Apéndice A

Gráficas.

Comparativa para la evolución del grado de cooperación entre modelos reestrictivos y modelos no reestrictivos.

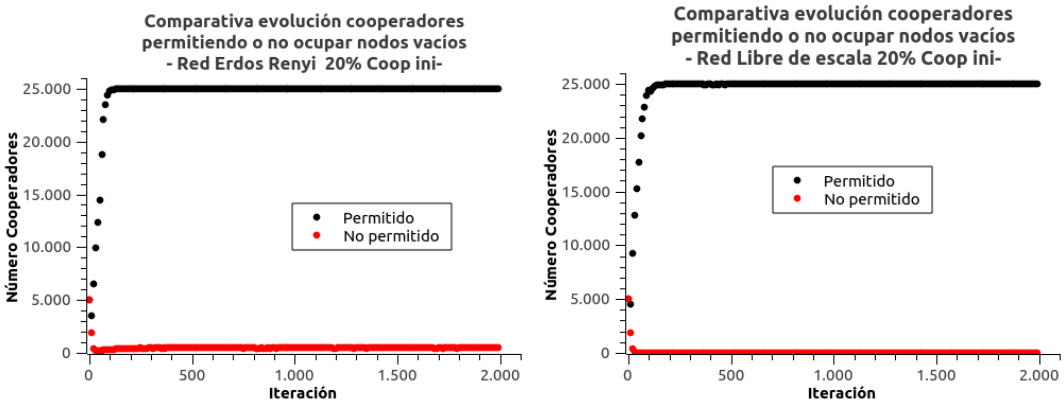


Figura A.1: Comparativa entre modelo de ocupación de nodos vacíos (permitido) y modelo reestrictivo (no permitido) en la evolución del número de cooperadores para un total de 2000 iteraciones en una red Erdös-Rényi (izqda) y en una red libre de escala (derecha). El número de cooperadores iniciales utilizado ha sido del 10 %

Para un porcentaje pequeño de cooperadores iniciales, obtenemos que el sistema no consigue invertir el comportamiento mayoritariamente desertor y acaba contagiando al resto del sistema de la misma estrategia, consiguiedo así que casi la totalidad de los agentes sean desertores.

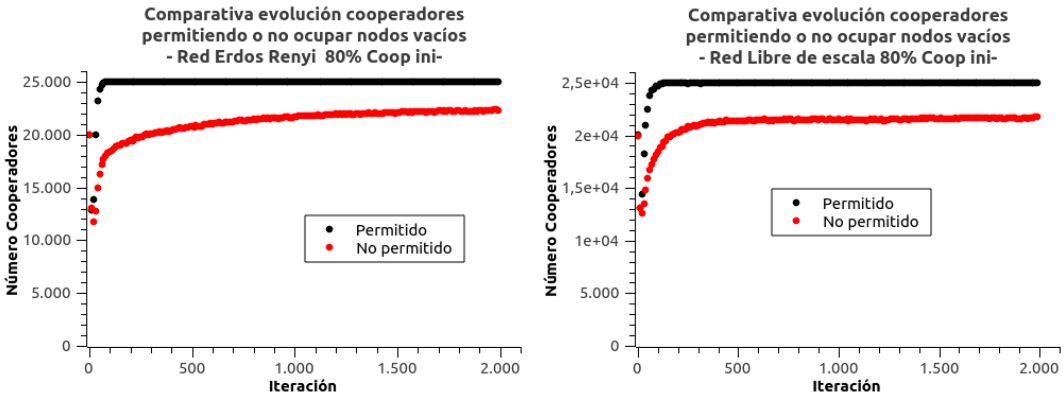


Figura A.2: Comparativa entre modelo de ocupación de nodos vacíos (permitido) y modelo restrictivo (no permitido) en la evolución del número de cooperadores para un total de 2000 iteraciones en una red Erdös-Rényi (izqda) y en una red libre de escala (derecha). El número de cooperadores iniciales utilizado ha sido del 80 %

