uso el algoritmo **graph-entropys** el cual calcula las medidas nodales para cada grafo, posteriormente normaliza los valores de estas medidas para que la suma de cada medida sobre todos los nodos de un grafo de 1, luego se calculo la entropía de shannon de estas distribuciones usando la ecuación 13. El código comentado de la función establecida para calcular las entropías se puede encontrar en 15.3 y una descripción de cada medida y como calcularla se encuentra en 2.4.

Entropías	B - A	Max Ent	W - S
Degree	4.736	4.908	5.001
Eigenvector	4.747	4.769	4.968
Betweenness	3.778	4.497	4.689
Closeness	5.004	5.005	5.007
Load	3.826	4.516	4.679

Cuadro 2: Entropias para las distribuciones de medidas nodales Degree, Eigenvector, Betweenness, Closeness y Load

A lo largo de todas las medidas es consistente que las redes generadas por el algoritmo de Watts-Strogatz tienen mayor entropía que las generadas por el algoritmo de Barabási y el de máxima entropía. A su vez la red generada por el algoritmo de máxima entropía presento en todas las medidas una mayor entropía que la red de Barabási. Resalta la medida Closeness en la cual las diferencias son del orden de 10^{-3}.

12.5.2. Ataques deliberados a los nodos

Para los análisis de robustez aquí presentados se usaron las funciones definidas en el tercer objetivo 12.3, 15.3. En este caso se usaron los mismos parámetros que en el articulo [10], Capacidad inicial = 1 y resiliencia = 1.1 . Para generar la red de máxima entropía se supuso que la distribución de grado de la red de Barabási es una distribución de Pareto, luego se midió el exponente según lo delineado en [2], [48] y luego se generó la red de máxima entropía imponiéndole el exponente medido de la red de Barabási, exponente que para este caso tomo el valor de $\gamma = 2.541$. Para las dos redes se impuso un grado mínimo $kmin = 3$ y para la red de Watts Strogatz se impuso un grado promedio de 6 y una probabilidad de reconexión de 0.1, tal como en [1]. Los tres grafos tenían 150 nodos y 395 ± 60 enlaces. En 15.5 se encuentra el código comentado que sustenta los resultados expuestos en este objetivo.

Se presenta la gráfica con la población de nodos sobrevivientes para cada ataque dirigido a los nodos de la red, en este caso se atacaron los nodos con mayor load usando el algoritmo **hub-cascade-failure**, descrito en 15.3.

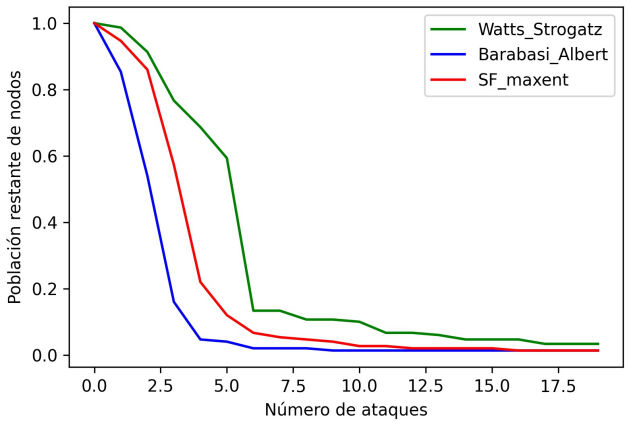

Figura 7: Porcentaje de sobrevivientes en la red vs número de ataques dirigidos

Donde la suma de los % de población sobrevivientes para cada ataque fueron: Barabási-Albert = 45.033, Máxima entropía = 52.233 y Watts-Strogatz = 55.539.

Las redes de W-S son las más robustas ante ataques deliberados, seguidas de las redes libres de escala con máxima entropía y finalmente las generadas por preferential attachment, esto a su vez correlaciona con las entropías nodales para la mayoría de las medidas, en este caso a mayor entropía mayor robustez.

12.5.3. Ataques aleatorios a los nodos

Se presenta la gráfica con la población de nodos sobrevivientes para cada ataque aleatorio a los nodos de la red, donde se uso el algoritmo **aleatory-cascade-failure** descrito en 15.3. Dado el carácter aleatorio de este tipo de ataque los datos presentados son el promedio a lo largo de 60 realizaciones.

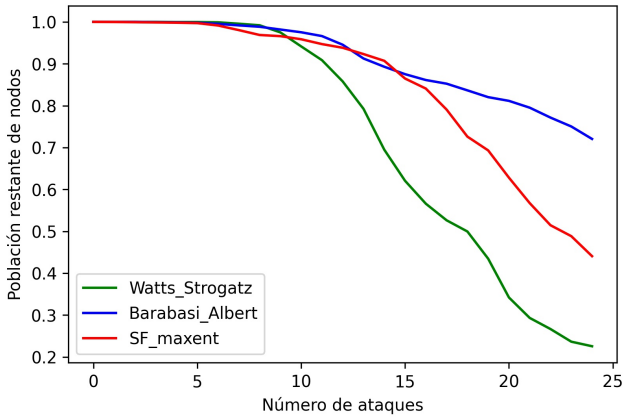

Figura 8: Porcentaje de sobrevivientes en la red vs número de ataques aleatorios

Donde la suma de los % de población sobrevivientes para cada ataque fueron: Barabási-Albert = 65.972, Máxima entropía = 47.566 y Watts-Strogatz = 31.703.

En este caso observamos que las redes libres de escala tienen mayor robustez ante ataques aleatorios que la de Watts-Strogatz a su vez la red de Barabási se presenta más robusta que la de máxima entropía. Note que la red con mayor entropía es la menos robusta ante ataques aleatorios a los nodos y la red con menor entropía es la más robusta, en este caso la relación entropía - robustez es opuesta a la encontrada para ataques dirigidos a los nodos.

12.5.4. Ataques deliberados a los enlaces

Se presenta la gráfica con la población de nodos sobrevivientes para cada ataque dirigido a los enlaces de la red, en este caso se atacaron los enlaces con mayor load usando el algoritmo **edge-hub-cascade-failure** descrito en 15.3.

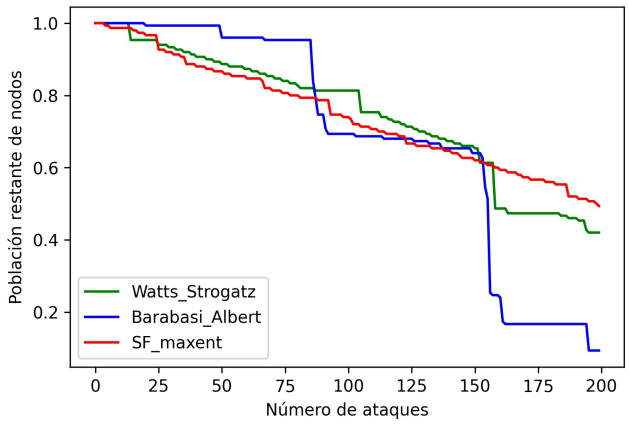

Figura 9: Porcentaje de sobrevivientes en la red vs número de ataques aleatorios

Donde la suma de los % de población sobrevivientes para cada ataque fueron: Barabási-Albert = 352.034, Máxima entropía = 348.359 y Watts-Strogatz = 320.761.

En esta gráfica se distingue que para pocos ataques la red de Barabasi-Albert se evidencia más robusta que las otras dos, sin embargo en la medida en que aumenta el número de ataques llega un punto donde la población de nodos en la componente principal de la red de Barabási-Albert decae mucho más rápidamente de lo que lo hacen las poblaciones de las redes de Watts-Strogatz y máxima entropía. En este caso la suma de los % de poblaciones de nodos sobrevivientes para cada ataque presenta a la red de Barabási como la más robusta, esto asociado a su robustez ante los primeros 80 ataques, luego la red de máxima entropía y finalmente la red de Watts-trogatz. En este caso no se observa

ninguna correlación con las entropías.

12.5.5. Ataques aleatorios a los enlaces

Se presenta la gráfica con la población de nodos sobrevivientes para cada ataque aleatorio a los enlaces de la red, donde se usó la función **edge-aleatory-cascade-failure** descrita en 15.3. Dado el carácter aleatorio de este tipo de ataque los resultados presentados son el promedio sobre 15 realizaciones.

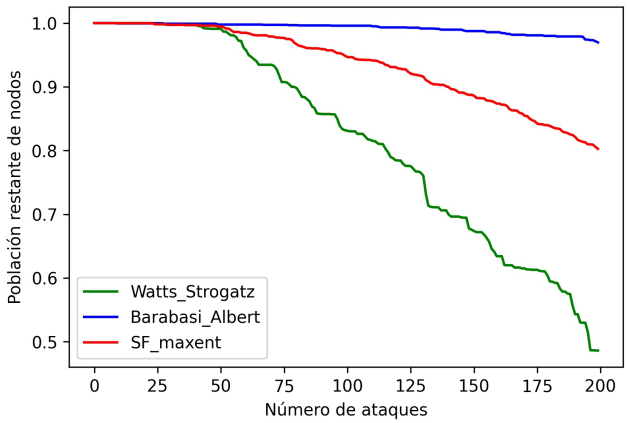

Figura 10: Porcentaje de sobrevivientes en la red vs número de ataques aleatorios

Donde la suma de los % de población sobrevivientes para cada ataque fueron: Barabási-Albert = 213.047, Máxima entropía = 198.959 y Watts-Strogatz = 170.814.

En este caso observamos el mismo patrón que en ataques aleatorios a los nodos, las redes libres de escala presentan mayor robustez que las redes de Watts-Strogatz y la red de Barabási-Albert presenta más robustez que la de máxima entropía. En este caso si se presenta relación entropía robustez, a mayor entropía menor robustez ante ataques aleatorios a los enlaces.

13. Análisis de resultados

13.1. Análisis primer objetivo específico

Como vemos, en la ecuación 20, el termino α es siempre mayor a 2 teniendo en cuenta que a/c siempre es una cantidad positiva como se sugiere en otros desarrollos conocidos en la literatura de network science [16], esto es debido a las condiciones impuestas en donde desde la ecuación 23 sesgamos la red con el fin de que se observe un comportamiento en donde poco nodos tienen muchas conexiones y muchos nodos tienen pocas conexiones, derivando esto en un comportamiento de ley de potencias; en el termino a/c la constante c representa el grado promedio de la red; en el caso en el que a tenga la restricción de presentar su valor máximo como el grado promedio de la red c, tendremos que el coeficiente α se encuentra entre el rango $[2, 3]$ como se sugiere en [3].

Otro factor importante que hay que tener en cuenta es que debido a la aproximación que se aplicó para obtener la forma de la ley de potencias vista en la ecuación 20, en la que se asumió que q es mucho mayor a a para representarla de esta forma, se debe tener en cuenta que esta expresión solamente es valida para grados altos, por lo tanto, la forma de la distribución de ley de potencias de la ecuación 20 solo es valida a partir de cierto valor, mientras que la ecuación 46 ajustaría mejor para todos los valores.

Sin embargo hay que recordar que aún así la ecuación 46 fue obtenida a través del uso de la aproximación de Stirling 44 sobre la función beta de Euler, en donde esta aproximación solo funciona para valores de x grandes, razón por la cual vemos que la ley de potencias en su forma 20 solo se ajusta en valores puestos en la cola de la función, razón por la que consideramos que la expresión 42 representa la distribución de grado en su forma más precisa, aunque su representación por medio del uso de funciones gamma la hace una función desventajosa para trabajar desde la parte analítica también es importante recalcar que para trabajos numéricos en donde se requiera precisión y se trabaje con pocos nodos no presentaría mayores problemas.

