Universidad de Zaragoza

Trabajo de Fin de Grado

Desarrollo de una aproximación variacional para la cinética de procesos markovianos en grafos. Aplicación a redes de regulación genética.

Autor:
Pablo Pérez Lázaro

Director:
Dr. Pierpaolo Bruscolini

Universidad Zaragoza

Proyecto del grado en Matemáticas

realizado en el departamento de Física Teórica

12 de septiembre de 2022

Resumen

Gene regulatory networks come in different forms and each has its own characteristics. In this project we consider the application of a statistical physics method (the Cluster Variation Method) to the study of the kinetics of a class of gene regulatory networks, namely activity flow gene networks that also behave as Markov processes.

We define the class of models we use, and for the sake of clearness we also describe in some detail a couple of examples of gene networks that we study later on: a toy model and a cell-cycle model found in the *GINsim* repository. We also describe briefly an approach, alternative to ours, that uses a Kinetic Monte Carlo algorithm (the Gillespie algorithm) to simulate the kinetics of gene regulatory networks: this approach, implemented in the software MaBoss, will represent a benchmark to evaluate our approach.

Methodology plays an important role in our project, so we devote the second and third chapters to describe the rationale that allows to study a dynamical problem within the CVM approach from equilibrium statistical physics, and to introduce a particular approximation (the M-approximation) within the possible CVM choices. In the second chapter, we start by defining the stochastic models we aim at and proposing a standard notation. Those models must be Markovian processes and each node's state in a given time must be determined by itself and its nearest neighbours on the previous time step. Then, we lay down the thermodynamic formalism so that we can later use the statistical physics tools available. This means coming up with a variational free energy expression for the models defined, so that we can obtain the system's dynamic by minimizing that variational. In order to achieve this, we will introduce a fair amount of basic statistical physics theory in a way that it looks familiar to the reader. Finally in that same chapter, we get to derive the generalized *Cluster Variational Method* (CVM). In order to achieve it, we give an appropiate definition of cluster and a brief subsection about Möbius numbers. The goal of the CVM method is to approximate the variational free energy so that we can handle it when looking for its minimum. It does this by truncating the cumulant expansion of the entropy to a set of chosen maximal clusters. Applying a Möbius inversion to this expansion we get a valid entropy approximation. The choice of maximal clusters characterizes the approximation.

In the third chapter, we derive an explicit algorithm for the M approximation so that it can be applied to the toy and cell cycle models. First, we derive the Möbius numbers given the choice of the M clusters as maximal, which determines the approximated entropy. Then, we lay the compatibility constraints between clusters, which are nothing but marginalization identities between probability distributions with respect to the missing variables. Finally, we derive an algorithm that obtains the system dynamic given the initial probability distribution and the transition probabilities. We do so by proving two equivalent expressions for the approximated variational free energy, which can be expressed in terms of *Kullback-Leiber divergence* terms. These divergence terms give a natural and direct way of obtaining the probability distribution of the maximal clusters, from which we can derive the system's evolution.

In the fourth chapter, we apply the method developed to the forementioned models, along with the PQR approximation from the same CVM method and *MaBoSS*. This will be useful in order to check the performance of the M approximation. We discuss three possible trajectories from the toy model where we check if the new method works accordingly. Finally, we check qualitatively whether we can replicate the oscillating behaviour of the cell cycle with the M approximation or not and we draw some conclusions and perspectives.

Índice general

Capítulo 1 Introducción

En este Trabajo de Fin de Grado se trata un método para la cinética de procesos markovianos, que pueda ser aplicado a redes de regulación genética. Estas redes se suelen estudiar desde la física y matemáticas en el campo de la biología de sistemas, que trata de esclarecer propiedades y comportamientos de sistemas presentes en la biología. Este campo ha experimentado en las últimas décadas un crecimiento sustancial, gracias a la gran cantidad de datos experimentales que está necesitando procesar la biología y a la relevancia biomédica de la comprensión de mecanismos de regulación.

Este trabajo extiende los resultados presentados en mi anterior TFG en Física [1], adoptando un enfoque más formal: se introduce y desarrolla la aproximación M, comparándola a la aproximación PQR previamente utilizada, y a los resultados de simulaciones de Monte Carlo Cinético (algoritmo de Gillespie), obtenidas con el software *MaBoSS*.

La memoria está organizada de la siguiente manera: en la introducción, se presentan las redes de regulación genética y las formas más comunes de describirlas y modelar su dinámica. Se introduce luego en una clase de modelos estocásticos markovianos, y se describe brevemente el enfoque implementado en el programa *MaBoSS*, que se utilizará para validar los resultados.

En la sección siguiente, se trata la aplicación de un método variacional al estudio de la cinética de una clase general de modelos markovianos, que abarca los ejemplos mencionados de redes genéticas, y se discute cómo la evolución cinética se puede traducir en el estudio termodinámico de otra red, subrayando los aspectos claves de las aproximaciones. A continuación, se implementan los detalles de la aproximación M, de la que solamente se dan pinceladas en el artículo [2]. Finalmente, se presentan los resultados de aplicar el método sobre algunas redes de regulación sencillas, y se discuten lo hallazgos.

1.1. Redes de regulación genética

Las redes de regulación genética representan las colecciones de segmentos de ADN que interactúan entre sí y con las biomoléculas que regulan su transcripción a proteínas. La biología de sistemas se ocupa de dar una descripción matemática de este tipo de redes, siguiendo el enfoque adecuado.

Su modelado presenta muchas dificultades: en principio, se podría considerar un enfoque microscópico detallado, en el que se representan la secuencia de ADN con sus posibles genes, las ARN polimerasas que los transcriben y su ARNm producto, los ribosomas que ensamblan las cadenas de aminoácidos, los ARNt que llevan los aminoácidos y la concentración de aminoácidos en el citoplasma. También habría que incluir en la descripción el plegamiento de las proteínas, que a continuación puede interactuar con otros genes o con otras proteínas, en el marco de la red red de regulación génica. Está claro que enfoques de este tipo son muy complejos y requieren de muchos datos experimentales para simular todas las reacciones enzimáticas.

Por esta razón, según el comportamiento intracelular que se quiera estudiar, es normal usar un enfoque macroscópico, menos detallado. Por ejemplo, tomando solamente las proteínas que participan en la regulación, con sus interacciones, en una red en que los nodos describen globalmente una proteína, y el gen y ARN necesarios para producirla.

A la hora de determinar el enfoque a seguir para modelizar una red de regulación génica, hay que determinar la naturaleza de cada nodo de la red, que se puede describir con una variable real o booleana. En el caso lógico cada nodo representa una variable cualitativa, no requiere tantos datos como el cinético, aunque la información que se extrae de los resultados es limitada y cualitativa. Por ejemplo, se puede

llegar a conclusiones sobre la estructura del sistema biológico.

En segundo lugar, si la red del modelo se puede representar en una matriz de adyacencia, se denomina grafo secuencial. También, el grafo que representa el modelo puede tener aristas dirigidas o no, según cómo sean las interacciones entre los nodos. Además, si a lo largo de la evolución del sistema hay un estado del que se puede pasar a varios estados distintos, se trata de un grafo estocástico. En caso contrario, si nunca hay problemas de ambigüedad en la evolución, es un grafo determinista.

En función de qué cualidades anteriores cumplan las interacciones del modelo, existen distintas formas de representar la red. En particular, este trabajo se enfoca en modelos booleanos, con dinámica secuencial estocástica y de corrientes de actividad: cada nodo se asociará a una variable booleana, que describe si el gen es activo, la proteína asociada está expresada y ejerce su función)¡, o está apagado. Las aristas entre los nodos son dirigidas y pueden representar activación o represión. Por lo tanto , las interacciones mutuas entre nodos se describen con unas parejas de aristas entre ellos, lo que se puede ver en la figura 1.1 como ejemplo. El estado de cada nodo evoluciona de acuerdo a ecuaciones booleanas, distintas según se encuentren activados o apagados, que devuelven el ritmo al que cambia el nodo o se mantiene en el estado actual.

Modelo de juguete Como primer ejemplo de aplicación, se va a comprobar el correcto funcionamiento del método en un modelo sencillo de juguete del que ya se conoce la dinámica [3][1]. El grafo del modelo consta de tres nodos A, B y C relacionados entre sí mediante las reglas booleanas presentadas en el apéndice A. Existen tres posibles trayectorias en su dinámica, descritas en la figura 1.1. Se ve claramente que es un modelo estocástico pues del estado $ABC \equiv 001$ se puede pasar a dos estados distintos.

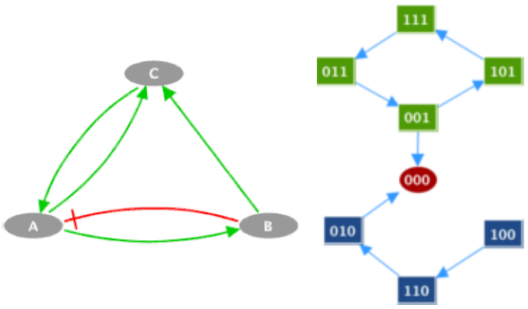

Figura 1.1: Izquierda: diagrama de interacción entre nodos. Derecha: evolución dinámica de los estados. A partir del estado $ABC \equiv 100$, arranca una trayectoria determinista (azul) hasta el atractor $ABC \equiv 000$ (rojo). En verde, un bucle con escape estocástico hacia el atractor, debido al decaimiento de C desde $ABC \equiv 001$. La transición al 101 tiene un ritmo $Au = 1$, mientras que C decae como $Cd = 10$ (escape rápido) ó $Cd = 0{,}5$ (escape lento).

Ciclo celular El ciclo celular es el proceso mediante el cual las células crecen y se duplican, dando lugar a dos células con el mismo contenido genético que la madre [4]. Este se desarrolla en cinco fases: fase G0 (señal extracelular que desencadena el ciclo celular), fase G1 (crecimiento celular, síntesis de proteínas y ARN), fase S (síntesis del material genético), fase G2 (comprobación de errores en la replicación del ADN), fase M (mitosis, se da la división celular).

Las ciclinas son una familia de proteínas esenciales para la regulación del ciclo celular. Su concentración varía en cada cambio de fase, aumentando o disminuyendo. Por ello, las ciclinas son utilizadas como indicadoras del transcurso del ciclo celular. Se tendrán en cuenta: Ciclina D, Ciclina E, Ciclina A y Ciclina B. Su relación con las fases del ciclo celular se presenta en la figura 1.2.

Además, para modelizar correctamente el ciclo celular hacen falta otras proteínas que cooperan con las ciclinas: proteínas del retinoblastoma (Rb), p27, E2F, complejo de promoción de la anafase (APC), Cdc20, Cdh1 y UbcH10. Para definir la red que describe el ciclo celular, nos hemos dirigido al repositorio *GINsim* que contiene modelos validados, además con información sobre condiciones iniciales y tipo de enlaces (promoción o inhibición) entre las diferentes proteínas. Sin embargo, *GINsim* no proporciona información sobre las tasas de evolución: solo un modelo lógico de una red de corrientes de actividad.

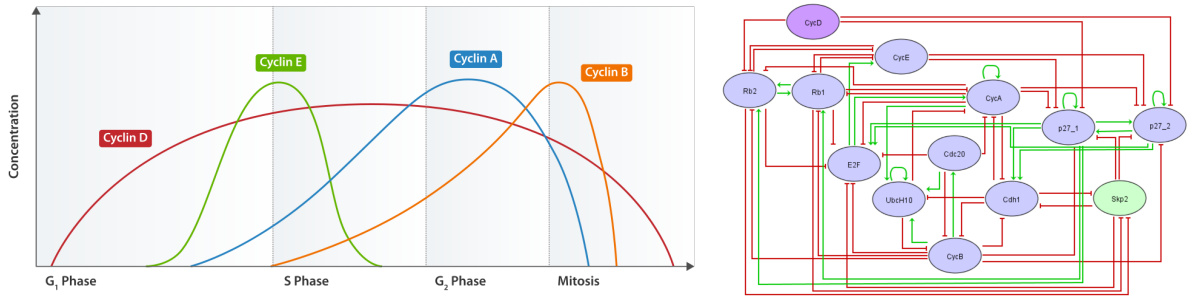

Figura 1.2: A la izquierda, se presentan las concentraciones de las ciclinas en la célula. Podemos observar cómo la ciclina D está presente durante todo el ciclo, pues se activa por agentes externos e inicia el ciclo celular. También, la ciclina E inicia la fase S y la ciclina A se mantiene durante toda la replicación del ADN y su comprobación posterior. Finalmente, la ciclina B está presente durante la mitosis, hasta que la célula se divide [5]. A la derecha, una representación del modelo del ciclo celular.

Este tipo de modelos requieren poca información sobre la interacción entre nodos, pero solo son válidos para hacer análisis cualitativos del sistema. Para completar el modelo con las informaciones sobre la evolución de cada nodo, nos hemos basado en el modelado del ciclo celular en el artículo [3], donde se estudia mediante la herramienta *MaBoSS* con las reglas booleanas expuestas en el apéndice B.

1.2. MaBoSS

MaBoSS [3] es un entorno de simulación para modelos booleanos, creado para el uso en redes biológicas con las características descritas en las secciones anteriores.

El entorno simula un número dado s de trayectorias del sistema. En cada trayectoria, se ve la evolución aplicando el algoritmo de Monte Carlo Cinético (Gillespie) para procesos de Markov a tiempo continuo, que determina qué transición de estado se realiza y cuánto tiempo tarda el sistema en realizarla, donde en cada transición se cambia un único nodo cada vez. El algoritmo continúa hasta llegar a una transición que ocurriría más allá de un tiempo final determinado por el usuario. A pesar de trabajar con Gillespie de tiempos continuos, *MaBoSS* finalmente discretiza el tiempo, para poder promediar trayectorias diferentes, considerando los histogramas de frecuencias de cada estado en cada ventana temporal.

Para obtener la dinámica del sistema hay que introducir como datos los nodos del sistema, las condiciones iniciales del sistema en forma de probabilidad de que al inicio cada nodo esté activo, las reglas lógicas para la activación o inhibición de los nodos y los ritmos de transición para cada activación o inhibición, pudiendo ser estos últimos experimentales o meras estimaciones. Por ello, es un método válido para el análisis cualitativo de modelos lógicos con red de corrientes de actividad, como los modelos de ciclo celular y de juguete que consideramos en este trabajo

Como resultados se obtienen varias salidas: un fichero de texto con todas las s trayectorias que se han simulado y el tiempo en el que se ha dado cada transición de estado, y dos ficheros de excel con las probabilidades de cada estado posible en cada ventana temporal. Además existe la opción de que el entorno trabaje con nodos internos ocultos, que es útil cuando hay muchos nodos pues los 2^N estados pueden dificultar la extracción de conclusiones. En este caso, el cálculo de la evolución se realiza de la forma usual, pero en los resultados aparecen solamente los nodos no ocultos.

El algoritmo tiene algunas limitaciones de uso: por ejemplo, el número máximo de nodos es $N = 24$ y para obtener buena resolución en los resultados hace falta aumentar bastante el número de trayectorias.

Capítulo 2 Aproximación Variacional a la Dinámica Estocástica

En este capítulo, se va a plantear todo el formalismo teórico necesario para poder aplicar el método de Variación de Clúster (CVM) a modelos lógicos con redes de corrientes de actividad, como los presentados en el capítulo anterior. En la primera sección, se da una definición probabilística del tipo general de modelos de evolución estocásticas que se pueden tratar. Más adelante, se establece una relación entre la cinética de estos modelos y el equilibrio termodinámico de un modelo espaciotemporal contruido a partir de ellos, lo que permite la implementación de un formalismo termodinámico sobre estos modelos, para poder usar herramientas de la física estadística. Finalmente, sobre este formalismo realiza la aproximación a los clústeres maximales, introduciendo un funcional que se minimiza en el método CVM para dar con la evolución del sistema resultante.

El método CVM fue originalmente propuesto en 1951 por Kikuchi [6], como método para aproximar la entropía en sistemas termodinámicos modelados con redes, usándolo para resolver el modelo de Ising lineal. La idea principal es restringir el rango de las interacciones entre nodos a un conjunto de clústeres maximales, cuya elección determina la bondad de la aproximación. De esta forma por ejemplo, escoger los pares de primeros vecinos es equivalente a la aproximación Bethe-Peierls y tomando los nodos por separado se recupera el campo medio usual.

Desde entonces, han aparecido distintas reformulaciones del método. Una de ellas es la propuesta por An [7] en 1988, en relación con la inversión de Möbius.

En particular, una extensión del CVM al estudio fue detallada en 2017 por Pelizzola y Pretti [2]. Siguiendo los pasos de esta formulación, se obtiene la forma de estudiar un modelo de dinámica estocástica, interpretándolo como un problema de equilibrio termodinámico en un sistema con una dimensión extra t, que representa las réplicas de dicho sistema en tiempos sucesivos.

A continuación se detalla el método y se va a ver cómo hay distintos tipos de aproximación según la elección de los clústeres maximales. En concreto, habiendo implementado la aproximación PQR en un trabajo previo [1], en esta memoria expondremos una descripción detallada de la aproximación M, más compleja, desarrollando los pasos intermedios y demostraciones que se dejan implícitas en [8][2].

2.1. Modelo de dinámica estocástica

Se considera un proceso estocástico discreto, donde el vector aleatorio $X^{(t)} = (X_1^{(t)}, ..., X_N^{(t)})$ con N un número natural finito, representa el estado de un sistema en el tiempo $t = 0, ..., \tau$ finito. En este vector, las variables aleatorias $X_i^{(t)}$ representan los valores que puede tomar cada nodo i en el tiempo t.

Se asume que la evolución temporal del sistema $X^{(0)}, ..., X^{(\tau)}$ es un proceso de Markov [9] , de forma que el estado del sistema en un tiempo $t+1$ depende solo del estado en el tiempo t. Esta propiedad es común en los sistemas biológicos a los que nos interesa aplicar el método. Sobre esta evolución, suponemos que se conocen la *función de masa de probabilidad* (fmp) de la configuración inicial $p^{(0)}(x^{(0)})$ y las probabilidades de transición de un estado $x^{(t)}$ al siguiente $x^{(t+1)}$, denotadas como $\omega^{(t)}(x^{(t+1)}|x^{(t)})$. Estas probabilidades condicionadas cumplen por construcción que la probabilidad de transicionar a los posibles $x^{(t+1)}$ está normalizada

$$\sum_{x^{(t+1)}} \omega^{(t)}\left(x^{(t+1)} \middle| x^{(t)}\right) = 1, \quad t = 0, ..., \tau. \tag{2.1}$$

Toda esta información es inherente al sistema.

El vector aleatorio (v.a.) de las posibles trayectorias que puede tomar el sistema $\mathbb{X} := (X^{(0)}, ..., X^{(\tau)})$ toma valores en el conjunto contable de puntos $\{\mathbb{x}_n\} = \{(x_1^{(0)}, ..., x_N^{(0)}, ..., x_1^{(\tau)}, ..., x_N^{(\tau)})_n\} \in \mathbb{R}^{N\tau}$ y se define en un espacio de probabilidad $(\Omega, \mathcal{F}, \mathbb{P})$. Estos son el conjunto de estados $\{\mathbb{x}_n\}_{n=1}^{N\tau} \in \Gamma$. La fmp de $\mathbb{X}$ es

$$p\left(x^{(0)}, ..., x^{(\tau)}\right) = \mathbb{P}\left(X^{(0)} = x^{(0)}, ..., X^{(\tau)} = x^{(\tau)}\right). \tag{2.2}$$

Proposición 1. *La fmp del sistema descrito toma la forma*

$$p\left(x^{(0)}, ..., x^{(\tau)}\right) = p^{(0)}\left(x^{(0)}\right) \prod_{t=0}^{\tau-1} \omega^{(t)}\left(x^{(t+1)} \middle| x^{(t)}\right).$$

Demostración. Se prueba por inducción sobre $t \in \mathbb{N}$. Para $t = 1$, utilizando la relación entre probabilidad condicionada y conjunta, tenemos

$$\mathbb{P}\left(X^{(1)} = x^{(1)}, X^{(0)} = x^{(0)}\right) = \mathbb{P}\left(X^{(0)} = x^{(0)}\right)\mathbb{P}\left(X^{(1)} = x^{(1)} \middle| X^{(0)} = x^{(0)}\right) = p^{(0)}\left(x^{(0)}\right)\omega^{(0)}\left(x^{(1)} \middle| x^{(0)}\right).$$

Y ahora veamos que para $t \in \mathbb{N}$ la probabilidad conjunta se puede escribir en función de la probabilidad condicionada

$$\mathbb{P}\left(X^{(t+1)} = x^{(t+1)}, ..., X^{(0)} = x^{(0)}\right) = \mathbb{P}\left(X^{(t+1)} = x^{(t+1)} \middle| X^{(t)} = x^{(t)}, ..., X^{(0)} = x^{(0)}\right)\mathbb{P}\left(X^{(t)} = x^{(t)}, ..., X^{(0)} = x^{(0)}\right).$$

Como se trata de un proceso de Markov el estado en el tiempo $t+1$ depende únicamente del estado en el tiempo t anterior

$$\mathbb{P}\left(X^{(t+1)} = x^{(t+1)} \middle| X^{(t)} = x^{(t)}, ..., X^{(0)} = x^{(0)}\right) = \mathbb{P}\left(X^{(t+1)} = x^{(t+1)} \middle| X^{(t)} = x^{(t)}\right) = \omega\left(x^{(t+1)} \middle| x^{(t)}\right),$$

y suponiendo cierta la hipótesis para $t-1$

$$p\left(x^{(t+1)}, ..., x^{(0)}\right) = \omega\left(x^{(t+1)} \middle| x^{(t)}\right) p\left(x^{(t)}, ..., x^{(0)}\right) = p\left(x^{(0)}\right) \prod_{i=0}^{t} \omega\left(x^{(i+1)} \middle| x^{(i)}\right).$$

El enunciado se obtiene tomando $t = \tau$. □

A menudo, en modelos físicos y biológicos el estado para un nodo i en un paso temporal viene determinado solamente por el estado de este nodo y sus vecinos ∂i, unidos a i mediante una arista no dirigida, sin importar la dirección de la actividad, en el paso temporal anterior. Nos limitaremos a este tipo de modelos, en los que las probabilidades de transición $\omega^{(t)}(x^{(t+1)}|x^{(t)})$ se pueden expresar factorizadas

$$\omega^{(t)}(y|x) = \prod_i \omega_i^{(t)}(y_i|x_{i,\partial i}), \tag{2.3}$$

dada la notación en cada paso temporal t para los v.a. $X^{(t+1)} \equiv Y$ y $X^{(t)} \equiv X$. Además, las v.a. $X_1^{(0)}, ..., X_N^{(0)}$ se toman independientes, de forma que para cualquier estado $x^{(0)}$ se tiene

$$p^{(0)}(x) = \prod_i p_i^{(0)}(x_i). \tag{2.4}$$

Es interesante observar como la proposición 1 proporciona una receta para calcular exactamente la probabilidad en cada tiempo: definiendo un espacio vectorial de dimensión 2^N donde cada vector de la base canónica describe un estados $\mathbb{x} \in \Gamma$, y las coordenadas representan las probabilidades p(x), la omegas(t+1|t) se describen como matrices $2^N \times 2^N$, y la proposición 1 permite calcular la probabilidad al tiempo t como producto de matrices por el vector de probabilidad inicial. Sin embargo, este enfoque se hace rápidamente impracticable para N grande, y las consideraciones (2.3) y (2.4) no eliminan la dificultad. Por eso, el resto del trabajo está dedicado a presentar una aproximación tratable del problema.