Por el contrario, en caso de altos porcentajes de cooperadores iniciales, la dinámica del sistema tiende hacia la cooperación. Sí que es cierto que el tiempo que debe pasar hasta que el sistema se estabilice es mucho mayor en el modelo reestrictivo en comparación con el modelo no reestrictivo. Como podemos ver, para un mismo tiempo t (mismo número de iteraciones), el porcentaje de agentes que son cooperadores es menor para el modelo reestrictivo, necesitando un mayor número de iteraciones (todavía vemos que el sistema no es estable y tiene una tendencia al alza) para conseguir por completo la estrategia cooperadora.

Estudio de la evolución de la componente gigante.

En caso de realizar los histogramas estratégicos de la componente gigante para distintos valores de r, el resultado que se obtienen para ambos tipos de redes son los siguientes:

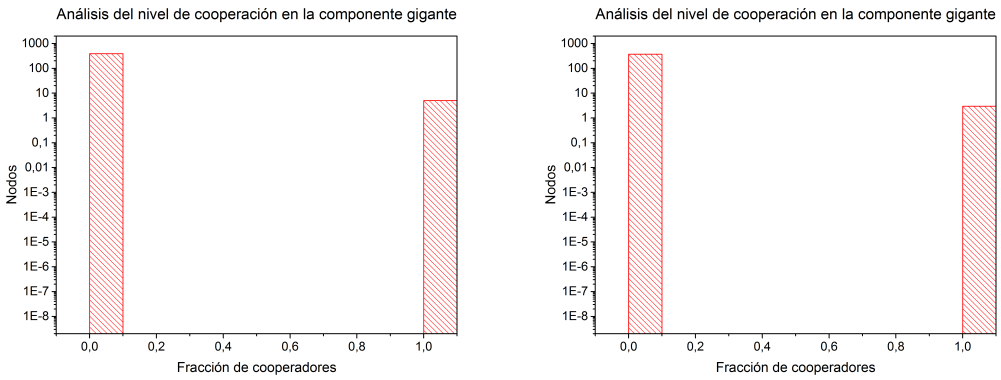


Figura A.3: Estudio de la fracción de cooperadores por nodo en la componente gigante para una red Erdös-Renyi (izquierda) y una red libre de escala (derecha), para un porcentaje inicial de cooperdordes del 50 %, valor de $r = 5$ y $a = 1$.

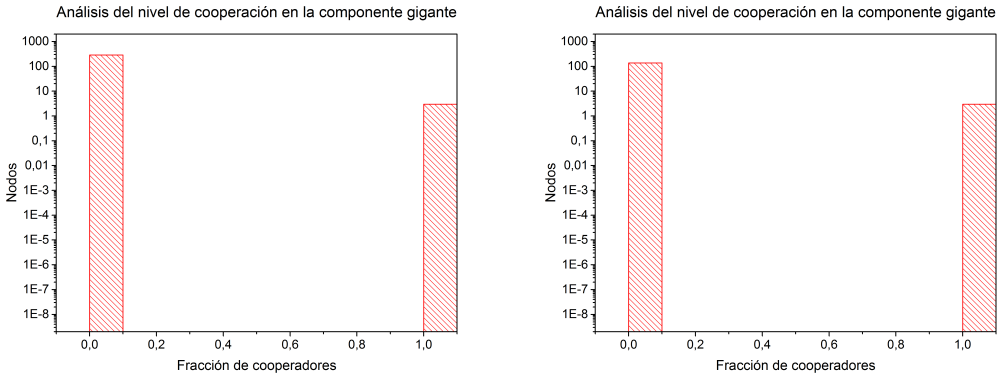
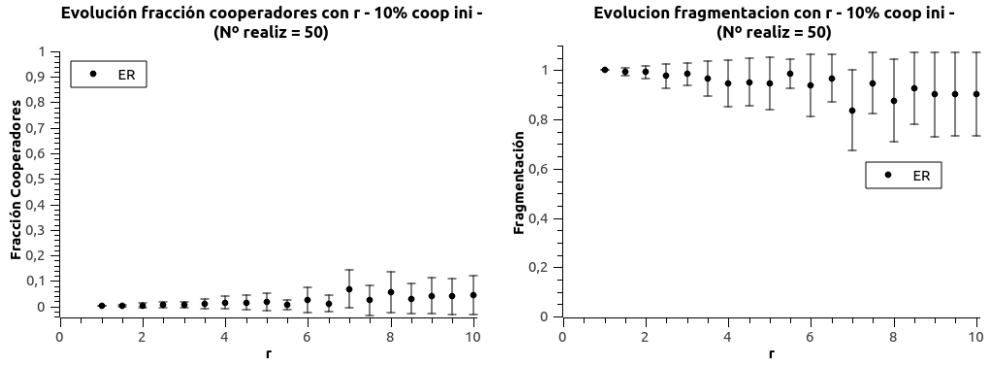