13.2. Análisis segundo objetivo específico

A partir de las funciones establecidas en este objetivo, las cuales incorporan los resultados analíticos desarrollados se puede partiendo de una secuencia de grado obtener la matriz de adyacencia promedio predicha por el ensamble canónico 22.

13.3. Análisis tercer objetivo específico

En el repositorio de Github 12.3 se presenta NetStPY repositorio de mecánica estadística enfocado en redes, aquí se presentan todas las funciones usadas en este trabajo, por lo cual este repositorio permitirá a sus usuarios en pocas lineas de código muestrear ensambles de redes (i.e. generar grafos) imponiendo una distribución de grado elegida en promedio, calcular la matriz de adyacencia promedio del ensamble de forma numérica, ejecutar ataques y fallas en cascada a nodos y enlaces de forma deliberada y dirigida. Sin embargo para hacer uso de estas funciones hay que tener en cuenta que el desempeño de los algoritmos para generar redes aleatorias con una secuencia de grado dada depende fuertemente

de como se muestree esta secuencia [49], por lo cual, tal como está presentado los algoritmos es necesario modificar la función **Degree-Sec-Generator** con el fin de muestrear otro tipo de distribuciones, esto porque para el caso particular de este trabajo (Pareto con exponente entre -2 y -3) la distribución posee una varianza divergente [47], cuestión que sesga los muestreos al generar muchos valores atípicos en los mismos. Estos valores atípicos resultan en la generación de redes libres de escala artificialmente débiles ante ataques dirigidos [47], razón por la cual fue necesario imponer un cutoff 15.3. Las funciones para ejecutar la falla en cascada y la medida de entropía están listas para usar en cualquier grafo no dirigido.

13.4. Análisis cuarto objetivo específico

En 3 se distingue una región reducida de la matriz en la cual es mucho más probable encontrar un enlace que en el resto, si consideramos que los nodos están ordenados de menor a mayor grado se concluye que entre los nodos de mayor grado es más probable encontrar un enlace que entre nodos de bajo grado. El comportamiento de ley de potencias se ve reflejado en lo reducida de la región de alta probabilidad de enlace en la matriz, puesto que los nodos con alto grado son pocos, entonces la región de alta probabilidad de enlace es reducida, esto concuerda con el comportamiento cualitativo esperado de la matriz de adyacencia de redes con distribución de grado libre de escala.

Comparando 4 y 3 cualitativamente observamos bastantes similitudes, las diferencias capturadas en 5 muestran que las dos matrices en efecto son similares, por otra parte 6 nos muestra claramente como la predicción de la probabilidad de conectividad del nodo 90 con los otros nodos encontrada insilico oscila alrededor de la predicción numérica producto de 48, validando así la equivalencia entre el algoritmo para generar redes de máxima entropía dada una distribución de grado y el formalismo del ensamble canónico imponiendo como secuencia de grado promedio sobre el ensamble el promedio de las secuencias de grado muestreadas de la distribución.

13.5. Análisis quinto objetivo específico

Los resultados presentados en 7 y 8 evidencian la propiedad robustez pero fragilidad de las redes libres de escala ante ataques a los nodos que ha sido ampliamente reportada en la literatura [2],[10]. A su vez se encuentra mayor robustez ante ataques dirigidos en la red de máxima entropía que en la de Barabási, evidenciando así el sesgo hub-centrico del algoritmo Preferential Attachment al muestrear el ensamble de redes libres, tal como se reportó en [2].

Al tener en cuenta 10 y 9 se evidencia que ante ataques a los enlaces tanto dirigidos como aleatorios las redes libres de escala se presentan más robustaz que la red de Watts-Strogatz, tal como se evidencia en el valor de la suma de los % de población de nodos sobrevivientes para cada ataque. Sin embargo en la gráfica 9 se evidencia que el comportamiento de la red de Barabási es radicalmente distinto al de las otras dos redes, en especial porque a medida que se remueven los enlaces con mayor Load Centrality la población de nodos en la red de máxima entropía y Watts- Strogatz decae poco a poco, sin embargo en la red de Barabási se encuentra que dado cierto número de ataques la población se mantiene prácticamente igual, luego en relativamente pocos ataques cae la población rápidamente, este comportamiento se observa 3 veces en los 200 ataques realizados. Este comportamiento se asocia al carácter hub-centrico de la red de Barabási, puesto que los nodos hubs son por los que atravesarán más geodésicas y entonces puntuaran una mayor medida de Load Centrality, es de esperar que sus enlaces también posean un alto valor de Load Centrality, de tal forma que si se le quitan algunos enlaces a estos hubs la red se mantiene porque estos pueden redirigir las geodésicas a través de sus otros enlaces, esto hasta que se llega al punto en donde los enlaces de los hubs se sobrecargan causando una falla en cascada que causa la rápida caída de la población.

Al atacar aleatoriamente a los enlaces 10 la red de Watts-Strogatz es particularmente menos robusta que las redes libres de escala, esto lo asociamos con la robustez ante errores dado por los hubs de las redes libres de escala y al hecho de que la cohesión de la red de Watts-Strogatz se asegura por algunos pocos enlaces de largo rango que se han dispuesto aleatoriamente, por lo cual si el ataque aleatorio da con alguno de estos enlaces de largo rango, desembocará en una fuerte falla en cascada, puesto que a través de estos enlaces fluyen muchas geodésicas, puntuando así un alto valor en la medida Load Centrality y por lo tanto causando una fuerte redirección de las geodésicas en la red si se eliminan.

En [1] reportan un comportamiento opuesto de la robustez ante ataques deliberados de las redes de Watts-Strogatz y la de Barabási al atacar sus nodos y enlaces, en el estudio de robustez desarrollado en este trabajo se obtuvieron los mismos resultados 9, 7 tanto para la red de Barabási como para la de máxima entropía, concluyendo así que los resultados presentados en [1] a pesar de haber sido hechos con redes de Barabási si son representativos de la relación en términos de robustez entre las redes libres de escala y las de Watts-Strogatz. Este comportamiento inverso se asocia a que mientras la red de Barabási es hub-centrica es decir contiene nodos particularmente importantes para la cohesión de la red, se

puede considerar a la red de Watts-Strogatz edge-hub-centrica en el sentido de que contiene enlaces particularmente importantes para la cohesión de la red.

Finalmente, encontramos que la medida de entropía para las distribuciones de todas las medidas nodales concuerdan en que la red de Watts-Strogatz es más aleatoría que las redes libres de escala, a su vez la red de Barabási se presento como de menor entropía que la red con distribución de grado de ley de potencias y máxima entropía. Estos resultados de entropía correlacionan con la robustez ante ataques dirigidos a los nodos y anticorrelacionan con la robustez ante ataques aleatorios a los nodos. De tal forma que a mayor entropía mayor robustez ante ataques dirigidos y menor ante ataques aleatorios. En el caso de ataques a los enlaces tanto para ataques dirigidos o aleatorios hay una anticorrelación con la entropía de las medidas nodales, de tal forma que a menor entropía mayor robustez ante ataques a los enlaces. Sin embargo el comportamiento anómalo de la red de Barabási en 9 exige un estudio de está relación robustez entropía más sofisticado.

14. Conclusiones

En este trabajo se valido analíticamente las limitaciones del algoritmo Preferential Attachment para muestrear el ensamble de redes libres de escala, a su vez se valido el desempeño de otro algoritmo para muestrear el ensamble de redes libres de escala con máxima entropía, encontrando así una equivalencia entre este algoritmo y los desarrollos teóricos del ensamble canónico para grafos simples imponiendo una secuencia de grado promedio. Posteriormente se usaron estos dos algoritmos junto al de Watts-Strogatz para generar redes con topología de pequeño mundo y libre de escala; y se estudio su robustez ante fallas en cascada causadas por ataques aleatorios y dirigidos a los nodos y enlaces.

Como resultado de esta comparación se encontró que algunas de las propiedades de robustez de las redes generadas por Preferential Attachment no son típicas del ensamble de redes libres de escala, lo que se atribuye al hecho de que estas son particularmente hub-centricas, a su vez está propiedad de las redes de Barabási las hace particularmente robustaz ante ataques tanto dirigidos como aleatorios a los enlaces, este último hallazgo evidencia la importancia de ampliar los estudios que caractericen la robustez de las redes para también considerar ataques a los enlaces.

La propiedad de robustez pero fragilidad también se exhibió en las redes de máxima entropía en comparación con las de Watts-Strogatz pero en una menor medida que en la red de Barabási. Además se encontró que la red de Watts-Strogatz en comparación a las libres de escala es particularmente débil ante ataques aleatorios tanto a nodos como a enlaces.

Finalmente se calculo la entropía para la distribución de las medidas nodales Load, Degree, Betweenness, Closeness y Eigenvector, encontrándose consistentemente que la red de Watts-Strogatz presenta mayor entropía en estas distribuciones que la red de máxima entropía y a su vez está presenta mayor entropía que la red de Barabási. Al comparar estos resultados de entropía nodal con los de robustez se evidenció que a mayor entropía mayor robustez ante ataques dirigidos a los nodos, pero mayor fragilidad ante ataques aleatorios a los mismos. En el caso de ataques a lo enlaces se encontró que a menor entropía de las distribuciones de las medidas nodales mayor robustez ante ataques a los enlaces.

Finalmente se presenta NetStPY, un repositorio en Github de mecánica estadística de redes, en el cual se incorporaron todas las funciones desarrolladas en este trabajo.

15. Anexos

A continuación se presentan los desarrollos de cada uno de los objetivos específicos.