2.2. Formalismo termodinámico

Se va a demostrar que la función de masa de probabilidad de la proposición 1, que describe la dinámica del sistema dado, se puede escribir en forma variacional, como la probabilidad que minimiza un funcional con forma de divergencia de Kullback-Leiber.

Empezamos definiendo un nuevo grafo, constituido por la unión de los estados de N nodos en todos los tiempos posibles $t = 0, ..., \tau$, así cada estado de este sistema se puede entender como el apilamiento de $\tau + 1$ vectores aleatorios $X^{(t)}$, formando un vector aleatorio de estado espaciotemporal. Las trayectorias completas pasan a ser un subconjunto de los estados de este nuevo sistema, ya que no todos los estados corresponderán a trayectorias posibles.

La observación principal consiste en que las trayectorias solución de la evolución dinámica del sistema, enunciadas en la proposición 1, se trasladan al estado de equilibrio termodinámico del sistema definido en este grafo espaciotemporal. Para demostrarlo, es necesario primero recordar la definición de divergencia Kullback-Leiber.

En general, dadas dos distribuciones de probabilidad discretas P y Q definidas en un mismo espacio de probabilidad $(\Omega, \mathcal{F}, \mathbb{P})$, para Γ finito o infinito, se denota la divergencia Kullback-Leiber [10] de P y Q como

$$D_{KL}(P \| Q) = \sum_{\mathbb{x} \in \Gamma} P(\mathbb{x}) \ln\left(\frac{P(\mathbb{x})}{Q(\mathbb{x})}\right), \tag{2.5}$$

que está bien definida si para todo $\mathbb{x} \in \Gamma$, $Q(\mathbb{x}) = 0$ implica que $P(\mathbb{x})$ es idénticamente nula. Esta divergencia es siempre no negativa, y además se anula si y sólo si $P = Q$ en casi todo punto, como se demuestra usando la desigualdad de Gibbs. Sea Γ el espacio de estados finito, se cumple la desigualdad de Gibbs [11]

$$-\sum_{\mathbb{x} \in \Gamma} P(\mathbb{x}) \ln P(\mathbb{x}) \leq -\sum_{\mathbb{x} \in \Gamma} P(\mathbb{x}) \ln Q(\mathbb{x}), \tag{2.6}$$

donde la igualdad se da únicamente cuando $Q(\mathbb{x}) = P(\mathbb{x})$ para todo $\mathbb{x} \in \Gamma$.

Proposición 2. *La expresión*

$$\mathcal{F}[p] = \sum_{\mathbb{x} \in \Gamma} p\left(x^{(0)}, ..., x^{(\tau)}\right) \ln \frac{p\left(x^{(0)}, ..., x^{(\tau)}\right)}{p^{(0)}\left(x^{(0)}\right) \prod_{t=0}^{\tau-1} \omega^{(t)}\left(x^{(t+1)} \middle| x^{(t)}\right)} \geq 0,$$

es el funcional de argumento p, que alcanza su mínimo absoluto para la evolución temporal del modelo de dinámica estocástica.

Demostración. En primer lugar, p está normalizada por definición.

Además, si para un tiempo t se encuentra en un $x^{(t)}$ dado, la probabilidad de transicionar a los distintos $x^{(t+1)}$ cumple (2.1) pues es una distribución de probabilidad condicionada. De esta forma, para cualquier $x^{(0)}$ posible

$$\sum_{x^{(1)},...,x^{(\tau)}} \prod_{t=0}^{\tau-1} \omega^{(t)}(x^{(t+1)}|x^{(t)}) = \sum_{x^{(1)},...,x^{(\tau-1)}} \left[\prod_{t=0}^{\tau-2} \omega^{(t)}(x^{(t+1)}|x^{(t)}) \sum_{x^{(\tau)}} \omega^{(\tau-1)}(x^{(\tau)}|x^{(\tau-1)})\right] = ... = \sum_{x^{(1)}} \omega^{(0)}(x^{(1)}|x^{(0)}) = 1,$$

y así está claro que

$$\sum_{\mathbb{x} \in \Gamma} p^{(0)}\left(x^{(0)}\right) \prod_{t=0}^{\tau-1} \omega^{(t)}\left(x^{(t+1)} \middle| x^{(t)}\right) = \sum_{x^{(0)}} p^{(0)}\left(x^{(0)}\right) \sum_{x^{(1)},...,x^{(\tau)}} \prod_{t=0}^{\tau-1} \omega^{(t)}(x^{(t+1)}|x^{(t)}) = \sum_{x^{(0)}} p^{(0)}\left(x^{(0)}\right) = 1,$$

está normalizado. Así, p y $p^{(0)}\prod_{t=0}^{\tau-1}\omega^{(t)}$ son fmp. Además, como el tiempo τ y el número de nodos N son finitos, Γ también lo es.

Entonces, por la desigualdad de Gibbs (2.6), queda claro que $\mathcal{F}[p] \geq 0$. Además, el mínimo absoluto se obtiene si y solo si $p = p^{(0)}\prod_{t=0}^{\tau-1}\omega^{(t)}$, que es la condición que cumple el modelo de dinámica estocástica en la proposición 1. □

A continuación, se va a tratar de ver cómo la expresión de la proposición 2 se corresponde con la energía libre del sistema termodinámico definido como la unión de los N nodos en todos los tiempos posibles $t = 0, ..., \tau$. Entonces, la probabilidad que resuelve el sistema dinámico será la del equilibrio termodinámico. Para este desarrollo, se usa la Mecánica Estadística del Equilibrio.

El problema básico a resolver de la Física Estadística consiste en determinar las relaciones termodinámicas fundamentales de cualquier sistema macroscópico a partir de las interacciones entre sus grados de libertad. La solución al problema viene dada por el cumplimiento de los siguientes postulados:

I) Un sistema termodinámico se representa mediante un espacio de probabilidad $\{\Omega, \mathcal{F}, \mathbb{P}\}$. En el caso discreto, se define una v.a. $\mathbb{X}$, que toma valores en el conjunto de posibles estados del sistema $\{\mathbb{x}\}_{n\geq 1} = \Gamma$. Para cada medida de probabilidad posible $\mathbb{P}$ que se pueda considerar, a la v.a. $\mathbb{X}$ le corresponde una fmp p de la forma (2.2).

II) Las variables extensivas definen el estado termodinámico de equilibrio. Son los valores esperados de ciertas funciones sobre Γ, según la fmp p_{eq} que presenta el sistema en el equilibrio.

III) La fmp del equilibrio p_{eq} viene dada por la medida de probabilidad $\mathbb{P}_{eq}$ que maximiza la desinformación (entropía) de expresión variacional

$$S[p] := -\sum_{\mathbb{x}\in\Gamma} p(\mathbb{x}) \ln p(\mathbb{x}), \tag{2.7}$$

condicionada a las restricciones de los parámetros extensivos en el equilibrio.

El ensemble canónico es un formalismo de la física estadística que permite describir los estados de sistemas termodinámicos con paredes diatérmanas, pero restrictivas con el exterior respecto del resto de variables posibles. Es decir, la energía de los estados $E = E(\mathbb{x})$ puede intercambiarse con el entorno y tomar valores arbitrarios, pero el número de partículas o nodos y el volumen que ocupan se mantienen constantes. Se define energía interna variacional como el valor esperado de la energía

$$U[p] = \sum_{\mathbb{x}\in\Gamma} p(\mathbb{x})E(\mathbb{x}), \tag{2.8}$$

en el equilibrio es una variable extensiva $U := U[p_{eq}]$ denominada energía interna.

Ahora, teniendo en cuenta el postulado III, se puede obtener la expresión de la distribución de probabilidad en el equilibrio p_{eq} para el formalismo canónico.

Lema 3. *Para el ensemble canónico, la distribución del sistema termodinámico en el equilibrio se expresa como la distribución de Boltzmann*

$$p_{eq}(\mathbb{x}) = \frac{1}{Z}e^{-E(\mathbb{x})}, \quad \mathbb{x} \in \Gamma,$$

donde $Z = \sum_{\mathbb{x}\in\Gamma} e^{-E(\mathbb{x})}$ *es la función de partición canónica.*

Demostración. El postulado III define $\{p_{eq}(\mathbb{x})\}$ como la distribución $\{p(\mathbb{x})\}$ que maximiza la entropía $S(\{p(\mathbb{x})\})$ bajo la restricción de los parámetros extensivos. Entonces es un problema de optimización de (2.7) bajo las restricciones de la energía interna U

$$U - U[p] = 0, \tag{2.9}$$

y de normalización

$$1 - \sum_{\mathbb{x}\in\Gamma} p(\mathbb{x}) = 0. \tag{2.10}$$

Los problemas de óptimos restringidos se pueden resolver mediante el método de multiplicadores de Lagrange, tomando $\ln Z - 1$ como multiplicador asociado a (2.10) y λ asociado a (2.9). El teorema de Lagrange asegura que en el óptimo restringido buscado, se anula la variación de la función

$$S(\{p(\mathbb{x})\}) + (\ln Z - 1)\left(1 - \sum_{\mathbb{x}\in\Gamma} p(\mathbb{x})\right) + \lambda\left(U - \sum_{\mathbb{x}\in\Gamma} p(\mathbb{x})E(\mathbb{x})\right),$$

respecto de sus variables $(\{p(\mathbb{x})\}_{\mathbb{x}\in\Gamma}, \ln Z - 1, \lambda)$. Así, derivando respecto de $p(\mathbb{x})$, la distribución en el equilibrio cumple

$$\ln p_{eq}(\mathbb{x}) = -\ln Z - \lambda E(\mathbb{x}),$$

que despejando p_{eq}, tiene la forma de la expresión del enunciado, donde usando (2.10) se obtiene

$$Z = \sum_{\mathbb{x}\in\Gamma} e^{-\lambda E(\mathbb{x})}.$$

En cuanto al parámetro λ, la entropía se puede escribir como

$$S(\{p(\mathbb{x})\}) = -\sum_{\mathbb{x}\in\Gamma} p(\mathbb{x}) \ln p(\mathbb{x}) = \sum_{\mathbb{x}\in\Gamma} p(\mathbb{x}) \ln Z + \lambda \sum_{\mathbb{x}\in\Gamma} p(\mathbb{x})E(\mathbb{x}) = \ln Z + \lambda U,$$

de forma que $\lambda = \partial S/\partial U$. Teniendo presente la definición termodinámica de temperatura [12], encontramos que $\lambda = 1/T$. Sin embargo, como consideraremos la entropía Shannon [13], se puede tomar 1. Como p_{eq} está claramente normalizada y todos sus términos son positivos, está bien definida como fmp del vector aleatorio $\mathbb{X}$. Así, la fmp del vector aleatorio $\mathbb{X}$ es la distribución de probabilidad del sistema en el equilibrio termodinámico $p(x^{(0)}, ..., x^{(\tau)}) = p_{eq}(\mathbb{x})$. □

A menudo, es demasiado complicado utilizar la expresión de Boltzmann del lema 3 para obtener la fmp en el equilibrio, pues es una suma sobre todos los estados posibles Γ. También, suele ser complicado encontrar el máximo de la entropía (2.7). Sin embargo, se pueden construir aproximaciones a partir de la formulación equivalente de la probabilidad de equilibrio, como aquella que minimiza la energía libre de Helmholtz. Esta en el equilibrio termodinámico se define como

$$F := U - \mathcal{S}[p_{eq}].$$

Entonces, para encontrar F hay que minimizar el variacional de la energía libre $\tilde{\mathcal{F}}$, que dadas las posibles distribuciones de probabilidad $\{p(\mathbb{x})\}_{\mathbb{x}\in\Gamma}$ para los estados del sistema toma la forma

$$\tilde{\mathcal{F}}(p) := \sum_{\mathbb{x}\in\Gamma} [p(\mathbb{x})E(\mathbb{x}) + p(\mathbb{x}) \ln p(\mathbb{x})] = U[p] - \mathcal{S}[p]. \tag{2.11}$$

La relación fundamental (lema 3) en el formalismo canónico viene naturalmente expresada en términos de la energía libre de Helmholtz como

$$F := -\ln Z. \tag{2.12}$$

Lema 4. *Para toda* $\{p(\mathbb{x})\}_{\mathbb{x}\in\Gamma}$*, se tiene*

$$\tilde{\mathcal{F}}(p) \geq F.$$

Demostración. En primer lugar, la energía interna variacional se puede expresar como valor esperado de la energía $U[p] = \mathbb{E}E$.

Supongamos que se tiene otro sistema cualquiera donde los estados tienen energías $E_0(\mathbb{x})$, y una distribución de probabilidad que tiene la forma de Boltzmann

$$p_0(\mathbb{x}) = \frac{1}{Z_0} e^{-E_0(\mathbb{x})}.$$

Entonces, veamos que la divergencia de Kullback-Leiber

$$D_{KL}(p_0 \| p_{eq}) = \sum_{\mathbb{x}\in\Gamma} \left(\frac{e^{-E_0(\mathbb{x})}}{Z_0} \ln \frac{e^{-E_0(\mathbb{x})}}{Z_0} - \frac{e^{-E_0(\mathbb{x})}}{Z_0} \ln \frac{e^{-E(\mathbb{x})}}{Z} \right) = -\ln Z_0 - \mathbb{E}_0 E_0 + \ln Z + \mathbb{E}_0 E = F_0 + \mathbb{E}_0(E - E_0) - F,$$

que es no negativa por definición. Así, $F \leq F_0 + \mathbb{E}_0(E - E_0) = \tilde{\mathcal{F}}(p_0)$. □

Como además se alcanza en p_{eq} la cota $\tilde{\mathcal{F}}(p_{eq}) = F$, se puede dar una definición alternativa $F := \min_p \tilde{\mathcal{F}}(p)$. Entonces, la probabilidad de equilibrio minimiza el variacional de energía libre pues $p_{eq} = \arg\min \tilde{\mathcal{F}}$.

Partiendo de (2.12) y del lema 3 se puede obtener una expresión para $\tilde{\mathcal{F}}$.

Proposición 5. *La energía libre variacional se puede expresar como*

$$\tilde{\mathcal{F}}(p) = F + \sum_{\mathbb{x}\in\Gamma} p(\mathbb{x}) \ln \frac{p(\mathbb{x})}{p_{eq}(\mathbb{x})},$$

cuyo mínimo absoluto se obtiene para $p = p_{eq}$*.*

Demostración. Introduciendo la expresión de $p_{eq}(\mathbb{x})$ obtenida en el lema 3 al lado derecho de la expresión del enunciado, se obtiene fácilmente

$$F + \sum_{\mathbb{x}\in\Gamma} p(\mathbb{x}) \ln\left(Z \frac{p(\mathbb{x})}{e^{-E(\mathbb{x})}} \right) = F + \ln Z \sum_{\mathbb{x}\in\Gamma} p(\mathbb{x}) + \sum_{\mathbb{x}\in\Gamma} [p(\mathbb{x}) \ln p(\mathbb{x}) + p(\mathbb{x}) E(\mathbb{x})] = F + \ln Z + \tilde{\mathcal{F}}(p) = \tilde{\mathcal{F}}(p),$$

usando (2.10) para la suma a todos los estados Γ de $p(\mathbb{x})$, (2.11) para la expresión de $\tilde{\mathcal{F}}(p)$ y (2.12) para la definición de F.

Ahora, como dada la expresión del lema 3 p_{eq} es una distribución normalizada y no negativa, entonces se puede reformular la expresión del enunciado en función de un término de divergencia Kullback-Lieber. Es decir,

$$\tilde{\mathcal{F}}(p) = F + D_{KL}(p \| p_{eq}),$$

que por definición de D_{KL} se tiene $\tilde{\mathcal{F}}(p) \geq F$ y el único mínimo se da cuando la divergencia se anula, en $p = p_{eq}$. □

Ahora, podemos ver cómo el funcional de la proposición 2 se puede interpretar como une energía libre variacional $\tilde{\mathcal{F}}$. Reescribiéndolo de una forma similar a (2.11)

$$\mathcal{F}[p] = \sum_{\mathbb{x}\in\Gamma} \left[-p\left(x^{(0)}, ..., x^{(\tau)}\right) \ln\left(p^{(0)}\left(x^{(0)}\right) \prod_{t=0}^{\tau-1} \omega^{(t)}\left(x^{(t+1)} \middle| x^{(t)}\right) \right) + p\left(x^{(0)}, ..., x^{(\tau)}\right) \ln p\left(x^{(0)}, ..., x^{(\tau)}\right) \right], \quad (2.13)$$

e imponiendo la identificación $\mathcal{F}[p] = \mathbb{E}E - \mathcal{S}[p]$ se ve que la energía para cada estado del sistema termodinámico espaciotemporal vendría dada por

$$E(x^{(0)},...,x^{(\tau)}) := -\ln p^{(0)}(x^{(0)}) - \sum_{t=0}^{\tau-1} \ln \omega^{(t)}(x^{(t+1)}|x^{(t)}), \quad \mathbb{x} \in \Gamma, \tag{2.14}$$

y la entropía (variacional) sería

$$\mathcal{S}[p] = -\sum_{\mathbb{x}\in\Gamma} p\left(x^{(0)},...,x^{(\tau)}\right) \ln p\left(x^{(0)},...,x^{(\tau)}\right). \tag{2.15}$$

Además, de la proposición 2 se deduce que la fmp en el equilibrio tomaría la forma $p_{eq} = \arg\min \mathcal{F} = p^{(0)} \prod_{t=0}^{\tau-1} \omega^{(t)}$.

Así, el funcional $\mathcal{F}$ de la proposición 2 se identifica con un variacional de energía libre de un sistema termodinámico con ensemble canónico, pues tiene la forma de la proposición 5 con mínimo absoluto en $p = p^{(0)} \prod_{t=0}^{\tau-1} \omega^{(t)}$. Entonces también, (2.14) es la energía de cada estado y (2.15) es la entropía variacional de dicho sistema termodinámico espaciotemporal. Esto significa que se ha conseguido trasladar el sistema dinámico a un sistema termodinámico, cuyo equilibrio se da para la probabilidad de la proposición 1 que resuelve el sistema dinámico. Gracias a este mapeo del sistema dinámico, se pueden aprovechar todas las tecnologías de la física estadística.

2.3. Aproximación CVM

La aproximación CVM es un método que proporciona una expresión más manejable del variacional de energía libre $\mathcal{F}$ para aquellos modelos donde la interacción involucra un número razonablemente pequeño de primeros vecinos, como es el caso de las aplicaciones a la biología de sistemas que consideramos. La idea principal reside en considerar la información de cada nodo y la interacción con sus primeros vecinos como suficiente para describir perfectamente la energía mientras que se aproxima el término de la entropía (2.7) truncando su expansión en cumulantes, calculados sobre clústeres oportunos.