Figura A.4: Estudio de la fracción de cooperadores por nodo en la componente gigante para una red Erdös-Renyi (izquierda) y una red libre de escala (derecha), para un porcentaje inicial de cooperdordes del 50 %, valor de $r = 10$ y $a = 1$.

Como se puede observar, conforme el valor de r se hace más grande, la componente gigante está conformada por un número de nodos totales menor. A pesar de ello, el remanente de nodos con estrategia cooperadora se sigue manteniendo constante, por lo que se consigue que el nivel de cooperación del sistema aumente.

Estudio de la evolución del nivel de cooperación y fragmentación de la red con r.

Análisis individualizado para cada porcentaje de cooperadores iniciales:



(a) Evolución cooperación en red Erdös Renyi con 10 % de cooperadores iniciales.

(b) Evolución fragmetnación de red Erdös Renyi con 10 % de cooperadores iniciales.

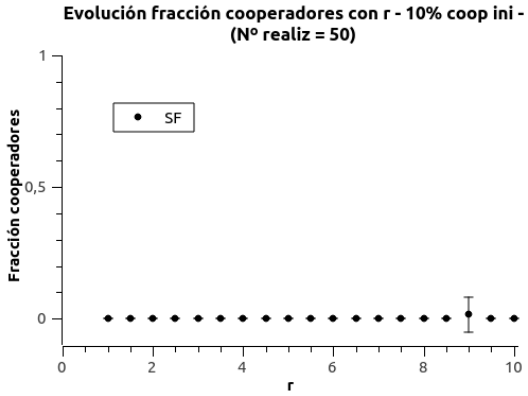
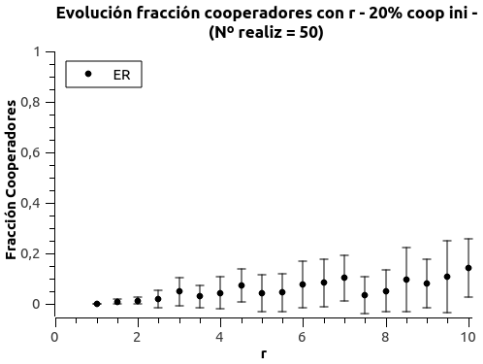
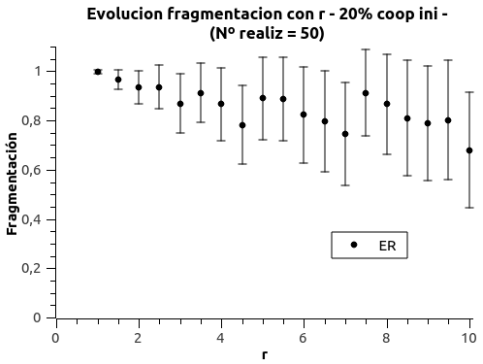


Figura A.6: Evolución cooperación en red libre de escala con 10 % de cooperadores iniciales.