15.1. Desarrollo analítico del proceso fuera del equilibrio preferential attacthment

Consideramos alguna de la notación usada para el desarrollo analítico de una red clasificada como preferential attachtment, el grado de un nodo i en la red se representa como q_i, $p_q(n)$ representa la fracción de nodos en la red que tienen grado q y por lo tanto representa la distribución de grado sobre la red. [3]

Examinamos el caso en el que decidimos añadir un solo nodo a la red, consideremos una de las conexiones que puede llagar a hacer este nodo, siguiendo la definición del modelo de preferential attachtment, la probabilidad de que la conexión sea con un nodo i de la red es proporcional a una cantidad $q_i + a$ con a una constante positiva, debido a que la conexión es con otro nodo esta probabilidad debió ser normalizada de tal manera que la suma sobre todos los posibles nodos de la red i resulta ser 1. Para la deducción de esta forma debemos tener en cuenta que estamos trabajando sobre un escenario de probabilidad laplaciano en donde una probabilidad es proporcional al cociente entre casos favorables y casos posibles [50]. Considerando la probabilidad de que un nodo aleatorio j existente en la red tenga una conexión con otro nodo i diferente de si mismo, decimos que la probabilidad de que el nodo j tenga una conexión mediante un eje con el nodo i es proporcional a la cantidad de ejes que tenga el nodo i, de esta forma nos

aseguramos de que si el nodo i en cuestión tiene muchas conexiones sea cada vez más probable que una de esas conexiones sea con el nodo j conectando de esta forma ambos nodos. [3]

Ahora que vimos que esta probabilidad es proporcional al grado del nodo i, denominado como k_i, consideramos que esta proporcionalidad relacionada con el número de conexiones que representaría los casos posibles de que un nodo aleatorio se encuentre enlazado con un nodo i en la probabilidad de laplace, ahora, los casos totales, serían tomados como todos los posibles escenarios que se pueden dar en la red, es decir, se tendrían que tener en cuenta cada uno de los ejes de cada uno de los nodos que conforman la red, de esta forma se apoya el hecho de que la normalización encontrada en la forma de esta probabilidad, ya que si en el caso límite de que un nodo tenga un número alto de conexiones y los demás un número pequeño veremos que la probabilidad de que un nodo j se encuentre conectado con un nodo i se aproximara a un valor unitario ya que lo más probable en este escenario seria que un nodo se encuentre conectado a ese único nodo central y no a otro nodo aledaño en la red. La forma que tendría esta probabilidad sería la siguiente, con k_i el grado del nodo número i y N el número de nodos en la red: [3]

$$P_i = \frac{k_i}{\sum_j^N k_j} \tag{23}$$

En el caso en el que se consideren nuevos nodos en la red, incluiremos esta información como un aumento en el número de conexiones representado por una constante a, esto significa que el grado k_i ahora tendrá un aumento, de tal manera que tendremos que $k_i \to k_i + a$, con lo cual si representamos es grado k_i como p_j, obtenemos la forma de esta probabilidad:

$$P_i = \frac{q_i + a}{\sum_j (q_j + a)} \tag{24}$$

Tomamos el termino $< q >= n^{-1} \sum_i q_i = c$ como el grado promedio de la red y remplazamos sobre la ecuación 24 una vez la multiplicamos por un factor n/n, obtuvimos que:

$$\frac{q_i + a}{\sum_i (q_i + a)} = \frac{q_i + a}{n < q > + na} = \frac{q_i + a}{n(c + a)} \tag{25}$$

Cada nuevo nodo que decidimos introducir en la red tiene conexiones con otros c nodos en promedio, entonces el número esperado de nuevas conexiones de un nodo existente i tras la aparición de un nuevo nodo sería c veces la ecuación 25, si tenemos en cuenta que este se calcula mediante $(promedio) \times (Probabilidad)_i$. La siguiente ecuación representa el número de nodos con grado q en la red, por lo tanto el número esperado de nuevas conexiones a todos los nodos con grado q es:

$$np_q(n) \times c \times \frac{q_i + a}{n(c + a)} = c \frac{q_i + a}{n(c + a)} p_q(n) \tag{26}$$

Con los desarrollos anteriores fue posible utilizar la master equation para trabajar la evolución de la distribución de grado. Cuando añadimos un solo nodo a una red con n nodos, el número de nodos con grado q aumenta en uno por cada nodo previo en la red con grado $q - 1$ que recibe una conexión, volviéndose un nodo de grado q; por la ecuación 26 sabemos que el numero esperado de tales nodos es el siguiente:

$$\frac{c(q - 1 + a)}{c + a} P_{q-1}(n) \tag{27}$$

De forma similar, perdemos un nodo de grado q cuando este recibe una nueva conexión volviéndose un nodo de grado $q + 1$, luego el número esperado de tales nodos es el siguiente:

$$\frac{c(q + a)}{c + a} P_q(n) \tag{28}$$

El número de nodos con grado q en la red después de la adición de un nuevo nodo $(n + 1)p_q(n + 1)$ pudo ser considerado como la componente P_j que cambia con el tiempo visto en la master equation 11, de la misma forma todos lo términos de la forma $Np_q(N)$ considerados en las ecuaciones 26, y 28 representarían los términos representativos del vector P_j, mientras que los otros términos que acompañaban a estas ecuaciones representan la matriz (Para este caso vector) de transición de estados, utilizando la master equation presentamos el siguiente desarrollo:

$$\frac{dP_o}{dt} = \sum_j A_{oj} P_j \tag{29}$$
$$\frac{dP_o}{dt} = A_{o1} P_1 + A_{o2} P_2 + A_{o3} P_3$$

Junto a los siguientes remplazos:

$$\frac{dP_o}{dt} \longrightarrow (n + 1)p_q(n + 1) \tag{30}$$
$$A_{o1} P_1 \longrightarrow np_q(n)$$

$$A_{o2}P_2 \longrightarrow \frac{c(q-1+a)}{ca}p_{q-1}(n)$$
$$A_{o3}P_3 \longrightarrow -\frac{c(q+a)}{c+a}p_q(n) \tag{31}$$

Obtuvimos la evolución en el tiempo del número de nodos sobre la red:

$$(n+1)p_q(n+1) = np_q(n) + \frac{c(q-1+a)}{ca}p_{q-1}(n) - \frac{c(q+a)}{c+a}p_q(n) \tag{32}$$

El primer termino del lado derecho de la ecuación representa el número de nodos previamente de grado q, el segundo termino representa los nodos ganados y el tercer termino los nodos perdidos.

La ecuación 32 aplica para cualquier valor de q excepto cuando este es 0, en el caso de que $q = 0$ no hay nodos de un grado más bajo que puedan ganar un conexión para volverse nodos de grado cero, por lo tanto el segundo termino de la ecuación 32 no debería aparecer. Por otro lado, ganamos un nodo de grado cero siempre y cuando un nuevo nodo sea añadido a la red, debido a que los nodos no tienen conexiones cuando entran a la red. la ecuación apropiada para el caso de $q = 0$ es la siguiente:

$$(n+1)p_o(n+1) = np_o(n) + 1 - \frac{ac}{c+a}p_o(n) \tag{33}$$

Si consideramos el limite en el que tenemos un numero muy grande de nodos ($n \to \infty$), las ecuaciones 32 y 33 toman las siguientes formas:

$$p_q = \frac{c}{c+a}[(q-1+a)p_{q-1} - (q+a)p_q] \tag{34}$$
$$p_q = \frac{q+a-1}{q+a+1+a/c}p_{q-1}$$

$$p_0 = 1 - \frac{ca}{c+a}p_o = \frac{1+a/c}{a+1+a/c} \tag{35}$$

Usando la ecuación 34 pudimos conocer el termino p_1, pero para esto se nos exigió conocer p_o, de tal forma que para esta exigencia usamos la ecuación 35, una vez se conoció el terminó p_1 es posible conocer el termino p_2 de manera análoga, y luego el termino p_3, y así sucesivamente:

$$p_1 = \frac{a}{a+2+a/c}\frac{1+a/c}{a+1+a/c} \tag{36}$$

$$p_2 = \frac{(a+1)a(1+a/c)}{(a+3+a/c)(a+2+a/c)(a+1+a/c)} \tag{37}$$

$$p_3 = \frac{(a+2)(a+1)a(1+a/c)}{(a+4+a/c)(a+3+a/c)(a+2+a/c)(a+1+a/c)} \tag{38}$$

A partir de la visualización de los anteriores términos, se supo que el termino p_q, tenia la siguiente forma:

$$p_q = \frac{(q+a-1)\cdots a(1+a/c)}{(q+a+1a/c)\cdots(a+2+a/c)(a+1+a/c)} \tag{39}$$

La anterior ecuación es la solución completa para el modelo de prefential attachment, sin embargo, si queremos visualizar una distribución de grado en forma de ley de potencias debemos de usar la función gamma definida como:

$$\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt \tag{40}$$

Esta función cumple con $\Gamma(x+1) = x\Gamma(x)$, de manera iterativa haciendo $x \to x+1$, tendríamos $\Gamma(x+2) = x(x+1)\Gamma(x)$, en el limite en que consideramos $x \to n$, tenemos:

$$\frac{\Gamma(x+n)}{\Gamma(x)} = (x+n-1)(x+n-2)\cdots(x+1)x \tag{41}$$

Debido a que la ecuación 39 puede llegarse a escribir al igual que la ecuación 41 como una productoria de términos, podemos expresar la ecuación 39 en términos de funciones gamma de la siguiente forma:

$$p_q = (1+a/c)\frac{\Gamma(q+a)\Gamma(a+1+a/c)}{\Gamma(a)\Gamma(q+a+2+a/c)} \tag{42}$$

Esta expresión puede ser simplificada en términos de la función beta de Euler, ($B(x,y) = \Gamma(x)\Gamma(y)/\Gamma(x+y)$) si multiplicamos el numerador y denominador de la ecuación 42 por $\Gamma(2+a/c) = (1+a/c)\Gamma(1+a/c)$, encontramos que:

$$p_q = \frac{\Gamma(q+a)\Gamma(2+a/c)}{\Gamma(q+a+2+a/c)} \times \frac{\Gamma(a+1+a/c)}{\Gamma(a)\Gamma(1+a/c)} \tag{43}$$
$$p_q = \frac{B(q+a, 2+a/c)}{B(a, 1+a/c)}$$

Reescribimos la anterior ecuación mediante una aproximación de la función gamma usando la aproximación de Stirling, la aproximación que usaremos será la siguiente: [46]

$$\Gamma(x) \approx \sqrt{2\pi} e^{-x} x^{x-\frac{1}{2}} \tag{44}$$

Mediante el uso de la anterior ecuación pudimos reescribir la función beta de Euler para después utilizarla en la ecuación 43 de esta manera:

$$B(x,y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)} \tag{45}$$
$$B(x,y) \approx \frac{\sqrt{2\pi} e^{-x} x^{x-\frac{1}{2}} \Gamma(y)}{\sqrt{2\pi} e^{-(x+y)} (x+y)^{(x+y)-\frac{1}{2}}}$$
$$B(x,y) \approx \frac{e^{-x} x^{x-\frac{1}{2}}}{e^{-(x+y)} (x+y)^{(x+y)-\frac{1}{2}}} \Gamma(y)$$
$$B(x,y) \approx \frac{e^{-x} x^{x-\frac{1}{2}}}{e^{-(x+y)} x^{x+y-\frac{1}{2}} e^{y}} \Gamma(y)$$
$$B(x,y) \approx x^{-y} \Gamma(y)$$

Utilizando la ecuación 45 sobre nuestra distribución de grado 43 obtuvimos lo siguiente tomando todos los términos que dependieran de a y c como una constante:

$$p_q \approx \frac{(q+a)^{-(2+a/c)} \Gamma(2+a/c)}{a^{-(1+a/c)} \Gamma(1+a/c)} \tag{46}$$
$$p_q \approx \frac{(q+a)^{-(2+a/c)}}{a^{-(1+a/c)}} \frac{e^{-1}(2+\frac{a}{c})^{3/2+\frac{a}{c}}}{(1+\frac{a}{c})^{\frac{1}{2}+\frac{a}{c}}}$$
$$p_q \approx \frac{e^{-1}(2+\frac{a}{c})^{\frac{3}{2}+\frac{a}{c}}}{(1+\frac{a}{c})^{\frac{1}{2}+\frac{a}{c}}} (q+a)^{-(2+\frac{a}{c})}$$
$$p_q \approx K(a,c)(q+a)^{-(2+\frac{a}{c})}$$

Consideramos que q es mucho mayor que a, por lo tanto, nuestra distribución de grado tiene la forma de una distribución de ley de potencias como se esperaba con $\alpha = 2 + \frac{a}{c}$: [3]

$$p_q \approx k q^{-\alpha} \tag{47}$$

El valor de α encontrado debe ser mayor o igual a 2, este resultado es consistente con el modelo de Price y otros desarrollos vistos en la literatura encontrada[16].

