Paisajes de energía libre en biomoléculas: modelos de redes de Markov

Alejandro Camón Fernández

Trabajo de fin de grado en Física
Universidad de Zaragoza

Codirectores del trabajo:
Fernando Falo Forniés y Alessandro Fiasconaro

9 de septiembre de 2021

Resumen

El estudio del plegado y desplegado de polímeros es de especial interés en la física biológica por la relación entre estructura y función. Por ello, la investigación de la estabilidad de las estructuras tiene un carácter relevante en el conocimiento de los sistemas biológicos. Una aproximación al problema es la observación del paisaje de energía libre del polímero para encontrar sus estados más estables y los caminos puede llegar a ellos. En este trabajo se presenta un algoritmo que pretende facilitar esta observación discretizando el sistema para convertirlo en una red de Markov en la que podamos ver los estados más relevantes estudiando el grafo. También se muestra otro algoritmo, no del todo perfeccionado, que toma la red generada por el primero y agrupa sus nodos en atractores (basins) que se identifican con estados concretos. El trabajo concluye con la aplicación de los algoritmos a un modelo de un sistema real: la estructura de ADN G-quadruplex, presente cerca de telómeros y en zonas relacionadas con la regulación de genes, y muy estudiada en las últimas décadas. Se sacarán conclusiones sobre la validez del modelo utilizado que demuestran la eficacia de los algoritmos.

Índice general

Capítulo 1

Introducción

1.1. Problema folding-unfolding

Muchos biopolímeros, como las proteínas o algunas conformaciones de ácidos nucléicos, realizan funciones biológicas que dependen fuertemente de la forma que toma su estructura al plegarse, su capacidad de recuperar esta forma después de que se deshaga o la manera en que esta organización cambia bajo unas determinadas condiciones. Por tanto, para entender adecuadamente las bases del funcionamiento de esos polímeros, conviene alcanzar cierta comprensión de sus procesos de plegamiento y desplegamiento, que no son triviales.

El pliegue de los biopolímeros es un problema que se ha enfocado desde muchos ángulos distintos y que hoy en día sigue en proceso de investigación debido a lo complicado que supone entender con precisión la configuración estable que tomará un polímero independiente en un medio acuoso en función de su composición, la temperatura del medio y la configuración inicial de la que partiese. A este problema se le suma el intento de comprender también si ese polímero, en caso de ser forzado a abandonar el estado estable alcanzado o salir de él debido a alguna fluctuación, será capaz de recuperar la misma forma que consiguió al principio o adoptará una nueva configuración debido a algún cambio que hayan sufrido las condiciones de dependencia antes mencionadas.

La paradoja de Levinthal señala que, si las proteínas alcanzasen su forma nativa probando una a una todas las conformaciones posibles, incluso tardando picosegundos en cambiar de un estado a otro, los tiempos de búsqueda serían mayores que la edad del universo. La paradoja radica en que, sin embargo, las proteínas se pliegan correctamente en tiempos del orden de los milisegundos o microsegundos, lo que indicaría que no siguen un procedimiento aleatorio, sino un proceso guiado de alguna manera que las lleva hasta su conformación más estable. [1]

1.1.1. Free Energy Landscape (FEL)

Un método muy interesante que se puede utilizar para estudiar el plegamiento de polímeros es la búsqueda de estados estables entre todas las configuraciones que toma la cadena. Para hallar estos estados habrá que asignar a todas las configuraciones una energía libre basándose en los conocimientos que tengamos del polímero y después explorar el paisaje de energía libre resultante en busca de sus mínimos locales, que se corresponderán con aquellos estados que, en determinadas condiciones, pueden resultar estables.

El estado más interesante no siempre tiene por qué ser el de menor energía, ya que puede darse la situación de que el objeto de estudio tienda atascarse en determinados mínimos locales, resultando en que son estos estados los que nos interesa más conocer. Esto sucederá sobre todo a bajas temperaturas, donde las fluctuaciones estadísticas son menores y resulta más fácil quedarse atrapado en un mínimo local. [1, 2, 3, 4]

1.2. Redes de Markov

Un proceso estocástico con tiempo discreto se dice de Markov cuando carece de memoria, es decir, cuando la distribución de probabilidades de pasar a cada uno de los posibles estados en el siguiente paso temporal depende únicamente del estado en el que estemos actualmente, no guarda relación con los pasos anteriores de la trayectoria.

Una red de Markov es una representación de un proceso estocástico de Markov en un espacio de estados discreto, donde los nodos se corresponden con los estados del sistema y los enlaces dirigidos indican las probabilidades de transición de unos estados a otros. [2, 5, 6, 7]

1.2.1. Modelo de red

Nuestro espacio de estados es un continuo, así que para hacerlo más fácil de analizar vamos a discretizarlo definiendo un mallado tan fino como convenga a la situación. En esta discretización, todas las posiciones del polímero cuyas variables caigan en el mismo conjunto de intervalos (el mismo bin) se considerarán el mismo estado y trabajaremos únicamente con aquellos bins que estén ocupados, es decir, que en nuestro conjunto de datos aparezcan al menos una vez. Finalmente, estos estados discretos se conectarán por enlaces dirigidos que representarán la probabilidad de que el polímero pase de un estado al otro con la intención de formar así, si la cantidad de datos es suficiente, una red de Markov que represente fielmente el espacio de estados del sistema. [2, 3, 4, 5]

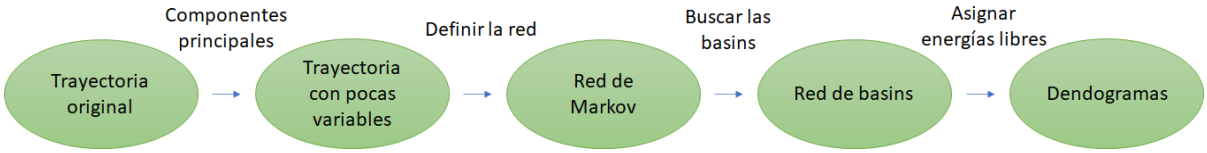


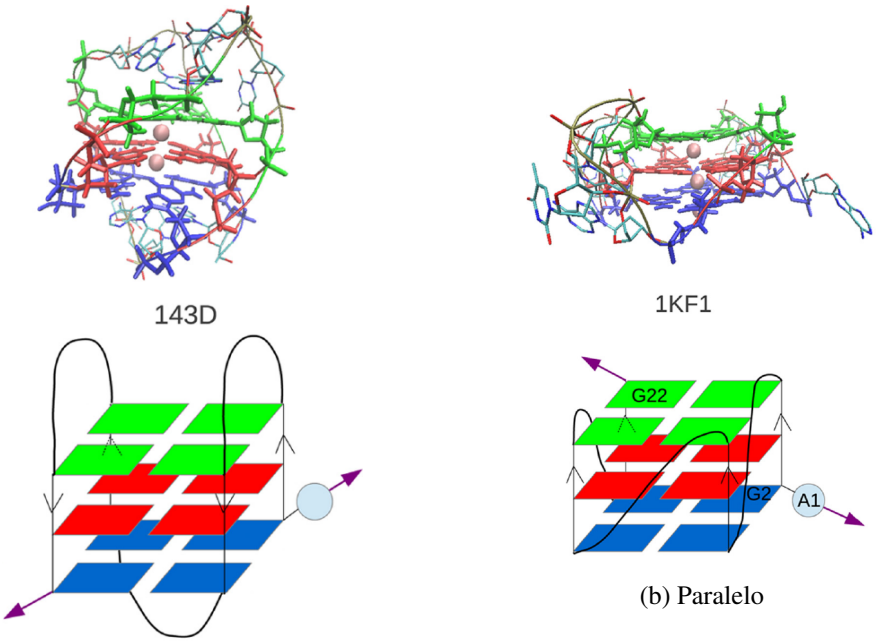
Figura 1.1: Esquema de los pasos seguidos en el método de análisis de la trayectoria de un polímero desarrollado en la memoria, a fin de extrapolar el paisaje de energía libre del sistema.

1.3. Introducción a G-quadruplex

Tras desarrollar este método lo vamos a aplicar en un polímero concreto: la estructura de DNA G-quadruplex (G4), en la que una cadena de nucleótidos rica en guaninas se pliega varias veces en torno a cationes monovalentes formando varios planos paralelos ligeramente rotados uno respecto a otro y en cada uno de los cuales hay un cuadrado compuesto por cuatro nucleótidos de guanina de distintas partes de la cadena, lo que da un formación con cuatro aristas de guaninas alineadas y los loops de monómeros que las unen. En nuestro caso trabajaremos con dos estructuras de G4 que poseen 3 planos de guaninas, lo que implica dos iones y una cadena de 21 monómeros consistente en alternar tríos de guaninas con tríos de otros nucleótidos que formarán los loops entre una arista de la estructura y la siguiente. Las dos estructuras serán: una paralela, en la que la cadena recorre todas las aristas en el mismo sentido, y una antiparalela, donde dos aristas se recorren en un sentido y las otras dos en el opuesto.

El estudio del G4 es de interés actualmente porque se ha detectado su presencia en muchas partes del DNA, especialmente cerca de los telómeros y de zonas relacionadas con la regulación de genes, lo que hace pensar que las estructuras G4 podrían estar relacionadas con diversas funciones biológicas que influirían en la transcripción de genes y en la replicación del DNA. De hecho, se ha observado que la aplicación de diversos ligandos que se unen al G4 puede inhibir la expresión de determinados genes.

Además, se ha descubierto una mayor cantidad de estructuras G4 en células cancerosas, lo que permitiría intentar usar la abundancia de G4 como un marcador en la detección de esta clase de células. Este hecho, unido a que algunos de los genes antes mencionados que pueden ser inhibidos por ligandos específicos del G4 son, precisamente, oncogenes (genes relacionados con la transformación de una célula sana en una célula cancerosa), ha provocado que también se intente buscar un método de tratamiento para los tumores utilizando ese tipo de ligandos. [11, 12, 13, 14]



(a) Antiparalelo

(b) Paralelo

Figura 1.2: Estructura molecular y modelo mesoscópico de las estructuras antiparalela y paralela del G-Quadruplex con las que trabajaremos.[15]

1.4. Objetivo general y desarrollo del trabajo

La intención principal de este trabajo es desarrollar, explicar y aplicar adecuadamente el método de redes de Markov para la caracterización de paisajes de energía libre a partir de trayectorias generadas por simulaciones de dinámica molecular, método ya utilizado en algunos trabajos anteriores [1, 4]. Se revisarán en detalle los algoritmos utilizados para ello y se aplicarán a un modelo de ejemplo y a un caso real (modelo mesoscópico de G-quadruplex) para comprobar su eficiencia y su versatilidad.

En el Capítulo 2, se profundizará en las propiedades de las redes de Markov. Después, se explicarán paso a paso los algoritmos de formación de la red y de búsqueda de basins, así como los archivos de datos que obtendremos de nuestra aplicación concreta de estos algoritmos (programas en los anexos). Por último, se aplicará el método a una trayectoria simulada en un potencial sencillo para ver su funcionamiento general.

En el Capítulo 3, se presentará un modelo mesoscópico de G4, todavía en desarrollo, del que se estudiarán a fondo varias simulaciones para aprender más sobre esos sistemas y comprobar la eficacia de los algoritmos en un caso complejo real. Finalmente, se resumirán las conclusiones de todo el trabajo, tanto respecto a los algoritmos como al modelo del G4.

Capítulo 2

Modelo de la red

2.1. Propiedades

En un proceso de Markov, el estado del sistema en un paso depende únicamente del estado en el paso anterior, por lo que el avance de un paso es equivalente a aplicar una matriz de transición que transforme el vector de coordenadas de todas las partículas del sistema en un momento en las coordenadas del sistema en el instante siguiente.

En una situación estacionaria, los elementos de esta matriz serán constantes en el tiempo y el sistema cumplirá la condición de balance detallado.

2.1.1. Balance detallado

Nuestra intención con este algoritmo era construir, a partir de una trayectoria o conjunto de trayectorias suficientemente largas, una red de Markov en la que la probabilidad P_i de estar en un nodo concreto es el peso de ese nodo normalizado de forma que la suma a todos los nodos sea 1 ($\sum_i P_i = 1$), mientras que la probabilidad W_{ji} de, estando en el nodo i, pasar por el enlace que va de i a j es el peso del enlace normalizado para que la suma a todos los enlaces que salen de i sea 1 ($\sum_j W_{ji} = 1$). [2]

Para asegurarnos de que estamos trabajando con una situación estacionaria del sistema queremos ver que se cumple la condición de balance detallado [6, 7]:

$$W_{ji}P_i = W_{ij}P_j$$

Para comprobar numéricamente que se cumple esta propiedad hemos propuestos tres estimadores para el balance detallado de un par de nodos i y j, que están descritos en el Anexo 3.

2.2. Algoritmo de formación de la red

El algoritmo de formación de la red (programa network_building en el Anexo 1) consta de dos partes bien diferenciadas: una en la que se definen los nodos y otra en la que se construye la red definiendo sus enlaces.

2.2.1. Definir los nodos

Los pasos que sigue el algoritmo para discretizar el espacio en que trabajamos y definir ahí los nodos de la nueva red son:

1. Se realiza una lectura inicial de los datos para separar las distintas variables en trayectorias individuales y analizar sus valores máximos y mínimos.

2. Se discretiza la primera variable acotada por sus valores extremos.

3. Se comprueba, en una nueva lectura de las trayectorias, qué bins de la discretización están realmente ocupados, es decir, por qué intervalos pasa la trayectoria o trayectorias de los datos.

4. Se asigna a los bins ocupados una numeración de índices a modo de identificador de cada estado y se escribe la traducción de estos índices a estados en un fichero (trad_aux.txt).

5. Se guarda la trayectoria en clave de la numeración que se ha realizado de los bins.

6. Se define una nueva discretización añadiendo la segunda variable (si la hay, si no ya ha terminado esta parte del algoritmo), con lo que ahora tendremos un número de bins potencialmente ocupados igual al número de bins ocupados definidos anteriormente por el número de intervalos en que se discretiza la segunda variable.

7. Se repiten los pasos 3, 4 y 5, identificando los bins ocupados, numerándolos, almacenando la nueva numeración en un fichero que sustituye al antiguo y guardando la trayectoria en términos de los índices definidos.

8. En sucesivas iteraciones se va añadiendo una variable cada vez y repitiendo con ella el mismo proceso que con la segunda hasta haber terminado con todas las variables, momento en que tendremos la clave completa de las celdas de nuestro espacio discreto que están ocupadas, es decir, cuáles serán los nodos de nuestra red.

Los ficheros de salida que nosotros obtendremos de esta parte del algoritmo son los siguientes:

- ***tray_xj.txt***: Los ficheros de las trayectorias individuales de cada variable, con j siendo el número que indica la variable correspondiente.
- ***max.txt***: Almacena los valores mínimo y máximo de todas las variables.
- ***trad_aux.txt***: Indica la relación de cada índice asignado a un bin con su conjunto de coordenadas asociado.
- ***tray_int_bins.oup***: Guarda la trayectoria escrita en clave de los índices asignados.

2.2.2. Construcción de la red

La segunda parte del algoritmo es la construcción de la red, en la cuál se trabaja con la trayectoria traducida, por lo que ya no se habla en términos de los bins en los que se discretizan las coordenadas, sino de los estados que corresponden al conjunto de valores de las coordenadas agrupados en esos bins.

Aquí, el algoritmo cuenta las veces que la trayectoria pasa por un nodo i y de un nodo i a otro j para asignar así el peso de i y el del enlace que une i y j, respectivamente. La primera idea al intentar almacenar los pesos de todos los enlaces posibles entre n nodos podría ser una matriz de $n \times n$ dimensiones en la que el elemento de la fila i y la columna j es el peso del enlace que va de i a j, sin embargo, esto puede ocupar mucho más espacio de almacenamiento del necesario si la red dista mucho de ser completamente conexa, o sea, si hay muchos nodos no conectados entre sí, lo que se traduciría en muchos valores nulos en la matriz de pesos de los enlaces. Para solucionar este inconveniente, el algoritmo propone la utilización de arrays con longitud igual al número de estados ocupados cuyos componentes serán otros arrays de longitud variable, de manera que tendremos un array que almacena, para cada estado, los índices de todos los estados a los que va (sus salidas) y otro array que almacena, de forma equivalente, los pesos de estos enlaces. Todo esto sin ocupar más espacio del necesario.

De esta manera, los pasos a seguir para calcular la frecuencia de ocupación de los estados (*peso de los nodos*) y la frecuencia de tránsito de los enlaces (*peso de los enlaces*) con este método serán los siguientes:

1. Se toman los índices de los dos primeros estados de la trayectoria. El primero será el *estado de partida* y el segundo el *estado de llegada*.

2. Se suma 1 al peso del estado de partida.

3. Se comprueban todas las salidas conocidas del estado de partida para saber si se ha visitado antes el estado de llegada o, dicho de otra forma, si el enlace tiene peso no nulo hasta ahora porque ya se ha pasado por él. En caso afirmativo se suma uno al peso del enlace. Si no es así, se añade la existencia del nuevo enlace a las listas de datos preexistentes aumentando en 1 la longitud de los dos arrays. Para hacer esto se usan dos arrays auxiliares de longitud fija para almacenar temporalmente los valores del array que guarda los índices de los nodos alcanzables desde el estado de partida y los valores del array que guarda los pesos de los enlaces que van a esos nodos. Después se redefinen los dos arrays con un hueco más cada uno, se copian todos los valores de los arrays auxiliares y se llenan los huecos nuevos con el índice del estado de llegada y un 1 en el peso del enlace a ese estado.