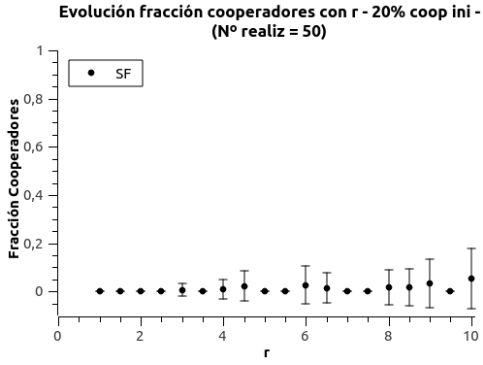
No se añade la gŕafica correspondiente a la fragmentación, ya que dadas las condiciones iniciales descritas, para este tipo de red no exise fragmentación alguna.



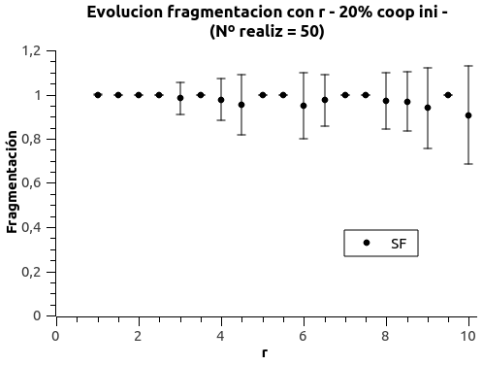
(a) Evolución cooperación en red Erdös Renyi con 20 % de cooperadores iniciales.



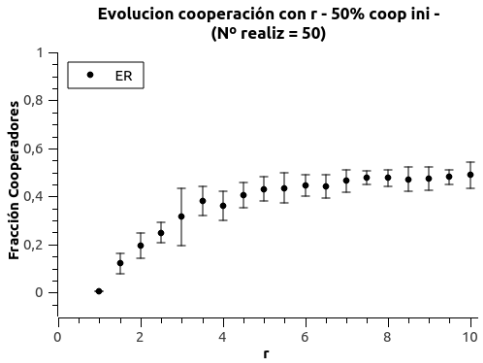
(b) Evolución fragmetnación de red Erdös Renyi con 20 % de cooperadores iniciales.



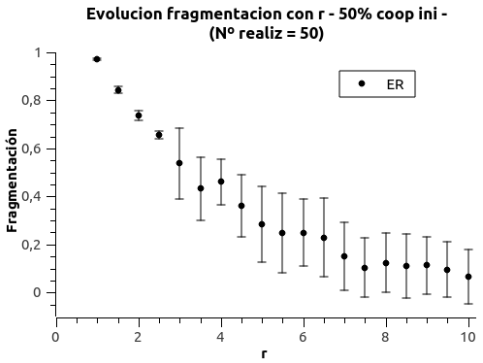
(a) Evolución cooperación en red libre de escala con 20 % de cooperadores iniciales.



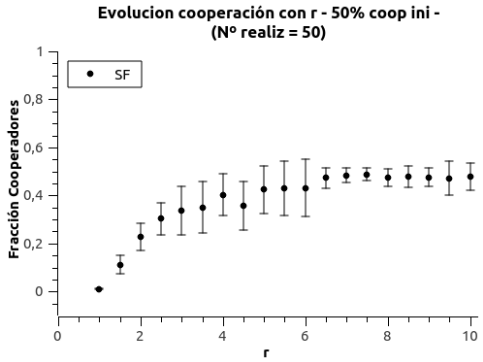
(b) Evolución fragmetnación de red libre de escala con 20 % de cooperadores iniciales.