15.2. Desarrollo analítico del ensamble canónico y calculo numérico de matriz de adyacencia promedio

El algoritmo presentado en [2] muestrea secuencias de grado a partir de una distribución de grado dada usando el método montecarlo para posteriormente generar grafos con estas secuencias de grado. Para la validación analítica, por medio de muestreo se aproximo la secuencia de grado promedio resultante al imponer una distribución de grado dada, traduciendo así la imposición de tener en promedio una distribución de grado a la imposición de tener en promedio una secuencia de grado. La secuencia de grado $\boldsymbol{k_i}$ que se impondrá se extrajo muestreando una distribución de grado con forma funcional de ley de potencias y exponente -2.4.

Consideremos el conjunto de todos los grafos simples $\wp$ con n vértices y definamos el ensamble dándole una probabilidad a cada grafo tal que:

$$\sum_{G \in \wp} P(G) = 1$$

El valor esperado del grado del nodo i-ésimo $\langle k_i \rangle$ al interior de este ensamble está dado por:

$$\langle k_i \rangle = \sum_{G \in \wp} P(G) k_i(G)$$

Ahora impongamos la condición de que los valores esperados de los grados de los nodos correspondan a la secuencia de grado extraída del muestreo

$$\boldsymbol{k_i} = \langle k_i \rangle = \sum_{G \in \wp} P(G) k_i(G)$$

donde $\boldsymbol{k_i}$ corresponde a los valores de la secuencia de grado extraída del muestreo.

El siguiente paso es usar el método de multiplicadores de Lagrange para encontrar la función P(G) de probabilidad que cumpla las siguientes condiciones:

$$S_{max} = -\sum_{G \in \wp} P(G) ln P(G)$$
$$\sum_{G \in \wp} P(G) = 1$$
$$\boldsymbol{k_i} = \sum_{G \in \wp} P(G) k_i(G)$$

Esto nos lleva a que la distribución P(G) buscada es tal que maximiza la cantidad

$$-\sum_{G\in\wp} P(G)lnP(G) + \alpha\left[1-\sum_{G\in\wp} P(G)\right] + \\ +\sum_i \theta_i\left[\boldsymbol{k_i} - \sum_{G\in\wp} P(G)k_i(G)\right]$$

Donde α y θ_i corresponden a los multiplicadores de lagrange. Derivando respecto a la probabilidad P(G) e igualando a 0

$$lnP(G) + 1 + \alpha + \sum_i \theta_i k_i(G) = 0$$

de donde obtenemos

$$P(G) = exp\left[-(\alpha+1) - \sum_i \theta_i k_i(G)\right]$$

que podemos reescribir como:

$$P(G) = \frac{exp\left[-\sum_i \theta_i k_i(G)\right]}{Z}$$

Donde Z es la función de partición y toma la siguiente forma:

$$Z = e^{\alpha+1} = \sum_{g\in G} e^{-H} = \sum_{g\in G} exp\left[-\sum_i \theta_i k_i(G)\right]$$

Antes de calcular la energía libre sera necesario trabajar con la función de partición, primero note que el hamiltoniano puede reescribirse como:

$$H = \sum_i \theta_i k_i = \sum_{i<j}(\theta_i + \theta_j)\sigma_{ij}$$

Esto posible gracias a que $k_i = \sum_j \sigma_{ij}$ y $\sigma_{ij} = \sigma_{ji}$.

entonces la función de partición toma la forma:

$$Z = \sum_{\sigma_{ij}} exp\left(-\sum_{i<j}(\theta_i + \theta_j)\sigma_{ij}\right)$$

Donde la suma sobre los grafos se cambio por la suma sobre las matrices de adyacencia. Si consideramos que cada matriz de adyacencia no es mas que la asignación de unos y ceros a los elementos σ_{ij}, y teniendo en cuenta las propiedades de la exponencial podemos reescribir la función de partición como:

$$Z = \prod_{i<j}\sum_{\sigma_{ij}=0}^{1} e^{-(\theta_i+\theta_j)\sigma_{ij}} = \prod_{i<j}(1 + e^{(\theta_i+\theta_j)})$$

Definamos Θ_{ij} tal que si $i \neq j$ entonces $\Theta_{ij} = \theta_i + \theta_j$, de otra forma $\Theta_{ij} = 0$. A partir de 15 podemos escribir la energía libre como:

$$F = -\sum_{i<j} ln(1 + e^{-\Theta_{ij}})$$

A partir de la energía libre se calcularon las componentes de la matriz de adyacencia promedio, es decir la probabilidad de que dos nodos estén conectados en ese ensamble i.e. la probabilidad de que el elemento σ_{ij} de un grafo extraído aleatoriamente del ensamble sea igual a 1:

$$p_{ij} = \langle\sigma_{ij}\rangle = \frac{\partial F}{\partial\Theta_{ij}} = \frac{1}{e^{\Theta_{ij}} + 1} \tag{48}$$

Para calcular los multiplicadores de Lagrange θ_i es necesario imponer las N ligaduras, una por cada nodo, para esto consideremos lo siguiente:

$$\boldsymbol{k_i} = \langle k_i\rangle = \langle\sum_j \sigma_{ij}\rangle = \sum_j\langle\sigma_{ij}\rangle = \sum_{j\neq i}\langle\sigma_{ij}\rangle$$

Donde en la última igualdad se usó que para los grafos considerados en este ensamble $\langle\sigma_{ii}\rangle = 0$ y en la penúltima el hecho de que el promedio es sobre el ensamble de grafos. La expresión con la que finalmente se calcularon los multiplicadores de Lagrange fue:

$$\boldsymbol{k_i} = \sum_G k_i(G)P(G) = \sum_{j\neq i}\langle\sigma_{ij}\rangle = \sum_{j\neq i}\frac{1}{e^{\theta_i+\theta_j}+1} \tag{49}$$

Para resolver este conjunto de ecuaciones no lineales y calcular la matriz de adyacencia promedio predicha por el ensamble canónico se establecieron dos funciones, las cuales se presentan a continuación.

Dado que aquí es la primera parte donde se presenta codigo en este trabajo, se mostrará así mismo las librerías a importar, se asumirán importadas ya en todo el resto del código mostrado aquí. A continuación el código comentado.

```
#networkx para teoria de grafos, matplotlib
    para graficar, random para generar numeros
    aleatorios, stats para la entropia y solve
    para optimizar.
import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
import scipy as sp
from scipy import stats as st
import math
from scipy.optimize import fsolve
import powerlaw
```

multp-lagrange-canonico: Está función recibe como input una secuencia de grado y te entrega los multiplicadores según el ensamble canónico habiendo impuesto como ligadura que el grado promedio de los nodos este dado por la secuencia de grado dada como input. Esto lo hace resolviendo numéricamente la ecuación 49 usando la librería scipy.optimize.

```
# este programa te calcula los multiplicadores
    de lagrange dada una secuencia de grado
    acorde al ensamble canonico.
def multp_lagrange_canonico(degree_sequence):
  """
  (degree_sequence) ----> (
    lagrange_multiplicators_canonicalensemble)
  esta funcion te recibe una secuencia de grado
    , te la impone como ligadura en un ensamble
     canonico de grafos simples y
  te entrega un vector con los multiplicadores
    de lagrange ordenados de menor a mayor
    acorde al grado de la restricci n asociada
  """
  #la siguiente funcion contiene el sistema de
    ecuaciones y se define para solucionar
  def lagrange_eq(lagrange_multiplicators,
    degree_sequence):
    n = len(degree_sequence)
    matrix = np.zeros((n,n))
    for i in range(n):
      for j in range(n):
        #esta es la expresion de la matriz de
    adyacencia promedio segun el ensamble
    canonico
        matrix[i][j] = 1/(math.exp(
    lagrange_multiplicators[i] +
    lagrange_multiplicators[j]) + 1)
      matrix[i][i] = 0
    sums = []
    for i in range(n):
      sum = 0
      for j in range(n):
        sum = sum + matrix[i][j]
      sums.append(sum)
    resultado = np.zeros(n)
    for i in range(n):
      resultado[i] = sums[i] - degree_sequence[
    i]
    return resultado

  x0 = np.zeros(len(degree_sequence))
  def lagrange(lagrange_multiplicators):
    k = lagrange_eq(lagrange_multiplicators,
    degree_sequence)
    return k
  #hasta aca lo que se ha hecho es definir las
    funciones a resolver acorde a los
    resultados del ensamble canonico
  #la siguiente linea resuelve el sistema de
    ecuaciones y entrega el resultado en una
    lista
  multiplicators = fsolve(lagrange,x0)
  return multiplicators
```

canonical-matrix: Está función construye una lista de listas con las componentes de la matriz de adyacencia promedio predicha por el ensamble canónico tal como se establece en la ecuación 22. El código comentado de esta función se presenta a continuación.

```
#Esta funcion recibe un vector con los
    multiplicadores de lagrange (ordenados de
    tal manera que el grado correspondiente
    como ligadura este ordenado de menor a
    mayor)
#Entrega una lista de listas con las
    componentes de la matriz de adyacencia
    promedio segun el ensamble canonico
def canonical_matrix(Multiplicadores_Lagrange):
  #se genera la lista de listas
  Adyacencia_promedio = []
  for i in range(len(Multiplicadores_Lagrange))
    :
    # se agrega una fila/lista por cada
    multiplicador
    Adyacencia_promedio.append([])
    for j in range(len(Multiplicadores_Lagrange
    )):
      Adyacencia_promedio[i].append(1/(1 + math
    .exp((Multiplicadores_Lagrange[i] +
    Multiplicadores_Lagrange[j]))))
  #ahora nos aseguramos que no haya autoloops
  for i in range(len(Multiplicadores_Lagrange))
    :
    Adyacencia_promedio[i][i] = 0
  return Adyacencia_promedio
```

15.3. Programas para el estudio de la robustez

A continuación se presenta una descripción de las funciones resultado del tercer objetivo y su código comentado.

pareto-cum-probabilites
esta función genera un vector con las probabilidades acumuladas de una distribución de Pareto para puntos en una grilla uniforme. Recibe como input el número de particiones, el punto final de la grilla y el exponente de la distribución de Pareto.

```
#Esta funcion recibe un xfinal hasta donde
    sumar y un numero de particiones.
#Juntos definen la longitud y el valor maximo
    del vector resultante
def pareto_cumm_probabilities(particiones,xfin,
     alpha, distribution = st.pareto.cdf):
  """
  (particiones,xfin,alpha) -----> (
    probability_cum_vector)
  Esta funcion recibe un xfinal hasta donde
    sumar, un exponente para la ley de
    potencias
```

```
  y un numero de particiones. Juntos definen la longitud y el valor maximo del vector resultante.
  Entrega como resultado un vector con las probabilidades acumuladas en los puntos de las particiones
  """
  dx = (xfin-1)/particiones
  x = []
  probability_cum_vector = []
  for i in range(particiones):
    equis = 1 + dx*i
    x.append(equis)
    pes = st.pareto.cdf(x[i],alpha-1)
    probability_cum_vector.append(pes)
  return probability_cum_vector, x # entrega como resultado el vector y los valores de x asociados
```