4. En la próxima iteración, el que hasta ahora era el estado de llegada pasa a ser el nuevo estado de partida y tomaremos el siguiente paso de la trayectoria como el nuevo estado de llegada, para después repetir todo desde el paso 2.

5. Si hay más trayectorias en los datos proporcionados, se repite el proceso con ellas añadiendo las salidas y pesos necesarios a los ya almacenados en los arrays de longitud variable. En caso contrario, ha terminado el algoritmo.

El único fichero de salida que obtenemos en esta segunda parte del algoritmo (y el único necesario para el algoritmo de búsqueda de basins) es ***net_oldstyle.oup***, que muestra en su primera línea el número de nodos que tiene la red, el máximo número de enlaces que salen de un nodo y el número total de enlaces de la red. Las siguientes líneas almacenan, el índice de cada nodo seguido del número de enlaces de salida que tiene, el peso total del nodo sin normalizar, es decir, el número de veces que la trayectoria o trayectorias pasan por ese estado, y, por último, la sucesión de duplas que indican el índice del nodo al que se dirige cada enlace seguido del peso del enlace sin normalizar, es decir, el número de veces que la trayectoria pasa, a través de ese enlace, de un nodo al otro. Es relevante mencionar que todas estas listas de enlaces incluyen también los autoloops, correspondientes a esos pasos temporales en los que la trayectoria permanece en el mismo estado.

Para terminar de obtener la información completa de la red falta un paso extra que consistiría en normalizar los pesos de los nodos dividiéndolos por el peso total de todos los nodos y normalizar los pesos de los enlaces dividiendo cada uno por el peso total de todos los enlaces que salen del mismo nodo que ese. De esta forma tendremos la probabilidad de estar en un nodo i concreto (P_i) y la probabilidad de pasar por un enlace concreto que va del nodo i al nodo j condicionada a que estamos en i (W_{ji}). Esto lo hemos hecho al comienzo del programa de búsqueda de basins (saco_basins_red_original), donde creamos el fichero ***Pe.oup***, que asocia el índide de cada nodo de la red con su peso normalizado.

2.3. Algoritmo de búsqueda de ‘basins’ (atractores o cuencas)

El algoritmo de búsqueda de basins (programa saco_basins_red_original en el Anexo 2) utiliza un método de descenso de gradiente para asociar nodos que se encuentran en un mismo atractor, es decir, nodos desde los cuales la trayectoria tiende a moverse hacia un *mínimo local de la energía libre*, que estará en un nodo al que llamaremos representante de la comunidad. De nuevo, podemos distinguir dos partes bien diferenciadas en este algoritmo: identificación de los representantes de las basins y asignación del resto de nodos a estas basins.

2.3.1. Identificación de representantes

Para identificar los mínimos locales de energía límite vamos a barrer todos los nodos de la red y ver cuáles no están conectados a nodos con menor energía libre. Los pasos del algoritmo para encontrar estos

mínimos locales de energía libre en nuestro espacio discreto y definir de esta manera los representantes de las basins son:

1. Se leen y guardan todos los datos de la red obtenidos del algoritmo anterior (el fichero net_oldstyle.oup).
2. Se toma un nodo que no se haya comprobado ya.
3. Para buscar los mínimos de energía libre se sigue un descenso de gradiente siguiendo el razonamiento de que cuanto más pesado sea un nodo o un enlace, menor será su energía libre. De esta forma se procede a buscar, para el nodo actual, el enlace más pesado que sale de él y que no haya sido descartado. Si hay varios enlaces no eliminados con peso máximo se priorizará aquel de ellos que vaya al nodo más pesado.
4. Si el nodo al que va el enlace seleccionado es más pesado que el nodo actual, sabremos que el nodo actual no es un mínimo local, así que habrá terminado esta iteración y se puede pasar a estudiar el siguiente nodo de la red, volviendo al paso 2. De no ser así, se descartará este enlace como posible camino de energía libre decreciente y se volverá al paso 3. El bucle 3-4 se repetirá hasta que el nodo candidato sea más pesado que el nodo actual o hasta que miremos y rechacemos un número de enlaces igual a un cierto cutoff preestablecido. Si se alcanza el cutoff, el nodo que estamos estudiando se considerará un mínimo local de energía libre y será el representante de su propia basin.

En este procedimiento es muy relevante la elección del valor adecuado para el cutoff, ya que puede afectar significativamente al número de basins que se definan. En general, un valor recomendable de este parámetro es el doble del número de variables con el que estemos trabajando.

2.3.2. Asignación de los nodos

Asumiremos que los sistemas con los que se trabaje van a tener siempre al menos un mínimo local detectable. En ese caso, una vez identificados los representantes de las basins toca asignar cada nodo de la red a una de estas basins. Para ello se realizará otro proceso de descenso de gradiente con todos los nodos restantes utilizando la misma metodología en la selección de los enlaces y nodos candidatos, o sea, buscar el enlace o enlaces más pesados y elegir, entre los enlaces de peso máximo, el que vaya al nodo más pesado. Los pasos que sigue esta segunda mitad del algoritmo son los siguientes:

1. Se toma un nodo que no haya sido asignado aún a una basin como *nodo de partida* y se selecciona un candidato entre los enlaces que salen de él utilizando el mismo criterio que en la primera mitad del algoritmo. Denominaremos al nodo hacia el que va el enlace seleccionado *nodo de llegada*.
2. Si el nodo de partida es más pesado que el de llegada se descarta el enlace y se elige el siguiente que mejor cumpla las condiciones. Esta sucesión de descarte y reselección se repite hasta que el nodo de llegada pese más que el de partida o hasta que se haya descartado todos los enlaces. En este último caso se seleccionará el primer nodo que se había elegido por ser el que mejor cumple las condiciones.
3. Una vez obtenido el candidato final a nodo de llegada se guarda el nodo de partida en un array y, si el nodo de llegada no está asignado aún a una basin ni se ha pasado por él previamente en la trayectoria actual, se convertirá en el nuevo nodo de partida, reiniciando el proceso de selección con sus enlaces. De esta manera, se va recorriendo un camino por la red, almacenando todos los nodos por los que se pase hasta llegar a uno que ya esté asociado a una basin o en un nodo por el que la trayectoria ya haya pasado antes, lo que significaría que se ha caído en un ciclo.
4. Si el nodo final de llegada está asignado a una basin se asignarán todos los nodos recorridos a la misma basin a la que pertenezca este. Si, por el contrario, no está asignado a una basin se considerará que es el representante de una nueva basin.

5. Se repite todos los pasos desde el principio con el siguiente nodo no asignado hasta que todos los nodos de la red pertenezcan a alguna basin.

Los archivos que obtenemos en nuestra aplicación concreta de este algoritmo son:

- ***minimos.oup***: Almacena los índices y los pesos normalizados de los representantes de todas las basins detectadas en la primera mitad del algoritmo o, equivalentemente, de todos los nodos de la red donde hay mínimos locales de la energía libre hallados en esa primera parte.

- ***lista_nodos_basins.dat***: Asocia el índice de cada nodo con el índice del representante de la basin a la que pertenece.

- ***info_basins.dat***: Tiene dos partes. Primero asocia los índices de los nodos representantes de cada basin con el número de nodos en esa basin. Después, se muestra una lista de los índices de todos los nodos ordenados por basin.

- ***Krates.oup***: Indica, para cada basin, el tiempo de espera para escapar de ella a una basin diferente (calculado según se explica en [2]) y el peso total normalizado de la basin, es decir, la probabilidad de estar en ella.

- ***KAS***: Asocia a cada par de basins i y j el peso total normalizado de todos los enlaces que van de i a j, o sea, la probabilidad de ir a j condicionada a que se empieza en i. Luego, una segunda parte relaciona cada basin con su representante.

- ***nodos.csv*** y ***conectividades.csv***: Muestran la información de la red en el formato adecuado para importar los archivos en Gephi y representar el grafo.

- ***nodos_basins.csv*** y ***conectividades.csv***: Muestran la información de la red de basins en el formato adecuado para importar los archivos en el programa utilizado para representar el grafo (Gephi).

Ninguno de estos ficheros es estrictamente necesario para el algoritmo, solo son diferentes maneras de agrupar y presentar la información obtenida.

2.4. Modelo de ejemplo

2.4.1. Potencial

Para comprobar el funcionamiento de los algoritmos con un ejemplo cuyos resultados conocemos vamos a aplicarlos a un modelo de ejemplo: una función en 2 dimensiones de un triple pozo o doble pozo con un estado intermedio.

$$V(x,y) = -A_N \mathrm{e}^{-\frac{(x-X_N)^2+(y-Y_N)^2}{S_N}} - A_I \mathrm{e}^{-\frac{(x-X_I)^2+(y-Y_I)^2}{S_I}} - A_D \mathrm{e}^{-\frac{(x-X_D)^2+(y-Y_D)^2}{S_D}} + \frac{x^2}{5} + \frac{y^2}{5} \tag{2.1}$$

Hemos utilizado los siguientes parámetros:

X_N	-2	Y_N	-2	A_N	25	S_N	2
X_D	2,5	Y_D	2,5	A_D	20	S_D	4
X_I	0	Y_I	0	A_I	10	S_I	1,5

De esta manera, el potencial con el que trabajaremos adquiere la siguiente forma:

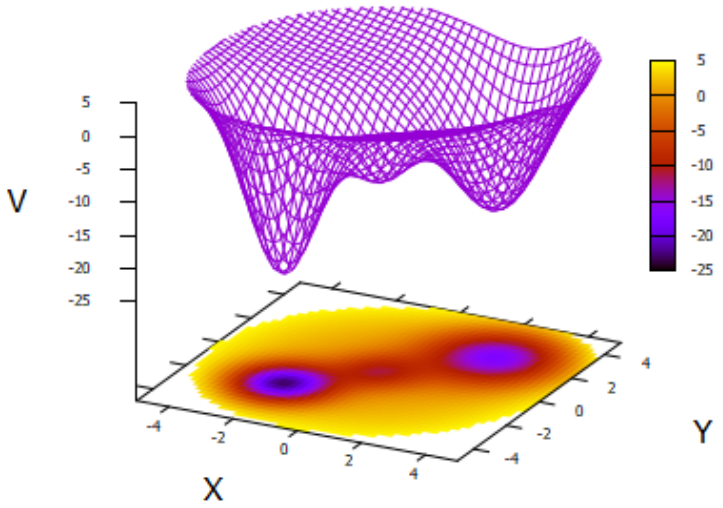


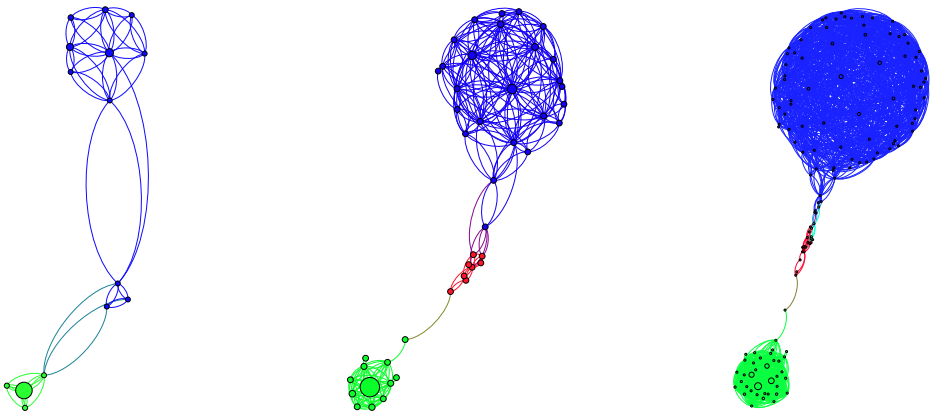
Figura 2.1: Potecial de ejemplo.

Con este potencial, lo que esperamos obtener al aplicar los algoritmos de formación de la red y de búsqueda de basins es que se detecten dos basins principales, con una un poco más grande que la otra, y una tercera basin más pequeña que las anteriores correspondiente al estado intermedio. La presencia de la contribución cuadrática impide que la partícula escape de la zona de los pozos.

2.4.2. Resultados

Los datos introducidos a los algoritmos corresponden a 10 trayectorias de 1.200.000 pasos temporales. Queremos observar como cambian los resultados de los algoritmos al definir la red a partir de distintos mallados de bins o al cambiar el salto de tiempo de los pasos de las trayectorias.

Con este propósito, comenzamos observando los resultados para el conjunto de datos original con mallados de 5×5 bins, 10×10 bins y 20×20 bins:



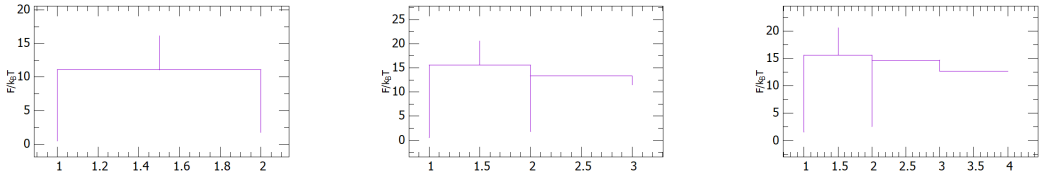
(a) 5×5 (b) 10×10 (c) 20×20

Figura 2.2: Redes para distintos mallados.

A continuación, haremos los dendogramas que relacionan la energía libre de los representantes de cada basin y los enlaces que las unen utilizando un programa que calcula la energía libre con la siguiente fórmula:

$$\frac{F_I}{k_B T} = -\log P_I, \quad \frac{F_{JI}}{k_B T} = -\log W_{JI}$$

donde F_I y F_{JI} son las energías libres del nodo representante de la basin I y del enlace que conecta las basins I y J, respectivamente; P_I es el peso normalizado del nodo representante de la basin I y W_{JI} es el peso del enlace que une las basins I y J.



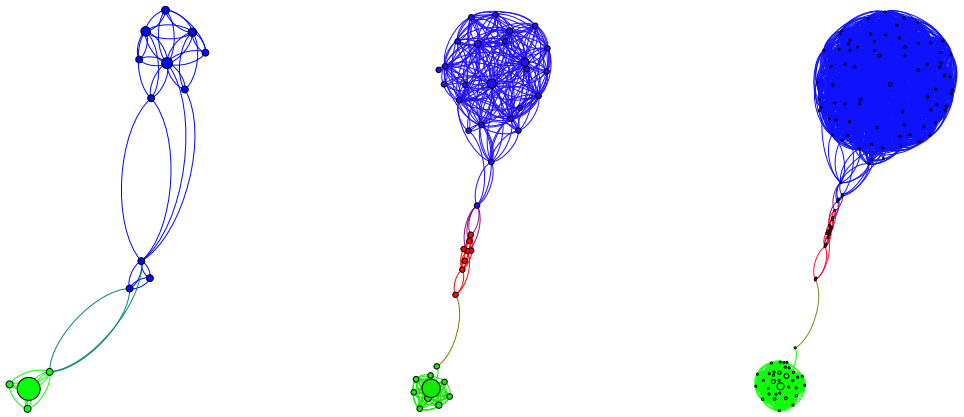
(a) 5×5 (b) 10×10 (c) 20×20

Figura 2.3: Dendogramas de la energía libre de las basins para distintos mallados. La altura de las líneas horizontales representa la energía libre de los enlaces (F_{JI}), la altura del extremo inferior de las líneas verticales indica la energía libre de los nodos representantes de las basins (F_I) cuyos índices están indicados en el eje x.

Se puede observar como al hacer mallados más finos (con más bins) se obtienen muchos más estados ocupados (nodos) y aparecen nuevas basins de menor tamaño que los primeros mallados no nos permitían ver debido a su baja definición. Esto se ve claramente comparando la red y el dendograma del mallado de 5×5 bins, donde no se detecta el estado intermedio, con los correspondientes del mallado de 10×10 bins.

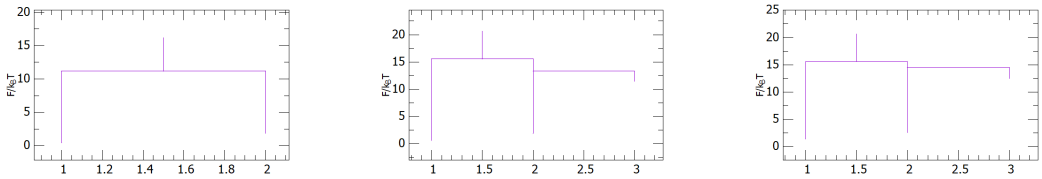
Sin embargo, este aumento en los estados conlleva que cada vez sea más difícil extraer información útil de la representación de la red al no poder distinguirse bien los enlaces. Además, mallados demasiado finos podrían llevar a que el algoritmo de búsqueda de basins considerase como una basin un conjunto de estados por los que en realidad la trayectoria solo ha pasado debido a fluctuaciones estadísticas, como parece suceder en este ejemplo para el mallado de 20×20 bins, en el que aparecen dos estados intermedios con el mismo peso en vez de uno.