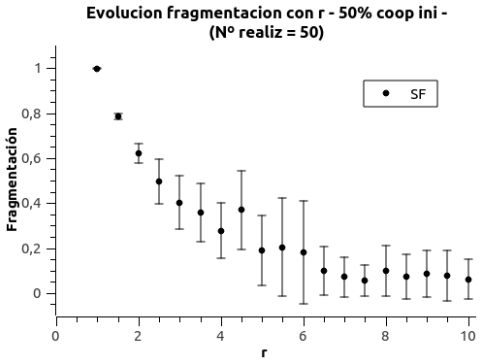
(a) Evolución cooperación en red Erdös Renyi con 50 % de cooperadores iniciales.



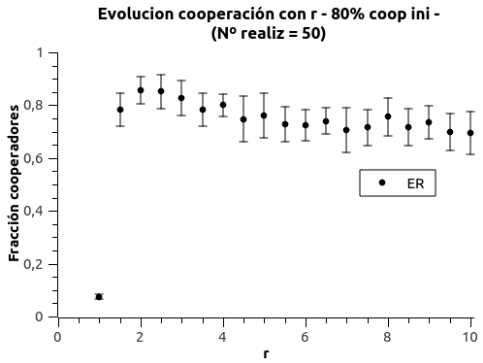
(b) Evolución fragmetnación de red Erdös Renyi con 50 % de cooperadores iniciales.



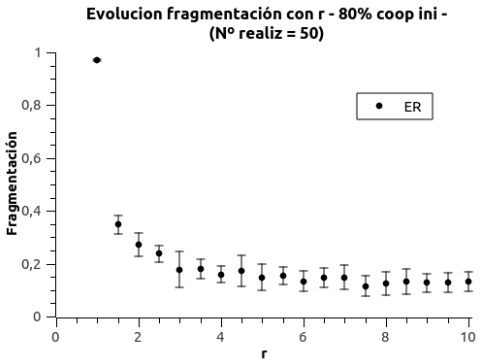
(a) Evolución cooperación en red libre de escala con 50 % de cooperadores iniciales.



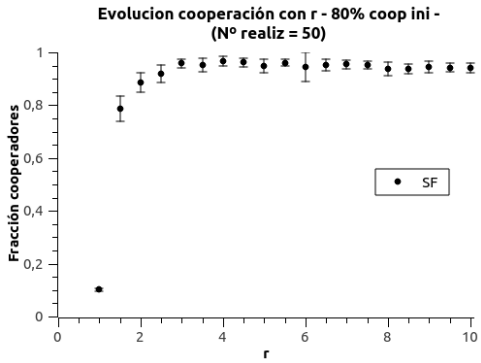
(b) Evolución fragmetnación de red libre de escala con 50 % de cooperadores iniciales.



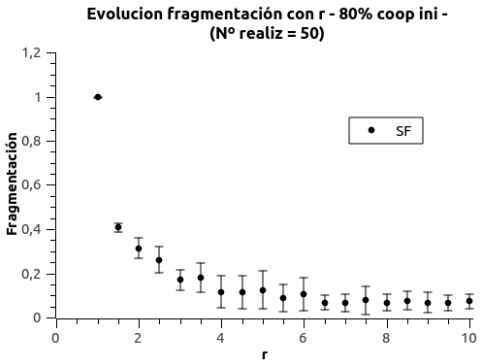
(a) Evolución cooperación en red Erdös Renyi con 80 % de cooperadores iniciales.



(b) Evolución fragmetnación de red Erdös Renyi con 80 % de cooperadores iniciales.



(a) Evolución cooperación en red libre de escala con 80 % de cooperadores iniciales.



(b) Evolución fragmetnación de red libre de escala con 80 % de cooperadores iniciales.

Apéndice B

Segmentos de código relevantes.