Degree-Sec-Generator

Este algoritmo esta hecho para muestrear secuencias de enteros con una distribución de ley de potencias, en particular está hecha para exponentes entre -2 y -3 donde la distribución tiene un segundo momento (varianza) divergente, esto causa un sesgo en los grafos generados con las secuencias muestreadas a partir de esta distribución de varianza divergente, sesgo que se ve reflejado en correlaciones entre grados [49]. Para solucionar este problema y eliminar las correlaciones en [49] proponen imponer un cutoff para el grado máximo, donde dan un criterio de elección de este cutoff para el configuration model ($kmax = \sqrt{N}$), tomando como base este cutoff en este trabajo se tomó como cutoff para el muestreo $\sqrt{N} + 10$, donde N es el número de nodos.

```
#Este algoritmo es para muestrar secuencias de enteros on una distribucion dada. ( Montecarlo
def Degree_Sec_Generator(Probabilidad_Acumulada,longitud_Secuencia,kmin):
  """
  (Vector Probabilidada Acumulada, Longitud Secuencia,kmin) -------> (Secuencia de enteros con la distribucion del vector)
  Este algoritmo es para muestrar secuencias de enteros con una distribucion de ley de potencias con exponente entre 2 y 3. ( Montecarlo)
  recuerde que si quiere una secuencia de grado longitud_Secuencia = len( Probabilidad_Acumulada)
  kmin corresponde al valor minimo desde el cual se muestrear  la distribucion, si no es necesario poner 0.
  """
  Degree_Sequence = []
  sum = 0
  #por la condicion del cutoff no todos los enteros son aceptados, por lo cual se hacen 100000 intentos, para asegurar
  # que se encuentre la distribucion de grado de la longitud requerida
  intentos = 10000*longitud_Secuencia
  for i in range(intentos):
    k = np.random.rand()*Probabilidad_Acumulada[len(Probabilidad_Acumulada)-1]
    for j in range(len(Probabilidad_Acumulada)):
      #la siguiente condicion busca la posicion en el vector de probabilidades acumuladas que es justo mayor
      # que el numero extraido aleatoriamente, para asi tomar el numero de esta posicion como un grado de la secuencia
      if k <= Probabilidad_Acumulada[j]:
        #la siguiente condicion solo acepta el entero si es menor que el cutoff
        if j <= round(len(Probabilidad_Acumulada)**(1/2) + 10):
          Degree_Sequence.append(j+kmin-1)
          break
    #cuando la longitud de la secuencia sea la longitud requerida rompe
    if len(Degree_Sequence) == longitud_Secuencia:
      break
  return Degree_Sequence
```

maxent-generator

Esta función recibe un vector con probabilidades acumuladas según cierta distribución y dos enteros, uno correspondiente al número de nodos y el otro al grado mínimo respectivamente. Hace uso de la función **Degree-Sec-Generator** para generar secuencias de grado, luego valida si es compatible para generar un grafo simple conectado y finalmente por medio de Edge Switching muestrea uniformemente el ensamble de grafos con esa secuencia de grado, tal como se delinea en 2.8.1 y [2].

```
#esta funcion recibe vector de probabilidades acumulada y un entero, te entrega un grafo generado con el algoritmo expuesto en el proyecto que sustenta este respositorio
def maxent_generator(cum_probability,number_nodes,kmin):
  """
  (cum_probability,number_nodes,kmin) -----> G
  esta funcion recibe vector de probabilidades acumulada, un numero de nodos y un grado minimo.
  te entrega un grafo generado con el algoritmo expuesto de maxima entropia expuesto en el proyecto
  """
  for k in range(1000000000000):
        #generamos la secuencia y validamos que permita generar un grafo simple conectado
        Degree_Sequence = Degree_Sec_Generator(cum_probability,number_nodes,kmin)
        if nx.is_valid_degree_sequence_havel_hakimi(Degree_Sequence) == True:
          if nx.is_connected(nx.havel_hakimi_graph(Degree_Sequence)) == True:
            #la ordenamos, esto con el fin de darle una identidad a los nodos acorde a su puesto en el ranking de medida
            Degree_Sequence.sort()
            #creamos el grafo con el algoritmo havel hakimi
            Grafo = nx.havel_hakimi_graph(Degree_Sequence)
            break
  #muestreamos los grafos con una secuencia de
```

```
    grado dada a partir de edge switching
  nx.double_edge_swap(Grafo,nswap=1000,
    max_tries=150000)
  return Grafo
```

remove-hubs-load

lEsta función se construyo para usarse posteriormente en la falla en cascada, recibe un grafo y un entero correspondiente a la cantidad de nodos que quiere que se remuevan y entrega como resultado el grafo sin los nodos con mayor valor en la medida Load Centrality 5.

edge-remove-hub-load

Esta función recibe un grafo y un entero correspondiente a la cantidad de enlaces que se quiere remover y entrega como resultado el grafo sin los enlaces con mayor valor en la medida Load Centrality5.

remove-aleatory

Esta función recibe un grafo y un entero correspondiente a la cantidad de enlaces que se quiere remover aleatoriamente, entrega el grafo con los enlaces removidos.

edge-remove-aleatory

Esta función recibe un grafo y un entero correspondiente a la cantidad de enlaces que se quiere remover aleatoriamente. Entrega como resultado el grafo sin los enlaces.

```
#las siguientes 4 funciones remueven N nodos o
    enlaces de forma aleatoria y dirigida de un
     grafo dado como input

#remueve nodos con mayor load
def remove_hubs_load(G,nodos_removidos):
  """
  (G, #nodos_removidos) ---------> G
  """
  for i in range(nodos_removidos):
    keys = list(nx.load_centrality(G).keys())
    values = list(nx.load_centrality(G).values
    ())
    maxval = max(values)
    casilla_nodo = values.index(maxval)
    G.remove_node(keys[casilla_nodo])
  return G


#remueve edges con mayor load
def edge_remove_hub_load(G,edges_removidos):
  """
  (G,#edges_removidos) -------> G
  """
  for i in range(edges_removidos):
    keys = list(nx.edge_load_centrality(G).keys
    ())
    values = list(nx.load_centrality(G).values
    ())
    maxval = max(values)
    casilla_edge = values.index(maxval)
    G.remove_edge(keys[casilla_edge][0],keys[
    casilla_edge][1])
  return G

#remueve nodos aleatoriamente
def remove_aleatory(G,nodos_removidos):
  """
  (G,#nodos_removidos) -----------> G
  """
  keys = list(G.nodes())
  for i in range(nodos_removidos):
    remove_node = np.random.randint(0,G.
    number_of_nodes())
    remnode = keys[remove_node]
    if (remnode in G) == True:
      G.remove_node(remnode)
  return G

#remueve edges aleatoriamente
def edge_remove_aleatory(G,edges_removidos):
  """
  (G,#edges_removidos) ------> G
  """
  for i in range(edges_removidos):
    remove_edge = np.random.randint(0,len(nx.
    edges(G)))
    if (remove_edge in G) ==True:
      G.remove_edge(list(nx.edges(G))[
    remove_edge][0],list(nx.edges(G))[
    remove_edge][1])
  return G
```

hub-cascade-failure

Está función recibe como input un grafo, número de ataques y los parámetros del modelo de falla en cascada (capacidad inicial, resiliencia). Haciendo uso de la función **remove-hub-load** se elimina el nodo con mayor Load y posteriormente se recalcula Load centrality para cada nodo, finalmente se decide que nodos colapsan acordé a su resiliencia, capacidad inicial y Load, tal como se expuso en 2.8.2. Finalmente te entrega el grafo luego de la falla en cascada y dos vectores, uno conteniendo el número de ataques y otro conteniendo el % de nodos sobrevivientes en la componente principal para cada ataque.

edge-hub-cascade-failure

Este algoritmo hace exactamente lo mismo que el algoritmo anterior, pero ya no ataca nodos, sino enlaces por lo cual hace uso de **edge-remove-hub-load**. Esto es posible gracias a que el algoritmo presentado en 2.8.2 es válido para nodos y enlaces.

```
#estas funciones atacan nodos y enlaces en base
     al ranking en la medida fload y causan una
     falla en cascada
# la falla en cascada se realiza recalculando
    el load en la red y eliminando con cierta
    probabilidad los nodos que tengan un load
    mayor
# a su load_inicial*resiliencia, si el load
    supera un valor dado la probabilidad de
    eliminacion es 100%

def hub_cascade_failure(G,Initial_Capacity,
    Resiliencia,Number_Attacks):
  """
  (Graph,float,float,int) -------> (G)
  Esta funcion ataca nodos acorde a su ranking
    en la medida load y luego aplica una falla
    en cascada
  acorde a que nodos soportan un mayor load que
     el dado por su capacidad inicial y
    resiliencia
  """
  load1 = nx.load_centrality(G)
```

```
  keys1 = list(load1.keys())
  nodes = len(G.nodes())
  Capacity = []
  #se definen condiciones iniciales
  for i in range(len(keys1)):
    q = (1+Initial_Capacity)*load1[keys1[i]]
    Capacity.append(q)
  DAMAGE = []
  ATTACK = []
  #en este for se ejecutan los ataques
  for i in range(Number_Attacks):
    NodosBC = len(max(nx.connected_components(G
    ), key=len))
    #registramos el da o con el cambio en el
    tama o de la componente principal y los
    ataques
    DAMAGE.append(NodosBC/nodes)
    ATTACK.append(i)
    #se remueve el nodo con mayor load
    remove_hubs_load(G,1)
    DELETE_NODES = []
    load = nx.load_centrality(G)
    for j in G.nodes():
      #calculamos el valor de load sobre el
    cual un nodo colapsara con 100% de
    seguridad
      val = Resiliencia*Capacity[keys1.index(j)
    ]
      if load[j] > val:
        ##enlistamos los nodos que colapsaran
    con 100% de probabilidad
        DELETE_NODES.append(j)
      else:
        #se elige si un nodo que ha superado su
     capacidad colapsa
        if load[j] > Capacity[keys1.index(j)]:
          k = st.uniform.rvs()
          P = (1/(Resiliencia-1))*(load[j]/
    Capacity[keys1.index(j)] -1)
          if k <= P:
            ##enlistamos los nodos colapsados
            DELETE_NODES.append(j)
    #se eliminan los nodos colapsados
    G.remove_nodes_from(DELETE_NODES)
  return G,ATTACK,DAMAGE

def edge_hub_cascade_failure(G,Initial_Capacity
    ,Resiliencia,Number_Attacks):
  """
  (Graph,float,float,int) -------> (G)
  Esta funcion ataca nodos acorde a su ranking
    en la medida load y luego aplica una falla
    en cascada
  acorde a que nodos soportan un mayor load que
     el dado por su capacidad inicial y
    resiliencia
  """
  load1 = nx.edge_load_centrality(G)
  keys1 = list(load1.keys())
  Capacity = []
  #se definen condiciones iniciales
  for i in range(len(keys1)):
    q = (1+Initial_Capacity)*load1[keys1[i]]
    Capacity.append(q)
  DAMAGEEDGE = []
  ATTACK = []
  #en este for se ejecutan los ataques
  for i in range(Number_Attacks):
    NodosBC = len(max(nx.connected_components(G
    ), key = len))
    #registramos el da o con el cambio en el
    tama o de la componente principal y los
    ataques
    DAMAGEEDGE.append(NodosBC/len(G.nodes()))
    ATTACK.append(i)
    #se remueve el edge con mas load
    edge_remove_hub_load(G,1)
    DELETE_EDGES = []
    load = nx.edge_load_centrality(G)
    for j in G.edges():
        #calculamos el valor de load sobre el
    cual un nodo colapsara con 100% de
    seguridad
      val = Resiliencia*Capacity[keys1.index(j)
    ]
      #enlistamos los nodos que colapsaran con
    100% de seguridad
      if load[j] > val:
        DELETE_EDGES.append(j)
      else:
        #se elige si un nodo que ha superado su
     capacidad colapsa
        if load[j] > Capacity[keys1.index(j)]:
          k = st.uniform.rvs()
          P = (1/(Resiliencia -1))*(load[j]/
    Capacity[keys1.index(j)] - 1)
          if k <= P:
            #enlistamos los nodos colapsados
            DELETE_EDGES.append(j)
    #se eliminan los nodos colapsados
    G.remove_edges_from(DELETE_EDGES)
  return G, ATTACK, DAMAGEEDGE
```

aleatory-cascade-failure Este algoritmo es análogo a **hub-cascade-failure**, por lo cual reciben los mismos inputs y dan los mismos outputs, solo que ya no hace uso de **remove-hub-load** para remover nodos con más load sino usa a **remove-aleatory** para removerlos aleatoriamente.