La noción general de clúster es un conjunto de nodos agrupados según un criterio. Para el caso particular del sistema descrito en las secciones 2.1 y 2.2, los clústeres son un subconjunto de nodos vecinos en la red espaciotemporal, a los cuales se asocia un vector aleatorio que describe su estado. La distribución de probabilidad de un clúster α es la marginal del vector aleatorio $\mathbb{x}_\alpha$ formado por los nodos $i \in \alpha$ sobre el vector aleatorio $\mathbb{X}$ de todo el sistema

$$p_\alpha(\mathbb{x}_\alpha) = \sum_{\mathbb{x}\setminus\mathbb{x}_\alpha} p(\mathbb{x}),$$

y consecuentemente su entropía toma la forma

$$\mathcal{S}_\alpha = -\sum_{\mathbb{x}_\alpha} p_\alpha(\mathbb{x}_\alpha) \ln p_\alpha(\mathbb{x}_\alpha).$$

Para describir la aproximación que relaciona la $\mathcal{S}$ exacta con la $\mathcal{S}_\alpha$ de los clústeres, es necesario introducir antes el concepto de función de Möbius.

2.3.1. Números de Möbius

El uso de la inversión de Möbius permite una formulación más clara y simple del CVM que la introducida en 1951 por Kikuchi [6].

Un conjunto $\mathscr{P}$ es parcialmente ordenado [14] si está dotado de una relación binaria $\leq$ que satisface

- reflexividad: $\alpha \leq \alpha$ para $\alpha \in \mathscr{P}$,
- transitividad: $\alpha \leq \beta$ y $\beta \leq \gamma$ implican que $\alpha \leq \gamma$ para $\alpha,\beta,\gamma \in \mathscr{P}$,
- antisimetría: $\alpha \leq \beta$ y $\beta \leq \alpha$ implican que $\alpha = \beta$ para $\alpha,\beta \in \mathscr{P}$.

Se dice que $\leq$ define un orden parcial en $\mathscr{P}$, y dos elementos son comparables si están relacionados. De esta forma, en $\mathscr{P} \times \mathscr{P}$ está bien definida la siguiente función, que dice si dos elementos de $\mathscr{P}$ son comparables o no

$$\zeta(\beta,\alpha) = \begin{cases} 1, & \beta \leq \alpha \\ 0, & resto \end{cases} \quad \text{para } \alpha,\beta \in \mathscr{P}. \tag{2.16}$$

La función de Möbius μ en un conjunto parcialmente ordenado $\mathscr{P}$ con dominio $\mathscr{P} \times \mathscr{P}$ se define a partir de

$$\sum_{\substack{\alpha \leq \beta \leq \gamma \\ \beta \in \mathscr{P}}} \zeta(\alpha,\beta)\mu(\beta,\gamma) = \delta(\alpha,\gamma), \quad \alpha,\gamma \in \mathscr{P}, \tag{2.17}$$

donde δ es la delta de Kronecker. Es decir, μ es la inversa de ζ en $\mathscr{P}$.

En general, para dos conjuntos A, S y dos funciones f, g tales que

$$g(A) = \sum_{S \subseteq A} f(S),$$

el *principio de inclusión-exclusión* [15] dice que entonces

$$f(A) = \sum_{S \subseteq A} (-1)^{|A|-|S|} g(S), \tag{2.18}$$

donde $|\cdot|$ da la cardinalidad de un conjunto.

En una red con N nodos, se define un clúster α como un subconjunto de la red con n_α nodos. Entre ellos, por definición existe un orden parcial, donde si α está contenido en β, entonces $\alpha \leq \beta$. Este orden define un conjunto $\mathscr{P}$ parcialmente ordenado con todos los clústeres posibles de la red.

Corolario 6. *En el caso descrito, la función de Möbius es*

$$\mu(\alpha,\beta) = (-1)^{n_\beta - n_\alpha}, \quad \alpha,\beta \in \mathscr{P}.$$

Demostración. Para cualquier par de funciones reales f, g definidas en el conjunto ordenado $\mathscr{P}$, relacionadas mediante

$$f(\alpha) = \sum_{\beta \leq \alpha} g(\beta) = \sum_{\beta \leq \alpha} \zeta(\beta,\alpha) g(\beta)$$

donde $\alpha,\beta \in \mathscr{P}$ y (2.16). Entonces, como la función de Möbius se define como la inversa de ζ (2.17),

$$g(\alpha) = \sum_{\beta \leq \alpha} \mu(\beta,\alpha) f(\beta),$$

y comparando con el principio de inclusión-exclusión (2.18) dada la cardinalidad de cada clúster $|\alpha| = n_\alpha$ y $|\beta| = n_\beta$, queda probado el enunciado. □

2.3.2. Funcional de energía libre

Ahora, sea $\mathcal{S}_\alpha$ la entropía de un clúster α: nos proponemos expresarla utilizando cantidades definidas en clústeres $\beta \leq \alpha$. Para ello, sea $\tilde{\mathcal{S}}_\beta$ un cumulante entrópico para cualquier clúster β, la expansión en cumulantes de $\mathcal{S}_\alpha$ se expresa

$$\mathcal{S}_\alpha = \sum_{\beta \leq \alpha} \tilde{\mathcal{S}}_\beta. \tag{2.19}$$

Proposición 7. *Los cumulantes entrópicos se pueden expresar como*

$$\tilde{\mathcal{S}}_\beta = \sum_{\alpha \leq \beta} (-1)^{n_\alpha - n_\beta} \mathcal{S}_\alpha,$$

para cualquier β clúster.

Demostración. De (2.19), sabemos

$$\mathcal{S}_\alpha = \sum_{\beta \leq \alpha} \zeta(\alpha,\beta) \tilde{\mathcal{S}}_\beta,$$

entonces por la inversión de Möbius y el corolario 6

$$\tilde{\mathcal{S}}_\beta = \sum_{\alpha \leq \beta} \mu(\alpha,\beta) \mathcal{S}_\alpha = \sum_{\alpha \leq \beta} (-1)^{n_\alpha - n_\beta} \mathcal{S}_\alpha.$$

□

Tomando como clúster α todo el grafo, se obtiene la expansión en cumulantes de la entropía $\mathcal{S} = \sum_\beta \tilde{\mathcal{S}}_\beta$. Entonces, la energía libre se puede expresar de forma exacta como

$$\mathcal{F}[p] = \mathbb{E}E - \sum_\beta \tilde{\mathcal{S}}_\beta.$$

El método CVM consiste en aproximar el segundo término de esta aprovechando que los cumulantes $\tilde{\mathcal{S}}_\beta$ suelen anularse rápidamente para clústeres más grandes de la longitud de correlación [8].

Se considera un conjunto de clústeres $\mathscr{P} = \{\gamma_1, ..., \gamma_k\}$ no comparables entre sí, es decir, cumplen $\zeta(\gamma_i, \gamma_j) = 0$ para cualquier $i \neq j$. Los clústeres que componen $\mathscr{P}$ se llaman clústeres maximales. Este conjunto permite definir otro $\mathscr{P}' = \{\alpha \leq \gamma | \gamma \in \mathscr{P}\}$ parcialmente ordenado con $\leq$, en el que los clústeres más *grandes* son los maximales. Entonces, se realiza la aproximación de que la entropía

$$\mathcal{S} = \sum_\alpha \tilde{\mathcal{S}}_\alpha \approx \sum_{\alpha \in \mathscr{P}'} \tilde{\mathcal{S}}_\alpha, \tag{2.20}$$

donde se pasa de tener al total como único clúster no comparable (el más *grande*), a sumar sobre un conjunto en el que los clústeres maximales son los no comparables.

Corolario 8. *La aproximación de la entropía se puede expresar como*

$$\mathcal{S} \approx \sum_{\beta \in \mathscr{P}'} a_\beta \mathcal{S}_\beta,$$

donde $a_\beta = \sum_{\alpha \in \mathscr{P}'} (-1)^{n_\alpha - n_\beta} \zeta(\beta, \alpha)$ son los números de Möbius.

Demostración. Simplemente usando la proposición 7 y la definición (2.16)

$$\mathcal{S} \approx \sum_{\alpha \in \mathscr{P}'} \tilde{\mathcal{S}}_\alpha = \sum_{\alpha \in \mathscr{P}'} \sum_{\beta \leq \alpha} (-1)^{n_\alpha - n_\beta} \mathcal{S}_\beta = \sum_{\alpha \in \mathscr{P}'} \sum_\beta (-1)^{n_\alpha - n_\beta} \zeta(\beta, \alpha) \mathcal{S}_\beta.$$

□

Para clústeres como los maximales no es complicado el cálculo de los números de Möbius, pues para cualquier $\gamma \in \mathscr{P}$ está claro que $a_\gamma = \zeta(\gamma,\gamma) = 1$ porque en $\mathscr{P}'$ solo es comparable con sí mismo. Sin embargo, para el resto de clústeres en $\mathscr{P}'$ puede complicarse su cálculo mediante la definición.

Lema 9. *Los números de Möbius se pueden calcular mediante las reglas implícitas*

$$\sum_{\alpha\leq\beta\in\mathscr{P}'} a_\beta = 1, \quad \alpha \in \mathscr{P}'.$$

Demostración. De la definición de los números de Möbius en el corolario 8 y (2.17)

$$\sum_{\alpha\leq\beta\in\mathscr{P}'} a_\beta = \sum_{\gamma\in\mathscr{P}'}\sum_{\alpha\leq\beta\leq\gamma} \zeta(\alpha,\beta)\mu(\beta,\gamma) = \sum_{\gamma\in\mathscr{P}'} \delta(\alpha,\gamma) = 1.$$

□

Este lema indica que cada subclúster en $\mathscr{P}'$ ha de contarse una única vez en la expansión en cumulantes de la entropía.

Gracias a la hipótesis de factorización de las probabilidades de transición $\omega_i^{(t)}$ (2.3) y de las condiciones iniciales (2.4), por la que cada nodo i solo interacciona con sus primeros vecinos ∂i, el término de energía U se trata de forma exacta. Entonces, la aproximación a clústeres maximales que contienen un nodo i y sus vecinos ∂i no corta la información sobre interacciones entre nodos del modelo. Por ejemplo, si por el contrario las interacciones fueran a todos los N cuerpos, habría que usar $p(\mathbb{x})$ para expresar la energía U y cualquier elección de clúster maximal cortaría las interacciones del modelo. Finalmente, la expresión encontrada para la energía libre aproximada es

$$\mathcal{F}_{\mathscr{P}}[p_\alpha, \alpha \in \mathscr{P}'] = \sum_{\mathbb{x}\in\Gamma} p(\mathbb{x})E(\mathbb{x}) - \sum_{\alpha\in\mathscr{P}'} a_\alpha \mathcal{S}_\alpha.$$

La elección de los clústeres maximales en general reside en la intuición física, pues el truncamiento limita las interacciones que se tienen en cuenta a las que se dan entre elementos de los clústeres maximales. Como en los modelos que se tratan aquí las interacciones entre nodos vienen dadas por (2.3), los clústeres maximales deben abarcar nodos a lo largo de pasos temporales consecutivos y vecinos. Las posibles elecciones de clústeres maximales que cumplen este criterio se diferencian en el tiempo de ejecución y bondad de la aproximación.

En detalle, siguiendo las indicaciones de [2], los clústeres a lo largo de uno o dos pasos consecutivos, para un nodo y algunos vecinos suyos se muestran en el cuadro 2.1, junto a la letra del alfabeto latino que lo representa. La elección más simple que se toma es la de lo clústeres P, pues contiene exactamente a todos los elementos presentes en las factorizaciones de la interacción energética (2.3), entonces el método se denomina aproximación P. A este se le puede añadir los clústeres Q, que representan las aristas del sistema y su traslación al siguiente paso $t+1$, resultando la aproximación PQ. Esta aproximación se puede mejorar incluyendo los clústeres R, que también cumplen el criterio mencionado antes, dando lugar a la aproximación PQR. Todos estos clústeres están contenidos en al menos un clúster M, por lo que la aproximación más precisa de las presentadas es la M.

La aproximación PQR ya se ha estudiado anteriormente [2],[1], de ahí el interés en estudiar una alternativa más precisa pero con a priori mayores tiempos de ejecución como es la aproximación M.

Una vez realizada la aproximación CVM, hay que resolver el problema de minimizar el variacional de energía libre $\mathcal{F}$. Se trata de una minimización con restricciones de compatibilidad, pues se tienen que cumplir las condiciones de marginalización y normalización. En el caso del CVM, se suelen utilizar métodos iterativos del tipo *message passing* [16][8], como por ejemplo *generalized belief propagation*,

	t	t+1	Distribución marginal de probabilidad
M	$i, \partial i$	$i, \partial i$	$M_i^{(t)}(y_{i,\partial i}, x_{i,\partial i}) = \mathbb{P}\left\{X_{i,\partial i}^{(t+1)} = y_{i,\partial i}, X_{i,\partial i}^{(t)} = x_{i,\partial i}\right\}$
P	$i, \partial i$	i	$P_i^{(t)}(y_i, x_{i,\partial i}) = \mathbb{P}\left\{X_i^{(t+1)} = y_i, X_{i,\partial i}^{(t)} = x_{i,\partial i}\right\}$
Q	i, j	i, j	$Q_{\langle ij\rangle}^{(t)}\left(y_{i,j}, x_{i,j}\right) = \mathbb{P}\left\{X_{i,j}^{(t+1)} = y_{i,j}, X_{i,j}^{(t)} = x_{i,j}\right\}$
R	i	$i, \partial i$	$R_i^{(t)}(y_{i,\partial i}, x_i) = \mathbb{P}\left\{X_{i,\partial i}^{(t+1)} = y_{i,\partial i}, X_i^{(t)} = x_i\right\}$
S	$i, \partial i$	-	$S_i^{(t)}(x_{i,\partial i}) = \mathbb{P}\left\{X_{i,\partial i}^{(t)} = x_{i,\partial i}\right\}$
T	i, j	i	$T_{i,\langle ij\rangle}^{(t)}\left(y_i, x_{i,j}\right) = \mathbb{P}\left\{X_i^{(t+1)} = y_i, X_{i,j}^{(t)} = x_{i,j}\right\}$
U	i	i, j	$U_{\langle ij\rangle}^{(t)}\left(y_{i,j}, x_i\right) = \mathbb{P}\left\{X_{i,j}^{(t+1)} = y_{i,j}, X_i^{(t)} = x_i\right\}$
V	i	i	$V_i^{(t)}(y_i, x_i) = \mathbb{P}\left\{X_i^{(t+1)} = y_i, X_i^{(t)} = x_i\right\}$
Z	i, j	-	$Z_{\langle ij\rangle}^{(t)}\left(x_{i,j}\right) = \mathbb{P}\left\{X_{i,j}^{(t)} = x_{i,j}\right\}$
A	i	-	$A_i^{(t)}(x_i) = \mathbb{P}\left\{X_i^{(t)} = x_i\right\}$

Cuadro 2.1: En las primeras columnas se indican los nodos que componen cada tipo de clúster. En la última columna, las distribuciones marginales de probabilidad $p_\alpha(\mathbb{x}_\alpha)$ para cada clúster α. La notación $\langle ij\rangle$ denotan los pares de vecinos, sin importar el orden en que se tomen, pues la distribución de probabilidad es la misma. Es decir, $Z_{\langle ij\rangle}^{(t)}(x_{i,j}) = \mathbb{P}\{X_i^{(t)} = x_i, X_j^{(t)} = x_j\} = Z_{\langle ji\rangle}^{(t)}(x_{j,i})$. Además, $X_{i,\partial i}^{(t)}$ es una notación compacta de $X_i^{(t)}, \{X_j^{(t)}\}_{j\in\partial i}$.

warning propagation o *survey propagation*. Todos estos mecanismos son exactos cuando se trata de un modelo de tipo árbol, pero este no es el usual en biología. Además, cuando no pueden aprovechar ningún tipo de regularidad en el grafo del sistema no son tan rápidos y necesitan ser adaptados. También, para tener en cuenta toda la información del grafo, van actualizando toda fmp a cada paso, lo cual no debería ser necesario pues el paso de la información sigue un orden temporal.

Sin embargo, estos métodos son generales y no aprovechan al completo el modelo termodinámico y la aproximación. Dado que se trata de un modelo termodinámico espaciotemporal con la propiedad de Markov, se puede idear un procedimiento mucho más simple. Una secuencia natural de iteraciones, que parte de las condiciones iniciales del sistema y sigue la evolución temporal como se hizo para la aproximación PQR [2][1]. Ésta surge de la expresión de $\mathcal{F}$, sin tener que refinar la fmp obtenida mediante iteraciones sucesivas.

Capítulo 3 Desarrollo e implementación de la aproximación M

3.1. Números de Möbius

En la sección 2.3 se ha visto que la elección de los clústeres maximales determina la aproximación sobre la entropía que se realiza. La aproximación M consiste en escoger como clústeres maximales los $\mathscr{P} = \{\mathcal{M}_i^{(t)} = (y_{i,\partial i}, x_{i,\partial i})^{(t)} | i = 1,...,N; t = 0,...,\tau-1\}$, que al ser maximales tendrán número de Möbius $a_{\mathcal{M}} = 1$. Además, los números de Möbius de todos los subclústeres $\mathscr{P}' = \{\alpha \leq \beta | \beta \in \mathscr{P}\}$ se pueden calcular mediante el lema 9. La prueba completa se presenta en el apéndice C.

Proposición 10. *Los números de Möbius de los clústeres en $\mathscr{P}'$ son, para $t = 0,...,\tau-1$*

	$\mathcal{M}_i^{(t)}$	$\mathcal{P}_i^{(t)}$	$\mathcal{Q}_{\langle ij\rangle}^{(t)}$	$\mathcal{S}_i^{(t)}$	$\mathcal{T}_{i,\langle ij\rangle}^{(t)}$	$\mathcal{Z}_{\langle ij\rangle}^{(t)}$	$\mathcal{R}_i^{(t)}$	$\mathcal{U}_{\langle ij\rangle,i}^{(t)}$	$\mathcal{V}_i^{(t)}$	$\mathcal{A}_i^{(t)}$
a_β	*1*	*0*	*-1*	*-1 (0)*	*0*	*1 (0)*	*0*	*0*	*0*	*0*

y entre paréntesis los valores que cambian para $t = 0$.

Demostración. Partimos de los clústeres maximales $\mathcal{M}_i^{(t)} = (x_i, x_{\partial i}, y_i, x_{\partial i})^{(t)}$, con número $a_{\mathcal{M}} = 1$ y utilizando la regla del lema 9 se obtiene la tabla. Para los clústeres $\mathcal{Q}_{\langle ij\rangle}^{(t)} = (x_i, x_j, y_i, y_j)^{(t)}$

$$\mathcal{Q}_{\langle ij\rangle}^{(t)} \leq \mathcal{M}_i^{(t)}, \mathcal{M}_j^{(t)}, \quad \text{entonces } a_{\mathcal{Q}} = 1 - 2a_{\mathcal{M}} = -1.$$

Para los clústeres $\mathcal{S}_i^{(t)} = (x_i, x_{\partial i})^{(t)}$ en $t = 1,...,\tau-1$

$$\mathcal{S}_i^{(t)} \leq \mathcal{M}_i^{(t)}, \mathcal{M}_i^{(t-1)}, \quad \text{entonces } a_{\mathcal{S}} = 1 - 2a_{\mathcal{M}} = -1,$$

y en $t = 0$

$$\mathcal{S}_i^{(0)} \leq \mathcal{M}_i^{(0)}, \quad \text{entonces } a_{\mathcal{S}} = 1 - a_{\mathcal{M}} = 0.$$

Para los clústeres $\mathcal{Z}_{\langle ij\rangle}^{(t)} = (x_i, x_j)^{(t)}$ en $t = 1,...,\tau-1$

$$\mathcal{Z}_{\langle ij\rangle}^{(t)} \leq \mathcal{M}_i^{(t)}, \mathcal{M}_j^{(t)}, \mathcal{M}_i^{(t-1)}, \mathcal{M}_j^{(t-1)}, \mathcal{Q}_{\langle ij\rangle}^{(t)}, \mathcal{Q}_{\langle ij\rangle}^{(t-1)}, \mathcal{S}_i^{(t)}, \mathcal{S}_j^{(t)}, \quad \text{entonces } a_{\mathcal{Z}} = 1 - 4a_{\mathcal{M}} - 2a_{\mathcal{Q}} - 2a_{\mathcal{S}} = 1,$$

y en $t = 0$

$$\mathcal{Z}_{\langle ij\rangle}^{(0)} \leq \mathcal{M}_i^{(0)}, \mathcal{M}_j^{(0)}, \mathcal{Q}_{\langle ij\rangle}^{(0)}, \quad \text{entonces } a_{\mathcal{Z}} = 1 - 2a_{\mathcal{M}} - a_{\mathcal{Q}} = 0.$$

□

3.2. Restricciones de compatibilidad

Los vectores aleatorios definidos por las distribuciones de probabilidad de los clústeres ($P_i^{(t)}$, $S_i^{(t)}$, $Q_{\langle ij\rangle}^{(t)}$, $Z_{\langle ij\rangle}^{(t)}$, $T_{i,\langle ij\rangle}^{(t)}$, $M_i^{(t)}$) son claramente no independientes. Para cualquier clúster, las distribuciones de sus subclústeres se pueden obtener marginalizando sobre las variables descartadas. Se van a enumerar las relaciones que se utilizarán, para todo $t = 0,...,\tau-1$.