A continuación, para comprobar cómo afecta cambiar la longitud de los pasos temporales vamos a tomar el mismo conjunto de datos saltándonos un paso de cada dos y dos pasos de cada tres, de forma que tendremos trayectorias equivalentes a las originales con pasos el doble y el triple de largos, respectivamente.



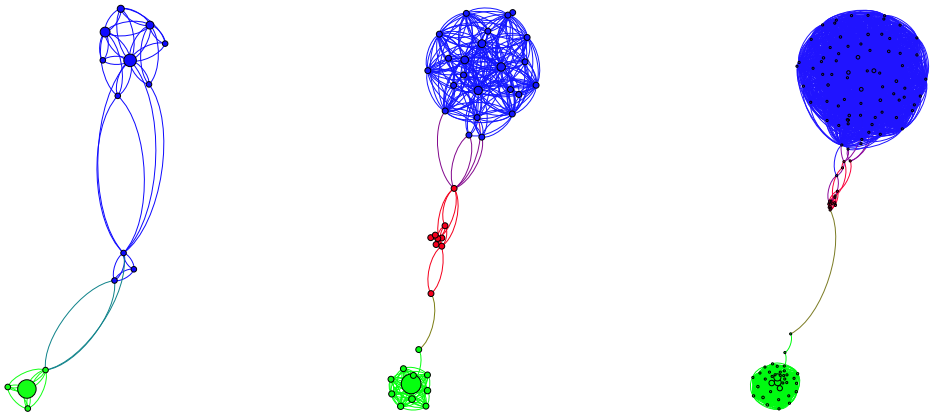
(a) 5×5 (b) 10×10 (c) 20×20

Figura 2.4: Redes para distintos mallados del caso con pasos el doble de largos que los originales.



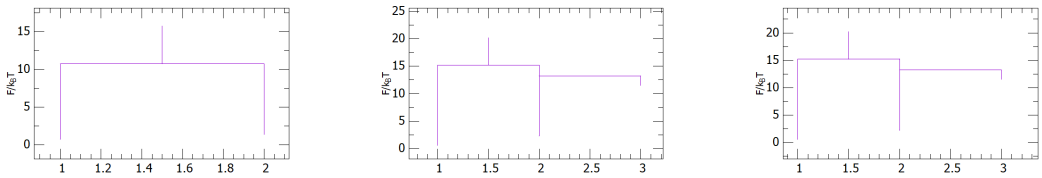
(a) 5×5 (b) 10×10 (c) 20×20

Figura 2.5: Dendogramas de la energía libre de las basins para distintos mallados del caso con pasos el doble de largos que los originales.



(a) 5×5 (b) 10×10 (c) 20×20

Figura 2.6: Redes para distintos mallados del caso con pasos el triple de largos que los originales.



(a) 5×5 (b) 10×10 (c) 20×20

Figura 2.7: Dendogramas de la energía libre de las basins para distintos mallados del caso con pasos el triple de largos que los originales.

En ambos casos, el único cambio notable es que el mallado de 20×20 bins ya no presenta el doble estado intermedio que aparecía en el caso original. Solo esto no nos da mucha información de como afecta a los resultados la pérdida de información que supone la eliminación de pasos realizada, pero, como mínimo, sirve para demostrar que, efectivamente, existe una cierta dependencia con la longitud de los pasos para trayectorias equivalentes. Hay que tener en cuenta que realmente se ha perdido información, porque las trayectorias empiezan y terminan en los mismos puntos a pesar de tener distintos pasos temporales, es decir, que la primera variante tiene la mitad de pasos temporales que los datos originales y la segunda variante un tercio.

Capítulo 3

Aplicación al G-quadruplex

3.1. Modelo

Ahora aplicaremos el método de estudio que se ha desarrollado a un modelo mesoscópico, ideado en un artículo anterior ([11, 16]), de dos estructuras de G4 concretas. Las estructuras G4 con las que trabajaremos, como se explica en la introducción, son cadenas de 21 nucleótidos (al menos 12 de ellos guaninas) plegadas de una forma muy concreta en torno a dos iones. En nuestro modelo se simularán 24 objetos: una partícula fija que servirá de referencia para el sistema e impedirá, mediante una fuerza de Hooke, que este se aleje demasiado del punto de partida; los 21 monómeros, correspondientes a todos los nucleótidos; y los 2 iones.

Estos objetos se verán sometidos a las siguientes interacciones:

- Una fuerza de Hooke entre los monómeros consecutivos de la cadena.

$$U_{str} = \sum_i \frac{1}{2} k_s \left(l_i - l_0\right)^2$$

  donde k_s es la constante elástica entre dos monómeros consecutivos, l_i las distancias entre monómeros y l_0 una longitud de relajación establecida mediante datos bibliográficos.

- Un potencial de Morse entre guaninas contiguas de un mismo plano para simular los enlaces de hidrógeno entre ellas, con un factor D_G que tiene en cuenta el efecto de cooperatividad que fortalece los enlaces de hidrógeno cuando las cuatro guaninas están unidas.

$$U_{GG} = \sum_p \sum_g D_G \left(e^{-\alpha_G\left(d_{GGg} - r_0\right)} - 1\right)^2, \quad \text{con } D_G = D_0 \left[1 + 2e^{-\delta\left(\sum_g d_{GGg} - 4r_0\right)}\right]$$

  donde p indica los diferentes planos, g se refiere a las distintas guaninas de cada plano, d_{GGg} es la distancia entre guaninas contiguas de un plano, r_0 es la longitud de equilibrio del lado de cada plano y δ es la longitud de decaimiento del factor D_G.

- Un potencial entre cada ión y todas sus guaninas cercanas (U_{IG}) que aúna el potencial de Coulomb con una fuerza repulsiva que simula el volumen excluido de los iones. Así mismo también se tiene en cuenta la repulsión coulombiana entre los dos iones ($U_{I_1I_2}$).

$$U_{IG} = \sum_i \left[\sum_g \left(\frac{A}{d_{ig}^{12}} - \frac{Q_{ig}}{d_{ig}}\right)\right], \quad U_{I_1I_2} = \frac{Q_{I_1I_2}}{d_{I_1I_2}}$$

  donde los índices i indican cada uno de los dos iones y los índices g identifican a las 8 guaninas vecinas del ion correspondiente. Además, la interacción coulombiana ion-guanina, es más débil la interacción ion-ion, por lo que $Q_{I_1I_2}$ será mayor que Q_{ig} y la interacción Q_{ig}/d_{ig} se anulará en cuando d_{ig} esté en cierto intervalo $[a,b]$ que actúa a modo de cutoff.

- La energía de doblado entre las tres guaninas de una misma arista.

$$U_{ben} = \sum_i \frac{1}{2} k_b \left[1 - \cos\left(\theta_i - \theta_0\right)\right]$$

  donde k_b es la constante elástica de doblado, θ_i es el ángulo entre los dos enlaces de monómeros de la arista (l_i y l_{i+1}) y θ_0 es el ángulo de equilibrio. A esta componente se le sumará también una *energía de bending de los enlaces que forman los loops* entre aristas de la estructura. Su fórmula será como la anterior, pero su constante elástica será $B \leq k_b$ y tomará dos posibles valores: 0 cuando consideremos sistemas sin bending en los loops o 2 cuando incluyamos el bending.

- Un potencial de Lennard-Jones repulsivo entre todos los objetos para representar el efecto del volumen excluido cuando las partículas se acerquen entre sí a distancias inferiores a $1{,}122\sigma$.

$$U_{LJ} = \sum_i \sum_{j>i} 4\varepsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right]$$

Obviamente, este conjunto de interacciones solo podrá ser una buena aproximación a la realidad cuando el sistema tome la forma prevista de la estructura, ya que si la posición relativa de los nucleótidos o de los iones distase mucho de la del G4 podría haber otras interacciones más relevantes que las que se están teniendo en cuenta, como las fuerzas de Coulomb que se están despreciando o las energías de doblado del resto de la cadena.

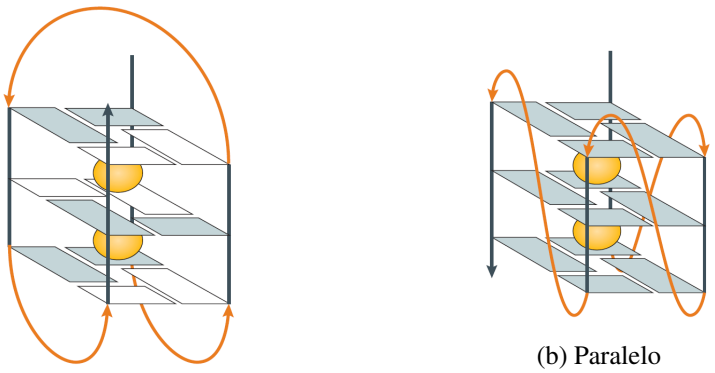
Las trayectorias se generan con una dinámica de Langevin sobreamortiguada, por lo que el término inercial es despreciable con respecto al término cinético. De manera que tenemos dos expresiones: una para los monómeros y otra para los iones, respectivamente.

$$\gamma_j \dot{r}_j = -\nabla_{r_j} U_g + \sqrt{2k_B T \gamma_j}\eta_j(t)$$

$$\gamma_i \dot{I}_i = -\nabla_{I_i} U_I + \sqrt{2k_B T \gamma_i}\eta_i(t)$$

donde γ es el coeficiente de amortiguamiento y los potenciales utilizados son $U_g = U_{str} + U_{GG} + U_{IG} + U_{ben} + U_{LJ}$ y $U_I = U_{IG} + U_{I_1 I_2}$. Ambas expresiones tienen además una componente estocástica.

Lo que estudiaremos serán simulaciones que utilizan este modelo en dos estructuras de G4: una *paralela*, en la que la cadena recorre todas las aristas en el mismo sentido, por lo que lo que los loops irán de un extremo a otro de la estructura; y una *antiparalela*, donde dos aristas se recorren en un sentido y las otras dos en el opuesto de forma alterna, lo que hace que los loops unan distintos vértices de un mismo extremo de la formación.



(a) Antiparalelo

(b) Paralelo

Figura 3.1: Visión esquemática de las estructuras antiparalela y paralela del G-quadruplex con las que trabajaremos.[13]

3.2. Reducción de variables

Aunque ignoraremos las coordenadas de la partícula fija porque no aportan información y solo sirve para mantener el resto del sistema en una misma zona, seguimos teniendo 23 partículas con 3 dimensiones de coordenadas cada una, es decir, 69 variables, una cantidad demasiado alta como para trabajar cómodamente. Para reducir el número de variables de los datos a algo manejable manteniendo la mayor cantidad de información relevante posible vamos a aplicar el método de componentes principales (PCA) a las trayectorias simuladas antes de introducirlas en los algoritmos.

3.2.1. PCA

El estudio del paisaje de energía libre no es algo sencillo en este caso, ya que supone trabajar con el espacio de los posibles estados de la macromolécula, cuyas dimensiones son todas las variables que definen la configuración exacta del polímero, es decir, 69. Trabajar con este volumen de datos puede resulta computacionalmente prohibitivo o, como mínimo, incómodo, por lo que trataremos de reducir el número de variables al mínimo que resulte de interés observar siguiendo el método de componentes principales (PCA, Principal Component Analysis).

La matriz de covarianza de una n-tupla ordenada de variables es una matriz simétrica que tiene en la diagonal la varianza de cada variable y, en el resto de posiciones, las covarianzas de las variables asociadas a la fila y la columna correspondientes, es decir:

$$A_{ij} = A_{ji} = Covar(i,j) = \overline{ij} - \bar{i}\bar{j}$$

donde i y j las variables i-ésima y j-ésima, respectivamente, y $\bar{i}$ indica el promedio temporal de la variable i, en el caso de $\overline{ij}$, el promedio del producto ij.

El método de componentes principales toma la matriz de covarianza de todas las variables a estudiar y calcula sus autovalores, aprovechando el hecho de que algunas de estas variables están mucho más interrelacionadas que otras, lo que resultará en que unos pocos autovalores de la matriz serán mucho mayores que los demás, por lo que nos centraremos en las nuevas variables que podemos definir a partir de esta matriz, que acumularán la mayor parte de la información necesaria para distinguir unos estados de otros y que se llamarán componentes principales. Los valores de cada una de estas variables se obtendrán proyectando los datos originales sobre los autovectores correspondientes a cada autovalor. Finalmente, nos quedaremos únicamente con las componentes correspondientes a los autovalores más altos, teniendo en cuenta que cuantas menos variables tomemos más fácil será trabajar con ellas pero más información se perderá, haciendo peor la aproximación. Tendremos así un nuevo conjunto de datos que describirá nuestro sistema con un número mucho más reducido de variables, permitiéndonos, ahora sí, trabajar con un espacio de estados más razonable. [8, 9, 10]

Cuando las variables de un sistema se pueden reducir adecuadamente, es habitual observar que los autovalores de mayor valor son órdenes de magnitud más altos que todos los demás, lo que indica que sus componentes principales asociadas almacenan la mayor parte de la información relevante. En nuestro caso particular hemos decidido tomar las 3 primeras componentes principales, que suponen entre un 60 % y un 95 % del valor de la suma total de todos los autovalores según el caso.

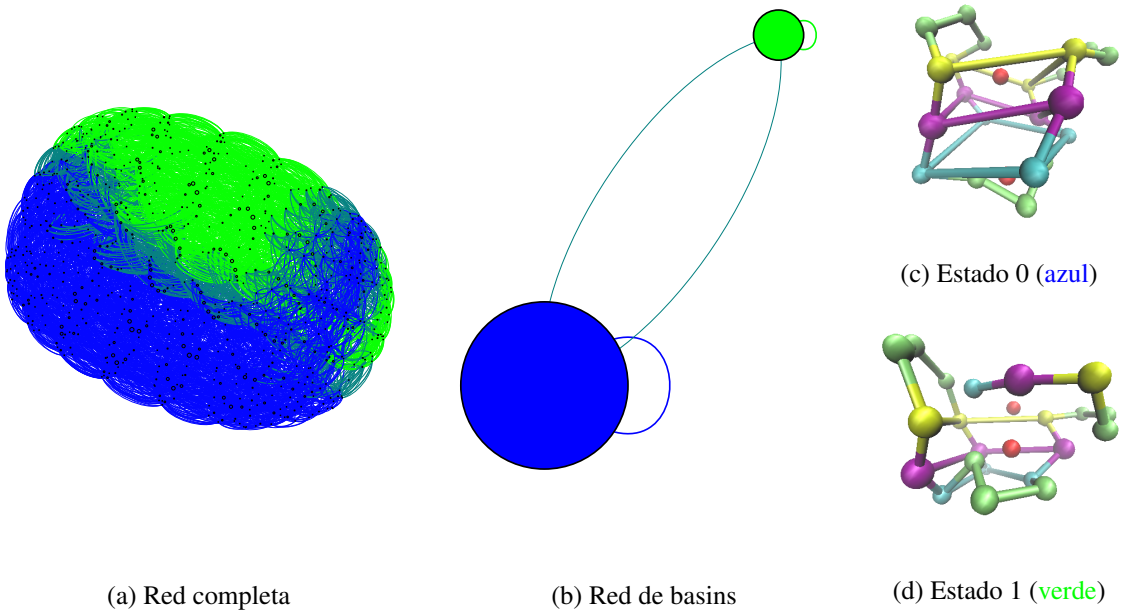
3.3. Resultados

Sabemos que la temperatura de melting del G4 es $T_m = 65^oC = 338K$. En la escala que utiliza nuestra simulación, la temperatura de melting de ambas estructuras sin tener en cuenta el bending de los loops $T_m^* = 0{,}4875$ ([11]). Sin embargo, el efecto del bending disminuye la temperatura de melting, sobre todo en la estructura paralela, que es mucho más sensible a ello ([16]), así que, como el sistema con bending debería ser más cercano a la realidad, entendemos que la verdadera temperatura de melting debería ser menor que 0,4875. Nos interesa estudiar las estructuras a una temperatura cercana a la de melting para intentar observar situaciones de desplegamiento y replegamiento de la cadena, así que las simulaciones

principales se realizarán a $T^* = 0{,}4$. Estas simulaciones consistirán en trayectorias de 4.500.000 pasos temporales del modelo presentado.

3.3.1. Temperatura 0,4 sin bending ($B = 0$)

Obtenemos las redes de estados individuales y de basins resultantes con nuestro método para un mallado de $10 \times 10 \times 10$ bins en la estructura antiparalela, sin considerar el bending de los loops. Sus representaciones gráficas son las siguientes:



(a) Red completa

(b) Red de basins

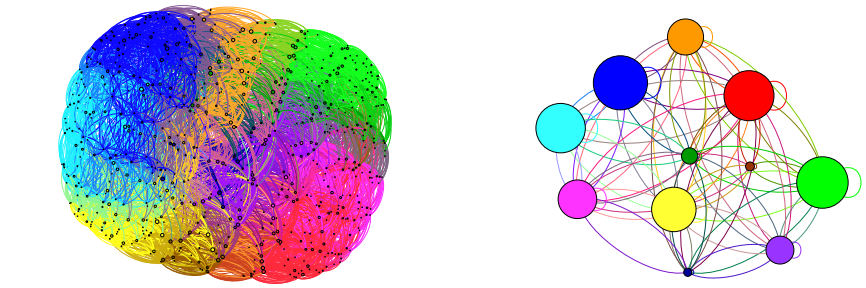
(c) Estado 0 (azul)

(d) Estado 1 (verde)

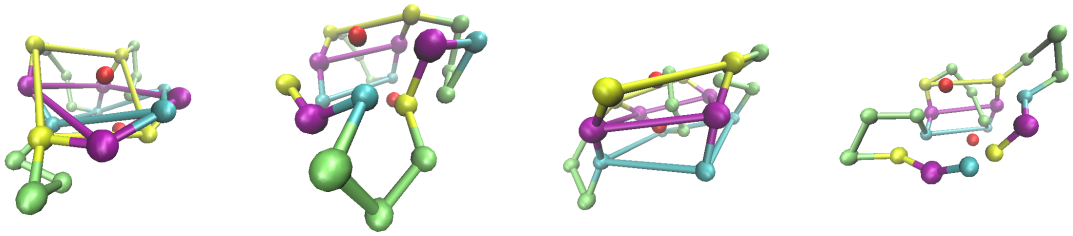
Figura 3.2: Grafos para un mallado $10 \times 10 \times 10$ del *sistema antiparalelo sin bending* a temperatura 0,4 y estructuras de los estados representantes de las basins (dibujadas utilizando el programa de visualización de moléculas VMD).