edge-aleatory-cascade-failure
Está función es análoga a **edge-hub-cascade-failure** la única diferencia es que en vez de usar **remove-aleatory** para extraer nodos usa **edge-remove-aleatory** para extraer enlaces y la falla en cascada se ejecuta de la misma forma que en los anteriores 3 algoritmos.

```
# estas funciones atacan a nodos y enlaces de
    forma aleatoria causando una falla en
    cascada
def aleatory_cascade_failure(G,Initial_Capacity
    ,Resiliencia,Number_Attacks):
  """
  (Graph,float,float,int) -------> (G,Vector
    con los da os, vector con el n mero de
    ataques)
  Esta funcion luego de atacar un nodo
    aleatoriamente ejecuta el algoritmo de
    falla en cascada y te entrega como
    resultado
  el grafo atacado, un vector con los da os
    por ataque y un vector con el n mero de
    ataques
  """
  load1 = nx.load_centrality(G)
  keys1 = list(load1.keys())
  Capacity = []
  #se definen condiciones iniciales
  for i in range(len(keys1)):
```

```
    q = (1+Initial_Capacity)*load1[keys1[i]]
    Capacity.append(q)
  DAMAGE = []
  ATTACK = []
  #en este for se ejecutan los ataques
  for i in range(Number_Attacks):
    NodosBC = len(max(nx.connected_components(G
    ), key = len))
    #registramos el da o con el cambio en el
    tama o de la componente principal y los
    ataques
    DAMAGE.append(NodosBC/len(G.nodes()))
    ATTACK.append(i)
    #se remueve un nodo aleatoriamente
    remove_aleatory(G,1)
    DELETE_NODES = []
    load = nx.load_centrality(G)
    for j in G.nodes():
      #calculamos el valor de load sobre el
    cual el nodo colapsara con 100% de
    probabilidad
      val = Resiliencia*Capacity[keys1.index(j)
    ]
      if load[j] > val:
        #enlistamos los nodos que colapsan con
    100% de probabilidad
        DELETE_NODES.append(j)
      else:
        #se decide si un nodo que ha superado
    su capacidad colapsa
        if load[j] > Capacity[keys1.index(j)]:
          k = st.uniform.rvs()
          P = (1/(Resiliencia-1))*(load[j]/
    Capacity[keys1.index(j)] -1)
          if k <= P:
            ##enlistamos los nodos colapsados
            DELETE_NODES.append(j)
    ##se eliminan los nodos colapsados
    G.remove_nodes_from(DELETE_NODES)
  return G,ATTACK,DAMAGE

def edge_aleatory_cascade_failure(G,
    Initial_Capacity,Resiliencia,Number_attacks
    ):
  """
  (Graph,float,float,int) -------> (G,Vector
    con los da os, vector con el numero de
    ataques)
  Esta funcion luego de atacar un enlace
    aleatoriamente ejecuta el algoritmo de
    falla en cascada y te entrega como
    resultado
  el grafo atacado, un vector con los da os
    por ataque y un vector con el n mero de
    ataques
  """
  load1 = nx.edge_load_centrality(G)
  keys1 = list(load1.keys())
  Capacity = []
  #se definen condiciones iniciales
  for i in range(len(keys1)):
    q = (1+Initial_Capacity)*load1[keys1[i]]
    Capacity.append(q)
  DAMAGEEDGE = []
  ATTACK = []
  #en este for se ejecutan los ataques
  for i in range(Number_attacks):
    NodosBC = len(max(nx.connected_components(G
    ), key = len))
    #registramos el da o con el cambio en el
    tama o de la componente principal y los
    ataques
    DAMAGEEDGE.append(NodosBC/len(G.nodes()))
    ATTACK.append(i)
    #se remueve un edge aleatoriamente
    edge_remove_aleatory(G,1)
    DELETE_EDGES = []
    load = nx.edge_load_centrality(G)
    for j in G.edges():
    #calcula el valor de load sobre el cual un
    nodo colapsara con 100% de probabilidad
      val = Resiliencia*Capacity[keys1.index(j)
    ]
      if load[j] > val:
        #enlistamos nodos colapsados
        DELETE_EDGES.append(j)
      else:
        #se elige si un nodo que ha superado
    su capacidad colapsa
        if load[j] > Capacity[keys1.index(j)]:
          k = st.uniform.rvs()
          P = (1/(Resiliencia -1))*(load[j]/
    Capacity[keys1.index(j)] - 1)
          if k <= P:
            DELETE_EDGES.append(j)
    #enlistamos nodos colapsados
    G.remove_edges_from(DELETE_EDGES)
  return G, ATTACK, DAMAGEEDGE
```

graph-entropys

Esta función recibe un grafo, posteriormente calcula su grado, eigenvector centrality, betweenness centrality, closeness centrality y load centrality con el fin de calcular la entropía de Shannon para cada una de estas listas de medidas nodales. Entrega como resultado los valores de las entropías para cada medida nodal.

```
# Esta funcion calcula la entropia de distintas
    distribuciones de medidas para un grafo
def graph_entropys(G):
  """
  Recibe un grafo y entrega numeros
  (Graph) -----> Degree_Entropy,
    Eigenvector_Entropy, Betweenness_Entropy,
    Closeness_Entropy
  """
  EIGENVECTOR = []
  DEGREE = []
  BETWEENNESS = []
  CLOSENESS = []
  LOAD = []
  #las siguientes 4 lineas producen 4
    diccionarios con los nodos y sus
    centralidades correspondientes
  eigenvector = nx.eigenvector_centrality(G,
    max_iter = 10000)
  degree = nx.degree(G)
  betweenness = nx.betweenness_centrality(G)
  closeness = nx.closeness_centrality(G)
  load = nx.load_centrality(G)
  #este for desempaqueta los diccionarios, para
     coger solo las centralidades en listas
  for i in range(len(eigenvector)):
    EIGENVECTOR.append(eigenvector[i])
    DEGREE.append(degree[i])
    BETWEENNESS.append(betweenness[i])
    CLOSENESS.append(closeness[i])
    LOAD.append(load[i])
  # aca se usa el modulo stats de scipy para
    normalizar las distribuciones y calcular
    las entropias
```

```
  Closeness_Entropy = st.entropy(CLOSENESS)
  Degree_Entropy = st.entropy(DEGREE)
  Eigenvector_Entropy = st.entropy(EIGENVECTOR)
  Betweenness_Entropy = st.entropy(BETWEENNESS)
  Load_Entropy = st.entropy(LOAD)
  return Degree_Entropy, Eigenvector_Entropy,
    Betweenness_Entropy, Closeness_Entropy,
    Load_Entropy
```

Las funciones presentadas aquí permiten generar grafos con máxima entropía acorde a una distribución dada2.8.1 y atacar esos grafos usando 2.8.2 entregándonos como resultado un vector conteniendo los ataques y la población de nodos sobreviviente para cada ataque, es precisamente en base a la población de nodos sobrevivientes que se comparará la robustez.

Finalmente las funciones para generar el grafo de Barabási y Watts-Strogatz vienen incorporadas en Networkx y se pueden encontrar en la documentación https://networkx.org/documentation/stable/_modules/networkx/generators/random_graphs.html#barabasi_albert_graph para el grafo de Barabási y https://networkx.org/documentation/stable/_modules/networkx/generators/random_graphs.html#watts_strogatz_graph Para el de Watts-Strogatz.

los cuales fueron incorporados en un repositorio en github, el cual se puede encontrar en 12.3

15.4. Comparación entre la matriz de adyacencia calculada numéricamente y la muestreada in-silico

Para el desarrollo de este objetivo se estableció además de las funciones ya expuestas las funciones:

deg-sec-prom

Está función hace uso de **Degree-Sec-Generator** para generar secuencias de grado acordé a la distribución de pareto y promediarlas. Recibe como input número de nodos por secuencia, un vector de probabilidades acumuladas, el grado mínimo y el número de secuencias a muestrear. Entrega como output la secuencia de grado promedio.

```
#Este algoritmo muestrea secuencias de grado
    con la distribucion del vector de
    probabilidades acumuladas, entrega como
    output secuencia de grado promedio
def Deg_Sec_Prom(Muestreo_seq,Number_nodes,
    Probability_distribution,kmin):
  """
  (num_muestras,number_nodes,
    probability_distribution,kimn)
    ----------------> (degree_sequence_prom)
  esta funcion hace uso de Degree_Sec_Generator
  """
  #matriz donde se guardaran las distintas
    secuencias a muestrear
  degree_sequences = []
  #por cada i generamos una secuencia
  for i in range(Muestreo_seq):
    degree_sequences.append([])
    for k in range(10000):
      #generamos y validamos la secuencia
      Degree_Sequence = Degree_Sec_Generator(
    Probability_distribution,Number_nodes,kmin)
      if nx.
    is_valid_degree_sequence_havel_hakimi(
    Degree_Sequence) == True:
        if nx.is_connected(nx.
    havel_hakimi_graph(Degree_Sequence)) ==
    True:
          #la ordenamos, esto con el fin de
    darle una identidad a los nodos acorde a su
     puesto en el ranking de medida
          Degree_Sequence.sort()
          #este for es para agregar la
    secuencia de grado a la matriz creada
    inicialmente
          for l in range(len(Degree_Sequence)):
            degree_sequences[i].append(
    Degree_Sequence[l])
          break
  Degree_sequence_prom = [] #esta parte del
    codigo calcular el promedio
  for i in range(Number_nodes):
    sum = 0
    for j in range(Muestreo_seq):
      sum = sum + degree_sequences[j][i]
    Degree_sequence_prom.append(sum/
    Muestreo_seq)
  return Degree_sequence_prom
```

muestra-ensamble-maxent

Está función hace uso de la función **maxent-generator** para muestrear el ensamble de grafos con máxima entropía y una distribución de grado dada 2.8.1. Luego halla las matrices de adyacencia para cada uno de estos grafos y las promedia. Recibe como input un vector con las probabilidades acumuladas, el numero de nodos por secuencia, el grado mínimo y el número de muestras. Entrega como output una lista de listas con las componentes de la matriz de adyacencia promedio resultante del muestreo.

```
#este algoritmo muestrea el ensamble de redes
    por medio del algoritmo de maxima entropia
    y te entrega su matriz de adyacencia
    promedio
def muestra_ensamble_maxent(muestra,
    number_of_nodes,P,kmin):
  """
  (num_muestra, num_nodes,cum_probab_distribut)
     -------> (Degree_sequence_prom,
    adjacencia_insilico)
  """
  connectivity_matrix = np.zeros((muestra,
    number_of_nodes, number_of_nodes))
  #por cada l se genera un grafo
  for l in range(muestra):
      #generamos un grafo usando el algoritmo
    de maxima entropia
      G = maxent_generator(P,number_of_nodes,
    kmin)
      adjacency = nx.adjacency_matrix(G)
      for i in range(number_of_nodes):
        for j in range(number_of_nodes):
          #guardamos las matrices de adyacencias
```

```
    de los grafos muestreados
          connectivity_matrix[l][i][j] =
    adjacency[(i,j)]

  #ahora promediamos las matrices de adyacencia
     muestreadas muestreadas
  adjacencia_insilico = np.zeros((
    number_of_nodes,number_of_nodes))
  for i in range(number_of_nodes):
    for j in range(number_of_nodes):
      adjprom = 0
      for l in range(muestra):
        adjprom = adjprom + connectivity_matrix
    [l][i][j]
      adjprom = adjprom/muestra
      adjacencia_insilico[i][j] = adjprom
  return adjacencia_insilico
```

Ya definidas estas funciones se presenta el código en el cual se hallan la matriz de adyacencia numérica e insilico que sustentan los resultados expuestos en 12.4, en este esencialmente se hace uso de las funciones ya descritas. Se asume ya ejecutado todo el código mostrado previamente en este trabajo.