Para clústeres tipo M

$$\sum_{y_{\partial i}} M_i^{(t)}(y_{i,\partial i}, x_{i,\partial i}) = P_i^{(t)}(y_i, x_{i,\partial i}), \quad i = 1,...,N, \tag{3.1}$$

$$\sum_{y_{\partial i\setminus j}} \sum_{x_{\partial i\setminus j}} M_i^{(t)}(y_{i,\partial i}, x_{i,\partial i}) = Q_{\langle ij\rangle}^{(t)}\left(y_{i,j}, x_{i,j}\right), \quad i = 1,...,N,\ j \in \partial i, \tag{3.2}$$

$$\sum_{y_{i,\partial i}} \sum_{x_{\partial i\setminus j}} M_i^{(t)}(y_{i,\partial i}, x_{i,\partial i}) = Z_{\langle ij\rangle}^{(t)}\left(x_{i,j}\right), \quad i = 1,...,N,\ j \in \partial i, \tag{3.3}$$

$$\sum_{y_{\partial i}} \sum_{x_{\partial i\setminus j}} M_i^{(t)}(y_{i,\partial i}, x_{i,\partial i}) = T_{i,\langle ij\rangle}^{(t)}\left(y_i, x_{i,j}\right), \quad i = 1,...,N,\ j \in \partial i, \tag{3.4}$$

$$\sum_{y_{i,\partial i}} M_i^{(t)}(y_{i,\partial i}, x_{i,\partial i}) = S_i^{(t)}(x_{i,\partial i}), \quad i = 1,...,N, \tag{3.5}$$

$$\sum_{x_{i,\partial i}} M_i^{(t)}(y_{i,\partial i}, x_{i,\partial i}) = S_i^{(t+1)}(y_{i,\partial i}), \quad i = 1,...,N. \tag{3.6}$$

Para clústeres tipo Q

$$\sum_{y_{i,j}} Q_{\langle ij\rangle}^{(t)}\left(y_{i,j}, x_{i,j}\right) = Z_{\langle ij\rangle}^{(t)}\left(x_{i,j}\right), \quad \text{para todo par } \langle ij\rangle, \tag{3.7}$$

$$\sum_{x_{i,j}} Q_{\langle ij\rangle}^{(t)}\left(y_{i,j}, x_{i,j}\right) = Z_{\langle ij\rangle}^{(t+1)}\left(y_{i,j}\right), \quad \text{para todo par } \langle ij\rangle, \tag{3.8}$$

$$\sum_{y_j} Q_{\langle ij\rangle}^{(t)}\left(y_{i,j}, x_{i,j}\right) = T_{i,\langle ij\rangle}^{(t)}\left(y_i, x_{i,j}\right), \quad i = 1,...,N,\ j \in \partial i. \tag{3.9}$$

Para clústeres tipo P

$$\sum_{y_i} P_i^{(t)}(y_i, x_{i,\partial i}) = S_i^{(t)}(x_{i,\partial i}), \quad i = 1,...,N, \tag{3.10}$$

$$\sum_{x_{\partial i\setminus j}} P_i^{(t)}(y_i, x_{i,\partial i}) = T_{i,\langle ij\rangle}^{(t)}\left(y_i, x_{i,j}\right), \quad i = 1,...,N,\ j \in \partial i. \tag{3.11}$$

Y finalmente, para clústeres tipo T

$$\sum_{y_j} T_{i,\langle ij\rangle}^{(t)}\left(y_i, x_{i,j}\right) = Z_{\langle ij\rangle}^{(t)}\left(x_{i,j}\right), \quad i = 1,...,N,\ j \in \partial i. \tag{3.12}$$

Además, se asume que por construcción las probabilidades de transición, son distribuciones condicionadas de probabilidad, cumpliendo las condiciones de normalización

$$\sum_{y_i} \omega_i^{(t)}\left(y_i \middle| x_{i,\partial i}\right) = 1, \quad i = 1,...,N. \tag{3.13}$$

Todas las restricciones presentadas en esta sección se usan, no solo para el cálculo de distribuciones marginales, sino también para demostrar que la solución existe.

3.3. El funcional variacional

El objetivo es construir un algoritmo que permita obtener la distribución de probabilidad p del sistema que minimice la energía libre $\mathcal{F}_M$. Para ello, puede interesar buscar una expresión de $\mathcal{F}_M$ que se repita a lo largo del tiempo $t = 0,...,\tau$.

Empezamos por la expresión de la energía media (2.14), en término de las probabilidades de clúster.

Lema 11. *Se puede reescribir la energía interna del sistema como*

$$U = \sum_i \mathcal{S}[S_i^{(0)}] - \sum_{\langle ij\rangle} \mathcal{S}[Z_{\langle ij\rangle}^{(0)}] - \sum_{t=0}^{\tau-1} \sum_{y_{i,\partial i}, x_{i,\partial i}} M_i^{(t)}(y_{i,\partial i}, x_{i,\partial i}) \ln \omega_i^{(t)}\left(y_i \middle| x_{i,\partial i}\right).$$

Demostración. De la ecuación (2.14), sabemos que la energía interna del sistema viene dada por el valor esperado de $E(X^{(0)},...,X^{(\tau)}) = -\ln p^{(0)}(X^{(0)}) - \sum_{t=0}^{\tau-1} \ln \omega^{(t)}(X^{(t+1)}|X^{(t)})$. Para Γ el conjunto de todos los posibles estados del sistema,

$$U = \mathbb{E}E = \sum_{\mathbb{x}\in\Gamma} E(\mathbb{x}) p(\mathbb{x}) = -\sum_{x^{(0)}} \ln\left[p^{(0)}\left(x^{(0)}\right)\right] \sum_{\mathbb{x}\setminus x^{(0)}} p(\mathbb{x}) - \sum_{t=0}^{\tau-1} \sum_{x^{(t)}, x^{(t+1)}} \ln \omega^{(t)}\left(x^{(t+1)} \middle| x^{(t)}\right) \sum_{\mathbb{x}\setminus(x^{(t)}, x^{(t+1)})} p(\mathbb{x}),$$

donde se identifican claramente las distribuciones marginales $p(x^{(0)}) = \sum_{\mathbb{x}\setminus x^{(0)}} p(\mathbb{x})$ y $p(x^{(0)}, x^{(1)}) = \sum_{\mathbb{x}\setminus x^{(0)}, x^{(1)}} p(\mathbb{x})$.

Para el término de $p^{(0)}$, como las distribuciones de probabilidad iniciales son independientes $p^{(0)}(x) = \prod_i p_i^{(0)}(x_i)$, multiplicando y dividiendo por $\prod_{j\in\partial i} p_j^{(0)}(x_j)$ dentro del logaritmo, se puede separar en

$$-\sum_{x^{(0)}} \sum_i p\left(x^{(0)}\right) \ln p_i^{(0)}(x_i) = -\sum_i \sum_{x_{i,\partial i}} S_i^{(0)}(x_{i,\partial i}) \ln S_i^{(0)}(x_{i,\partial i}) + \sum_{x^{(0)}} p\left(x^{(0)}\right) \sum_i \left(\sum_{j\in\partial i} \ln p^{(0)}(x_j) \right),$$

donde se ha hecho la marginalización $S_i^{(0)}(x_{i,\partial i}) = \sum_{x^{(0)}\setminus x_{i,\partial i}} p^{(0)}(x^{(0)})$ y la ecuación (3.15) dentro del logaritmo, quedando $-\sum_i \sum_{x_{i,\partial i}} S_i^{(0)}(x_{i,\partial i}) \ln S_i^{(0)}(x_{i,\partial i}) = -\sum_i \mathcal{S}[S_i^{(0)}]$ en el primer término.

Además, se busca expresar el segundo término $\sum_{i,j\in\partial i} \ln p^{(0)}(x_j)$ con una suma sobre los pares $\langle ij\rangle$ de $\ln Z_{\langle ij\rangle}^{(0)}(x_{i,j})$, usando (3.16)

$$\prod_i \prod_{j\in\partial i} p_j^{(0)}(x_j) = \frac{1}{\prod_i p_i^{(0)}(x_i)^{|\partial i|}} \prod_i \prod_{j\in\partial i} p_i^{(0)}(x_i) p_j^{(0)}(x_j) = \frac{\prod_i \prod_{j\in\partial i} Z_{\langle ij\rangle}^{(0)}(x_{i,j})}{\prod_i p_i^{(0)}(x_i)^{|\partial i|}},$$

además como $Z_{\langle ij\rangle}^{(0)}(x_i, x_j) = p(x_i^{(0)}, x_j^{(0)}) = Z_{\langle ji\rangle}^{(0)}(x_j, x_i)$, entonces $\prod_i \prod_{j\in\partial i} Z_{\langle ij\rangle}^{(0)}(x_{i,j}) = \prod_{\langle ij\rangle} Z_{\langle ij\rangle}^{(0)}(x_{i,j})^2$ pues en grafos no dirigidos $\prod_i \prod_{j\in\partial i}$ recorre cada par $\langle ij\rangle$ en las dos direcciones posibles. También, en el denominador hay un factor por cada vecino para cada nodo, es decir, hay dos factores por cada arista $\langle ij\rangle$ y $\prod_i p_i^{(0)}(x_i)^{|\partial i|} = \prod_{\langle ij\rangle} p_i^{(0)}(x_i) p_j^{(0)}(x_j)$. De esta forma,

$$\sum_{x^{(0)}} p\left(x^{(0)}\right) \sum_i \left(\sum_{j\in\partial i} \ln p^{(0)}(x_j) \right) = \sum_{x^{(0)}} p\left(x^{(0)}\right) \ln \frac{\prod_{\langle ij\rangle} Z_{\langle ij\rangle}^{(0)}(x_{i,j})^2}{\prod_{\langle ij\rangle} Z_{\langle ij\rangle}^{(0)}(x_{i,j})} = \sum_{\langle ij\rangle} \sum_{x^{(0)}} p\left(x^{(0)}\right) \ln Z_{\langle ij\rangle}^{(0)}(x_{i,j}),$$

y por la marginalización a $Z_{\langle ij\rangle}^{(0)}(x_{i,j}) = \sum_{x^{(0)}\setminus x_{i,j}} p(x^{(0)})$, se llega a $-\sum_{\langle ij\rangle} \mathcal{S}[Z_{\langle ij\rangle}^{(0)}]$.

El término de las probabilidades de transición ω se maneja en su forma factorizada (2.3) sobre todos los sitios $i = 1,...,N$, y marginalizando a la distribución de clústeres tipo M

$$\sum_{t=0}^{\tau-1} \sum_{x^{(t)}, x^{(t+1)}} p\left(x^{(t)}, x^{(t+1)}\right) \ln \omega^{(t)}\left(x^{(t+1)} \middle| x^{(t)}\right) = \sum_{t=0}^{\tau-1} \sum_i \sum_{y_{i,\partial i}, x_{i,\partial i}} M_i^{(t)}(y_{i,\partial i}, x_{i,\partial i}) \ln \omega_i^{(t)}\left(y_i \middle| x_{i,\partial i}\right).$$

□

La entropía truncada $\mathcal{S}_M$, dados los números de Möbius de la proposición 10, toma la forma del corolario 8

$$\mathcal{S}_M[M,Q,S,Z] = \sum_{t=0}^{\tau-1}\left\{\sum_i \mathcal{S}[M_i^{(t)}] - \sum_{\langle ij\rangle}\mathcal{S}[Q_{\langle ij\rangle}^{(t)}]\right\} + \sum_{t=1}^{\tau-1}\left\{-\sum_i \mathcal{S}[S_i^{(t)}] + \sum_{\langle ij\rangle}\mathcal{S}[Z_{\langle ij\rangle}^{(t)}]\right\}. \tag{3.14}$$

Observamos que en la proposición 10 se ha visto que $\mathcal{S}_i^{(0)}$ y $Z_{\langle ij\rangle}^{(0)}$ tienen números de Möbius nulos, por lo que sus distribuciones no aparecen en la entropía $\mathcal{S}_M$. Estas, dada la independencia de las condiciones iniciales (2.4), toman las formas

$$S_i^{(0)}(x_{i,\partial i}) = p_i^{(0)}(x_i)\prod_{j\in\partial i} p_j^{(0)}(x_j), \tag{3.15}$$

y

$$Z_{\langle ij\rangle}^{(0)}(x_{i,j}) = p_i^{(0)}(x_i)p_j^{(0)}(x_j). \tag{3.16}$$

Sin embargo, se ha visto en el lema 11 que estas distribuciones sí que aparecen en la energía interna, lo que permite escribir la energía libre aproximada $\mathcal{F}_M[M,Q,S,Z] = U - \mathcal{S}_M[M,Q,S,Z]$ con una suma a partir del tiempo $t = 0$:

$$\begin{aligned}\mathcal{F}_M[M,Q,S,Z] = \sum_{t=0}^{\tau-1}\Bigg\{&-\sum_i \mathcal{S}[M_i^{(t)}] + \sum_{\langle ij\rangle}\mathcal{S}[Q_{\langle ij\rangle}^{(t)}] + \sum_i \mathcal{S}[S_i^{(t)}] - \sum_{\langle ij\rangle}\mathcal{S}[Z_{\langle ij\rangle}^{(t)}] \\ &- \sum_i \sum_{y_{i,\partial i}, x_{i,\partial i}} M_i^{(t)}(y_{i,\partial i}, x_{i,\partial i}) \ln \omega_i^{(t)}\left(y_i \middle| x_{i,\partial i}\right)\Bigg\}.\end{aligned} \tag{3.17}$$

Sin embargo, esta expresión no es la más cómoda para determinar su mínimo, ni para calcular las probabilidades de clúster que corresponden a ese mínimo.

Para obtener un algoritmo, se puede seguir un esquema similar al de la aproximación *PQR*. En primer lugar, como se dispone de las probabilidades de transición factorizadas (2.3), se pasa la información a clústeres P pues del paso $t+1$ contienen solo al estado y_i del nodo i. Después, como ya se tiene la información de cada nodo i por separado, se pueden juntar en pares de nodos $\langle ij\rangle$ en los clústeres Q. Finalmente, se puede hacer lo mismo para cada nodo i y sus vecinos ∂i obteniendo las distribuciones marginales de M.

Entonces, se busca llegar a una expresión de $\mathcal{F}_M$ de la que deducir un algoritmo que la minimice como el mencionado. Para ello, es necesario sumar y restar ciertos términos entrópicos a la entropía truncada $\mathcal{S}_M$ (3.14), quedando

$$\begin{aligned}\mathcal{S}_M[M,Q,S,Z] = \sum_{t=0}^{\tau-1}\Bigg\{&\sum_i\left[\mathcal{S}[M_i^{(t)}] + \mathcal{S}[P_i^{(t)}] - \mathcal{S}[P_i^{(t)}] + \sum_{j\in\partial i}\left\{\mathcal{S}[T_{j,\langle ij\rangle}^{(t)}] - \mathcal{S}[T_{j,\langle ij\rangle}^{(t)}]\right\}\right] \\ &+ \sum_{\langle ij\rangle}\left[-\mathcal{S}[Q_{\langle ij\rangle}^{(t)}] + \mathcal{S}[Z_{\langle ij\rangle}^{(t)}] - \mathcal{S}[Z_{\langle ij\rangle}^{(t)}]\right]\Bigg\} + \sum_{t=1}^{\tau-1}\left\{\sum_{\langle ij\rangle}\mathcal{S}[Z_{\langle ij\rangle}^{(t)}] - \sum_i \mathcal{S}[S_i^{(t)}]\right\}.\end{aligned} \tag{3.18}$$

Con esta expresión de $\mathcal{S}_M$ y (3.10), (3.7), (3.1), (3.4), (3.9) y (3.3) se llega a la forma de $\mathcal{F}_M$ buscada.

Proposición 12. *El funcional de energía libre M toma la forma*

$$\begin{aligned}\mathcal{F}_M[M,Q,S,Z] = &\sum_{t=0}^{\tau-1}\sum_i\sum_{y_i,x_{i,\partial i}} P_i^{(t)}(y_i,x_{i,\partial i})\ln\frac{P_i^{(t)}(y_i,x_{i,\partial i})}{\omega_i^{(t)}\left(y_i\middle|x_{i,\partial i}\right)S_i^{(t)}(x_{i,\partial i})}\\ &-\sum_{t=0}^{\tau-1}\sum_{\langle ij\rangle}\sum_{y_{i,j},x_{i,j}} Q_{\langle ij\rangle}^{(t)}\left(y_{i,j},x_{i,j}\right)\ln\frac{Q_{\langle ij\rangle}^{(t)}\left(y_{i,j},x_{i,j}\right)Z_{\langle ij\rangle}^{(t)}\left(x_{i,j}\right)}{T_{i,\langle ij\rangle}^{(t)}\left(y_i,x_{i,j}\right)T_{j,\langle ji\rangle}^{(t)}\left(y_j,x_{j,i}\right)}\\ &+\sum_{t=0}^{\tau-1}\sum_i\sum_{y_{i,\partial i},x_{i,\partial i}} M_i^{(t)}(y_{i,\partial i},x_{i,\partial i})\ln\frac{M_i^{(t)}(y_{i,\partial i},x_{i,\partial i})}{P_i^{(t)}(y_i,x_{i,\partial i})}\prod_{j\in\partial i}\frac{Z_{\langle ij\rangle}^{(t)}\left(x_{i,j}\right)}{T_{j,\langle ji\rangle}^{(t)}\left(y_j,x_{j,i}\right)}.\end{aligned}$$

Demostración. Antes de juntar los términos entrópicos en (3.18), hay que tratar algunos de ellos.

En primer lugar, dada la ecuación de marginalización (3.10)

$$-\sum_i \mathcal{S}[S_i^{(t)}] = -\sum_i\sum_{x_{i,\partial i}} S_i^{(t)}(x_{i,\partial i})\ln S_i^{(t)}(x_{i,\partial i}) = -\sum_i\sum_{y_i,x_{i,\partial i}} P_i^{(t)}(y_i,x_{i,\partial i})\ln S_i^{(t)}(x_{i,\partial i}),$$

y se procede de forma análoga para $-\sum_{\langle ij\rangle}\mathcal{S}[Z_{\langle ij\rangle}^{(t)}]$ y $-\sum_{\langle ij\rangle}\mathcal{S}[P_i^{(t)}]$, usando (3.7) y (3.1) respectivamente.

También procediendo de forma parecida y usando (3.4)

$$-\sum_{i,j\in\partial i}\mathcal{S}[T_{j,\langle ij\rangle}^{(t)}] = -\sum_{\mathbb{x}\in\Gamma}p(\mathbb{x})\sum_{i,j\in\partial i}\ln T_{j,\langle ij\rangle}^{(t)}(y_j,x_{j,i}) = -\sum_i\sum_{y_{i,\partial i},x_{i,\partial i}} M_i^{(t)}(y_{i,\partial i},x_{i,\partial i})\ln\prod_{j\in\partial i}T_{j,\langle ij\rangle}^{(t)}(y_j,x_{j,i}).$$

De forma análoga, usando que $\prod_i\prod_{j\in\partial i}T_{i,\langle ij\rangle}^{(t)} = \prod_{\langle ij\rangle}T_{i,\langle ij\rangle}^{(t)}T_{j,\langle ij\rangle}^{(t)}$ y (3.9)

$$\sum_{i,j\in\partial i}\mathcal{S}[T_{j,\langle ij\rangle}^{(t)}] = \sum_{\mathbb{x}\in\Gamma}p(\mathbb{x})\sum_{i,j\in\partial i}\ln T_{j,\langle ij\rangle}^{(t)}(y_j,x_{j,i}) = \sum_{\langle ij\rangle}\sum_{y_{i,j},x_{i,j}} Q_{\langle ij\rangle}^{(t)}(y_{i,j},x_{i,j})\ln T_{i,\langle ij\rangle}^{(t)}(y_i,x_{i,j})T_{j,\langle ij\rangle}^{(t)}(y_j,x_{j,i}).$$

Y por último, usando que $\prod_i\prod_{j\in\partial i}Z_{\langle ij\rangle}^{(t)} = \prod_{\langle ij\rangle}Z_{\langle ij\rangle}^{(t)}Z_{\langle ij\rangle}^{(t)}$ y (3.3), está claro que

$$\sum_{\langle ij\rangle}2\mathcal{S}[Z_{\langle ij\rangle}^{(t)}] = \sum_{\mathbb{x}\in\Gamma}p(\mathbb{x})\ln\prod_{\langle ij\rangle}\left(Z_{\langle ij\rangle}^{(t)}(x_{i,j})\right)^2 = \sum_i\sum_{y_{i,\partial i},x_{i,\partial i}} M_i^{(t)}(y_{i,\partial i},x_{i,\partial i})\ln\prod_{j\in\partial i}Z_{\langle ij\rangle}^{(t)}(x_{i,j}).$$

Juntando los términos de la entropía $\mathcal{S}_M[M,Q,S,Z]$ tratados con la expresión de U obtenida en el lema 11, se llega a la expresión de $\mathcal{F}_M$ buscada. □

Se puede probar que la expresión de la proposición 12 se expresa como suma de divergencias de Kullcak-Leiber (2.5)

$$\mathcal{F}_M = \sum_{t=0}^{\tau-1}\left[\sum_i D_{KL}\left(P_i^{(t)}\middle\|\omega_i^{(t)}S_i^{(t)}\right) - \sum_{\langle ij\rangle}D_{KL}\left(Q_{\langle ij\rangle}^{(t)}\middle\|\frac{T_{i,\langle ij\rangle}^{(t)}T_{j,\langle ji\rangle}^{(t)}}{Z_{\langle ij\rangle}^{(t)}}\right) + \sum_i D_{KL}\left(M_i^{(t)}\middle\|P_i^{(t)}\prod_{j\in\partial i}\frac{T_{j,\langle ji\rangle}^{(t)}}{Z_{\langle ij\rangle}^{(t)}}\right)\right]. \tag{3.19}$$

Sin embargo, uno de los términos D_{KL} va acompañado de un signo negativo. Esto hace que a priori no se pueda aprovechar la no negatividad de las divergencias Kullback-Leiber para encontrar el mínimo de la energía libre $\mathcal{F}_M$. Entonces, hay que buscar otra expresión equivalente en función de divergencias KL que se encuentren siempre sumando. Para ello, se busca conseguir un término de KL sumando y con la distribución $Q_{ij}^{(t)}$ en el numerador, por lo que habrá que sumar y restar términos a la entropía. Además, para puedan ser consideradas D_{KL}, habrá que introducir también distribuciones como $T_{i,\langle ij\rangle}^{(t)}$ para que se cumpla la condición de normalización.