Como se puede observar, el estado fundamental, efectivamente, tiene la forma esperada, aunque los planos se doblan ligeramente y los iones no están del todo centrados entre un plano y otro. El otro estado, sin embargo, no forma la estructura completa debido a que la arista compuesta por los tres primeros monómeros no se alinea con las demás, permitiendo únicamente 3 guaninas en cada plano, lo que da más libertad de movimiento a los iones.

Ahora realizaremos las observaciones correspondientes para nuestro otro objeto de estudio. Los grafos análogos para la estructura paralela son estos:



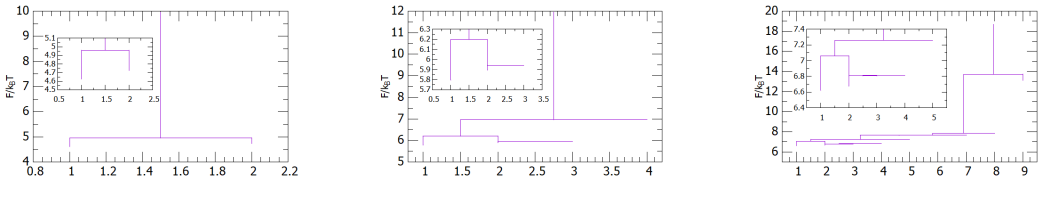
(a) Red completa (b) Red de basins



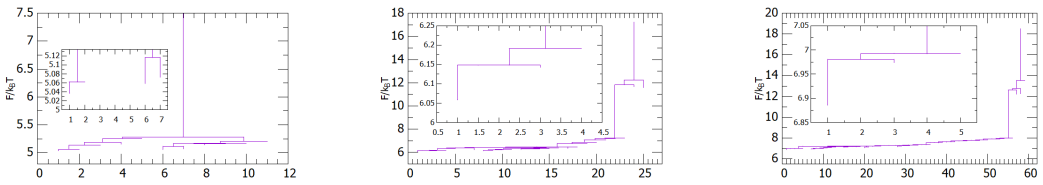
(c) Estado 0 (azul) (d) Estado 1 (verde) (e) Estado 2 (rojo) (f) Estado 3 (cian)

Figura 3.3: Grafos para un mallado $10\times10\times10$ del *sistema paralelo sin bending* a temperatura 0,4 y estructuras de los estados representantes de las basins principales.

En los grafos se observan varias basins de peso similar, lo que podría significar una degeneración de estados nativos. En las estructuras, esta vez vemos que, aunque la cadena permanece rodeando los iones y, en algunos caso, llegan a alinearse algunas aristas, ninguno de los estados principales presenta la estructura esperada. A continuación, usamos un programa que asigna valores de energía libre a los nodos representantes de las basins y a los enlaces entre ellas en función de su peso para obtener los dendogramas que comparan la energía libre de los representantes de las basins de las trayectorias estudiadas con distintos mallados:



(a) $10\times10\times10$ antiparalelo (b) $15\times15\times15$ antiparalelo (c) $20\times20\times20$ antiparalelo



(d) $10\times10\times10$ paralelo (e) $15\times15\times15$ paralelo (f) $20\times20\times20$ paralelo

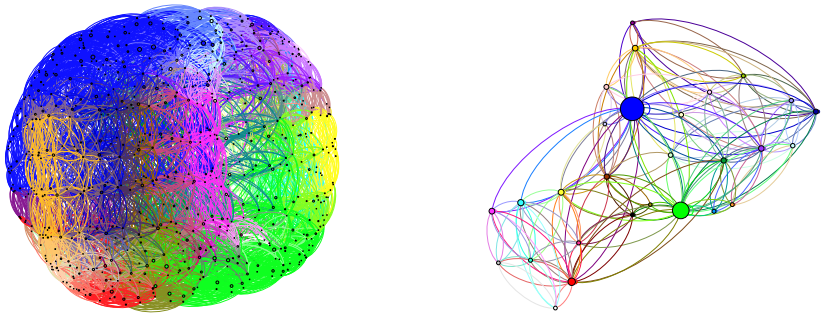
Figura 3.4: Dendogramas de la energía libre de las basins para distintos mallados de los sistemas sin bending a temperatura 0,4.

Al probar mallados más finos que el de $10 \times 10 \times 10$ vemos que los estados se desdoblan en varias basins con pequeñas diferencias de energía libre entre ellas sin afectar a la estructura global del dendograma. Esto hace que no siempre sea evidente un estado fundamental individual, y que, al aumentar la precisión del mallado, aparezcan subestados en torno a los mínimos observados a mallados más bajos que representan configuraciones menos relevantes que los estados fundamentales más importantes, evidentes ya en el mallado más bajo.

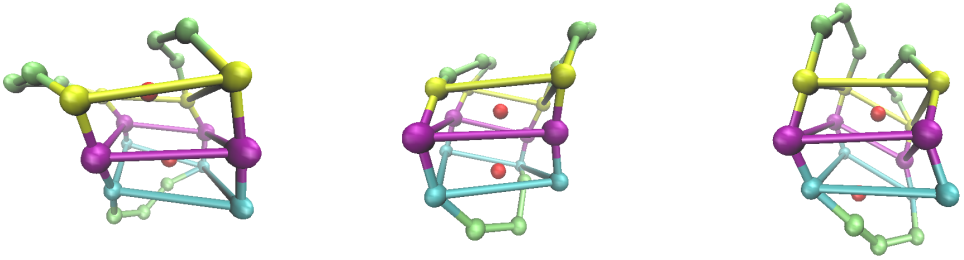
3.3.2. Temperatura 0,4 con bending ($B = 2$)

Comprobemos ahora cómo cambian los resultados al añadir el bending de los loops a la simulación. Esto, idealmente, debería forzar los loops a tomar una cierta forma más tensa, evitando que queden tan libres por la estructura como en algunos de los ejemplos anteriores y obligando también a la propia estructura a abrirse un poco más.

Veamos pues los grafos de estados individuales y de basins resultantes para un mallado de $10 \times 10 \times 10$ bins en la estructuras antiparalela:



(a) Red completa (b) Red de basins

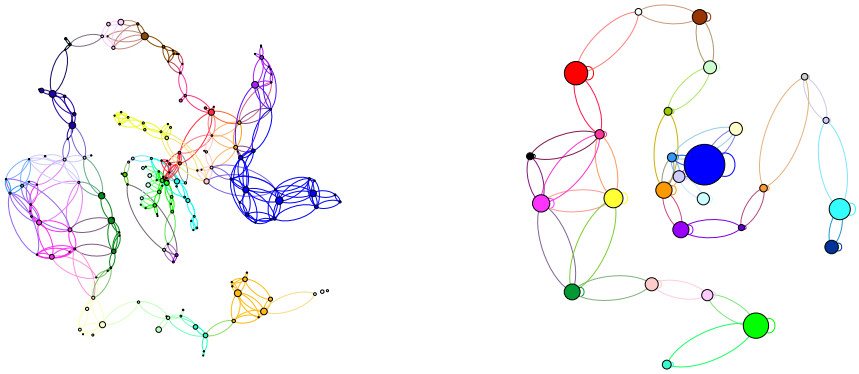


(c) Estado 0 (azul) (d) Estado 1 (verde) (e) Estado 2 (rojo)

Figura 3.5: Grafos para un mallado $10 \times 10 \times 10$ del *sistema antiparalelo con bending* a temperatura 0,4 y estructuras de los estados representantes de las basins principales.

Está claro que en los tres estados representados se ha formado la estructura antiparalela esperada correctamente, incluso si en algunos casos los planos parecen doblarse un poco. Además, los iones parecen estar mejor contenidos entre los planos que en el caso sin bending y, como preveíamos, los loops toman ahora una forma más arqueada. Estas observaciones son especialmente relevantes en los dos primeros estados (0 y 1), ya que se corresponden con los representantes de las dos basins más pesadas de la red con mucha diferencia.

Veamos si esta nueva adición mejora también la representación de la estructura paralela:



(a) Red completa (b) Red de basins

Figura 3.6: Grafos para un mallado $10\times10\times10$ del *sistema paralelo con bending* a temperatura 0,4.

Esta vez, al mirar la red de basins, nos encontramos con un grafo casi unidimensional, con un nodo especialmente destacable y algunas otras basins relevantes por debajo de él. Es importante mencionar en este caso que ninguna de las basins más relevantes ha sido definida en la primera parte del algoritmo de búsqueda de basins, sino en la segunda. Solamente 4 de las 32 basins se definen en la primera mitad del algoritmo.

Una posible explicación de esto sería una abundancia de mínimos locales en los que el nodo tiene menos enlaces de salida que el valor del cutoff (en nuestro caso $cutoff = 6$). En esa circunstancia, por cómo está estructurado el algoritmo, la primera parte no detectaría estos nodos como mínimos locales de energía libre, pero es fácil que en la segunda mitad, para cada uno de estos mínimos, se produzca un ciclo en el que la trayectoria caiga dos veces en él, reconociéndolo entonces como el representante de una nueva basin. Esta no es necesariamente la única manera de que se detecte un mínimo de energía libre en la segunda mitad del algoritmo, ya podría haber otras situaciones que explicarían que se formase un ciclo y se detectase como representante de una basin a un nodo que realmente no sea estrictamente un mínimo de energía libre. De hecho, en este caso particular, ninguna de las basins más pesadas cumple la hipótesis de que su representante tenga menos de 6 enlaces de salida, lo que no significa que esa no pueda ser la situación de alguna de las otras basins.

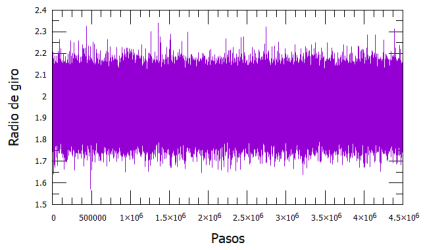
En cualquier caso, este resultado hace que volvamos la mirada hacia las posibles limitaciones del algoritmo de búsqueda de basins y hacia pequeñas variaciones que tal vez se podrían realizar para mejorarlo.

Tal vez ver los estados el aspecto de los representantes de las basins principales nos ayude a entender qué está pasando en este sistema:

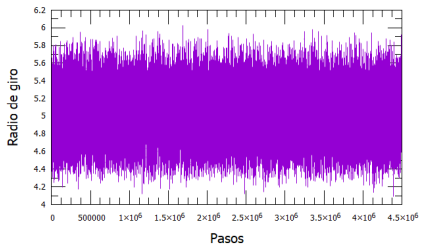
(a) Estado 0 (azul) (b) Estado 1 (verde) (c) Estado 2 (rojo)

Figura 3.7: Representantes de los estados principales de la red de basins del *sistema paralelo con bending* a temperatura 0,4.

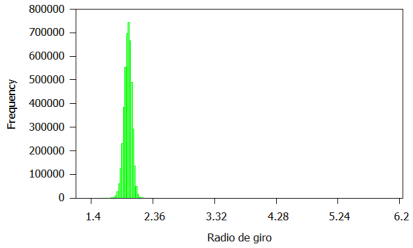
Está claro que la cadena se ha desplegado. Que esto se observe en los tres estados más estables hace razonable pensar que sucederá en todo el sistema, lo cual se puede comprobar viendo la evolución del radio de giro a lo largo de la trayectoria y comparándola con otra en la que sepamos que el despliegue no sucede:



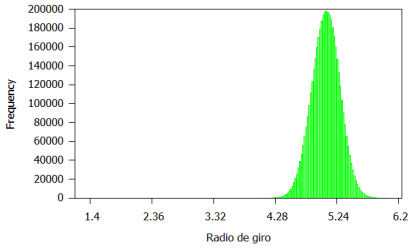
(a) Paralelo sin bending



(b) Paralelo con bending



(c) Histograma del paralelo sin bending



(d) Histograma del paralelo con bending

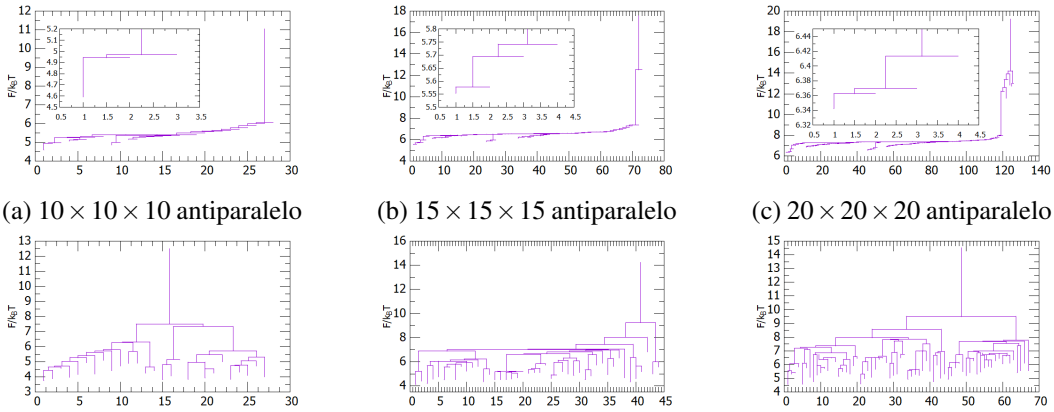
Figura 3.8: Representación e histogramas del radio de giro a lo largo de las trayectorias para la estructura paralela sin bending y con bending a temperatura 0,4.

Efectivamente, podemos ver que el radio de giro del sistema paralelo con bending toma valores considerablemente mayores que el del sistema paralelo sin bending durante todo el transcurso de ambas trayectorias, lo que, unido a nuestras observaciones anteriores, confirma definitivamente que la cadena permanece desplegada todo el tiempo. Esto indica que la temperatura de melting de la estructura paralela con bending es menor que 0,4, lo que cuadra con lo que ya sabíamos de trabajos previos ([16]), que muestran que la temperatura de melting disminuye al aumentar el bending y que la estructura paralela es mucho más sensible a esta disminución que la antiparalela.

Originalmente teníamos expectativas de poder observar el proceso de desplegado térmico seguido del replegado de la estructura, sin embargo, en la representación de la molécula vemos que cuando el G-quadruplex se despliega por completo pierde los iones. Al alejarse los iones, el desplegado del modelo resulta irreversible.

La conclusión de este resultado particular es que la simulación utilizada falla en replicar el replegado de la estructura, ya que, en una situación real, es probable que los iones que se pierdan durante el desplegado sean reemplazados por otros iones del medio, permitiendo así que el replegado se produzca a pesar de que los iones originales se hayan alejado de la molécula. Para representar esto correctamente haría falta una simulación que tuviese en cuenta la concentración de iones del medio y contemplase la posible sustitución de alguno de los iones del G-quadruplex por un ion del medio en caso de desplegarse la estructura.

Veamos los dendogramas de estas dos nuevas trayectorias estudiadas con distintos mallados:



(a) $10 \times 10 \times 10$ antiparalelo (b) $15 \times 15 \times 15$ antiparalelo (c) $20 \times 20 \times 20$ antiparalelo

(d) $10 \times 10 \times 10$ paralelo (e) $15 \times 15 \times 15$ paralelo (f) $20 \times 20 \times 20$ paralelo

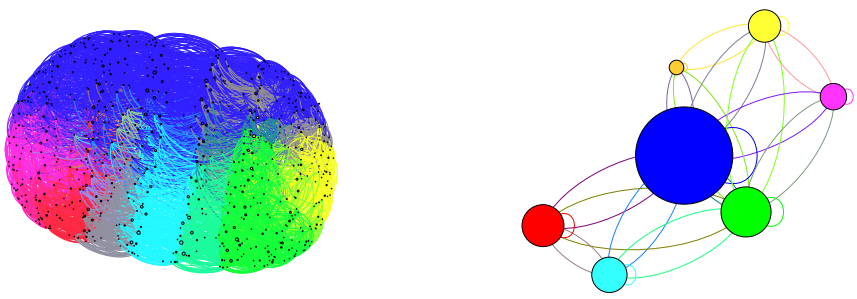
Figura 3.9: Dendogramas de la energía libre de las basins para distintos mallados de los sistemas con bending a temperatura 0,4.

En el caso del sistema paralelo podemos ver cómo, que la estructura esté desplegada, le da a la cadena la libertad de movimiento suficiente como para que el sistema se despliegue en una gran cantidad de estados no interconectados, muchos de ellos degenerados o, al menos, de energía libre similar.