- Termalización -

El procedimiento que se sigue para realizar la distribución de agentes termalizada es la siguiente:

Algoritmo 1: Termalización

1: Para cada agente repito
2: Localizo el nodo donde habita
3: **Compruebo** si el nodo esta vacio
4: Si el nodo está vacío, el agente no varía su posición. Actualizo población nodo
5: Si el nodo no está vacío
6: Número aleatorio entre 0 y el grado del nodo donde habita el agente
7: El número aleatorio marca la nueva posición del agente (el nodo destino)
8: Actualizo posición del agente
9: Actualizo población nodo
10: **Fin de la comprobación**
11: Fin de la repetición
12: Fin del algoritmo

- Dinámica Evolutiva -

Una vez termalizado el sistema, asignamos a cada agente el tipo de estrategia a seguir: O bien es un agente cooperador o por el contrario es un agente desertor. El porcentaje de cooperadores y desertores iniciales es una variable que puede ser modificada con el fin de proponer posibles variaciones de las condiciones iniciales en nuestro sistema y ver así cómo afectan a la dinámica evolutiva.

El procedimiento que se ha seguido para la implementación de la dinámica evolutiva ha sido el siguiente:

Algoritmo 2 Dinámica de red

```
1: Para cada iteración repito
2:     Para todos los agentes
3:         Calculo su pay off personal
4:     Para todos los nodos
5:         Calculo su pay off asociado
6:     Para cada agente
7:         Calculo la probabilidad de movimiento
8:         Compruebo si se mueve o no
9:             No se mueve
10:                Mantiene su estrategia de juego
11:                Actualizo población nodo
12:            Se mueve
13:                Determino nodo destino
14:                Calculo probabilidad de cambio de estrategia
15:                Compruebo cambio de estrategia
16:                    No cambia de estrategia
17:                        Actualizo población nodo
18:                    Cambio de estrategia
19:                        Actualizo estrategia agente
20:                        Actualizo población nodo
21:                Fin comprobación
22:            Fin comprobación
23:    Actualizo matriz conexiones
24:    Calculo cluster y componentes gigantes
25: Fin repetición
26: Fin algoritmo
```

Los pasos 23 y 24 son de gran importancia ya que determinan la dinámica de la red. Conforme los nodos se van quedando vacíos, la red se va fragmentantando. El estudio de la fragmentación de la misma, determinara si esta se realiza con la formación de pequeñas comunidades o si por el contrario tenemos un sola componente gigante y nodos completamente independientes. Conseguiremos así averiguar qué tipos de estrategias serán más útiles para permitir que el sistema no se destruya por completo.

Para acabar con esta parte, queríamos introducir el algoritmo que nos permite calcular los diferentes cluster que forman mi sistema, así como la componente gigante del mismo. Será este

algoritmo el que nos permitirá estudiar cómo evoluciona la fragmentación de la red y ver si se forman pequeñas comunidades en las que, con el paso del tiempo, triunfará un tipo de estrategia, o si por el contrario, tengamos una sola componente gigante mientras que el resto de nodos se van desligando de esta conforme quedan vacíos.

Algoritmo 3 Cálculo Cluster

1: Creo vector label de dimensión nodos
2: Inicializo variable cluster a 1
3: Para cada nodo repito
4: Inicio contador a 1
5: **Compruebo** que label[nodo] sea cero
6: Es cero
7: Miro vecinos del nodo
8: Marco label[vecinos[nodo]] = cluster
9: **Compruebo** que he marcado algún nodo
10: He marcado nuevos nodos
11: Sumo a contador el número de vecinos de nodo
12: Miro nodos vecinos de mis vecinos
13: Repito proceso desde 5
14: No he marcado nuevos nodos
15:: Escribo nodos marcado en vector label y valor de contador.
Obtengo así el numero de nodos en un cluster y los nodos de cluster
16: **Fin comprobación**
17: Es distinto de cero
18: Cambio nodo
19: **Fin comprobación**
20: Sumo 1 a la variable cluster
21: **Fin repetición**
22: Fin algoritmo