Partiendo de (3.17), el funcional de energía libre se puede reescribir en dos términos, usando (3.2), (3.5), (3.7), (3.9) y sumando y restando los términos entrópicos $\sum_i\sum_{j\in\partial i}\mathcal{S}[T_{i,\langle ij\rangle}^{(t)}]$ y $\sum_{\langle ij\rangle}\mathcal{S}[Q_{\langle ij\rangle}^{(t)}]$.

Proposición 13. *El funcional de energía libre M toma la forma*

$$\mathcal{F}_M[M,Q,S,Z] = \sum_{t=0}^{\tau-1}\sum_i \sum_{y_{i,\partial i},x_{i,\partial i}} M_i^{(t)}(y_{i,\partial i},x_{i,\partial i}) \ln\left(\frac{M_i^{(t)}(y_{i,\partial i},x_{i,\partial i})}{\omega_i^{(t)}\left(y_i\middle|x_{i,\partial i}\right)S_i^{(t)}(x_{i,\partial i})}\prod_{j\in\partial i}\frac{T_{i,\langle ij\rangle}^{(t)}\left(y_i,x_{i,j}\right)}{Q_{\langle ij\rangle}^{(t)}\left(y_{i,j},x_{i,j}\right)}\right)$$
$$+\sum_{t=0}^{\tau-1}\sum_{\langle ij\rangle}\sum_{y_{i,j},x_{i,j}} Q_{\langle ij\rangle}^{(t)}\left(y_{i,j},x_{i,j}\right)\ln\frac{Q_{\langle ij\rangle}^{(t)}\left(y_{i,j},x_{i,j}\right)Z_{\langle ij\rangle}^{(t)}\left(x_{i,j}\right)}{T_{i,\langle ij\rangle}^{(t)}\left(y_i,x_{i,j}\right)T_{j,\langle ji\rangle}^{(t)}\left(y_j,x_{j,i}\right)}.$$

Demostración. En primer lugar, como $Q_{\langle ij\rangle}^{(t)}(y_{i,j},x_{i,j}) = Q_{\langle ji\rangle}^{(t)}(y_{j,i},x_{j,i})$ y $\sum_i\sum_{j\in\partial i}$ recorre cada par $\langle ij\rangle$ en las dos direcciones posibles, entonces $\sum_i\sum_{j\in\partial i}\mathcal{S}[Q_{\langle ij\rangle}^{(t)}] = 2\sum_{\langle ij\rangle}\mathcal{S}[Q_{\langle ij\rangle}^{(t)}]$ pues cada elemento se suma dos veces. Sumando y restando los términos entrópicos $\sum_{j\in\partial i}\mathcal{S}[T_{i,\langle ij\rangle}^{(t)}]$ y $\sum_{\langle ij\rangle}\mathcal{S}[Q_{\langle ij\rangle}^{(t)}]$ se llega a una expresión de $\mathcal{S}_M$ equivalente a (3.14)

$$\mathcal{F}_M = \sum_{t=0}^{\tau-1}\left\{\sum_i\left(\mathcal{S}[S_i^{(t)}] - \mathcal{S}[M_i^{(t)}] - \sum_{y_{i,\partial i},x_{i,\partial i}} M_i^{(t)}(y_{i,\partial i},x_{i,\partial i})\ln\omega_i^{(t)}\left(y_i\middle|x_{i,\partial i}\right) - \sum_{j\in\partial i}\left(\mathcal{S}[T_{i,\langle ij\rangle}^{(t)}] - \mathcal{S}[Q_{\langle ij\rangle}^{(t)}]\right)\right)\right.$$
$$\left. - \sum_{\langle ij\rangle}\left(\mathcal{S}[Q_{\langle ij\rangle}^{(t)}] + \mathcal{S}[Z_{\langle ij\rangle}^{(t)}]\right) + \sum_i\sum_{j\in\partial i}\mathcal{S}[T_{i,\langle ij\rangle}^{(t)}]\right\}. \tag{3.20}$$

En primer lugar, en la ecuación (3.20) los primeros términos en los que se suma sobre $\sum_i$ se pueden agrupar usando simplemente la marginalización (3.5)

$$\mathcal{S}[S_i^{(t)}] - \mathcal{S}[M_i^{(t)}] - \sum_{y_{i,\partial i},x_{i,\partial i}} M_i^{(t)}(y_{i,\partial i},x_{i,\partial i})\ln\omega_i^{(t)}\left(y_i\middle|x_{i,\partial i}\right) = \sum_{y_{i,\partial i},x_{i,\partial i}} M_i^{(t)}(y_{i,\partial i},x_{i,\partial i})\ln\frac{M_i^{(t)}(y_{i,\partial i},x_{i,\partial i})}{\omega_i^{(t)}\left(y_i\middle|x_{i,\partial i}\right)S_i^{(t)}(x_{i,\partial i})},$$

para todo $i = 1,...,N$. Y para los que además se suma sobre $\sum_{j\in\partial i}$, usando primero la marginalización (3.9) y después (3.2) se tiene

$$\sum_{j\in\partial i}\mathcal{S}[T_{i,ij}^{(t)}] - \mathcal{S}[Q_{ij}^{(t)}] = \sum_{j\in\partial i}\sum_{y_{i,j},x_{i,j}} Q_{ij}^{(t)}(y_{i,j},x_{i,j})\ln\frac{T_{i,\langle ij\rangle}^{(t)}(y_i,x_{i,j})}{Q_{\langle ij\rangle}^{(t)}(y_{i,j},x_{i,j})} = \sum_{y_{i,\partial i},x_{i,\partial i}} M_i^{(t)}(y_{i,\partial i},x_{i,\partial i})\ln\prod_{j\in\partial i}\frac{T_{i,\langle ij\rangle}^{(t)}(y_i,x_{i,j})}{Q_{\langle ij\rangle}^{(t)}(y_{i,j},x_{i,j})},$$

para todo $j\in\partial i$ y todo $i = 1,...,N$. Juntando esta última expresión con los términos anteriores se llega a la primera línea del enunciado.

En segundo lugar, usando que $\prod_i\prod_{j\in\partial i}T_{i,\langle ij\rangle}^{(t)} = \prod_{\langle ij\rangle}T_{i,\langle ij\rangle}^{(t)}T_{j,\langle ij\rangle}^{(t)}$ y (3.9), se tiene

$$\sum_{i,j\in\partial i}\mathcal{S}[T_{j,\langle ij\rangle}^{(t)}] = \sum_{\mathbb{x}\in\Gamma}p(\mathbb{x})\sum_{i,j\in\partial i}\ln T_{j,\langle ij\rangle}^{(t)}(y_j,x_{j,i}) = \sum_{\langle ij\rangle}\sum_{y_{i,j},x_{i,j}} Q_{\langle ij\rangle}^{(t)}(y_{i,j},x_{i,j})\ln T_{i,\langle ij\rangle}^{(t)}(y_i,x_{i,j})T_{j,\langle ij\rangle}^{(t)}(y_j,x_{j,i}).$$

Juntando este término el resto que se suman sobre $\sum_{\langle ij\rangle}$ y usando la marginalización (3.7) de forma que

$$\mathcal{S}[Z_{\langle ij\rangle}^{(t)}] = \sum_{x_{i,j}} Z_{\langle ij\rangle}^{(t)}(x_{i,j})\ln Z_{\langle ij\rangle}^{(t)}(x_{i,j}) = \sum_{y_{i,j},x_{i,j}} Q_{\langle ij\rangle}^{(t)}(y_{i,j},x_{i,j})\ln Z_{\langle ij\rangle}^{(t)}(x_{i,j}),$$

para todo par $\langle ij\rangle$, se llega a la segunda línea del enunciado. □

Se puede probar que la expresión obtenida en la proposición 13, que tiene la forma de divergencias de Kullback-Leiber, se puede expresar como tal

$$\mathcal{F}_M[M,Q,S,Z] = \sum_{t=0}^{\tau-1}\left\{\sum_i D_{KL}\left(M_i^{(t)}\middle\|\omega_i^{(t)}S_i^{(t)}\prod_{j\in\partial i}\frac{Q_{\langle ij\rangle}^{(t)}}{T_{i,\langle ij\rangle}^{(t)}}\right) + \sum_{\langle ij\rangle}D_{KL}\left(Q_{\langle ij\rangle}^{(t)}\middle\|\frac{T_{i,\langle ij\rangle}^{(t)}T_{j,\langle ij\rangle}^{(t)}}{Z_{\langle ij\rangle}^{(t)}}\right)\right\}. \tag{3.21}$$

Esta es una expresión equivalente a la dada en (3.19), pues provienen de la misma $\mathcal{F}_M$ y no se ha realizado ninguna aproximación adicional. Sin embargo, no se puede usar esta expresión directamente en un algoritmo de minimización de $\mathcal{F}_M$, pues en cada paso temporal la información de $Z^{(t)}_{\langle ij\rangle}$, $S^{(t)}_i$ y $\omega^{(t)}_i$, habría que pasarla a $Q^{(t)}_{\langle ij\rangle}$ y $M^{(t)}_i$ simultáneamente.

Corolario 14. *El funcional $\mathcal{F}_M$ es no negativo y se minimiza para las condiciones en $t = 0, ..., \tau - 1$*

I) $M^{(t)}_i(y_{i,\partial i}, x_{i,\partial i}) = \omega^{(t)}_i\left(y_i \middle| x_{i,\partial i}\right) S^{(t)}_i(x_{i,\partial i}) \prod_{j\in\partial i} \frac{Q^{(t)}_{\langle ij\rangle}(y_{i,j}, x_{i,j})}{T^{(t)}_{i,\langle ji\rangle}(y_i, x_{i,j})}, \quad i = 1, ..., N.$

II) $Q^{(t)}_{\langle ij\rangle}\left(y_{i,j}, x_{i,j}\right) = \frac{T^{(t)}_{i,\langle ij\rangle}(y_i, x_{i,j}) T^{(t)}_{j,\langle ji\rangle}(y_j, x_{j,i})}{Z^{(t)}_{\langle ij\rangle}(x_{i,j})}, \quad$ *para todo par* $\langle ij\rangle$.

Demostración. Partiendo de la expresión (3.21) en función de divergencias KL, está claro que $\mathcal{F}_M \geq 0$ pues todas las divergencias KL son no negativas por definición. Además, se alcanza el mínimo $\mathcal{F}_M = 0$ si y sólo si todos los términos se anulan, que por la propiedad de las divergencias (2.5) se da si y sólo si se cumplen las dos identidades del enunciado.

Queda solo probar la expresión (3.21). Para ello, primero se prueba que $\omega^{(t)}_i S^{(t)}_i \prod_{j\in\partial i} Q^{(t)}_{\langle ij\rangle} / T^{(t)}_{i,\langle ij\rangle}$ y $T^{(t)}_{i,\langle ij\rangle} T^{(t)}_{j,\langle ij\rangle} / Z^{(t)}_{\langle ij\rangle}$ son distribuciones de probabilidad. Es decir, que están normalizadas. En primer lugar, usando (3.9)

$$\sum_{y_{\partial i}} \prod_{j\in\partial i} \frac{Q^{(t)}_{\langle ij\rangle}(y_{i,j}, x_{i,j})}{T^{(t)}_{i,\langle ij\rangle}(y_i, x_{i,j})} = \prod_{j\in\partial i} \frac{\sum_{y_j} Q^{(t)}_{\langle ij\rangle}(y_{i,j}, x_{i,j})}{T^{(t)}_{i,\langle ij\rangle}(y_i, x_{i,j})} = 1,$$

y usando también (3.13)

$$\sum_{y_{i,\partial i}, x_{i,\partial i}} \omega^{(t)}_i(y_i|x_{i,\partial i}) S^{(t)}_i(x_{i,\partial i}) \prod_{j\in\partial i} \frac{Q^{(t)}_{\langle ij\rangle}(y_{i,j}, x_{i,j})}{T^{(t)}_{i,\langle ij\rangle}(y_i, x_{i,j})} = \sum_{x_{i,\partial i}} \Big(\sum_{y_i} \omega^{(t)}_i(y_i|x_{i,\partial i})\Big) S^{(t)}_i(x_{i,\partial i}) = 1,$$

para todo $i = 1, ..., N$ y $t = 0, ..., \tau - 1$. En segundo lugar, de (3.12)

$$\sum_{y_{i,j}, x_{i,j}} \frac{T^{(t)}_{i,\langle ij\rangle}\left(y_i, x_{i,j}\right) T^{(t)}_{j,\langle ji\rangle}\left(y_j, x_{j,i}\right),}{Z^{(t)}_{\langle ij\rangle}\left(x_{i,j}\right)} = \sum_{x_{i,j}} \frac{\sum_{y_i} T^{(t)}_{i,\langle ij\rangle}\left(y_i, x_{i,j}\right) \sum_{y_j} T^{(t)}_{j,\langle ji\rangle}\left(y_j, x_{j,i}\right)}{Z^{(t)}_{\langle ij\rangle}\left(x_{i,j}\right)} = \sum_{y_j, x_{j,i}} T^{(t)}_{j,\langle ji\rangle}\left(y_j, x_{j,i}\right) = 1,$$

para todo par $\langle ij\rangle$. Y como $Q^{(t)}_{\langle ij\rangle}$ y $M^{(t)}_i$ son distribuciones de probabilidad, ya se ha probado la normalización de todos los términos.

Finalmente, falta mencionar que en I) cuando una $T^{(t)}_{i,\langle ij\rangle}(y_i, x_i, j)$ se anula, por (3.9) también lo hace la probabilidad $Q^{(t)}_{\langle ij\rangle}(y_{i,j}, x_{i,j})$ correspondiente. En II) se razona análogamente con (3.12). □

A partir del corolario 14, ya estamos en condiciones de minimizar la expresión (3.19) de $\mathcal{F}_M$, obteniendo las identidades que la minimizan y forman parte del algoritmo.

Corolario 15. *El funcional de energía libre se minimiza para las condiciones en $t = 0, ..., \tau - 1$*

I) $P^{(t)}_i(y_i, x_{i,\partial i}) = \omega^{(t)}_i\left(y_i \middle| x_{i,\partial i}\right) S^{(t)}_i(x_{i,\partial i}), \quad i = 1, ..., N,$

II) $Q^{(t)}_{\langle ij\rangle}\left(y_{i,j}, x_{i,j}\right) = \frac{T^{(t)}_{i,\langle ij\rangle}(y_i, x_{i,j}) T^{(t)}_{j,\langle ji\rangle}(y_j, x_{j,i}),}{Z^{(t)}_{\langle ij\rangle}(x_{i,j})}, \quad$ *para todo par* $\langle ij\rangle$,

III) $M^{(t)}_i(y_{i,\partial i}, x_{i,\partial i}) = P^{(t)}_i(y_i, x_{i,\partial i}) \prod_{j\in\partial i} \frac{T^{(t)}_{j,\langle ji\rangle}(y_j, x_{j,i})}{Z^{(t)}_{\langle ij\rangle}(x_{i,j})}, \quad i = 1, ..., N.$

Demostración. Partiendo de la expresión (3.19) y teniendo en cuenta que $\mathcal{F}_M \geq 0$ por el corolario 14, el mínimo se dará en el 0. Para ello, se tendrá que anular la suma de las divergencias Kullback-Leiber.

Sin embargo, la ecuación II) en el corolario 14 implica que en $\mathcal{F}_M = 0$ se anulan todos los términos de divergencia $D_{KL}(Q^{(t)}_{\langle ij\rangle} \| T^{(t)}_{i,\langle ij\rangle} T^{(t)}_{j,\langle ji\rangle} / Z^{(t)}_{\langle ij\rangle})$ en la ecuación (3.19) y los términos restantes son positivos por definición de divergencias KL, alcanzando el 0 en las identidades del enunciado de este corolario.

Queda solo probar que es cierta la expresión (3.19). Para ello, primero probamos que las distribuciones KL están normalizadas. De (3.13) se deduce

$$\sum_{y_i, x_{i,\partial i}} \omega_i^{(t)}\left(y_i \middle| x_{i,\partial i}\right) S_i^{(t)}(x_{i,\partial i}) = \sum_{x_{i,\partial i}} S_i^{(t)}(x_{i,\partial i}) \left[\sum_{y_i} \omega_i^{(t)}\left(y_i \middle| x_{i,\partial i}\right)\right] = 1, \quad i = 1, ..., N.$$

También, de (3.12)

$$\sum_{y_{i,j}, x_{i,j}} \frac{T^{(t)}_{i,\langle ij\rangle}\left(y_i, x_{i,j}\right) T^{(t)}_{j,\langle ji\rangle}\left(y_j, x_{j,i}\right),}{Z^{(t)}_{\langle ij\rangle}\left(x_{i,j}\right)} = \sum_{x_{i,j}} \frac{\sum_{y_i} T^{(t)}_{i,\langle ij\rangle}\left(y_i, x_{i,j}\right) \sum_{y_j} T^{(t)}_{j,\langle ji\rangle}\left(y_j, x_{j,i}\right)}{Z^{(t)}_{\langle ij\rangle}\left(x_{i,j}\right)} = \sum_{y_j, x_{j,i}} T^{(t)}_{j,\langle ji\rangle}\left(y_j, x_{j,i}\right) = 1,$$

para todo par $\langle ij\rangle$. Y finalmente, de (3.12)

$$\sum_{y_{i,\partial i}, x_{i,\partial i}} P_i^{(t)}(y_i, x_{i,\partial i}) \prod_{j \in \partial i} \frac{T^{(t)}_{j,\langle ji\rangle}\left(y_j, x_{j,i}\right)}{Z^{(t)}_{\langle ij\rangle}\left(x_{i,j}\right)} = \sum_{y_i, x_{i,\partial i}} P_i^{(t)}(y_i, x_{i,\partial i}) \prod_{j \in \partial i} \frac{\sum_{y_j} T^{(t)}_{j,\langle ji\rangle}\left(y_j, x_{j,i}\right)}{Z^{(t)}_{\langle ij\rangle}\left(x_{i,j}\right)} = \sum_{y_i, x_{i,\partial i}} P_i^{(t)}(y_i, x_{i,\partial i}) = 1,$$

para $i = 0, ..., N$.

En segundo lugar, por la condición (3.12), en los estados en que $Z^{(t)}_{\langle ij\rangle}(x_{i,j})$ se anula, son idénticamente nulas $T^{(t)}_{i,\langle ij\rangle}(y_i, x_{i,j})$ y $T^{(t)}_{j,\langle ji\rangle}(y_j, x_{j,i})$ para todo valor de $y_{i,j}$. Además, de forma análoga por (3.10), (3.9), (3.1) y (3.4), si se anula alguna distribución en los denominadores de los logaritmos de la proposición 12, se anula también su numerador. □

3.4. Algoritmo

En esta sección se describe el proceso iterativo que resuelve el problema de encontrar la distribución de probabilidad de cada clúster y la entropía del sistema a cada paso de tiempo. Los pasos del proceso se encuentran resumidos en la figura 3.1.

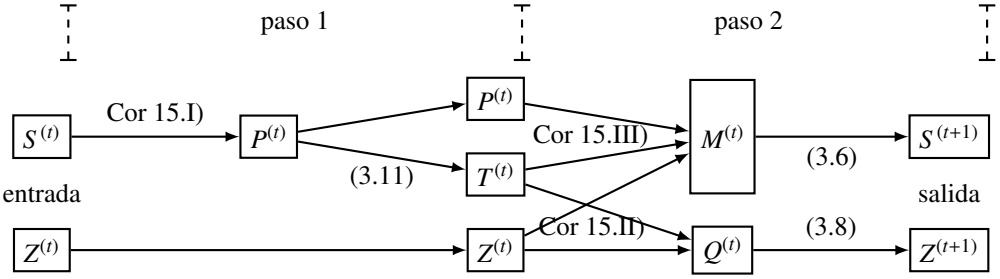

Figura 3.1: Esquema de flujo en un paso temporal del proceso computacional. Las cajas representan las distribuciones de probabilidad de los distintos clústeres que intervienen en el funcional de energía libre. Siendo las grandes para los clústeres maximales y las pequeñas para sus subclústeres. Las flechas pueden representar ecuaciones de minimización de $\mathcal{F}_M$ o de marginalización.