Código del calculo numérico de la matriz de adyacencia para el ensamble canónico.

```
#tomamos un valor lo suficientemente grande
    para que la probabilidad acumulada final
    sea muy proxima a 1
xfin = 40
Nnodes = 150
max_degree = Nnodes-1
kmin = 3
#creamos el vector de probabilidades acumuladas
P, x = pareto_cumm_probabilities(max_degree,
    xfin,alpha)

#Creamos la secuencia de grado promedio
Sec_prom = Deg_Sec_Prom(300,Nnodes,P,kmin)

#hallamos los multiplicadores de Lagrange
multipliers = multp_lagrange_canonico(Sec_prom)

#generamos la matriz de adyacencia promedio
adjacencia_numerica = canonical_matrix(
    multipliers)

#ploteamos y guardamos
fig = plt.figure(figsize=(9,6),dpi = 200)
plt.imshow(adjacencia_numerica,cmap = "inferno"
    ,vmin=0,vmax= 1)
plt.title("Matriz de adyacencia calculada
    numericamente")
plt.xlabel("Nodos ordenados de menor a mayor
    grado")
plt.ylabel("Nodos ordenados de menor a mayor
    grado")
plt.colorbar()
plt.savefig("adj_numerica_canonico.jpg",
    bbox_inches='tight', dpi=300)
plt.show()
```

Código del muestreo insilico del ensamble de grafos con máxima entropía y una distribución de grado dada.

```
#a continuacion se presenta el codigo que
    produce los resultados presentados en el
    objetivo 4
#parametros para generar el vector de
    probabilidades acumulada
xfin = 40
Nnodes = 150
max_degree = Nnodes-1
kmin = 3
#creamos el vector de probabilidades acumulada,
     se eligio exponente de -3.4
P, x = pareto_cumm_probabilities(max_degree,
    xfin,3.4)
#creamos la matriz insilico usando la funcion
    ya definida
adj_insilico = muestra_ensamble_maxent(150,
    Nnodes,P,kmin)


fig = plt.figure(figsize=(9,6), dpi = 200)
plt.imshow(adj_insilico,cmap = "inferno")
plt.xlabel("Nodos")
plt.ylabel("Nodos")
plt.title("Matriz de adjacencia muestreada
    insilico")
plt.xlabel("Nodos ordenados de menor a mayor
    grado")
plt.ylabel("Nodos ordenados de menor a mayor
    grado")
plt.colorbar()
plt.savefig("adj_muestreada_insilico.jpg",
    bbox_inches='tight', dpi=300)
plt.show()

#creamos la matriz de diferencias
DIFF = np.zeros((len(adj_insilico),len(
    adj_insilico)))
for i in range(len(adj_insilico)):
  for j in range(len(adj_insilico[i])):
    #calculamos las diferencias
    DIFF[i][j] = adjacencia_numerica[i][j] -
    adj_insilico[i][j]

#graficamos la matriz de diferencias
fig = plt.figure(figsize=(9,6), dpi = 200)
plt.imshow(DIFF,cmap = "seismic",vmin = -1,vmax
     = 1)
plt.xlabel("Nodos")
plt.ylabel("Nodos")
plt.title("Matriz diferencia")
plt.xlabel("Nodos ordenados de menor a mayor
    grado")
plt.ylabel("Nodos ordenados de menor a mayor
    grado")
plt.colorbar()
plt.savefig("adj_diff.jpg", bbox_inches='tight'
    , dpi=300)
plt.show()

#graficamos la fila 90 de las matrices de
    adyacencia para comparar
plt.plot(adjacencia_numerica[90],color = "
    yellow",label = "Prediccion numerica")
plt.title("Probabilidad de que el nodo 90 este
    conectado con cualquier nodo")
plt.xlabel("nodos")
plt.ylabel("probabilidad de conexion")
plt.plot(adj_insilico[90], color = "red", label
     = "Prediccion insilico")
plt.legend(loc = 1 )
plt.savefig("p_nodo90.jpg", bbox_inches='tight'
    , dpi=300)
```

Los outputs de este código se presentan en 12.4.

15.5. Comparaciones de robustez ante fallas en cascada

En el código a continuación se asume ya ejecutado todo el código presentado en los objetivos anteriores. A continuación una breve descripción de qué hace el código y las referencias de las funciones que usa. En este código se generan grafos de Barabási 15.3, de Watts-Strogatz 15.3 y máxima entropía 15.3. Se les calcula la entropía de algunas medidas nodales 15.3 y posteriormente se ataca de forma dirigida15.3 y aleatoria15.3 a nodos y enlaces. Finalmente se grafican las poblaciones de nodos sobrevivientes para cada ataque y se imprimen las sumas de estas poblaciones para cada modalidad de ataque y grafo.

```
#aqui se presenta la primra parte del codigo que sustenta los resultados del objetivo 5
#esta funciin se define para extraer secuencias de grado y facilitar el calculo del exponente de la distribucion de grado
def Degre_sec_extractor(G):
  degree_secuence = []
  deg = nx.degree(G)
  for i in range(len(deg)):
    degree_secuence.append(deg(i))
  return degree_secuence

#aca se usara la notacion GW = grafo watts-strogats, GB = grafo barabasi y GA = grafo maxima entropia
Nnodes = 150
#grado minimo para las redes libres de escala
m = 3
GW = nx.watts_strogatz_graph(Nnodes,6,0.1)
GB = nx.barabasi_albert_graph(Nnodes,m)
#Degre_sec_extractor lo unico que hace es extraer la secuencia de grado de un grafo dado como input
secuence = Degre_sec_extractor(GB)
#fiteamos el exponente de la dist. de grado usando maxima verosimilitud
results = powerlaw.Fit(secuence,discrete = True)

alpha = results.power_law.alpha
#xfin suficiente para que la probabilidad acumulada llegue por encima de 0.99
xfin = 50
max_degree = Nnodes -1
 #creamos el vector de probabilidades acumuladas
P, x = pareto_cumm_probabilities(max_degree, xfin,alpha)
#generamos el grafo usando maxent_generator
GA = maxent_generator(P,Nnodes,m)



#estas siguientes lineas son para calcular las entropias para cada medida usando la funcion previamente definida
wDegree_Entropy, wEigenvector_Entropy, wBetweenness_Entropy, wCloseness_Entropy, wLoad_entropy = graph_entropys(GW)
BDegree_Entropy, BEigenvector_Entropy, BBetweenness_Entropy, BCloseness_Entropy, BLoad_entropy = graph_entropys(GB)
ADegree_Entropy, AEigenvector_Entropy, ABetweenness_Entropy, ACloseness_Entropy, ALoad_entropy = graph_entropys(GA)

#se imprimen las entropias
print("Watts Strogatz", "Barabasi Albert" , " maxima entropia")
print(wDegree_Entropy,BDegree_Entropy, ADegree_Entropy,"Degree")
print(wEigenvector_Entropy,BEigenvector_Entropy, AEigenvector_Entropy,"Eigenvector")
print(wBetweenness_Entropy,BBetweenness_Entropy, ABetweenness_Entropy,"Betweenness")
print(wCloseness_Entropy,BCloseness_Entropy, ACloseness_Entropy,"Closeness")
print(wLoad_entropy,BLoad_entropy, ALoad_entropy,"load")

#ya aca creados los grafos se procede a atacarlos bajo las distintas modalidades, los codigos son muy parecidos entre si

#ataque deliberado a los nodos

GA2 = GA.copy() #se copian los grafos para no da ar los originales
GB2 = GB.copy()
GW2 = GW.copy()
num_attacks = 20 #los ataques a los hubs desconectan muy rapido las redes, por lo cual con pocos ataques basta
#se atacan las redes usando las funciones previamente definidas
GW2, ATTACKW,DAMAGEW = hub_cascade_failure(GW2,1,1.1,num_attacks)
GB2, ATTACKB,DAMAGEB = hub_cascade_failure(GB2,1,1.1,num_attacks)
GA2, ATTACK,DAMAGE = hub_cascade_failure(GA2,1,1.1,num_attacks)

#se grafica
plt.plot(ATTACKW,DAMAGEW,"green", label ="Watts_Strogatz")
plt.plot(ATTACKB,DAMAGEB,"blue", label = "Barabasi_Albert")
plt.plot(ATTACK,DAMAGE,color ="red", label = "SF_maxent")
plt.xlabel("Numero de ataques")
plt.ylabel("Poblacion restante de nodos")
plt.legend(loc = 1)
plt.savefig("hub_cascade_failure.jpg", bbox_inches = "tight", dpi= 300)
#el siguiente for es para sumar las poblaciones sobrevivientes de nodos para cada ataque
for i in range(len(DAMAGEW)):
  sumw = sumw + DAMAGEW[i]
  sumb = sumb + DAMAGEB[i]
  suma = suma + DAMAGE[i]
print(sumw,sumb,suma)
#ataques deliberados a los enlaces
#se copian los grafos
GA2 = GA.copy()
GB2 = GB.copy()
GW2 = GW.copy()
num_attacks = 200
# se atacan usando las funciones ya previamente definidas
GW2, ATTACKW,DAMAGEW = edge_hub_cascade_failure(GW2,1,1.1,num_attacks)
GB2, ATTACKB,DAMAGEB = edge_hub_cascade_failure(GB2,1,1.1,num_attacks)
```

```
GA2, ATTACK,DAMAGE = edge_hub_cascade_failure(GA2,1,1.1,num_attacks)
#se plotean los da os y ataques para cada grafo
plt.plot(ATTACKW,DAMAGEW,"green", label =" Watts_Strogatz")
plt.plot(ATTACKB,DAMAGEB,"blue", label = " Barabasi_Albert")
plt.plot(ATTACK,DAMAGE,color ="red", label = " SF_maxent")
plt.xlabel("N mero de ataques")
plt.ylabel("Poblaci n restante de nodos")
plt.legend(loc = 3)
plt.savefig("edge_hub_cascade_failure.jpg", bbox_inches = "tight", dpi= 300)
#se calculan e imprimen la suma de los nodos sobrevivientes en la componetne principal para cada ataque
for i in range(len(DAMAGEW)):
  sumw = sumw + DAMAGEW[i]
  sumb = sumb + DAMAGEB[i]
  suma = suma + DAMAGE[i]
print(sumw,sumb,suma)
```

```
#aqui se presenta la segunda parte del codigo que sustenta los resultados del objetivo quinto
#Dado el caracter aleatorio de este tipo de ataque fue necesario realizar el ataque varias veces y promediar los da os

# ataque aleatorio a los nodos, ejecutado 60 veces y promediado
muestra = 60
DW = []
DB = []
DA = []
for i in range(muestra):
# primero se copian los grafos
  GA2 = GA.copy()
  GB2 = GB.copy()
  GW2 = GW.copy()
  num_attacks = 25
  #se atacan los grafos
  GW2, ATTACKW,DAMAGEW = aleatory_cascade_failure(GW2,1,1.1, num_attacks)
  GB2, ATTACKB,DAMAGEB = aleatory_cascade_failure(GB2,1,1.1, num_attacks)
  GA2, ATTACK,DAMAGE = aleatory_cascade_failure(GA2,1,1.1,num_attacks)
  DW.append(DAMAGEW)
  DB.append(DAMAGEB)
  DA.append(DAMAGE)
DWP = []
DBP = []
DAP = []
#se promedian los da os atraves de todos los ataques
for j in range(len(DW[i])):
  sumw = 0
  sumb = 0
  suma = 0
  for i in range(muestra):
    sumw = sumw + DW[i][j]
    sumb = sumb + DB[i][j]
    suma = suma + DA[i][j]
  DWP.append(sumw)
  DBP.append(sumb)
  DAP.append(suma)
```

```
plt.plot(ATTACKW,DWP,"green", label =" Watts_Strogatz")
plt.plot(ATTACKB,DBP,"blue", label = " Barabasi_Albert")
plt.plot(ATTACK,DAP,color ="red", label = " SF_maxent")
plt.xlabel("Numero de ataques")
plt.ylabel("Poblacion restante de nodos")
plt.legend(loc = 3)
plt.savefig("aleatory_cascade_failureprom.jpg", bbox_inches = "tight", dpi= 300)
#el siguiente for suma las poblaciones de nodos sobrevivientes para cada ataque y las imprime
for i in range(len(DAMAGEW)):
  sumw = sumw + DWP[i]
  sumb = sumb + DBP[i]
  suma = suma + DAP[i]
print(sumw,sumb,suma)

#Ataque aleatorio a los enlaces, ejecutado 15 veces y promediado

muestra = 15
DW = []
DB = []
DA = []
for i in range(muestra):
  #se copian los grafos
  GA2 = GA.copy()
  GB2 = GB.copy()
  GW2 = GW.copy()
  num_attacks = 200
  #se atacan con la funcion previamente definida
  GW2, ATTACKW,DAMAGEW = edge_aleatory_cascade_failure(GW2,1,1.1, num_attacks)
  GB2, ATTACKB,DAMAGEB = edge_aleatory_cascade_failure(GB2,1,1.1, num_attacks)
  GA2, ATTACK,DAMAGE = edge_aleatory_cascade_failure(GA2,1,1.1, num_attacks)
  DW.append(DAMAGEW)
  DB.append(DAMAGEB)
  DA.append(DAMAGE)
DWP = []
DBP = []
DAP = []
#se promedian los ataques
for j in range(len(DW[3])):
  sumw = 0
  sumb = 0
  suma = 0
  for i in range(muestra):
    sumw = sumw + DW[i][j]
    sumb = sumb + DB[i][j]
    suma = suma + DA[i][j]
  DWP.append(sumw/muestra)
  DBP.append(sumb/muestra)
  DAP.append(suma/muestra)

plt.plot(ATTACKW,DWP,"green", label =" Watts_Strogatz")
plt.plot(ATTACKB,DBP,"blue", label = " Barabasi_Albert")
plt.plot(ATTACK,DAP,color ="red", label = " SF_maxent")
plt.xlabel("N mero de ataques")
plt.ylabel("Poblaci n restante de nodos")
```

```
plt.legend(loc = 3)
plt.savefig("edge_aleatory_cascade_failureprom.
    jpg", bbox_inches = "tight", dpi= 300)
#se suman las poblaciones de nodos
    sobrevivientes
for i in range(len(DAMAGEW)):
  sumw = sumw + DWP[i]
  sumb = sumb + DBP[i]
  suma = suma + DAP[i]
print(sumw,sumb,suma)
```