3.3.3. Temperatura menor

Estudiemos cómo cambian los sistemas observados a una temperatura menor, $T^* = 0{,}3$. En esta situación las estructuras deberían ser más estables, ya que nos alejamos de la temperatura de melting y las partículas tienen menos energía térmica.

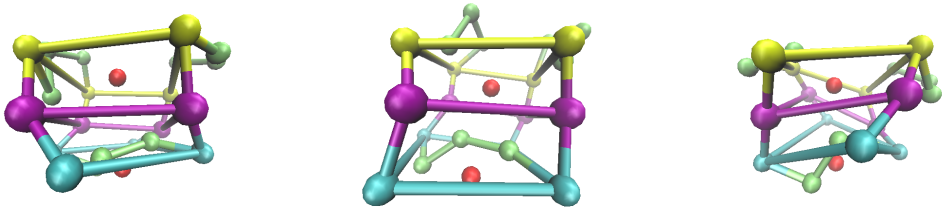
Los grafos de estados individuales y de basins resultantes para un mallado de $10 \times 10 \times 10$ bins en la estructura antiparalela son:



(a) Red completa (b) Red de basins

Figura 3.10: Grafos para un mallado $10 \times 10 \times 10$ del *sistema antiparalelo sin bending* a temperatura 0,3.

Y los representantes de las basins principales tienen el siguiente aspecto:

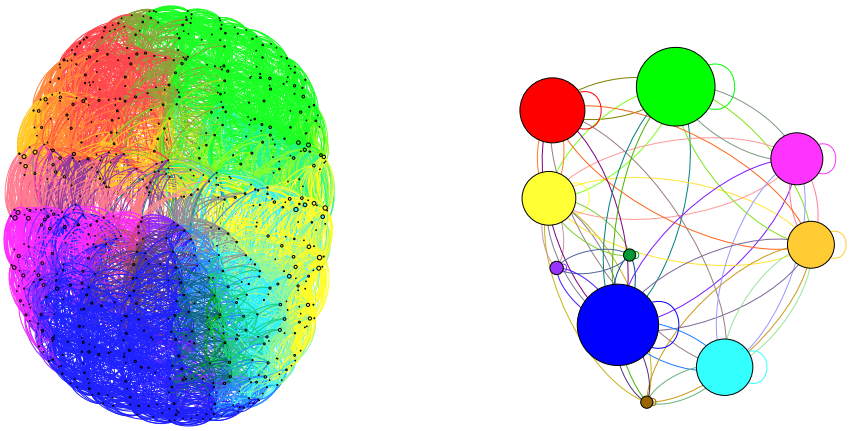


(a) Estado 0 (azul) (b) Estado 1 (verde) (c) Estado 2 (rojo)

Figura 3.11: Representantes de los estados principales de la red de basins del *sistema antiparalelo sin bending* a temperatura 0,3.

La estructura antiparalela se forma correctamente, pero algunos loops se meten dentro y los iones se alejan un poco de su posición central. Esperamos que esto se solucione cuando añadamos el bending al sistema.

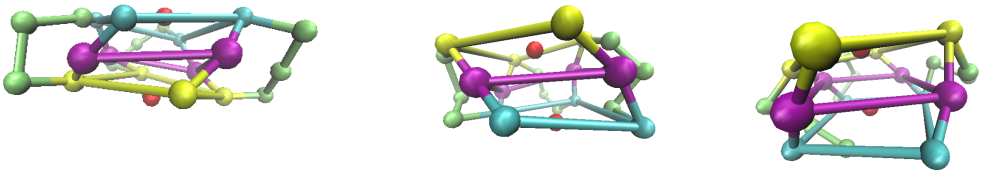
Veamos ahora los grafos del sistema paralelo sin bending:



(a) Red completa (b) Red de basins

Figura 3.12: Grafos para un mallado $10 \times 10 \times 10$ del *sistema paralelo sin bending* a temperatura 0,3.

Los representantes de sus basins principales son:

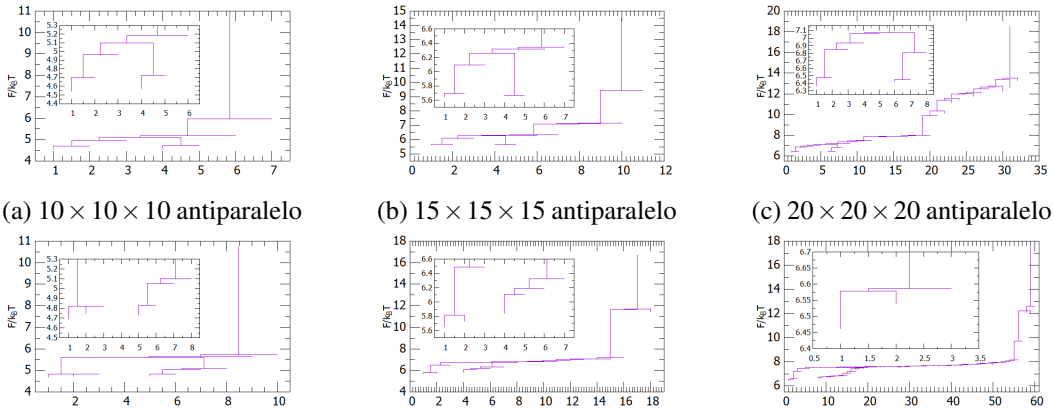


(a) Estado 0 (azul) (b) Estado 1 (verde) (c) Estado 2 (rojo)

Figura 3.13: Representantes de los estados principales de la red de basins del *sistema paralelo sin bending* a temperatura 0,3.

Esta vez la estructura paralela se forma correctamente y podemos ver como, en ella, los loops ocupan más espacio que en la antiparalela y fuerzan una mayor rotación relativa entre los planos.

Hacemos los dendogramas de las dos trayectorias:



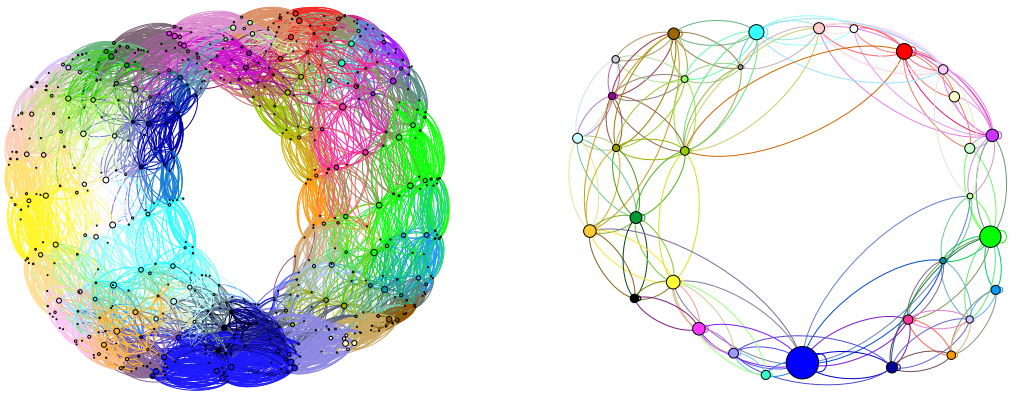
(a) $10 \times 10 \times 10$ antiparalelo (b) $15 \times 15 \times 15$ antiparalelo (c) $20 \times 20 \times 20$ antiparalelo

(d) $10 \times 10 \times 10$ paralelo (e) $15 \times 15 \times 15$ paralelo (f) $20 \times 20 \times 20$ paralelo

Figura 3.14: Dendogramas de la energía libre de las basins para distintos mallados de los sistemas sin bending a temperatura 0,3.

A esta temperatura se observan más basins que a 0,4, pero la diferencia de energía libre entre ellas es menor. Esta diferencia de energía representa una barrera pequeña entre los estados, lo que explica que a temperatura 0,4 parezcan una misma basin.

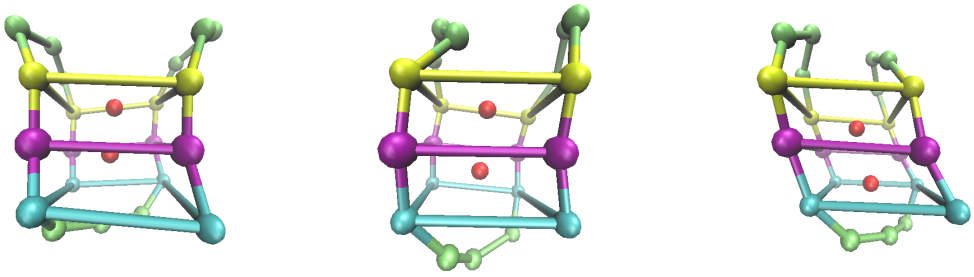
Veamos de nuevo los resultados de añadir el bending de los loops a nuestros sistemas, pero esta vez a esta temperatura más baja. Para el caso antiparalelo los grafos serán:



(a) Red completa (b) Red de basins

Figura 3.15: Grafos para un mallado $10 \times 10 \times 10$ del *sistema antiparalelo con bending* a temperatura 0,3.

Esta vez las redes toma una forma circular debido a la falta de conexión directa entre algunos nodos, pero no llega a ser un caso tan extremo como el del sistema paralelo con bending a temperatura 0,4. Veamos cómo son los representantes de las basins principales:

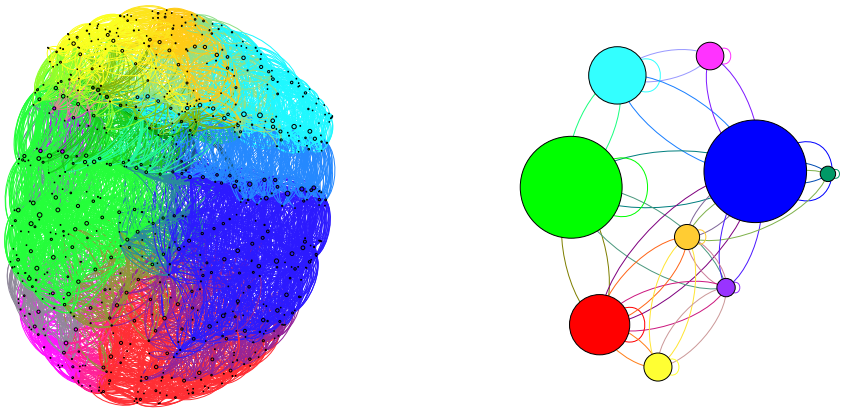


(a) Estado 0 (azul) (b) Estado 1 (verde) (c) Estado 2 (rojo)

Figura 3.16: Representantes de los estados principales de la red de basins del *sistema antiparalelo con bending* a temperatura 0,3.

Podemos ver que los tres estados forman la mejor versión de la estructura deseada vista hasta ahora, donde los loops no se meten en medio, los iones ocupan su debido lugar central y los planos están paralelos y nos se doblan. Además, en apenas se aprecian diferencias entre las estructuras de los distintos atractores, lo que muestra que la formación es muy estable.

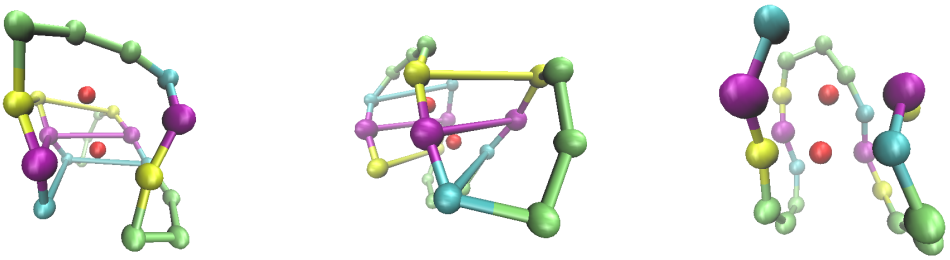
Veamos si se pueden obtener los mismos resultados en el caso paralelo. Empezamos viendo las representaciones gráficas de las redes:



(a) Red completa (b) Red de basins

Figura 3.17: Grafos para un mallado $10\times10\times10$ del *sistema paralelo con bending* a temperatura 0,3.

Aparecen dos estados especialmente destacables y algunas otras basins relevantes por debajo de ellos. Los representantes de los dos atractores principales y de la siguiente mayor basin son estos:

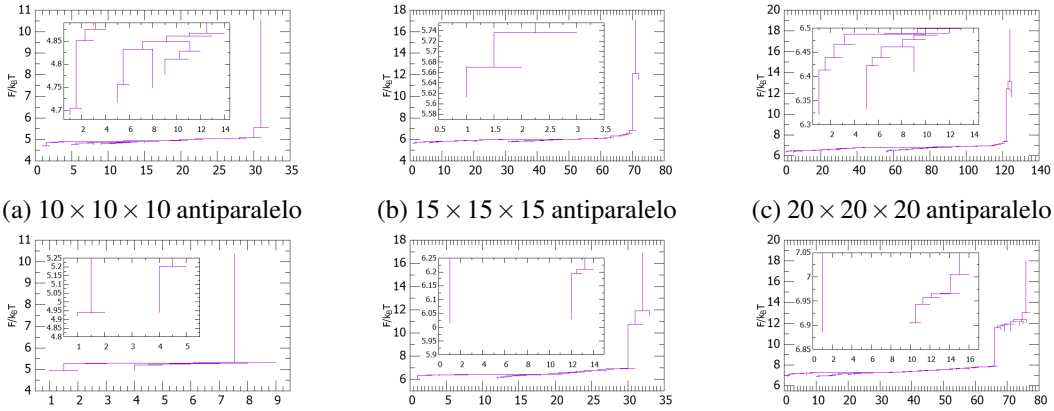


(a) Estado 0 (azul) (b) Estado 1 (verde) (c) Estado 2 (rojo)

Figura 3.18: Representantes de los estados principales de la red de basins del *sistema paralelo con bending* a temperatura 0,3.

Incluso aunque en algunos casos se llegan a alinear varias aristas para formar los enlaces de hidrógeno y la cadena gira siempre en torno a los iones sin perderlos, no se llega a formar nunca la estructura paralela deseada. Esto podría significar que todavía estamos demasiado cerca de la temperatura de melting del sistema paralelo con bending, pero los resultados de [16] situan la temperatura de melting de este sistema por encima de 0,3 y sería lógico esperar que, aunque la formación no fuese muy estable, al menos se pudiese ver la estructura paralela en alguno de los estados principales del sistema a esta temperatura. Esto, junto a algunas de las observaciones previas de los sistemas paralelos nos hace pensar que el modelo no representa correctamente la estructura paralela.

Por último, hagamos los dendogramas de estas dos trayectorias finales:



(a) $10 \times 10 \times 10$ antiparalelo (b) $15 \times 15 \times 15$ antiparalelo (c) $20 \times 20 \times 20$ antiparalelo

(d) $10 \times 10 \times 10$ paralelo (e) $15 \times 15 \times 15$ paralelo (f) $20 \times 20 \times 20$ paralelo

Figura 3.19: Dendogramas de la energía libre de las basins para distintos mallados de los sistemas con bending a temperatura 0,3.

3.4. Conclusiones

Se pueden extraer varias conclusiones de este trabajo. Por un lado, el algoritmo de construcción de la red parece funcionar correctamente y ha demostrado su utilidad para transformar un sistema más complejo en una red de Markov que se pueda estudiar desde el punto de vista de un grafo, sobre todo si se combina el algoritmo con el análisis de componentes principales para reducir la complejidad de los sistemas cuando sea necesario. Sin embargo, el algoritmo de búsqueda de basins ha presentado algunas imprecisiones que vendría bien revisar, como la planteada en el análisis del estado desplegado de la configuración paralela con bending a temperatura 0,4 (apartado 3.3.2). Es probable que estos problemas, que solamente aparecen en determinadas configuraciones, se puedan solucionar con pequeñas modificaciones en el algoritmo.

Respecto a la aplicación de este método de estudio al caso de las dos estructuras de G-quadruplex, se han podido explicar la mayoría de observaciones, ya fuesen debidas al comportamiento de los paisajes de energía libre, como algunos efectos vistos al cambiar el mallado de discretización, o debidas a las propiedades conocidas del G4 y sus simulaciones, como la dependencia de la temperatura de melting con el bending de los loops. Además, este estudio ha sido util para evidenciar dos problemas: uno es que el modelo utilizado no prevé la posibilidad de replegamiento de la estructura cuando esta se ha desplegado del todo, por lo que haría falta darle al modelo la capacidad, por ejemplo, de adquirir iones en el proceso de replegamiento; el otro inconveniente revelado es que el modelo no es capaz de describir correctamente el sistema paralelo, por lo que es probable que haga falta revisar las interacciones, ya sea refinando su parametrización o su forma funcional.

Bibliografía

[1] K. A. DILL, *Polymer principles and protein folding*, Protein Science **8**, 1166–1180 (1999).

[2] D. PRADA-GRACIA, J. GÓMEZ-GARDEÑES, P. ECHENIQUE, F. FALO, *Exploring the Free Energy Landscape: From Dynamics to Networks and Back*, PLoS Computational Biology **5** (6), e1000415 (2009).

[3] R. TAPIA-ROJO, J. J. MAZO, F. FALO, *Thermal versus mechanical unfolding in a model protein*, The Journal of Chemical Physics **151**, 185105 (2019).