A cada paso de tiempo es necesario normalizar las distribuciones de probabilidad de cada clúster S y Z, pues dejan de estarlo por error de redondeo. Es necesario para que las distribuciones sigan siendo *buenas* funciones de distribución, en el sentido de que el problema esté bien planteado.

Capítulo 4 Aplicación a las redes de regulación

Nos proponemos ahora aplicar la aproximación M a los ejemplos de redes de regulación mencionados en la introducción, con el fin de averiguar si mejora los resultados de la PQR (utilizando los resultados de MaBoss como referencia).

Al igual que se hizo en [1], se empieza comparando el comportamiento del método con el entorno *MaBoSS 2.0* en el modelo de juguete propuesto. Para hacerlo, se necesitan las distribuciones $A_i^{(t)}(1)$, definida en el cuadro 2.1, y la entropía en cada ventana temporal

$$H(t) = \sum_{\langle ij \rangle} \mathcal{S}[Z_{\langle ij \rangle}^{(t)}] - \sum_i \mathcal{S}[S_i^{(t)}],$$

sin tener en cuenta otros pasos temporales. Al igual que se señaló en el trabajo anterior [1], las distribuciones de transición $\omega_i^{(t)}$ se relacionan con los ritmos de *MaBoSS* $\rho_{\vec{S}\to\vec{S}'}$ mediante

$$\omega_i\left(x_i^{t+1}|x_{i,\partial i}^t\right) = \rho_{x_i\to y_i}\tau\left(1-\delta_{x_i,y_i}\right) + \left(1-\sum_{k\neq x_i}\rho_{x_i\to k}\tau\right)\delta_{x_i,y_i} \tag{4.1}$$

Donde δ_{x_i,y_i} es la delta de Kronecker. El parámetro τ hace el papel de ventana temporal para el *CVM*, marcando el paso de una escala temporal discreta al límite de procesos de Markov continuos en $\tau \to 0$.

Se han representado los resultados de MaBoSS y la aproximación M para las distintas posibles trayectorias. En la trayectoria determinista de la figura 4.1, coinciden perfectamente los tres métodos. Este es un resultado trivial desde el punto de vista físico, útil para comprobar el correcto funcionamiento. En los bucles con escape lento y rápido de la figura 4.2, se observa cualitativamente el comportamiento esperado que se había comentado en el apartado 1.1, similar entre las aproximaciones. Sin embargo, los resultados del CVM y de MaBoSS son distintos en el resto, estando más alejada la aproximación M que la PQR.

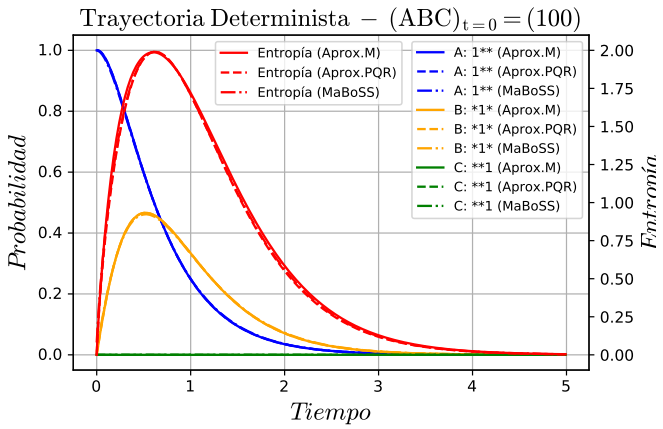

Figura 4.1: se puede observar claramente cómo va aumentando la probabilidad de B, baja la probabilidad de A y vuelve a bajar la de B. Esto sucede porque cuando se activa B llegando al estado $ABC \equiv 110$, pasa seguidamente al 010 y finalmente al 000, completando así la trayectoria.

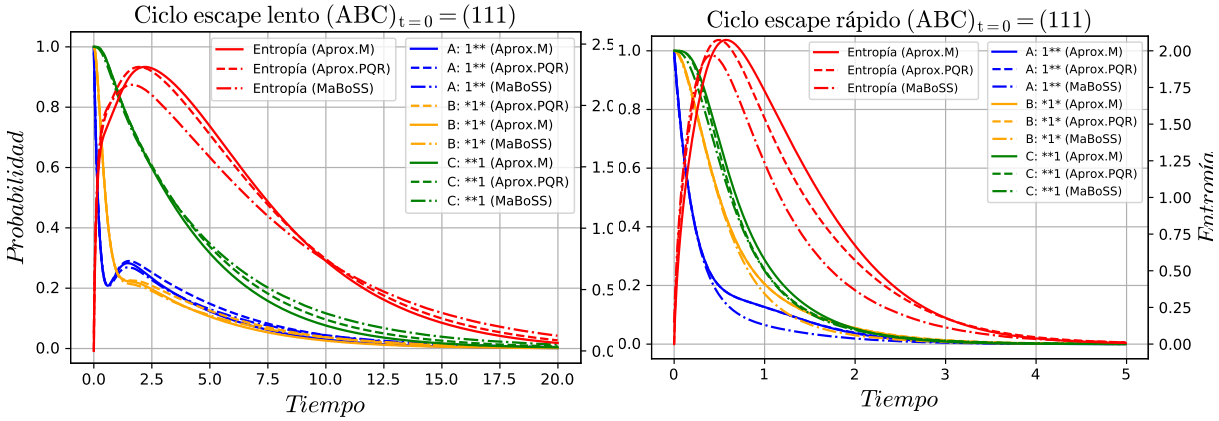

Figura 4.2: Trayectorias a partir de las condiciones iniciales 111 para las dinámicas de escape lento y rápido, como se detallan en la figura 1.1. La primera probabilidad en bajar es la del nodo A, seguida del B, lo que indica que está pasando del estado $ABC \equiv 111$ al 011 y al 001. En el escape rápido se notan los efectos del decaimiento de C rápidamente. Conforme el sistema llega al estado $ABC \equiv 001$, decae al atractor 000. En el escape lento, como el nodo C decae lentamente, aparece un pico en la probabilidad de A, el sistema sigue el bucle pasando al estado $ABC \equiv 100$. Después de este pico, las probabilidades de A y B empiezan a descender por el efecto del escape al estado 000, que evita el carácter asintótico del bucle en las probabilidades.

Se van a aplicar los dos métodos al modelo de ciclo celular expuesto en el apartado 1.1. Se observa perfectamente cómo tanto las probabilidades de las ciclinas como la entropía H son muy similares en ambas aproximaciones.

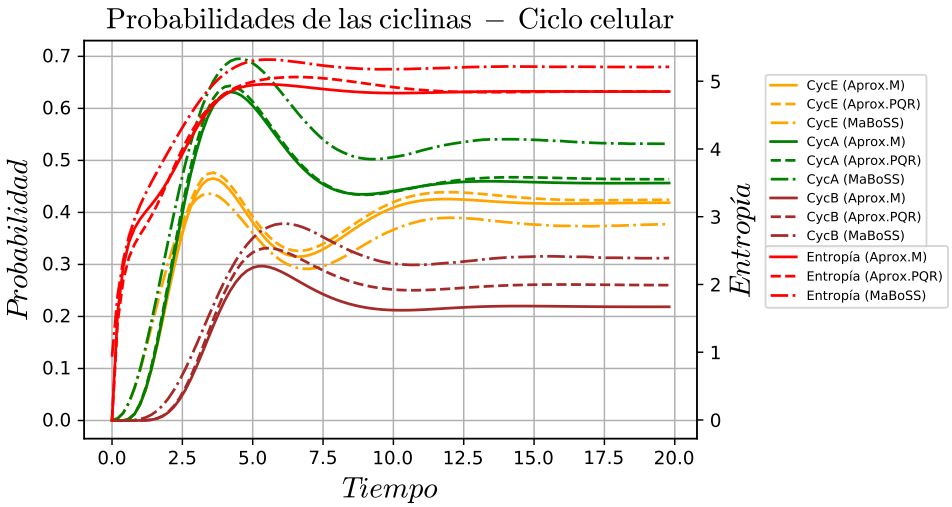

Figura 4.3: Los primeros picos de probabilidad en las ciclinas coinciden con las fases del ciclo celular. El primer pico en *CycE* coincide con la fase S, después viene la fase G2, para la que aparece un pico en *CycA*. En último lugar, al llegar a la mitosis aparece un pico en *CycB*. Tras el primer ciclo, se observa una oscilación amortiguada correspondiente a la desincronización de los ciclos celulares de las células hijas. Además, la ciclina *CycD* se encuentra siempre activa pues en ningún momento se ha eliminado la señal extracelular que desencadena el ciclo celular. Finalmente, tanto las probabilidades como la entropía H tienen carácter asintótico, que corresponde a un atractor cíclico.

Capítulo 5 Conclusiones

En un trabajo anterior [1] se consiguió aplicar por primera vez un método de Clúster Variacional a redes de regulación genética. Desde una perspectiva del campo de la biofísica, ese trabajo contribuía a ampliar el repertorio de métodos teóricos disponibles para aplicaciones biológicas. En esta ocasión, se ha conseguido repetir esto presentando e implementando otra aproximación variacional en clúster. Diferentemente del caso anterior, donde el foco estaba en la implementación de un algoritmo completamente detallado en literatura [2], ahora una primera parte del trabajo ha sido desarrollar el algoritmo de la aproximación M, solo esbozado en [2], derivando todos los pasos y demostrando rigurosamente la validez de su planteamiento. Luego se ha procedido a su aplicación a dos casos de redes de regulación genética: el primero, un sencillo modelo de juguete de tres nodos, que ha servido para ajustar la elección de los parámetros y comprobar la validez del código, y luego un modelo de ciclo celular de 10 nodos, costruido a partir del modelo de GINsim [3]. Obviamente, queda también allanado el camino de la aplicación de esta aproximación M a redes más grandes y complejas en el futuro.

Los resultados relativos a la aplicación de la aproximación M a los ejemplos analizados no son los deseados: esa aproximación en principio es más precisa (y costosa computacionalmente) de la PQR anterior; además, en [2], los autores destacan como, en presencia de tiempo continuo, la PQR degenera en otra aproximación de menor precisión. Aunque también en los resultados de [2], la aproximación M no destaca especialmente sobre la PQR, el hecho de tener que compararnos con simulaciones a tiempo continuo nos hacía esperar que, para nuestro caso de redes de regulación, la aproximación M pudiera resultar significativamente mejor. Sin embargo, las gráficas obtenidas son muy parecidas a las de PQR, cuando no peores en su comparación a los resultados de las simulaciones con el software MaBoss, que usamos como referencia. Hemos comprobado que esto no depende de la elección en la discretización del paso temporal, de acuerdo con la fórmula (4.1): el uso de un paso temporal más pequeño, en línea con la naturaleza continua del tiempo en las simulaciones de MaBoss, introduce oscilaciones (que también se daban en la aproximación PQR), sin que esto implique un mejor acuerdo con las simulaciones.

Es relevante subrayar que los resultados de M y PQR no son idénticos, lo que excluye que, por alguna razón accidental relacionada con las características de las dos redes consideradas, la aproximación M se reduzca trivialmente a la PQR para esos casos.

Por lo tanto, las razones de la falta de mejora de los resultados con la aproximación M no parece ser trivial, y debe estar relacionada con la estructura de las $\omega_i^{(t)}$ en estos problemas. Por lo tanto, una posible continuación de este trabajo sería tratar de entender la relación de las probabilidades entre los clústeres maximales de las dos aproximaciones, en ejemplos sencillos que se puedan estudiar exhaustivamente, mientras que para ejemplos más complejos, parece justificado seguir utilizando la aproximación PQR.

Bibliografía

[1] Pablo Pérez Lázaro, Pierpaolo Bruscolini y Joaquín Sanz Remón. "Estudios de redes de regulación genéticas con modelos discretos". En: (2021). [Unpublished manuscript].

[2] A Pelizzola y M Pretti. "Variational approximations for stochastic dynamics on graphs". En: *Journal of Statistical Mechanics: Theory and Experiment* 2017.7 (jul. de 2017), pág. 073406.

[3] G. Stoll y col. "Continuous time Boolean modeling for biological signaling: application of Gillespie algorithm". En: *BMC Syst Biol* 6 (ago. de 2012), pág. 116.

[4] "The cell cycle: An introduction, by Andrew Murray and Tim Hunt, W.H. Freeman&co., New York, distributed by oxford university press, 1993, 251pp, $22.95". En: *Molecular Reproduction and Development* 39.2 (1994), págs. 247-247.

[5] Wikimedia Commons. *Cyclin expression cycle*. 2011. URL: `https://en.wikipedia.org/wiki/File:Cyclin_Expression.svg`.

[6] Ryoichi Kikuchi. "A Theory of Cooperative Phenomena". En: *Phys. Rev.* 81 (6 mar. de 1951), págs. 988-1003.

[7] Guozhong An. "A note on the cluster variation method". En: *Journal of Statistical Physics* 52.3-4 (ago. de 1988), págs. 727-734.

[8] Alessandro Pelizzola. "Cluster variation method in statistical physics and probabilistic graphical models". En: *Journal of Physics A: Mathematical and General* 38.33 (ago. de 2005), R309-R339.

[9] G. Grimmett y col. *Probability and Random Processes*. Probability and Random Processes. OUP Oxford, 2001. ISBN: 9780198572220.

[10] D.J.C. MacKay y col. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003. ISBN: 9780521642989.

[11] P. Bremaud. *An Introduction to Probabilistic Modeling*. Springer Undergraduate Texts in Mathematics and Technology. Springer New York, 1988. ISBN: 9780387964607.

[12] H.B. Callen. *Thermodynamics and an Introduction to Thermostatistics*. Wiley, 1991.

[13] C. E. Shannon. "A mathematical theory of communication". En: *The Bell System Technical Journal* 27.3 (1948), págs. 379-423.

[14] Gian -Carlo Rota. "On the foundations of combinatorial theory I. Theory of Möbius Functions". En: *Zeitschrift fur Wahrscheinlichkeitstheorie und Verwandte Gebiete* 2.4 (1964), págs. 340-368.

[15] R.L. Graham, M. Grötschel y L. Lovász. *Handbook of Combinatorics*. Handbook of Combinatorics v. 1. Elsevier Science, 1995. ISBN: 9780444880024.

[16] Alfredo Braunstein, Marc Mézard y Riccardo Zecchina. "Survey propagation: an algorithm for satisfiability". En: *CoRR* cs.CC/0212002 (2002).

Apéndice A: Modelo de juguete con 3 nodos

A continuación se presenta el código para *MaBoSS* que define el modelo de juguete utilizado.

```
//Se define el primer nodo debajo
Node A
{
//Debajo se encuentra el operador ternario x?y:z,
//que resulta en el ritmo de activacion del nodo
rate_up=(C AND (NOT B))  ? $Au : 0.0;
//Este segundo operador ternario resulta
//en el ritmo de inhibicion del nodo
rate_down= B ? $Ad : 0.0;
}
Node B
{
rate_up= A ? $Bu : 0.0;
rate_down = A ? 0.0 : $Bd ;
}
Node C
{
rate_up=0.0;
rate_down=((NOT A) AND (NOT B)) ? $escape : 0.0 ;
}
```

Los parámetros del modelo presentado toman los valores:

```
$Au=1;   $Ad=4;
$Bu=2;   $Bd=3;
```

Apéndice B: Modelo del ciclo celular

A continuación se presenta el código para *MaBoSS* que define el modelo de juguete utilizado.

```
node CycD{
  logic = CycD;
  rate_up = (NOT $CycD_del) ? 0.0 : 0.0;
  rate_down = $CycD_del ? $fast : 0.0;
}
node CycE{
  logic = !Rb & E2F;
  rate_up = @logic ? $slow : 0.0;
  rate_down = @logic ? 0.0 : $fast;
}
node CycA{
  logic = !Rb & !Cdc20 & !(UbcH10 & cdh1) & (CycA | E2F);
  rate_up = @logic ? $slow : 0.0;
  rate_down = @logic ? 0.0 : $fast;
}
node CycB{
  logic = !Cdc20 & !cdh1;
  rate_up = @logic ? $slow : 0.0;
  rate_down = @logic ? 0.0 : $fast;
}
node Rb{
  logic = !CycD & !CycB & (p27 | !(CycA | CycE));
  rate_up = (@logic AND (NOT $Rb_del)) ? $fast : 0.0;
  rate_down = (@logic AND (NOT $Rb_del)) ? 0.0 : $fast;
}
node E2F{
  logic = !Rb & !CycB & (!CycA | p27);
  rate_up = @logic ? $slow : 0.0;
  rate_down = @logic ? 0.0 : $fast;
}
node p27{
  logic = !CycD & !CycB & (!(CycA | CycE) | (p27 & !(CycE & CycA)));
  rate_up = @logic ? $fast : 0.0;
  rate_down = @logic ? 0.0 : $fast;
}
node Cdc20{
  logic = CycB;
  rate_up = (@logic AND (NOT $Cdc20_del)) ? $slow : 0.0;
  rate_down = (@logic AND (NOT $Cdc20_del)) ? 0.0 : $fast;
}
node UbcH10{
  logic = (!(cdh1 & !UbcH10) & (CycA | CycB)) | (!CycA & !CycB & (!cdh1 | (Cdc20 &
    UbcH10)));
  rate_up = @logic ? $slow : 0.0;
  rate_down = @logic ? 0.0 : $fast;
}
node cdh1{
  logic = Cdc20 | (!CycB & (!CycA | p27));
  rate_up = @logic ? $fast : 0.0;
  rate_down = @logic ? 0.0 : $fast;
}
```

Los parámetros del modelo presentado toman los valores:

```
$fast=1;    $slow=1;    $CycD_del=$Cdc20_del=$Rb_del=0;
```

Apéndice C: Números de Möbius

En este apéndice se presenta la demostración completa de la proposición 10. La enunciamos de nuevo.

Proposición 16. *Los números de Möbius de los clústeres en $\mathscr{P}'$ son, para $t = 0, ..., \tau - 1$*

	$\mathcal{M}_i^{(t)}$	$\mathcal{P}_i^{(t)}$	$\mathcal{Q}_{\langle ij\rangle}^{(t)}$	$\mathcal{S}_i^{(t)}$	$\mathcal{T}_{i,\langle ij\rangle}^{(t)}$	$\mathcal{Z}_{\langle ij\rangle}^{(t)}$	$\mathcal{R}_i^{(t)}$	$\mathcal{U}_{\langle ij\rangle,i}^{(t)}$	$\mathcal{V}_i^{(t)}$	$\mathcal{A}_i^{(t)}$
a_β	*1*	*0*	*-1*	*-1 (0)*	*0*	*1 (0)*	*0*	*0*	*0*	*0*

y entre paréntesis los valores que cambian para $t = 0$.