Los resultados del estudio de robustez realizado con este código se pueden encontrar en 12.5.

Referencias

[1] ZJ Bao, YJ Cao, LJ Ding, and GZ Wang. Comparison of cascading failures in small-world and scale-free networks subject to vertex and edge attacks. *Physica A: Statistical Mechanics and its Applications*, 388(20):4491–4498, 2009.

[2] Linjun Zhang, Michael Small, and Kevin Judd. Exactly scale-free scale-free networks. *Physica A: Statistical Mechanics and its Applications*, 433:182–197, 2015.

[3] Mark Newman. *Networks*. Oxford university press, 2018.

[4] Douglas Brent West et al. *Introduction to graph theory*. Prentice hall Upper Saddle River, 2001.

[5] Mark Newman. *Networks, An Introduction*. Oxford university press, 2010.

[6] Ginestra Bianconi. The entropy of network ensembles (article). *The Physical Review*, 79, 2 2009.

[7] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[8] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

[9] Ye Cai, Yijia Cao, Yong Li, Tao Huang, and Bin Zhou. Cascading failure analysis considering interaction between power grids and communication networks. *IEEE Transactions on Smart Grid*, 7(1):530–538, 2015.

[10] et al Z.H. Bao. Dynamics of load entropy during cascading failure propagation in scale-free networks (article). *Physics Letters A*, 372, 07 2008.

[11] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and attack tolerance of complex networks. *nature*, 406(6794):378–382, 2000.

[12] Ginestra Bianconi. The entropy of randomized network ensembles (article). *A Letters Journal Exploring the Frontiers of Physics*, 81, 12 2007.

[13] Juyong Park and Mark EJ Newman. Statistical mechanics of networks. *Physical Review E*, 70(6):066117, 2004.

[14] Albert-László Barabási. Linked: The new science of networks, 2003.

[15] Hyunjeong Seo, Wanyeon Kim, Jihyung Lee, and Buhyun Youn. Network-based approaches for anticancer therapy (review). *International journal of oncology*, 43, 09 2013.

[16] Sergey N Dorogovtsev, Jose Ferreira F Mendes, and Alexander N Samukhin. Structure of growing networks: Exact solution of the barabási–albert's model. *arXiv preprint cond-mat/0004434*, 2000.

[17] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[18] Ryogo Kub. *Statistical Mechanics, an advanced course with problems and solutions*. North-Holland Physics Publishing, 2001.

[19] Paul D. Beale R.K. Pathria. *Statistical Mechanics, Third Edition*. Elsevier, 2011.

[20] Walter Greiner, Ludwig Neise, and Horst Stöcker. *Thermodynamics and statistical mechanics*. Springer Science & Business Media, 2012.

[21] Robert Zwanzig. *Nonequilibrium statistical mechanics*. Oxford university press, 2001.

[22] Radu Balescu. Equilibrium and nonequilibrium statistical mechanics. *NASA STI/Recon Technical Report A*, 76:32809, 1975.

[23] Harry Nyquist. Thermal agitation of electric charge in conductors. *Physical review*, 32(1):110, 1928.

[24] David Chandler. Introduction to modern statistical. *Mechanics. Oxford University Press, Oxford, UK*, 40, 1987.

[25] Rashmi C Desai and Robert Zwanzig. Statistical mechanics of a nonlinear stochastic model. *Journal of Statistical Physics*, 19(1):1–24, 1978.

[26] Robert van Leeuwen, Nils Erik Dahlen, Gianluca Stefanucci, C-O Almbladh, and Ulf von Barth. Introduction to the keldysh formalism. In *Time-dependent density functional theory*, pages 33–59. Springer, 2006.

[27] James R Norris and John Robert Norris. *Markov chains*. Number 2. Cambridge university press, 1998.

[28] Markus F Weber and Erwin Frey. Master equations and the theory of stochastic path integrals. *Reports on Progress in Physics*, 80(4):046601, 2017.

[29] Rongxiang Luo, Giuliano Benenti, Giulio Casati, and Jiao Wang. Onsager reciprocal relations with broken time-reversal symmetry. *Physical Review Research*, 2(2):022009, 2020.

[30] C.E. Shannon. Information theory and statistical mechanics (article). *The Physical Review*, 106, 10 1948.

[31] E.T. Jaynes. Information theory and statistical mechanics (article). *The Physical Review*, 106, 02 1957.

[32] John Harte. *Maximum entropy and ecology*. Oxford university press, 2011.

[33] Dénes König. Theory of finite and infinite graphs. In *Theory of Finite and Infinite Graphs*, pages 45–421. Springer, 1990.

[34] Frank Ball, Denis Mollison, Gianpaolo Scalla-Tomba, Mark Newman, Albert-László Barabási, and Duncan J Watts. Epidemics with two levels of mixing. In *The Structure and Dynamics of Networks*, pages 436–479. Princeton University Press, 2011.

[35] Mark EJ Newman and Duncan J Watts. Renormalization group analysis of the small-world network model. *Physics Letters A*, 263(4-6):341–346, 1999.

[36] George Udny Yule. Ii.—a mathematical theory of evolution, based on the conclusions of dr. jc willis, fr s. *Philosophical transactions of the Royal Society of London. Series B, containing papers of a biological character*, 213(402-410):21–87, 1925.

[37] Herbert A Simon. On a class of skew distribution functions. *Biometrika*, 42(3/4):425–440, 1955.

[38] Derek de Solla Price. A general theory of bibliometric and other cumulative advantage processes. *Journal of the American society for Information science*, 27(5):292–306, 1976.

[39] Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.

[40] DJC MacKay. Maximum entropy connections: neural networks. In *Maximum entropy and Bayesian methods*, pages 237–244. Springer, 1991.

[41] Hyunjune Sebastian Seung, Haim Sompolinsky, and Naftali Tishby. Statistical mechanics of learning from examples. *Physical review A*, 45(8):6056, 1992.

[42] Lester Ingber. Generic mesoscopic neural networks based on statistical mechanics of neocortical interactions. *Physical Review A*, 45(4):R2183, 1992.

[43] Duncan J Watts et al. A simple model of fads and cascading failures. *Preprint available at<http://www. santafe. edu/sfi/publications/Abstracts/00-12-062abs. html*, 2000.

[44] Duncan S Callaway, Mark EJ Newman, Steven H Strogatz, and Duncan J Watts. Network robustness and fragility: Percolation on random graphs. *Physical review letters*, 85(25):5468, 2000.

[45] Bing Wang, Huanwen Tang, Chonghui Guo, and Zhilong Xiu. Entropy optimization of scale-free networks' robustness to random failures. *Physica A: Statistical Mechanics and its Applications*, 363(2):591–596, 2006.

[46] Milton Abramowitz and Irene A Stegun. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*, volume 55. US Government printing office, 1964.

[47] Michele Catanzaro, Marián Boguná, and Romualdo Pastor-Satorras. Generation of uncorrelated random scale-free networks. *Physical review e*, 71(2):027103, 2005.

[48] Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.

[49] Javier Martin Hernandez, Tom Kleiberg, Huijuan Wang, and Piet Van Mieghem. A qualitative comparison of power law generators. *Power*, 24:26, 2006.

[50] Liliana Blanco Castañeda. Probabilidad. *Editorial UN*, 2013.