[4] R. TAPIA-ROJO, *Tesis doctoral. Modeling Biomolecules: Interactions, Forces and Free Energies*, Universidad de Zaragoza (2016).

[5] F. NOÉ, S. FISCHER, *Transition networks for modeling the kinetics of conformational change in macromolecules*, Current Opinion in Structural Biology **18**, 154–162 (2008).

[6] N. G. VAN KAMPEN, *Stochastic Processes in Physics and Chemistry*, North Holland, (2007).

[7] N. G. VAN KAMPEN, *Remarks on non-Markov processes*, Braz J Phys **28**, 90–96 (1998).

[8] I. T. JOLLIFFE, *Principal Component Analysis*, 2.ª ed., Springer-Velag New York (2002).

[9] A. AMADEI, A. B. M. LINSSEN, H. J. C. BERENDSEN, *Essential Dynamics of Proteins*, PROTEINS: Structure, Function, and Genetics **17**, 412-425 (1993).

[10] R. TAPIA-ROJO, J. J. MAZO, F. FALO, *Thermal and mechanical properties of a DNA model with solvation barrier*, Physical Review E **82**, 031916 (2010).

[11] A. E. BERGUES-PUPO, I. GUTIÉRREZ, J. R. ARIAS-GONZALEZ, F. FALO, A. FIASCONARO, *Mesoscopic model for DNA G-quadruplex unfolding*, Scientific Reports **7**, 11756 (2017).

[12] S. BALASUBRAMANIAN, L. H. HURLEY, S. NEIDLE, *Targeting G-quadruplexes in gene promoters: a novel anticancer strategy?*, Nature Reviews Drug Discovery **10**, 261-275 (2011).

[13] R. HÄNSEL-HERTSCH, M. DI ANTONIO, S. BALASUBRAMANIAN, *DNA G-quadruplexes in the human genome: detection, functions and therapeutic potential*, Nature Reviews Molecular Cell Biology **18**, 279–284 (2017).

[14] D. RHODES, H. J. LIPPS, *G-quadruplexes and their regulatory roles in biology*, Nucleic Acids Research **43** (18), 8627–8637 (2015).

[15] A. E. BERGUES-PUPO, J. R. ARIAS-GONZALEZ, M. C. MORÓN, A. FIASCONARO, F. FALO, *Role of the central cations in the mechanical unfolding of DNA and RNA G-quadruplexes*, Nucleic Acids Research **43** (15), 7638–7647 (2015).

[16] A. E. BERGUES-PUPO, F. FALO, A. FIASCONARO, *Modelling the DNA topology: the effect of the loop bending on Gquadruplex stability*, Journal of Statistical Mechanics, 094004 (2019).

[17] T. CRAGNOLINI, ET AL., *Multifunctional energy landscape for a DNA G-quadruplex: An evolved molecular switch*, The Journal of Chemical Phisycs 147, 152715 (2017).

.1. Anexo 1: programa network_building

Programa de formación de la red:

```
program red

!Lineas a editar si quieres cambiar
!el archivo de la trayectoria:52
!el numero de variables de la trayectoria:20,21,55-59,61,69-73,115-119,
!el numero de bins de cada variable:55-59
!el numero de pasos de la trayectoria:65(numero de simulaciones),66(pasos por
    simulacion)
implicit none

!VARIABLES DE LA PRIMERA PARTE: LECTURA Y TRADUCCION DE LA TRAYECTORIA

integer(KIND=8)::a,b,num_total,num_vars !a se usa para calcular el bin que se
    asocia a cada valor; b almacena el indice asociado al estado por el que estemos
     pasando en ese paso de la trayectoria; num_total es el numero de pasos de la
    trayectoria; num_vars es el numero de variables

character*20::f1 !f1 se usa, junto con variable para indicar la etiqueta de los
    ficheros de cada variable de la trayectoria

integer,dimension(:),allocatable::deantes !deantes almacena los valores de las
    variables anteriores de un estado para poder anadirle la nueva variable
    definiendo nuevos estados
integer(KIND=8)::llevamos,variable !variable indica la variable con la que estamos
    trabajando; llevamos lleva la cuenta de los estados ocupados hemos definido ya
    en la red

real::var,var2 !var se usa para leer los valores de las variables y var2 corrige
    estos valores restandole el valor minimo de la variable
real,dimension(3)::lim_inf,lim_sup,lim_test,delta_x !(hecho para un maximo de 5
    variables pero se podria aumentar de ser necesario);bins es el array que indica
     el numero de bins (celdas) de cada variable; lim_inf y lim_sup almacenan
    respectivamente los minimos y los maximos de cada variable; lim_test se utiliza
     para calcular lim_inf y lim_sup; delta_x indica la anchura de cada bin
integer,dimension(3)::bins

logical,dimension(:,:),allocatable::ocupado !array que indica que estados de la red
     aparecen en la trayectoria
integer,dimension(:,:),allocatable::indice !almacena los indices que asociamos a
    cada estado (la traduccion)

!VARIABLES DE LA SEGUNDA PARTE: CONSTRUCION DE LA RED

TYPE array_pointer !crea un vector de punteros para poder hacer luego una matriz de
     ellos
INTEGER,DIMENSION(:),POINTER::p1
END TYPE array_pointer
!usando el tipo creado antes se hace una matriz como vector de vectores de punteros
TYPE(array_pointer),DIMENSION(:),POINTER::C,WK_out !C es una matriz que relaciona
    el indice de cada nodo con un array de punteros con los indices de todas sus
    salidas;WK_out es una matriz que relaciona el indice de cada nodo con un array
    de punteros con los pesos de todos los enlaces que van de ese nodo a cada una
    de sus salidas (el numero de veces que la trayectoria pasa por cada enlace)

integer,DIMENSION(:),POINTER::aux_puntero,aux_puntero2 !arrays auxiliares
    utilizados para ir anadiendo los nodos de C y los pesos de WK_out,
    respectivamente, uno a uno
integer::g,kk,x,x2 !x y x2 almacenan, respectivamente, el estado de la trayectoria
    en el que estamos y a cual iremos en el siguiente paso durante la creacion de
    la red; g se usa para enumerar las coordenadas de los arrays de punteros p1
```

```
integer(KIND=8)::m,mm,minc !m, mm y minc se utilizan para ir mostrando el
    porcentaje de avance en el analisis de cada simulacion durante la creacion de
    la red

integer,dimension(:),allocatable::W,K_out !W es un array de los pesos de cada nodo
    (numero de veces que se pasa por el nodo en la trayectoria); K_out es un arrai
    que dice el numero de salidas distintas que tiene cada nodo
integer,dimension(:),allocatable::aux !almacena de forma alterna los indices de las
     salidas de cada nodo i (impares) y los pesos de los enlaces que van de i a la
    ultima salida mencionada (pares)
integer(KIND=8)::num_bins_ocupados,simus,num_simus,num_each,contador_steps!
    contador_steps lleva la cuenta de los pasos de la trayectoria; num_each es el
    numero de pasos de cada simulacion; num_simus indica el numero de simulaciones
    distintas que contiene nuestra trayectoria; simus indica con que simulacion
    estamos trabajando; num_bins_ocupados indica el numero de estados distintos que
     se ocupan en la trayectoria, es decir, el numero de estados que nos importan

logical::switch

!VARIABLES QUE APARECEN EN TODO EL PROGRAMA

integer::i,j
integer(KIND=8)::contador_bins !contador_bins lleva la cuenta del numero total de
    estados ocupados

integer,dimension(:),allocatable::tray !tray almacena los indices de cada estado en
     el orden en el que se pasa por ellos en la trayectoria


OPEN(99,FILE="tray_g4.txt",status="OLD",action="READ")

!numero de bins de cada variable
bins(1)=15
bins(2)=15
bins(3)=15
!bins(4)=20
!bins(5)=20

num_vars=3 !Numero de variables

!num_total=12000000
!Num pasos de la tray.
num_simus=1     !! num de tray a analizar
num_each=900000 !! num de pasos de cada tray
num_total=num_simus*num_each

OPEN(100,FILE="tray_x1.txt",status="REPLACE",ACTION="WRITE")
OPEN(101,FILE="tray_x2.txt",status="REPLACE",ACTION="WRITE")
OPEN(102,FILE="tray_x3.txt",status="REPLACE",ACTION="WRITE")
!OPEN(103,FILE="tray_x4.txt",status="REPLACE",ACTION="WRITE")
!OPEN(104,FILE="tray_x5.txt",status="REPLACE",ACTION="WRITE")
!escribimos los ficheros con las trayectorias para cada variable distinta

OPEN(999,File="max.txt",status="REPLACE",action="WRITE")

!buscamos los maximos y minimos de un fichero
READ(99,*) lim_test(:)
lim_inf=lim_test
lim_sup=lim_test

do j=1,num_vars
    WRITE(99+j,*) lim_test(j)
end do
```

```
86
do i=2,num_total
88     read(99,*)  lim_test(:)
       do j=1,num_vars
90         WRITE(99+j,*) lim_test(j)

92         if(lim_test(j)>lim_sup(j)) then
           lim_sup(j)=lim_test(j)
94         endif

96         if(lim_test(j)<lim_inf(j)) then
           lim_inf(j)=lim_test(j)
98         endif
       end do
100
end do
102
do j=1,num_vars
104    WRITE(999,*) lim_inf(j),lim_sup(j)
       CLOSE(99+j)
106 end do
CLOSE(999)
108 close(99)

110 print*,lim_inf,lim_sup

112 delta_x(:)=(lim_sup(:)-lim_inf(:))/(bins(:)*1.0d0) !Calculamos la anchura de cada
       bin

114 !abrimos los ficheros con las trayectorias para cada variable que hemos escrito
       antes
OPEN(100,FILE="tray_x1.txt",status="OLD",action="READ")
116 OPEN(101,FILE="tray_x2.txt",status="OLD",action="READ")
OPEN(102,FILE="tray_x3.txt",status="OLD",action="READ")
118 !OPEN(103,FILE="tray_x4.txt",status="OLD",action="READ")
!OPEN(104,FILE="tray_x5.txt",status="OLD",action="READ")
120
!Primero construyo el trad_aux y el tray_int_bin.oup
122 !      primero la trayectoria de la particula

124 llevamos=1
ALLOCATE(tray(num_total),ocupado(llevamos,bins(1)),indice(llevamos,bins(1)))
126 !Primero definiremos los estados con solo una variable, la primera, y veremos
       cuales estan ocupados

128 ocupado=.false. !Inicializamos los estados como no ocupados
DO i=1,num_total !Hacemos un barrido por todos los pasos de la trayectoria
130
   READ(100,*) var
132
    var2=0.0d0
134 var2=var-lim_inf(1)

136 a=CEILING(real(var2/delta_x(1)))!asignamos cada valor a una celda de la division


138
    IF (a<0) THEN !nos aseguramos de que el valor de a este en la lista de bins
140    print*,'problema'
       stop
142 END IF
    IF (a==0) a=1
144 IF (a==(bins(1)+1)) a=bins(1)
```

```
146     ocupado(llevamos,a)=.true.!marcamos todos los estados ocupados, es decir,
        aquellos por los que pasa la trayectoria
END DO
148
OPEN(21,FILE="trad_aux.aux",status="REPLACE",ACTION="WRITE") !Este archivo nos
    servira para traducir los indices a estados (dice que combinacion de bins de
    las variables se asocia con cada indice)
150
contador_bins=0
152 DO i=1,llevamos !miramos todos los estados definidos antes de la variable que
    estamos anadiendo (de momento es solo la primera)
    DO j=1,bins(1) !miramos todos los bins de esa variable
154     IF (ocupado(i,j)==.true.) THEN
            contador_bins=contador_bins+1
156         WRITE(21,'(2(1x,I4))') contador_bins,j!enumeramos los estados ocupados de
        menor a mayor (1x coloca un espacio))
            indice(i,j)=contador_bins !almacenamos el indice asignado a ese estado (
        los indices ordenan los estados por orden de aparicion en la trayectoria)
158       END IF
    END DO
160 END DO
DEALLOCATE(ocupado)
162 print*,'de la particula',llevamos,contador_bins,'bins'

164 REWIND(100)!volvemos a empezar la lectura del fichero

166 DO i=1,num_total
    READ(100,*) var
168
    var2=0.0d0
170 var2=var-lim_inf(1)

172 a=CEILING(real(var2/delta_x(1)))


174
    IF (a<0) THEN
176     print*,'problema'
        stop
178 END IF
    IF (a==0) a=1
180 IF (a==(bins(1)+1)) a=bins(1)

182 tray(i)=indice(llevamos,a)!anotamos la trayectoria con los estados numerados
END DO
184 CLOSE(21)
CLOSE(100)
186 DEALLOCATE(indice)
llevamos=contador_bins !guardamos el numero de estados ocupados de la variable para
     ver todas las combinaciones en la siguiente iteracion
188
!ahora las bases (las demas variables)
190 DO variable=2,num_vars !cada iteracion anadira una nueva variable a la definicion
    de los estados

192 CALL SYSTEM ('mv trad_aux.aux trad_aux_old.aux')!le cambiamos el nombre al
    fichero

194 OPEN(61,FILE="trad_aux_old.aux",status="OLD",ACTION="READ")

196 ALLOCATE(ocupado(llevamos,bins(variable)),indice(llevamos,bins(variable))) !
    redefinimos las dimensiones de ocupado e indice para ver todas las
    combinaciones de los estados definidos hasta ahora con la nueva variable
    anadida ahora
```

```
    ocupado=.false.
198
    DO i=1,num_total
200    READ(99+variable,*) var !leemos los valores del archivo correspondiente a la
     variable nueva

202    b=tray(i)

204    var2=0.0d0
       var2=var-lim_inf(variable)
206
       a=CEILING(real(var2/delta_x(variable)))
208
       IF (a==0) a=1
210    IF (a==(bins(variable)+1)) a=bins(variable)

212    !repetimos la asignacion de estados ocupados, pero esta vez para una matriz
       ocupado(b,a)=.true.
214    !marcamos los estados ocupados
       !b es el indice asociado antes al estado por el que pasamos ahora en la
     trayectoria
216    !a es el indice del bin que anadimos de la nueva variable
    END DO
218

220 OPEN(21,FILE="trad_aux.aux",status="REPLACE",ACTION="WRITE")
    ALLOCATE(deantes(variable-1)) !usaremos deantes para almacenar los valores de
     las variables anteriores y asi poder sobreescribir el estado anadiendo la nueva
      variable
222
    contador_bins=0
224 WRITE(f1,'(I2)') variable-1
    f1="(I,I4,"//TRIM(ADJUSTL(f1))//"I3)"!ADJUSTL mueve todos los espacios del
     string f1 al final y TRIM borra los espacios del final del string
226 DO i=1,llevamos !miramos todos los estados definidos hasta ahora
       READ(61,*) a,deantes(:) !leemos la traduccion realizada antes para saber que
     valores de las variables anteriores se asocian a cada estado
228    DO j=1,bins(variable) !miramos todos los bins de la nueva variable para
     comprobar cuales estan ocupados
          IF (ocupado(i,j)==.true.) THEN
230          contador_bins=contador_bins+1
             WRITE(21,f1) contador_bins,deantes(:),j !escribimos los nuevos estados
     y su traduccion a indices
232          indice(i,j)=contador_bins !anotamos el indice asignado a cada nuevo
     estado
          END IF
234    END DO
    END DO
236 DEALLOCATE(ocupado)

238 print*,'de la variable',variable,contador_bins,'bins'
    REWIND(99+variable) !reiniciamos la lectura del archivo
240 print*,llevamos,bins(variable)

242 print*,'entro en bucle',variable-1
    DO i=1,num_total
244
       READ(99+variable,*) var
246
       b=tray(i)
248
       var2=0.0d0
250    var2=var-lim_inf(variable)
```

```
252         a=CEILING(real(var2/delta_x(variable)))

254         IF (a==0) a=1
            IF (a==(bins(variable)+1)) a=bins(variable)
256
            tray(i)=indice(b,a) !introducimos los nuevos indices en la trayectoria por
        orden de aparicion
258       END DO
          print*,'salgo del bucle',variable-1,'con',contador_bins,'estados ocupados'
260       CLOSE(21)
          CLOSE(61)
262       CLOSE(99+variable)

264       DEALLOCATE(indice,deantes)
          llevamos=contador_bins !numero de estados definidos hasta ahora
266
          CALL SYSTEM ('rm trad_aux_old.aux') !eliminamos el archivo viejo
268
    END DO
270
    print*,'he salido de todo'
272


274 !! Escribo la trayectoria:
    print*,' '
276 PRINT*,'CONSTRUYENDO LA RED:'
    print*,' '
278
    OPEN(20,FILE="tray_int_bin.oup",status="REPLACE",action="WRITE")
280 !escribimos la trayectoria entera con los indices que han quedado definidos
        finalmente y cuya traduccion se encuentra en trad_aux.aux
    DO i=1,num_total
282     write(20,*) tray(i)
    END DO
284 CLOSE(20)

286 num_bins_ocupados=contador_bins !numero total de estados ocupados
    print*,'num_bins_ocupados=',contador_bins
288
    !Hacemos que el tamano de todos los arrays de interes sea el numero de bins
        ocupados; a los punteros auxiliares se les da un tamano fijo para ahorrar
        memoria, pero este se puede cambiar si fuese necesario
290 ALLOCATE(C(num_bins_ocupados),K_out(num_bins_ocupados))
    ALLOCATE(WK_out(num_bins_ocupados),W(num_bins_ocupados))
292 ALLOCATE(aux_puntero(50000),aux_puntero2(50000))

294 DO i=1,num_bins_ocupados
        ALLOCATE(C(i)%p1(1),WK_out(i)%p1(1)) !le damos dimension a los punteros de los
         arrays de punteros y los inicializamos
296     C(i)%p1(:)=0
        WK_out(i)%p1(:)=0
298 END DO

300 W=0
    K_out=0
302 kk=0
    x=0
304 x2=0
    aux_puntero(:)=0
306 aux_puntero2(:)=0

308 OPEN (13,FILE="net_oldstyle.oup",STATUS="REPLACE",ACTION="WRITE")
```

```
contador_steps=0
DO simus=1,num_simus !iteramos cada simulacion de la trayectoria

   print*,'en simu',simus

   minc=(num_each-2)/10
   m=minc
   mm=10

   !guardamos los pasos 1 y 2 de la trayectoria para tener el nodo en el que
    estamos y aquel al que iremos desde ahi
   contador_steps=contador_steps+1
   x=tray(contador_steps)
   contador_steps=contador_steps+1
   x2=tray(contador_steps)

   DO i=3,num_each !vemos todos los pasos de la simulacion del tercero en adelante

      IF(i==m) THEN
         PRINT*,'...',mm,'% !mostramos por pantalla cada vez que se analiza un 10%
     de una simulacion
         mm=mm+10
         m=m+minc
      END IF

      W(x)=W(x)+1 !sumamos 1 al peso del estado x porque pasamos por el una vez

      switch=.true.
      DO g=1,K_out(x) !miramos todas las salidas previamente registradas de x
         IF(C(x)%p1(g)==x2) THEN !Comprobamos si x2 coincide con algun nodo al que
    ya hemos ido desde x
            WK_out(x)%p1(g)=WK_out(x)%p1(g)+1 !sumamos uno al peso del enlace que
    va de x a x2
            switch=.false.
            EXIT
         END IF
      END DO

      IF (switch==.true.) THEN !si no se ha ido antes de x a x2 hay que anadir el
    nodo x2 a la lista de salidas de x
         IF (K_out(x)>0) THEN !si ya teniamos salidas de x anadimos x2 como una
    nueva
            g=K_out(x)
            IF (g>50000) THEN !nos avisa si superamos la capacidad de los arrays
    auxiliares
               print*,'tenemos un problema'
               stop
            END IF
            aux_puntero(1:g)=C(x)%p1(:) !almacenamos la lista de salidas de x
            aux_puntero2(1:g)=WK_out(x)%p1(:) !almacenamos los pesos de esos
    enlaces
            DEALLOCATE(C(x)%p1,WK_out(x)%p1)
            ALLOCATE(C(x)%p1(g+1),WK_out(x)%p1(g+1)) !redefinimos los arrays para
    meter un dato nuevo
            C(x)%p1(1:g)=aux_puntero(:) !copiamos los datos anteriores y anadimos
    el nodo nuevo
            WK_out(x)%p1(1:g)=aux_puntero2(:) !copiamos los datos anteriores y
    anadimos un peso de 1 para el enlace nuevo
            C(x)%p1(g+1)=x2
            WK_out(x)%p1(g+1)=1
            K_out(x)=K_out(x)+1 !sumamos 1 al numero de salidas de x
         ELSE !Si hasta ahora no habiamos ido a ningun sitio desde x, anadimos x2
    como el primer nodo al que vamos desde x
```

```
360             WK_out(x)%p1(1)=1 !le damos un peso de 1 al nuevo enlace
                C(x)%p1(1)=x2 !iniciamos la lista de salidas de x con x2
362             K_out(x)=1 !de momento x solo tiene 1 salida
            END IF
364       END IF

366       x=x2  !x pasa a ser x2 (el nodo en el que estamos ahora)

368       contador_steps=contador_steps+1
          x2=tray(contador_steps) !x2 pasa a ser el siguiente nodo de la trayectoria (
    el nodo al que vamos desde el que estamos ahora)
370
    END DO
372
!Ahora se realiza el paso del penultimo nodo al ultimo por separado (para evitar el
     comando x2=tray(contador_steps) cuando contador_steps se sale del tamano de
    tray)
374
    W(x)=W(x)+1
376
    switch=.true.
378
    DO g=1,K_out(x)
380     IF(C(x)%p1(g)==x2) THEN
            WK_out(x)%p1(g)=WK_out(x)%p1(g)+1
382         switch=.false.
            EXIT
384     END IF
    END DO
386
    IF (switch==.true.) THEN
388     IF (K_out(x)>0) THEN
            g=K_out(x)
390         IF (g>50000) THEN
                print*,'tenemos un problema'
392             stop
            END IF
394         aux_puntero(1:g)=C(x)%p1(:)
            aux_puntero2(1:g)=WK_out(x)%p1(:)
396         DEALLOCATE(C(x)%p1,WK_out(x)%p1)
            ALLOCATE(C(x)%p1(g+1),WK_out(x)%p1(g+1))
398         C(x)%p1(1:g)=aux_puntero(:)
            WK_out(x)%p1(1:g)=aux_puntero2(:)
400         C(x)%p1(g+1)=x2
            WK_out(x)%p1(g+1)=1
402         K_out(x)=K_out(x)+1
        ELSE
404         WK_out(x)%p1(1)=1
            C(x)%p1(1)=x2
406         K_out(x)=1
        END IF
408 END IF

410 x=x2

412 W(x)=W(x)+1 !Anadir 1 al peso (ocupacion) del nodo final

414 mm=100
    PRINT*,'...',mm,'%
416
END DO
418
print*,contador_steps,num_total !se muestran ambos numeros para ver que son iguales
```

```
420 print*,' '
    print*,'ESCRIBIENDO LA RED'
422 print*,' '

424 !Sacar a fichero los datos de la red al estilo penique
    WRITE(13,*) num_bins_ocupados,maxval(K_out(:)) ,sum(K_out(:)) !en la primera linea
        del fichero damos los datos de numero de nodos total (numero de bins o de
        estados ocupados), maximo numero de salidas de un nodo (numero de nodos
        distintos a los que va) ynumero total de enlaces distintos
426
    DO i=1,num_bins_ocupados !miramos todos los nodos uno a uno
428
       ALLOCATE(aux(K_out(i)*2)) !le damos un tamano igual al doble del numero de
        salidas del nodo i
430
       aux=0
432    kk=0

434    switch=.false.
       DO g=1,K_out(i)
436       IF (C(i)%p1(g)==i) THEN !Comprobamos si hay autoloop (el estado va a si mismo
        )
             switch=.true.
438          exit
          END IF
440    END DO

442    IF (switch==.true.) THEN !Ponemos como primer nodo el autoloop, si lo hay
          kk=kk+2
444       aux(kk-1)=i
          aux(kk)=WK_out(i)%p1(g)
446
          DO j=1,K_out(i)
448          IF (j/=g) THEN !Escribimos el resto de salidas en el orden que las ha
        encontrado
                kk=kk+2
450             aux(kk-1)=C(i)%p1(j) !En las coordenadas impares de aux se escriben los
         indices de las salidas del nodo i
                aux(kk)=WK_out(i)%p1(j) !En las coordenadas pares de aux se escriben
        los pesos de los enlaces (cuantas veces se va del nodo i a un nodo concreto)
452          END IF
          END DO
454    ELSE
          DO j=1,K_out(i)
456          kk=kk+2
             aux(kk-1)=C(i)%p1(j)
458          aux(kk)=WK_out(i)%p1(j)
          END DO
460    END IF

462    WRITE (13,*) i,K_out(i),W(i),aux !escribimos en cada linea el indice del nodo en
         el que estamos, el numero de salidas distintas que tiene, el peso del nodo (
        numero de veces que pasamos por el) y el array aux, que alterna los indices de
        las salidas con los pesos de los enlaces que van a ellas desde i

464    DEALLOCATE(aux)

466 END DO

468 !Sacar a fichero los datos para mi:
    PRINT*,'Num frames:', num_total
470 PRINT*,'Suma de pesos:',sum(W(:)) !suma de pesos de los nodos
    g=0
```

```
DO i=1,num_bins_ocupados
   DO j=1,K_out(i)
      g=g+WK_out(i)%p1(j)
   END DO
END DO

PRINT*,'Suma pesos de conectividades out', g !suma de pesos de los enlaces
PRINT*,'Numero maximo de vecinos',maxval(K_out(:))


END program red
```