Demostración. Partimos de los clústeres maximales $\mathcal{M}_i^{(t)} = (x_i, x_{\partial i}, y_i, x_{\partial i})^{(t)}$, con número $a_{\mathcal{M}} = 1$ y utilizando la regla del lema 9 se obtiene la tabla. Para los clústeres $\mathcal{P}_i^{(t)} = (x_i, x_{\partial i}, y_i)^{(t)}$

$$\mathcal{P}_i^{(t)} \leq \mathcal{M}_i^{(t)}, \quad \text{entonces } a_{\mathcal{P}} = 1 - a_{\mathcal{M}} = 0.$$

Análogamente para $\mathcal{R}_i^{(t)} = (x_i, y_i, y_{\partial i})^{(t)}$ se tiene $a_{\mathcal{R}} = 1 - a_{\mathcal{M}} = 0$. Para los clústeres $\mathcal{Q}_{\langle ij\rangle}^{(t)} = (x_i, x_j, y_i, y_j)^{(t)}$

$$\mathcal{Q}_{\langle ij\rangle}^{(t)} \leq \mathcal{M}_i^{(t)}, \mathcal{M}_j^{(t)}, \quad \text{entonces } a_{\mathcal{Q}} = 1 - 2a_{\mathcal{M}} = -1.$$

Para los clústeres $\mathcal{T}_{i,\langle ij\rangle}^{(t)} = (x_i, x_j, y_j)^{(t)}$

$$\mathcal{T}_{i,\langle ij\rangle}^{(t)} \leq \mathcal{M}_i^{(t)}, \mathcal{M}_j^{(t)}, \mathcal{Q}_{\langle ij\rangle}^{(t)}, \quad \text{entonces } a_{\mathcal{T}} = 1 - 2a_{\mathcal{M}} - a_{\mathcal{Q}} = 0.$$

Análogamente para $\mathcal{U}_{\langle ij\rangle,i}^{(t)} = (x_i, y_i, y_j)^{(t)}$ se tiene $a_{\mathcal{U}} = 1 - 2a_{\mathcal{M}} - a_{\mathcal{Q}} = 0$. Para los clústeres $\mathcal{S}_i^{(t)} = (x_i, x_{\partial i})^{(t)}$ en $t = 1, ..., \tau - 1$

$$\mathcal{S}_i^{(t)} \leq \mathcal{M}_i^{(t)}, \mathcal{M}_i^{(t-1)}, \quad \text{entonces } a_{\mathcal{S}} = 1 - 2a_{\mathcal{M}} = -1,$$

y en $t = 0$

$$\mathcal{S}_i^{(0)} \leq \mathcal{M}_i^{(0)}, \quad \text{entonces } a_{\mathcal{S}} = 1 - a_{\mathcal{M}} = 0.$$

Para los clústeres $\mathcal{Z}_{\langle ij\rangle}^{(t)} = (x_i, x_j)^{(t)}$ en $t = 1, ..., \tau - 1$

$$\mathcal{Z}_{\langle ij\rangle}^{(t)} \leq \mathcal{M}_i^{(t)}, \mathcal{M}_j^{(t)}, \mathcal{M}_i^{(t-1)}, \mathcal{M}_j^{(t-1)}, \mathcal{Q}_{\langle ij\rangle}^{(t)}, \mathcal{Q}_{\langle ij\rangle}^{(t-1)}, \mathcal{S}_i^{(t)}, \mathcal{S}_j^{(t)}, \quad \text{entonces } a_{\mathcal{Z}} = 1 - 4a_{\mathcal{M}} - 2a_{\mathcal{Q}} - 2a_{\mathcal{S}} = 1,$$

y en $t = 0$

$$\mathcal{Z}_{\langle ij\rangle}^{(0)} \leq \mathcal{M}_i^{(0)}, \mathcal{M}_j^{(0)}, \mathcal{Q}_{\langle ij\rangle}^{(0)}, \quad \text{entonces } a_{\mathcal{Z}} = 1 - 2a_{\mathcal{M}} - a_{\mathcal{Q}} = 0.$$

Para los clústeres $\mathcal{V}_i^{(t)} = (x_i, y_i)^{(t)}$

$$\mathcal{V}_i^{(t)} \leq \mathcal{M}_i^{(t)}, \mathcal{M}_j^{(t)}, \mathcal{Q}_{\langle ij\rangle}^{(t)}, \ j \in \partial i, \quad \text{entonces } a_{\mathcal{V}} = 1 - (1 + |\partial i|)a_{\mathcal{M}} - |\partial i| a_{\mathcal{Q}} = 0.$$

Finalmente, para los clústeres $\mathcal{A}_i^{(t)} = (x_i)^{(t)}$

$$\mathcal{A}_i^{(t)} \leq \mathcal{M}_i^{(t)}, \mathcal{M}_j^{(t)}, \mathcal{M}_i^{(t-1)}, \mathcal{M}_j^{(t-1)}, \mathcal{Q}_{\langle ij\rangle}^{(t)}, \mathcal{Q}_{\langle ij\rangle}^{(t-1)}, \mathcal{S}_i^{(t)}, \mathcal{S}_j^{(t)}, \mathcal{Z}_{\langle ij\rangle}^{(t)}, \quad j \in \partial i,$$

entonces $a_{\mathcal{A}} = 1 - 2(1 + |\partial i|)a_{\mathcal{M}} - 2|\partial i| a_{\mathcal{Q}} - (1 + |\partial i|)a_{\mathcal{S}} - |\partial i| a_{\mathcal{Z}} = 0$. Y en $t = 0$

$$\mathcal{A}_i^{(0)} \leq \mathcal{M}_i^{(0)}, \mathcal{M}_j^{(0)}, \mathcal{Q}_{\langle ij\rangle}^{(0)}, \ j \in \partial i, \quad \text{entonces } a_{\mathcal{A}} = 1 - (1 + |\partial i|)a_{\mathcal{M}} - |\partial i| a_{\mathcal{Q}} = 0.$$