network_building.f90

.2. Anexo 2: programa saco_basins_red_original

Programa de búsqueda de basins:

```
program network

implicit none

integer:: N,kmax,ktot,N_sets
integer:: i,j,h,sumk,ii,jj,g,gg,contador
integer:: vecinor,peso,final,suma
integer:: nada,hacia,desde
integer:: dim,cutoff

!!! Aqui van los vectores de conectividad y las matrices donde guardaremos los
    vecinos y el peso correspondiente al link
integer,dimension(:),allocatable:: Kout,Cout,Cout_start,Cout_W,Wout
integer,dimension(:),allocatable:: W,SL,auxiliar

!!! Cosicas:
character(10)::sufijo

!!! Distribuciones de probabilidad y mis cuentas:
double precision,dimension(:),allocatable:: Pe,Psalto
integer,dimension(:),allocatable::vect_aux1
logical::inter,inter2
logical,dimension(:),allocatable::label,label_glob_min,filtro
integer,dimension(:),allocatable::comunidades,sets,start_sets,representante
double precision,dimension(:),allocatable::P_set
double precision,dimension(:,:),allocatable::K_sets

!!! Definicion de T_tau
double precision,dimension(:),allocatable:: T_tau
integer,dimension(:),allocatable:: T_start, T_ind

double precision::estacionario,peso_salida
integer::posible,candidato

double precision::marca
integer::marca2

!Corregir con el numero coordenadas:
dim=2
cutoff=2*dim !Para un sistema dinamico "normal", el cutoff es 2 veces la dimension
    discretizada.

OPEN (20,FILE="net_oldstyle.oup",STATUS="OLD",ACTION="READ")
READ (20,*) N,kmax,ktot
```

```
!!!! alocato (no olvidar aplicar la funcion sizeof() a las cositas para hacerme una idea):
allocate(Kout(N),Cout(ktot),Cout_start(N),Cout_W(ktot),Wout(N))
allocate(W(N),SL(N))
allocate(Pe(N),Psalto(ktot))
allocate(T_tau(ktot+N),T_start(N+1),T_ind(ktot+N))
allocate(label(N),comunidades(N),label_glob_min(N))


Cout_start=0
Kout=0
W=0
Wout=0
Cout=0
Cout_W=0
label=.false.
label_glob_min=.false.
comunidades=0
sumk=0
suma=0
!Leo la red
!El fichero es: nodo, conect out, peso total, vecinos out, peso de cada transicion out
!(en este fichero esta contado el mismo nodo como vecino ajeno, es decir
!el auto loop aparece en la lista de vecinos con su correspondiente peso)
DO i=1,N

   READ (20,*) nada,Kout(i),W(i)
   suma=suma+W(i)

   Cout_start(i)=sumk !Cout_start(i) indica donde de Cout y Cout_W empiezan las salidas del nodo i. Si Cout_start(i)=j, la primera salida del nodo i es Cout(j+1) y el peso de su enlace es Cout_W(j+1)
   final=Kout(i) !numero de salidas del nodo i (contando autoloops si lo hay)

   ALLOCATE(auxiliar(2*final))
   BACKSPACE (20)
   READ (20,*) nada,Kout(i),nada,auxiliar(:)

   DO j=1,final !tras leer el fichero net_oldstyle.oup guardamos todos los datos en las variables correspondientes y distinguimos los autoloops

      vecinor=auxiliar(2*j-1)
      peso=auxiliar(2*j)

      if (vecinor/=i) then !si NO es un autoloop
         sumk=sumk+1
         Cout(sumk)=vecinor !lista de los indices de todas las salidas de cada nodo
         Cout_W(sumk)=peso !lista de los pesos de todos los enlaces
         Wout(i)=Wout(i)+peso !peso total de todas las salidas de cada nodo i
      else !si es un autoloop no lo contamos en las listas de Cout y Cout_W sino que lo almacenamos aparte
         SL(i)=peso
         Kout(i)=Kout(i)-1
         ktot=ktot-1
      end if
   END DO

   DEALLOCATE(auxiliar)
END DO

IF(sumk==ktot) print*, "Correcto... uno"
```

```
100
    print*,"Suma:",suma
102 print*,suma
    print*,'-------------'
104 print*,sum(W(:),DIM=1)
    print*,sum(SL(:),DIM=1)+sum(Wout(:),DIM=1)
106
    !Construyo Pe y Psalto normalizando Cout y Cout_W
108 Pe=0.0d0
    Psalto=0.0d0
110 DO i=1,N
        nada=Wout(i)+SL(i)
112     Pe(i)=(dble(W(i))/dble(suma)) !la probabilidad de estar en un nodo concreto
        DO j=1,Kout(i)
114         g=Cout_start(i)+j
            Psalto(g)=dble(Cout_W(g))/dble(nada) !la probabilidad condicionada de pasar
     por un enlace estando en el nodo del que sale
116     END DO
    END DO
118
    OPEN (900,FILE="Pe.oup",STATUS="REPLACE",ACTION="WRITE")
120 DO i=1,N
        WRITE(900,'(I,E)') i,Pe(i)
122 END DO

124 !Construyo la matriz T_tau con los indices antiguos
    g=0
126 T_start(1)=0 !T_start no indica la posicion del primer enlace de cada nodo sino la
        del ultimo del nodo anterior, igual que Cout_start
    DO i=1,N
128     T_start(i)=g
        DO j=1,Kout(i)
130         g=g+1
            gg=Cout_start(i)+j
132         T_ind(g)=Cout(gg) !indices de las salidas de cada nodo
            T_tau(g)=Psalto(gg) !probabilidades (pesos normalizados) de los enlaces que
     salen de cada nodo
134     END DO
        IF (SL(i)>0.0d0) THEN !si hay autoloop lo anadimos al final
136         g=g+1
            T_ind(g)=i
138         T_tau(g)=dble(SL(i))/dble(Wout(i)+SL(i))
        END IF
140 END DO
    T_start(N+1)=g
142 print*,g,ktot !escribimos ambos datos para ver que son iguales
    print*, 'sali del bucle'
144


146 label_glob_min=.false.
    OPEN (987,FILE="minimos.oup",STATUS="REPLACE",ACTION="WRITE")
148
    DO i=1,N
150
        h=0
152     g=0
        allocate(filtro(T_start(i+1)-T_start(i))) !definimos el filtro con longitud
     igual al numero de salidas del nodo i (incluyendo autoloop)
154     filtro=.true.
        IF (T_ind(T_start(i+1))==i) THEN !si hay autoloop en el nodo i lo ignoraremos
156         filtro(T_start(i+1)-T_start(i))=.false.
        END IF
158
```

```
   DO j=T_start(i)+1,T_start(i+1)
160
      !Selecciono candidato:
162      nada=T_start(i)+MAXLOC(T_tau((T_start(i)+1):T_start(i+1)),DIM=1,MASK=filtro)
     !MASK hace que no se tengan en cuenta los elementos false al buscar el maximo
         g=T_ind(nada) !nada es la posicion en T_tau del enlace mas pesado del nodo i,
     luego g sera el indice del nodo al que va ese enlace
164
         DO ii=1,Kout(i)
166
            gg=Cout_start(i)+ii
168         IF (Cout(gg)==g) THEN
               peso_salida=Cout_W(gg) !guardamos el peso entero sin normalizar del
     enlace
170            EXIT
            END IF
172      END DO

174      estacionario=0.0d0
         DO ii=1,Kout(i)
176         posible=Cout_start(i)+ii
            IF (Cout_W(posible)==peso_salida) THEN  !si hay varios enlaces de peso
     maximo que no hayamos descartado ya, elegimos como el que vaya al nodo mas
     pesado, ese nodo sera el candidato
178            IF ((Pe(Cout(posible))>estacionario).and.(filtro(ii)==.true.)) THEN
                  estacionario=Pe(Cout(posible))
180               candidato=Cout(posible)
               END IF
182         END IF
         END DO
184      DO ii=T_start(i)+1,T_start(i+1)
            IF (T_ind(ii)==candidato) THEN
186            nada=ii
               g=candidato
188            exit
            END IF
190      END DO

192      !Ya seleccionado
         IF (Pe(g)>Pe(i)) THEN !si el candidato es mas pesado que el nodo actual
     salimos del bucle de j (estamos suponiendo que se cumple el balance detallado)
194         exit
         ELSE
196         h=h+1
            filtro(nada-T_start(i))=.false.!lo descarta para la siguiente iteracion
198      END IF

200      IF (h==(cutoff)) THEN   !Cuando h alcanza el cutoff=2D+1 dejamos de mirar
     opciones y pasamos a considerar al nodo i como un minimo representante de su
     propio basin
            print*,'detecto el',i
202         WRITE(987,'(I,E)') i,Pe(i)
            label_glob_min(i)=.true.
204         !exit
         END IF
206
         IF (COUNT(filtro)==0) THEN !si no quedan salidas que mirar porque todas estan
      descartadas pasamos al siguiente nodo de la red
208         exit
         END IF
210   END DO

212 deallocate(filtro)
```

```
214 END DO

216
    allocate(vect_aux1(N))
218 vect_aux1=0
    label(:)=label_glob_min(:)
220
    DO i=1,N
222     IF (label_glob_min(i)==.true.) THEN !en la lista que identifica a cada nodo con
        su comunidad anotamos que, obviamente, cada representante pertenece a su propia
         comunidad (basin)
          comunidades(i)=i
224     ENDIF
    END DO
226
    OPEN (601,FILE="lista_nodos_basins.dat",STATUS="REPLACE",ACTION="WRITE")
228
    DO i=1,N
230
        desde=i
232     h=0
        DO WHILE (label(desde).eq..false.) !si i no es un minimo ni un nodo que ya haya
        sido "desde" en una iteracion anterior
234
           h=h+1
236        vect_aux1(h)=desde
           label(desde)=.true.
238
           allocate(filtro(T_start(desde+1)-T_start(desde)))
240        filtro=.true.
           IF (T_ind(T_start(desde+1))==desde) THEN !si hay autoloop lo descartamos
242           filtro(T_start(desde+1)-T_start(desde))=.false.
           END IF
244
           !!!Selecciono candidato:
246        nada=T_start(desde)+MAXLOC(T_tau((T_start(desde)+1):T_start(desde+1)),DIM=1,
        MASK=filtro)
           g=T_ind(nada) !buscamos el enlace mas pesado igual que en el bucle anterior
248
           DO ii=1,Kout(desde)
250
              gg=Cout_start(desde)+ii
252           IF (Cout(gg)==g) THEN
                 peso_salida=Cout_W(gg)
254              EXIT
              END IF
256        END DO

258        estacionario=0.0d0
           DO ii=1,Kout(desde) !igual que antes identificamos el nodo mas pesado al que
        se va con un enlace de peso maximo que salga de desde
260           posible=Cout_start(desde)+ii
              IF (Cout_W(posible)==peso_salida) THEN
262              IF ((Pe(Cout(posible))>estacionario).and.(filtro(ii)==.true.)) THEN
                    estacionario=Pe(Cout(posible))
264                 candidato=Cout(posible)
                 END IF
266           END IF
           END DO
268        DO ii=T_start(desde)+1,T_start(desde+1)
              IF (T_ind(ii)==candidato) THEN
270              nada=ii
```

```
            g=candidato
272             exit
            END IF
274       END DO

276       !Ya seleccionado
          hacia=g
278       inter=.false.
          DO WHILE (inter.eq..false.)
280
              inter2=.false.
282           IF (Pe(desde)<Pe(hacia)) THEN !si el candidato es mas pesado que el nodo
    actual esta sera la ultima iteracion
                  inter=.true.
284           END IF
              !otros casos:
286           IF (Wout(desde)==1) THEN !si el nodo actual solo tiene un enlace de salida
     esta sera la ultima iteracion
                  inter=.true.
288           END IF

290           IF (inter.eq..false.)THEN

292             filtro(nada-T_start(desde))=.false. !descartamos el candidato
                inter2=.false.
294             IF (COUNT(filtro)==0) THEN !si todos los enlaces estan descartados esta
    sera la ultima iteracion debido al bucle en la linea 335
                  inter2=.true.
296               filtro=.true.
                  IF (T_ind(T_start(desde+1))==desde) THEN !si hay autoloop lo
    descartamos
298                 filtro(T_start(desde+1)-T_start(desde))=.false.
                  END IF
300             END IF

302             !Selecciono candidato:
                nada=T_start(desde)+MAXLOC(T_tau((T_start(desde)+1):T_start(desde+1)),
    DIM=1,MASK=filtro)
304             g=T_ind(nada)

306             DO ii=1,Kout(desde)

308               gg=Cout_start(desde)+ii
                  IF (Cout(gg)==g) THEN
310                 peso_salida=Cout_W(gg)
                    EXIT
312               END IF
                END DO
314
                estacionario=0.0d0
316             DO ii=1,Kout(desde)
                  posible=Cout_start(desde)+ii
318               IF (Cout_W(posible)==peso_salida) THEN
                    IF ((Pe(Cout(posible))>estacionario).and.(filtro(ii)==.true.))
    THEN
320                   estacionario=Pe(Cout(posible))
                      candidato=Cout(posible)
322                 END IF
                  END IF
324             END DO
                DO ii=T_start(desde)+1,T_start(desde+1)
326               IF (T_ind(ii)==candidato) THEN
                    nada=ii
```

```
328                     g=candidato
                        exit
330                  END IF
               END DO
332
               !!!Ya seleccionado
334            hacia=g
               IF (inter2==.true.) THEN
336               inter=.true.
               END IF
338         END IF
         END DO
340
         desde=hacia !hacia sera el desde de la primera iteracion, asi vamos haciendo
       un descenso de gradiente hasta cumplir una de las condiciones para salir del
       bucle
342      DEALLOCATE(filtro)
      END DO
344
      IF (comunidades(desde)==0) THEN !si el ultimo nodo del bucle no tiene una basin
       ya asignada significa que es el representante de su propia basin no
       identificada hasta ahora
346      j=desde !esto solo puede pasar si el ultimo nodo es uno de los anteriormente
       visitados en el camino, es decir, si hemos caido en un ciclo
      ELSE
348      j=comunidades(desde)
      END IF
350
      DO jj=1,h !asociamos todos los nodos por los que hemos pasado en el bucle a la
       basin a la que pertenece el ultimo nodo al que hemos llegado
352      comunidades(vect_aux1(jj))=j
      END DO
354
      WRITE(601,*) i,comunidades(i)
356 END DO

358 label=.false.
    DO i=1,N !marcamos otra vez todos los representantes de basins por si hemos tenido
       que anadir alguno
360    label(comunidades(i))=.true.
    END DO
362 h=0

364 DO i=1,N
      IF (label(i).eq..true.) THEN !Si i es el minimo representante de una comunidad/
       basin
366      print*,'si'
         h=h+1
368      IF (h<10) WRITE(sufijo,'(I1)') h
         IF ((9<h).and.(h<100)) WRITE(sufijo,'(I2)') h
370      IF (99<h) WRITE(sufijo,'(I3)') h
         IF (999<h) WRITE(sufijo,'(I4)') h
372   END IF
    END DO
374
    CLOSE(601)
376
    N_sets=h !numero de basins
378 print*,N_sets

380 ALLOCATE(sets(N),start_sets(N_sets+1),P_set(N_sets),K_sets(N_sets,N_sets))
    ALLOCATE(filtro(N),representante(N_sets))
382 print*,'#########'
```

```
g=0
DO i=1,N
   IF (label(i)==.true.) THEN
      g=g+1
      representante(g)=i
   END IF
END DO


sets=0
start_sets=0
start_sets(1)=0
h=0
g=0
P_set=0.0d0
K_sets=0.0d0
DO i=1,N
   IF (label(i).eq..true.) THEN
      g=g+1
      start_sets(g)=h
      DO j=1,N
         IF (comunidades(j)==i) THEN
            h=h+1
            sets(h)=j
         END IF
      END DO
   END IF
END DO

start_sets(g+1)=h

OPEN (603,FILE="info_basins.dat",STATUS="REPLACE",ACTION="WRITE")

WRITE(603,*) N_sets
do i=1,N_sets
   WRITE(603,*) representante(i),start_sets(i+1)-start_sets(i)
END DO
DO i=1,N_sets
   DO j=start_sets(i)+1,start_sets(i+1)
      WRITE(603,*) sets(j),i
   END DO
END DO

print*,'calcula la informacion relativa a las basins'

DO i=1,N_sets
   filtro=.false.
   DO j=start_sets(i)+1,start_sets(i+1)
      filtro(sets(j))=.true.
   END DO
   P_set(i)=SUM(Pe,MASK=filtro)
END DO

DEALLOCATE(filtro)

OPEN (211,FILE="Krates.oup",STATUS="REPLACE",ACTION="WRITE")

DO ii=1,N_sets
   DO jj=1,N_sets
      DO i=start_sets(ii)+1,start_sets(ii+1) !elementos del basin

         g=sets(i)
```

```
446          DO j=T_start(g)+1,T_start(g+1) !salidas de g

448             gg=T_ind(j)
               DO h=start_sets(jj)+1,start_sets(jj+1) !enlaces que van del basin ii al
     basin jj
450                IF (gg==sets(h)) THEN
                      K_sets(jj,ii)=K_sets(jj,ii)+T_tau(j)*Pe(g) !sumamos el producto
    del peso del enlace normalizado con el peso del nodo en el que estamos del
    basin ii
452                   EXIT
                   END IF
454             END DO
           END DO
456       END DO

458       K_sets(jj,ii)=K_sets(jj,ii)/P_set(ii) !normalizamos con el peso total del
    basin (suma de los pesos de sus nodos)
    END DO
460 END DO

462 ALLOCATE(filtro(N_sets))

464 DO i=1,N_sets !info para Krateres.oup
    WRITE(211,*) i, 1.0d0/(1.0d0-K_sets(i,i)), P_set(i)
466 END DO


468
    OPEN(666,FILE="KAS", STATUS="REPLACE", ACTION="WRITE") !escribimos en KAS los pesos
         normalizados de los enlaces de los basins i a los basins ii
470
    do i=1,N_sets
472    do ii=1,N_sets
          WRITE(666,*) i, ii, K_sets(i,ii)
474    enddo
    enddo
476
    do ii=1,N_sets !escribimos en KAS los basins con su minimo de gradiente
        representante
478    WRITE(666,*)ii, representante(ii)
    enddo
480
    print*,' ****** '
482
    OPEN(114,FILE="nodos.csv",status="REPLACE",action="WRITE")
484 OPEN(115,FILE="conectividades.csv",status="REPLACE",action="WRITE") !Escribimos
        todos los nodos y sus enlaces en ficheros .csv y se importaran en gephi para
        hacer grafos

486 Write(114,*) "Id,Label,timeset,score,weight,comunidad"
    Write(115,*) "Source,Target,Type,Id,Label,timeset,weight,comunidad"
488 contador=0
    do ii=1,N
490    Write(114,*) "n",ii,",n",ii,",,,",Pe(ii),",",comunidades(ii) !cambiar formato de
         escritura para que se escriba todo seguido sin espacios ni saltos de linea
       do jj=1,Kout(ii)
492       contador=contador+1
          g=Cout_start(ii)+jj
494       Write(115,'(2(a,I4),2(a,I5),a,g15.5,a,I4)') "n",ii,",n",Cout(g),",Directed,e"
     ,contador,",e",contador,",,",Psalto(g),",",comunidades(ii)
       enddo
496 enddo

498 OPEN(116,FILE="nodos_basins.csv",status="REPLACE",action="WRITE")
```

```
OPEN(117,FILE="conectividades_basins.csv",status="REPLACE",action="WRITE") !
    Escribimos todos los nodos y sus enlaces de basins en fivheros .csv y se
    importaran en gephi para hacer grafos
500
Write(116,*) "Id,Label,timeset,score,weight"
502 Write(117,*) "Source,Target,Type,Id,Label,timeset,weight"
contador=0
504 do ii=1,N_sets
    Write(116,*) "n",ii,",n",ii,",,,",P_set(ii) !cambiar formato de escritura para
    que se escriba todo seguido sin espacios ni saltos de linea
506    do jj=1,N_sets
        if(K_sets(ii,jj).NE.0) then
508       contador=contador+1
          Write(117,'(2(a,I4),2(a,I5),a,g15.5)') "n",ii,",n",jj,",Directed,e",
    contador,",e",contador,",,",K_sets(ii,jj)
510       end if
    enddo
512 enddo

514 print*,' DONE '

516 END program network
```

saco_basins_red_original.f90

.3. Anexo 3: estimadores del balance detallado

Para asegurarnos de que estamos trabajando con una situación estacionaria del sistema queremos ver que se cumple la condición de balance detallado [6, 7]:

$$W_{ji}P_i = W_{ij}P_j$$

Con este fin hemos propuestos tres estimadores para el balance detallado de un par de nodos i y j:

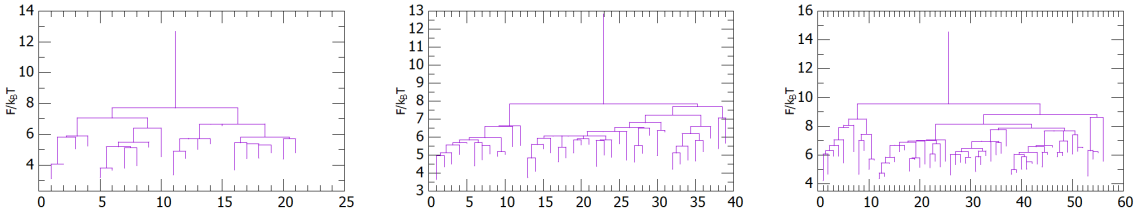
- **Estimador local 1:** $\frac{W_{ji}P_i}{W_{ij}P_j} - 1$
- **Estimador local 2:** $W_{ji}P_i - W_{ij}P_j$
- **Estimador local 3:** $\frac{W_{ji}P_i - W_{ij}P_j}{\sqrt{W_{ji}P_i W_{ij}P_j}} = \sqrt{\frac{W_{ji}P_i}{W_{ij}P_j}} - \sqrt{\frac{W_{ij}P_j}{W_{ji}P_i}}$

El estimador del balance detallado de la red será el promedio a todos los enlaces emparejados (aquellos enlaces de i a j para los que existe un recíproco de j a i) del estimador local. A parte también se observara el número de enlaces desparejados, ya que, si este es demasiado alto, no importa cómo se cumpla el balance detallado en los emparejados, porque de todas formas la propiedad no se cumplirá bien en toda la red. En el caso de que se cumpliese la condición de balance detallado de forma perfecta los estimadores deberían ser 0, pero al tener una trayectoria finita esto no se va a dar, así que hay que establecer una tolerancia por debajo de la cual consideraremos que se cumple la condición.

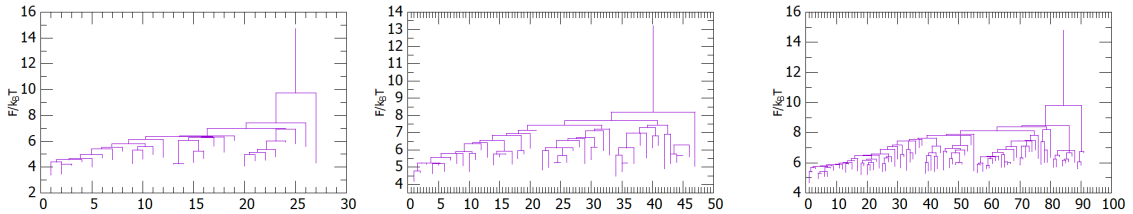
.4. Anexo 4: Temperatura mayor

Veamos cómo se comportan los sistemas observados a una temperatura mayor que la de melting, $T^* = 0{,}5$.

Dendogramas de las trayectorias sin bending a temperatura 0,5:



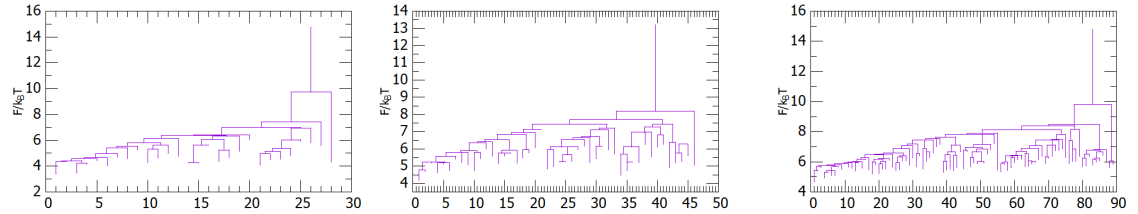
(a) 10x10x10 antiparalelo (b) 15x15x15 antiparalelo (c) 20x20x20 antiparalelo



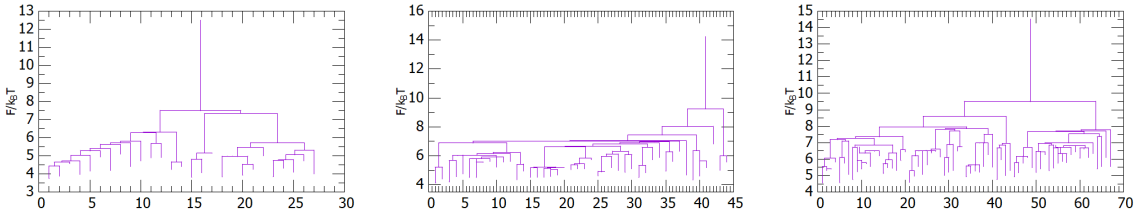
(d) 10x10x10 paralelo (e) 15x15x15 paralelo (f) 20x20x20 paralelo

Figura 20: Dendogramas de la energía libre de las basins para distintos mallados para el sistema a temperatura 0,5.

Dendogramas de las trayectorias con bending a temperaturas altas:



(a) 10x10x10 antiparalelo (b) 15x15x15 antiparalelo (c) 20x20x20 antiparalelo



(d) 10x10x10 paralelo (e) 15x15x15 paralelo (f) 20x20x20 paralelo

Figura 21: Dendogramas de la energía libre de las basins para distintos mallados para el sistema a temperatura 0,5 (antiparalelo) y temperatura 0,45 (paralelo).