□

Apéndice D: Código CVM

A continuación se presenta el código en C para la aproximación M aplicada al ciclo celular.

```
//EsteEsElBueno

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <stdbool.h>

#define N 10
//#define tao 0.2 //1/(fast+slow)
#define tao 0.5 //1/(fast+slow)
#define fast 1
#define slow 1
#define CycD_del 0
#define Cdc20_del 0
#define NodeStates 2
//#define tmax 100
#define tmax 40
#define eps 0

struct node{
  int V[N];
  int SNmax;
  double *S;//Tama o 2*SNmax, Primero xi=0, segundo xi=1
  double *P;//Tama o 4*SNmax
  double **M;//(system+i)->M[yi*((system+i)->SNmax) + SN][xi*((system+i)->SNmax) +
    SN]
  double *****Q;//j, yi, yj, xi, xj el n mero de j's es d_i
  double ***Z;//j\in V[N], xi, xj
  double ****T;//j, yi, xi, xj
  int d_i;//#vecinos
  //int d_ij;//#pares que alberga
};

struct entropy{
  double M;
  double Q;
  double S;
  double Z;
  double H;
};

void LecturaCondicionesIniciales(int *p);
void LecturaVecinos(struct node *p);
void iniScluster(struct node *p,int *q);
void iniZcluster(struct node *system,int *q);
int logic(int I, int xi, int* V, int SN);
double TransitionProb(int I, int yi, int xi, int* V, int SN);
void Pcluster(int I, int yi, int xi, int SN, struct node *Node);
void Tcluster(int I, int J, int yi, int xi, int xj, struct node *Node);
void Mcluster(int I, int yi, int xi, int SN1, int SN, struct node *Node);
void Qcluster(int I, int J, int yi, int yj, int xi, int xj, struct node *Node);
void Zcluster(int I, int J, int yi, int yj, struct node *Node);
void Scluster(int I, int yi, int SN, struct node *Node);
```

```
54 double Acluster(int I, int xi, struct node *Node);
55
56 int main(){
57
58   //ALLOC STRUCT DE NODOS
59   struct node *system;
60   system = (struct node*) malloc(N * sizeof(struct node));
61
62   //CONDICIONES INICIALES
63   int *xini=NULL;
64   xini = calloc(N,sizeof(int));
65   int i,j,t,yi,yj,xi,xj,SN,SN1,SNmax;
66   //int J;
67   LecturaCondicionesIniciales(xini);
68   LecturaVecinos(system);
69   //VOY A ALLOCAR LA MEMORIA EN MAIN, COMO DEBE SER
70   for(i=0;i<N;i++){
71     (system+i)->S = calloc(2*((system+i)->SNmax), sizeof(double));
72     (system+i)->P = calloc(4*((system+i)->SNmax), sizeof(double));
73     (system+i)->M = calloc(2*((system+i)->SNmax), sizeof(double*));
74     (system+i)->Q = calloc((system+i)->d_i, sizeof(double****));
75     (system+i)->Z = calloc((system+i)->d_i, sizeof(double**));
76     (system+i)->T = calloc((system+i)->d_i, sizeof(double***));
77     for(j=0;j<2*system[i].SNmax;j++){
78       (system+i)->M[j] = calloc(2*((system+i)->SNmax), sizeof(double));
79     }
80     for(j=0;j<system[i].d_i;j++){
81       (system+i)->Q[j] = calloc(NodeStates, sizeof(double***));
82       (system+i)->Z[j] = calloc(NodeStates, sizeof(double*));
83       (system+i)->T[j] = calloc(NodeStates, sizeof(double**));
84       for(yi=0;yi<NodeStates;yi++){
85         (system+i)->Q[j][yi] = calloc(NodeStates, sizeof(double**));
86         (system+i)->Z[j][yi] = calloc(NodeStates, sizeof(double));
87         (system+i)->T[j][yi] = calloc(NodeStates, sizeof(double*));
88         for(yj=0;yj<NodeStates;yj++){
89           (system+i)->Q[j][yi][yj] = calloc(NodeStates, sizeof(double*));
90           (system+i)->T[j][yi][yj] = calloc(NodeStates, sizeof(double));
91           for(xi=0;xi<NodeStates;xi++){
92             (system+i)->Q[j][yi][yj][xi] = calloc(NodeStates, sizeof(double));
93           }
94         }
95       }
96     }
97   }
98   iniScluster(system,xini);//HAGO EL S INICIAL
99   iniZcluster(system,xini);//Z INICIAL
100
101  //LIBERO MEMORIA DE C.I.
102  free(xini);
103
104  //COSAS PARA SACAR
105  FILE *g;
106  g=fopen("Results.txt","w");
107  if(g==NULL){
108    printf("No existe el fichero de salida\n");
109  }
110  fprintf(g, "t\tCycD\tCycE\tCycA\tCycB\tRb\tE2F\tp27\tCdc20\tUbcH10\tcdh1\tH\tTH\n"
       );
111  double A[N];
112  double norm=0;
113  double norm1=0;//para debugging
```

```
114   double sum;
115   //double sum1;
116   struct entropy S;
117   S.H=0;//INICIO AQU  S TOTAL, tengo que ir sumando en el tiempo
118   /*
119   //Para sacar transiciones
120   for(i=0;i<N;i++){
121     for(SN=0;SN<system[i].SNmax;SN++){
122       printf("Nodo %d\t", i);
123       printf("%.15lf", TransitionProb(i,0,0,system[i].V,SN));
124       printf("\n");
125     }
126   }
127   */
128   //Hasta aqu
129
130   //Para logic
131   /*
132   for(i=0;i<N;i++){
133     for(xi=0; xi<NodeStates;xi++){
134       for(SN=0;SN<system[i].SNmax;SN++){
135         printf("Nodo %d ", i);
136         if(xi==0){
137           printf("up\t");
138         }else{
139           printf("down\t");
140         }
141         printf("%d", logic(i, xi, system[i].V, SN));
142         printf("\n");
143       }
144     }
145   }
146   */
147   for(t=0;t<tmax;t++){
148     printf("Paso %d\n", t);
149     //INICIO ENTROP AS AUXILIARES
150     S.Q=0;
151     S.S=0;
152     S.Z=0;
153     S.M=0;
154
155     //SUMA AQU  PARA ENTROP A S Y Z, AL FINAL SER N DE t+1
156     for(i=0;i<N;i++){
157       for(xi=0;xi<NodeStates;xi++){
158         for(SN=0;SN<((system+i)->SNmax);SN++){
159           if((system+i)->S[((system+i)->SNmax)*xi+SN]==0){//HE PRINTEADO Y SIEMPRE
    ES 0, WTF
160             S.S-=0;
161           }else{
162             S.S-=(system+i)->S[((system+i)->SNmax)*xi+SN]*log10((system+i)->S[((
    system+i)->SNmax)*xi+SN]);
163           }
164         }
165         for(xj=0;xj<NodeStates;xj++){
166           for(j=0;j<(system[i].d_i);j++){
167             if((system+i)->Z[j][xi][xj]==0 || i>system[i].V[j]){
168               S.Z-=0;
169             }else{
170               S.Z-=(system+i)->Z[j][xi][xj]*log10((system+i)->Z[j][xi][xj]);
171             }
172           }
```

```
173         }
174       }/*
175       for(j=0;j<system[i].d_i;j++){
176         for(xi=0;xi<NodeStates;xi++){
177           for(xj=0;xj<NodeStates;xj++){
178             for(J=0;J<j;J++){
179             sum+=
180             }
181           }
182         }
183       }*/
184     }
185
186     //STEP 2
187     for(i=0;i<N;i++){//Paso c lculo P
188       norm1=0;
189       for(xi=0;xi<NodeStates;xi++){
190         for(SN=0;SN<((system+i)->SNmax);SN++){
191           sum=0;
192           for(yi=0;yi<NodeStates;yi++){
193             //PARA PRINT
194             //printf("Nodo %d\t", i);
195             Pcluster(i,yi,xi,SN,system);
196             //printf("i %d yi %d xi %d SN %d P=%.15lf\n",i,yi,xi,SN, (system + i)->P
    [2 * yi * ((system + i)->SNmax) + xi * ((system + i)->SNmax) + SN]);
197             norm1+=(system + i)->P[2 * yi * ((system + i)->SNmax) + xi * ((system +
    i)->SNmax) + SN];
198             sum+=(system + i)->P[2 * yi * ((system + i)->SNmax) + xi * ((system + i)
    ->SNmax) + SN];
199           }
200           /*
201           if(abs(sum-system[i].S[xi*system[i].SNmax + SN])>1E-14){
202             printf("i %d xi %d SN %d S=%.16lf P1=%.16lf P2=%.16lf sum=%.16lf\n", i,
    xi, SN, system[i].S[xi*system[i].SNmax + SN]
203             , system[i].P[xi*system[i].SNmax + SN], system[i].P[2*system[i].SNmax +
    xi*system[i].SNmax + SN], sum);
204           }
205           */
206         }
207       }/*
208       if(norm1!=1){
209         printf("NORM P t=%d\ti=%d\tP\n",t,i);
210       }*/
211       for(j=0;j<(system[i].d_i);j++){
212         norm1=0;
213         for(xi=0;xi<NodeStates;xi++){
214           for(xj=0;xj<NodeStates;xj++){
215             sum=0;
216             for(yi=0;yi<NodeStates;yi++){
217               //printf("Nodo %d\t", i);
218               Tcluster(i, system[i].V[j], yi, xi, xj, system);//checkear si lo de v[
    j] est  bien. s  est  bien
219               //printf("i %d j %d yi %d xi %d xj %d T=%.16lf\n", i, j,yi, xi, xj,
    system[i].T[j][yi][xi][xj]);
220               norm1+=system[i].T[j][yi][xi][xj];
221               sum+=system[i].T[j][yi][xi][xj];
222             }/*
223             if(sum!=system[i].Z[j][xi][xj]){
224               printf("i %d j %d xi %d xj %d Z=%.16lf T0=%.16lf T1=%.16lf\n", i, j,
    xi, xj, system[i].Z[j][xi][xj], system[i].T[j][0][xi][xj], system[i].T[j][1][xi
    ][xj]);
```

```
225             }*/
226           }
227         }/*
228         if(norm1!=1){
229           printf("NORM T t=%d\ti=%d\tj=%d\tnorm=%f\tT\n",t,i,j,norm1);
230         }*/
231       }
232     }
233
234     //STEP 3.1
235     for(i=0;i<N;i++){//Paso c lculo M
236       SNmax=(system+i)->SNmax;
237       norm=0;
238       norm1=0;
239       for(yi=0;yi<NodeStates;yi++){
240         for(SN1=0;SN1<SNmax;SN1++){
241           for(xi=0;xi<NodeStates;xi++){
242             for(SN=0;SN<SNmax;SN++){
243               //printf("Nodo %d\t", i);
244               Mcluster(i,yi,xi,SN1,SN,system);
245               //printf("i %d yi %d SN1 %d xi %d SN %d M=%.15lf\n",i,yi,SN1,xi,SN,
    system[i].M[yi*SNmax+SN1][xi*SNmax+SN]);
246               if((system+i)->M[yi*SNmax + SN1][xi*SNmax + SN]==0){
247                 S.M-=0;
248               }else{
249                 S.M-=system[i].M[yi*SNmax + SN1][xi*SNmax + SN]*log10(system[i].M[yi
    *SNmax + SN1][xi*SNmax + SN]);
250               }
251               norm1+=system[i].M[yi*SNmax+SN1][xi*SNmax+SN];
252               /*
253               if(system[i].M[yi*SNmax + SN1][xi*SNmax + SN]!=0){
254                 printf("%d\t%.15lf\n",i,system[i].M[yi*SNmax + SN1][xi*SNmax + SN]);
255               }*/
256             }
257           }
258           //printf("Nodo %d\t", i);
259           Scluster(i,yi,SN1,system);
260           //printf("i %d yi %d SN1 %d S=%.15lf\n",i,yi,SN1, (system+i)->S[((system+i
    )->SNmax)*yi+SN1]);
261           norm += system[i].S[SNmax*yi + SN1];
262         }
263       }/*
264       if(norm1!=1){
265         printf("NORM M t=%d\ti=%d\tnorm=%.15lf\tM\n",t,i,norm1);
266       }
267       if(norm!=1){
268         printf("NORM S t=%d\ti=%d\tnorm=%.15lf\tS\n",t,i,norm);
269       }*/
270       //printf("t=%d\ti=%d\tnorm=%f\tM\n",t,i,norm1);
271       for(yi=0;yi<NodeStates;yi++){
272         for(SN1=0;SN1<SNmax;SN1++){
273           system[i].S[SNmax*yi + SN1] = system[i].S[SNmax*yi + SN1] / norm;
274         }
275       }
276       A[i]=Acluster(i,1,system);
277     }
278
279     //STEP 3.2
280     for(i=0;i<N;i++){//Paso c lculo Q
281       for(j=0;j<system[i].d_i;j++){
282         norm1=0;
```

```
283         for(xi=0;xi<NodeStates;xi++){
284           for(xj=0;xj<NodeStates;xj++){
285             //sum=0;
286             for(yi=0;yi<NodeStates;yi++){
287               //sum1=0;
288               for(yj=0;yj<NodeStates;yj++){
289                 //printf("Nodo %d\t", i);
290                 Qcluster(i,j,yi,yj,xi,xj,system);
291                 //printf("i %d j %d yi %d yj %d xi %d xj %d Q=%.16lf\n", i, j, yi,yj
    , xi, xj, system[i].Q[j][yi][yj][xi][xj]);
292                 norm1+=(system+i)->Q[j][yi][yj][xi][xj];
293                 if((system+i)->Q[j][yi][yj][xi][xj]==0){
294                   S.Q-=0;
295                 }else{
296                   S.Q-=(system+i)->Q[j][yi][yj][xi][xj]*log10((system+i)->Q[j][yi][
    yj][xi][xj]);
297                 }
298                 //sum+=(system+i)->Q[j][yi][yj][xi][xj];
299                 //sum1+=(system+i)->Q[j][yi][yj][xi][xj];
300               }
301               /*
302               if(abs(sum1-system[i].T[j][yi][xi][xj])>1E-14){
303                 printf("i %d j %d yi %d xi %d xj %d T=%.16lf Q0=%.16lf Q1=%.16lf\n",
     i, j, yi, xi, xj,
304                 system[i].T[j][yi][xi][xj], system[i].Q[j][yi][0][xi][xj], system[i
    ].Q[j][yi][1][xi][xj]);
305               }
306               */
307             }
308             /*
309             if(sum!=system[i].Z[j][xi][xj]){
310               printf("i %d j %d xi %d xj %d Z=%.16lf sum=%.16lf\n", i, j, xi, xj,
    system[i].Z[j][xi][xj], sum);
311             }
312             */
313           }
314         }
315         for(yi=0;yi<NodeStates;yi++){//Paso c lculo U y Z
316           for(yj=0;yj<NodeStates;yj++){
317             Zcluster(i, j, yi, yj, system);
318             //printf("i %d j %d yi %d yj %d Z=%.16lf\n", i, j, yi, yj, system[i].Z[j
    ][yi][yj]);
319           }
320         }
321         norm = system[i].Z[j][0][0] + system[i].Z[j][0][1] + system[i].Z[j][1][0] +
    system[i].Z[j][1][1];
322         /*if(norm!=1){
323           printf("NORM Z t=%d\ti=%d\tj=%d\tnorm=%.15lf\tZ\n",t,i,j,norm);
324         }*/
325         system[i].Z[j][0][0] = system[i].Z[j][0][0] / norm;
326         system[i].Z[j][0][1] = system[i].Z[j][0][1] / norm;
327         system[i].Z[j][1][0] = system[i].Z[j][1][0] / norm;
328         system[i].Z[j][1][1] = system[i].Z[j][1][1] / norm;
329         /*if(norm1!=1){
330           printf("NORM Q t=%d\ti=%d\tj=%d\tnorm=%f\tQ\n",t,i,j,norm1);
331         }*/
332       }
333
334     }//Puede que sea posible meterlo dentro de bucle i, pues distintos nodos no
    afectan
335     // CHECK Zij=Zji
```

```
336     for(i=0;i<N;i++){
337       for(j=0;j<system[i].d_i;j++){
338         for(yi=0;yi<NodeStates;yi++){
339           for(yj=0;yj<NodeStates;yj++){
340             int k;
341             for(k=0;i!=system[system[i].V[j]].V[k];k++){}
342             //printf("%.15lf\t\t%.15lf\t", system[i].Z[j][yi][yj], system[system[i].
    V[j]].Z[k][yj][yi]);
343             if(abs(system[i].Z[j][yi][yj]-system[system[i].V[j]].Z[k][yj][yi])>1E
    -15){
344               printf("No son iguales los <ij>\n");
345             }
346           }
347         }
348       }
349     }
350 
351     S.H = S.M - S.Q - S.S + S.Z;//+= Porque suma en el tiempo
352 
353     //IMPRIMO AL TXT
354     fprintf(g, "%d\t", t);
355     for(i=0;i<N;i++){
356       fprintf(g, "%.15lf\t", A[i]);
357     }
358     fprintf(g, "%.15lf\t%.15lf\n", -S.S+2*S.Z, S.H);
359   }
360   fclose(g);
361 
362   //LIBERO TODA LA MEMORIA DIN MICA
363   for(i=0;i<N;i++){
364     free(system[i].S);
365     free((system+i)->P);
366     for(j=0;j<2*system[i].SNmax;j++){
367       free(system[i].M[j]);
368     }
369     free(system[i].M);
370     for(j=0;j<system[i].d_i;j++){
371       for(yi=0;yi<NodeStates;yi++){
372         for(yj=0;yj<NodeStates;yj++){
373           for(xi=0;xi<NodeStates;xi++){
374             free((system+i)->Q[j][yi][yj][xi]);
375           }
376           free((system+i)->Q[j][yi][yj]);
377           free((system+i)->T[j][yi][yj]);
378         }
379         free((system+i)->Q[j][yi]);
380         free((system+i)->Z[j][yi]);
381         free((system+i)->T[j][yi]);
382       }
383       free((system+i)->Q[j]);
384       free((system+i)->Z[j]);
385       free((system+i)->T[j]);
386     }
387     free((system+i)->Q);
388     free((system+i)->Z);
389     free((system+i)->T);
390   }
391   free(system);
392 
393 return 0;
394 }
```

```
395
396 void LecturaCondicionesIniciales(int *p){//REVISADO
397   char xbuffer[10];
398   FILE *f;
399   f=fopen("CondicionesIniciales.txt","r");
400   int i;
401   if(f==NULL){
402     printf("No existe el fichero de condiciones iniciales\n");
403   }
404   else{
405     for(i=0;!feof(f);i++){
406       fscanf(f,"%s%d",xbuffer,p+i);
407     }
408   }
409   fclose(f);
410 }
411
412 void LecturaVecinos(struct node *p){//REVISADO
413   int i,j,k;//Recuerda que no leo cuando el vecino es s   mismo
414   FILE *f;
415   f=fopen("Vecinos.txt","r");
416   if(f==NULL){
417     printf("No existe el fichero de vecinos\n");
418   }else{
419     for(j=0;j<N;j++){
420       //k=0;
421       for(i=0;(i<N)&&((p+j)->V[i-1]!=-1)&&!feof(f);i++){
422         fscanf(f,"%d",&(p+j)->V[i]);
423         //if(j<(p+j)->V[i]){k++;}
424       }
425       //(p+j)->d_ij=k;
426       //printf("%d", (p+j)->d_ij);Estas cosas eran para el cambio de <ij>
427       (p+j)->SNmax=1;
428       (p+j)->d_i=i-1;
429       for(k=0;k<i-1;k++){//AQU  HAGO 2^i
430         (p+j)->SNmax*=2;
431       }
432     }
433   }
434   fclose(f);
435 }
436
437 void iniScluster(struct node *system,int *xini){//REVISADO
438   int i, j;
439   int SN[N];
440   //Aqu  paso de xini a SN en cada nodo
441   for(j=0;j<N;j++){
442     SN[j]=0;
443     for(i=0;(((system+j)->V[i])!=-1)&&(i<N);i++){//ojo, para m  los SN son binarios
      al rev s
444       if(xini[(system+j)->V[i]]==1){
445         SN[j]+=(int)(pow(2,i)+0.5);//potencia en int
446       }
447     }
448   }
449
450   //Aqu  hago S
451   for(i=0;i<N;i++){
452     (system+i)->S[xini[i]*((system+i)->SNmax)+SN[i]]=1.;
453   }
454 }
```

```
455
456 void iniZcluster(struct node *system,int *xini){//REVISADO
457   int i,j;
458   for(i=0;i<N;i++){
459     for(j=0;j<(system+i)->d_i;j++){
460       (system+i)->Z[j][xini[i]][xini[(system[i].V[j])]] = 1.;
461     }
462   }
463 }
464
465 double TransitionProb(int I, int yi, int xi, int* V, int SN){//PARECE REVISADO
466   int delta;// DELTA DE KRONECKER
467   double w[2];
468   if(xi==yi){
469     delta=1;
470   }else {delta=0;}
471
472   /*
473   //Esto est  para ver la salida de transiciones
474   int *q=NULL;
475   q = calloc(N,sizeof(int));
476   int i;
477   int bin;
478   bin=SN;
479   for(i=0;(V[i]!=-1)&&(i<N);i++){
480     if(bin==0 || bin==1){
481       q[V[i]]=bin;
482       bin=bin/2;
483     }else{
484       q[V[i]]=bin%2; //PONGO EL ESTADO SEG N EL ORDEN DE LAS C. INI.
485       bin=bin/2;
486     }
487   }
488   q[I]=xi;
489   for(i=0;i<N;i++){
490     printf("%d ", q[i]);
491   }
492   printf("\t");
493   q[I]=yi;
494   for(i=0;i<N;i++){
495     printf("%d ", q[i]);
496   }
497   printf("\t");
498   //Hasta aqu
499   */
500
501   switch(I){
502     case 0://OJO CON ESTE, DEPENDE SOLO DE CTE.
503       if(!CycD_del){
504         w[0]=0;
505         w[1]=0;
506       }else{
507         w[0]=0;
508         w[1]=fast;
509       }
510       return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
511       break;
512     case 1://CycE
513       if(logic(I, xi, V, SN)){
514         w[0]=slow;
515         w[1]=0;
```

```
516       }else{
517         w[0]=0;
518         w[1]=fast;
519       }
520       return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
521       break;
522     case 2://CycA
523       if(logic(I, xi, V, SN)){
524         w[0]=slow;
525         w[1]=0;
526       }else{
527         w[0]=0;
528         w[1]=fast;
529       }
530       return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
531       break;
532     case 3://CycB
533       if(logic(I, xi, V, SN)){
534         w[0]=slow;
535         w[1]=0;
536       }else{
537         w[0]=0;
538         w[1]=fast;
539       }
540       return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
541       break;
542     case 4://Rb
543       if(logic(I, xi, V, SN)){
544         w[0]=fast;
545         w[1]=0;
546       }else{
547         w[0]=0;
548         w[1]=fast;
549       }
550       return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
551       break;
552     case 5://E2F
553       if(logic(I, xi, V, SN)){
554         w[0]=slow;
555         w[1]=0;
556       }else{
557         w[0]=0;
558         w[1]=fast;
559       }
560       return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
561       break;
562     case 6://p27
563       if(logic(I, xi, V, SN)){
564         w[0]=fast;
565         w[1]=0;
566       }else{
567         w[0]=0;
568         w[1]=fast;
569       }
570       return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
571       break;
572     case 7://Cdc20
573       if(logic(I, xi, V, SN)&&!Cdc20_del){
574         w[0]=slow;
575         w[1]=0;
576       }else{
```

```
577         w[0]=0;
578         w[1]=fast;
579       }
580       return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
581       break;
582     case 8://UbcH10
583       if(logic(I, xi, V, SN)){
584         w[0]=slow;
585         w[1]=0;
586       }else{
587         w[0]=0;
588         w[1]=fast;
589       }
590       return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
591       break;
592     case 9://cdh1
593       if(logic(I, xi, V, SN)){
594         w[0]=fast;
595         w[1]=0;
596       }else{
597         w[0]=0;
598         w[1]=fast;
599       }
600       return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
601       break;
602     default:
603       printf("No metiste nodo al TransitionProb\n");
604       return 0;//esto esta solo para que no salte advertencia
605   }
606 }
607
608 int logic(int I,int xi, int* V, int SN){//REVISADO Y CORREGIDO
609   int *q=NULL;
610   q = calloc(N,sizeof(int));
611   int i;
612   int bin;
613   bin=SN;
614   for(i=0;(V[i]!=-1)&&(i<N);i++){
615     if(bin==0 || bin==1){
616       q[V[i]]=bin;
617       bin=bin/2;
618     }else{
619       q[V[i]]=bin%2; //PONGO EL ESTADO SEG N EL ORDEN DE LAS C. INI.
620       bin=bin/2;
621     }
622   }
623   q[I]=xi;//METO ESTO PORQUE EN V NO VOY A METER A SU MISMO NODO (por problemas
       luego en sumatorios de vecinos)
624
625   //PARA MOSTRAR ESTADO
626   /*
627   for(i=0;i<N;i++){
628     printf("%d", q[i]);
629   }
630   printf("\t");
631   */
632
633   switch(I){
634     case 0:
635       if(*q){
636       return 1;
```

```
637       }else{return 0;}
638       break;
639     case 1:
640       if(!*(q+4) && *(q+5)){
641       return 1;
642       }else{return 0;}
643       break;
644     case 2:
645       if(!*(q+4) && !*(q+7) && !(*(q+8) && *(q+9)) && (*(q+2) || *(q+5))){
646       return 1;
647       }else{return 0;}
648       break;
649     case 3:
650       if(!*(q+7) && !*(q+9)){
651       return 1;
652       }else{return 0;}
653       break;
654     case 4:
655       if(!*q && !*(q+3) && (*(q+6) || !(*(q+2) || *(q+1)))){
656       return 1;
657       }else{return 0;}
658       break;
659     case 5:
660       if(!*(q+4) && !*(q+3) && (!*(q+2) || *(q+6))){
661       return 1;
662       }else{return 0;}
663       break;
664     case 6:
665       if(!*q && !*(q+3) && (!(*(q+2) || *(q+1)) || (*(q+6) && !(*(q+1) && *(q+2)))))
    {
666       return 1;
667       }else{return 0;}
668       break;
669     case 7:
670       if(*(q+3)){
671       return 1;
672       }else{return 0;}
673       break;
674     case 8:
675       if((!(*(q+9) && !*(q+8)) && (*(q+2) || *(q+3))) || (!*(q+2) && !*(q+3) && (!*(
    q+9) || (*(q+7) && *(q+8))))){
676       return 1;
677       }else{return 0;}
678       break;
679     case 9:
680       if(*(q+7) || (!*(q+3) && (!*(q+2) || *(q+6)))){
681       return 1;
682       }else{return 0;}
683       break;
684     default:
685       printf("Fallo en Logic\n");
686       return -1;
687       break;
688   }
689 }
690
691 void Pcluster(int I, int yi, int xi, int SN, struct node *Node){//OJO, ESTOY
    PENSANDO QUE IGUAL CONVIENE SACAR SN DEL STRUCT, PUES VA A IR VARIANDO MUCHO
692   (Node+I)->P[2*yi*((Node+I)->SNmax) + xi*((Node+I)->SNmax) + SN] = TransitionProb(I
    , yi, xi, (Node+I)->V, SN)*((Node+I)->S[xi*((Node+I)->SNmax)+SN]);//LE HE PASADO
     EL SYSTEM ENTERO A Pcluster
```

```
693
694   //PRINT DE ESTADO
695   /*
696   int *q = NULL;
697   q = calloc(N, sizeof(int));
698   int i;
699   int bin;
700   bin = SN;
701   for (i = 0; (Node[I].V[i] != -1) && (i < N); i++)
702   {
703     if (bin == 0 || bin == 1)
704     {
705       q[Node[I].V[i]] = bin;
706       bin = bin / 2;
707     }
708     else
709     {
710       q[Node[I].V[i]] = bin % 2; //PONGO EL ESTADO SEG N EL ORDEN DE LAS C. INI.
711       bin = bin / 2;
712     }
713   }
714   q[I] = xi;
715   for (i = 0; i < N; i++)
716   {
717     printf("%d ", q[i]);
718   }
719   printf("\t");
720   q[I] = yi;
721   for (i = 0; i < N; i++)
722   {
723     printf("%d ", q[i]);
724   }
725   printf("\t");
726   printf("w=%.15lf\tS=%.15lf\t", TransitionProb(I, yi, xi, (Node + I)->V, SN), (Node
        + I)->S[xi * ((Node + I)->SNmax) + SN]);
727
728   if((Node+I)->P[2*yi*((Node+I)->SNmax) + xi*((Node+I)->SNmax) + SN]>1.){
729     //exit(999);
730   }
731   */
732 }
733
734 void Tcluster(int I, int J, int yi, int xi, int xj, struct node *Node){//PASALE
       SYSTEM COMPLETO
735   double sum;
736   sum=0;
737   int i, j, SN, bin;
738   int *q=NULL;
739   q = calloc(N,sizeof(int));
740   //printf("I=%d J=%d yi=%d xi=%d xj=%d \n", I,J,yi,xi,xj);
741   for(SN=0;SN<((Node+I)->SNmax);SN++){
742     bin=SN;
743     for(i=0;(((Node+I)->V[i])!=-1)&&(i<N);i++){//SACO EL ESTADO AL COMPLETO PRIMERO
744       if(bin==0 || bin==1){
745         q[(Node+I)->V[i]]=bin;
746         bin=bin/2;
747       }else{
748         q[(Node+I)->V[i]]=bin%2;
749         bin=bin/2;
750       }
751     }
```

```
752     q[I] = xi;//Esto lo traje del Toymodel, creo que solo es  til  a la hora de
        printear
753     if(q[J]!=xj){//DESPU S COMPRUEBO QUE XJ NO HA CAMBIADO EN EL ESTADO
754       sum+=0;
755     }else{
756       sum+=(Node+I)->P[2*yi*((Node+I)->SNmax) + xi*((Node+I)->SNmax) + SN];
757       //printf("P(y=%d,x=%d%d%d%d%d%d%d%d%d%d)=%.15lf \n",yi, q[0],q[1],q[2], q[3],
        q[4], q[5], q[6], q[7], q[8], q[9], (Node+I)->P[2*yi*((Node+I)->SNmax) + xi*((
        Node+I)->SNmax) + SN]);
758     }
759   }
760   for(j=0;j<N;j++){//COMPROBADO EXTERNAMENTE
761     if(J==Node[I].V[j]){
762       (Node+I)->T[j][yi][xi][xj] = sum;//ESTO ES NUEVO, PARA QUE sea el j en el
        orden de vecinos, no el del orden global de N
763       break;
764     }
765   }
766   //printf("T=%.15lf\n\n", (Node+I)->T[j][yi][xi][xj]);
767   /*
768   if((Node+I)->T[j][yi][xi][xj] != sum){
769     printf("ERROR con T(j)");
770   }
771   if((Node+I)->T[j][yi][xi][xj]>1.){
772     //exit(999);
773   }
774   */
775 }
776
777 void Qcluster(int I, int J, int yi, int yj, int xi, int xj, struct node *Node){
778   //printf("I=%d J=%d yi=%d yj=%d xi=%d xj=%d ", I,J,yi,yj,xi,xj);
779   int i,j;
780   j=Node[I].V[J];
781   for(i=0;i<N;i++){//COMPROBADO EXTERNAMENTE
782     if(I==Node[j].V[i]){
783       break;//ESTO ES NUEVO, PARA QUE sea el j en el orden de vecinos, no el del
        orden global de N
784     }
785   }
786   //printf("i=%d j=%d\n",i,j);
787   /*if(i==N){
788     //printf("ERROR con Q(i)");
789   }*/
790   if((Node+I)->Z[J][xi][xj]<=eps){
791     (Node+I)->Q[J][yi][yj][xi][xj] = 0;
792     //printf("Q=0\n");
793   }else{
794     (Node+I)->Q[J][yi][yj][xi][xj] = (Node+I)->T[J][yi][xi][xj]*(Node+j)->T[i][yj][
        xj][xi]/(Node+I)->Z[J][xi][xj];
795     //printf("T1=%.15lf T2=%.15lf Z=%.15lf Q=%.15lf\n", (Node+I)->T[J][yi][xi][xj],
        (Node+j)->T[i][yj][xj][xi], (Node+I)->Z[J][xi][xj], (Node+I)->Q[J][yi][yj][xi][
        xj]);
796   }
797   /*
798   if((Node+I)->Q[J][yi][yj][xi][xj]>1.){
799     //exit(999);
800   }*/
801 }
802
803 void Zcluster(int I, int J, int yi, int yj, struct node *Node){
804   int i, j;
```

```
805   double sum=0;
806   //printf("I=%d J=%d yi=%d yj=%d ", I, J, yi, yj);
807   for(i=0;i<NodeStates;i++){
808     for(j=0;j<NodeStates;j++){
809       sum+=(Node+I)->Q[J][yi][yj][i][j];
810       //printf("Q(xi=%d, xj=%d) %.15lf\t", i, j, (Node+I)->Q[J][yi][yj][i][j]);
811     }
812   }
813   (Node+I)->Z[J][yi][yj]=sum;
814   /*
815   printf("Z %.15lf\n", (Node+I)->Z[J][yi][yj]);
816   if((Node+I)->Z[J][yi][yj]>1.){
817     //exit(999);
818   }*/
819 }
820
821 void Mcluster(int I, int yi, int xi, int SN1, int SN, struct node *Node){//He
      reutilizado el Rcluster
822   int j, i, J, bin, bin1;
823   double P;
824   int SNmax=Node[I].SNmax;
825   //printf("i=%d\tyi=%d\txi=%d\tSN1=%d\tSN=%d",I,yi,xi,SN1,SN);
826   P=Node[I].P[2*yi*SNmax + xi*SNmax + SN];
827   //printf("P=%.15lf\n",Node[I].P[2*yi*SNmax + xi*SNmax + SN]);
828   double prodZ,prodT;
829   prodZ=1;
830   prodT=1;
831   bin=SN;
832   //printf("SN1=%d\n",SN1);
833   bin1=SN1;
834   for(j=0;(((Node+I)->V[j])!=-1)&&(j<N);j++){//SACO EL ESTADO AL COMPLETO PRIMERO
      Este bucle va a ser lo que revisar a fondo
835     J=Node[I].V[j];
836     //printf("bin1=%d\n",bin1);
837     if(bin==0 || bin==1){
838       prodZ*=Node[I].Z[j][xi][bin];
839       //printf("Z=%.15lf\ti=%d\tj_i=%d\txi=%d\txj=%d\n",Node[I].Z[j][xi][bin],I,j,xi
      ,bin);
840       //apa o el j
841       for(i=0;(Node[J].V[i]!=-1)&&(i<N);i++){
842         if(I==Node[J].V[i]){
843           //printf("bin1=%d\n",bin1);
844           if(bin1==0||bin1==1){
845             prodT*=Node[J].T[i][bin1][bin][xi];
846             //printf("T=%.15lf\tj=%d\ti=%d\tyj=%d\txj=%d\txi=%d\n",Node[J].T[i][bin1
      ][bin][xi],J,i,bin1,bin,xi);
847           }else{
848             prodT*=Node[J].T[i][bin1%2][bin][xi];
849             //printf("T=%.15lf\tj=%d\ti=%d\tyj=%d\txj=%d\txi=%d\n",Node[J].T[i][bin1
      %2][bin][xi],J,i,bin1%2,bin,xi);
850           }
851           break;//ESTO ES NUEVO, PARA QUE sea el j en el orden de vecinos, no el del
       orden global de N
852           //El break este est  mal, te rompe el loop y no llego a dividir por 2 el
      bin1
853         }
854         //Adem s, si I est  en el 0, no llego aqu  nunca por culpa del break
855       }
856       //printf("U_%d(yj=%d) %.15lf\t", (Node+I)->V[j], bin, (Node+I)->U[j][yi][bin][
      xi]);
857       bin1=bin1/2;
```

```
858       bin=bin/2;
859     }else{
860       prodZ*=Node[I].Z[j][xi][bin%2];
861       //printf("Z=%.15lf\ti=%d\tj_i=%d\txi=%d\txj=%d\n",Node[I].Z[j][xi][bin%2],I,j,
    xi,bin%2);
862       //apa o el j
863       for(i=0;(Node[J].V[i]!=-1)&&(i<N);i++){
864         if(I==Node[J].V[i]){//solo a ado al producto en el i que toca
865           //printf("bin1=%d\n",bin1);
866           if(bin1==0||bin1==1){
867             prodT*=Node[J].T[i][bin1][bin%2][xi];
868             //printf("T=%.15lf\tj=%d\ti=%d\tyj=%d\txj=%d\txi=%d\n",Node[J].T[i][bin1
    ][bin%2][xi],J,i,bin1,bin%2,xi);
869           }else{
870             prodT*=Node[J].T[i][bin1%2][bin%2][xi];
871             //printf("T=%.15lf\tj=%d\ti=%d\tyj=%d\txj=%d\txi=%d\n",Node[J].T[i][bin1
    %2][bin%2][xi],J,i,bin1%2,bin%2,xi);
872           }
873           break;//ESTO ES NUEVO, PARA QUE sea el j en el orden de vecinos, no el del
     orden global de N
874         }
875       }
876       //printf("U_%d(yj=%d) %.15lf\t", (Node+I)->V[j], bin%2, (Node+I)->U[j][yi][bin
    %2][xi]);
877       bin1=bin1/2;
878       bin=bin/2;
879     }
880   }
881   if(prodZ==0){
882     Node[I].M[yi*SNmax + SN1][xi*SNmax + SN]=0;
883   }else{
884     Node[I].M[yi*SNmax+SN1][xi*SNmax+SN]=P*prodT/prodZ;
885   }
886     //printf("di %d\tR %.15lf", j-1, (Node+I)->R[xi*2*((Node+I)->SNmax) + yi*((Node+
      I)->SNmax) + SN]);
887   /*
888   printf("\n");
889   if((Node+I)->R[xi*2*((Node+I)->SNmax) + yi*((Node+I)->SNmax) + SN]>1.){
890     //exit(999);
891   }*/
892 }
893
894 void Scluster(int I, int yi, int SN1, struct node *Node){
895   int xi,SN;
896   double sum=0;
897   /*
898   //PRINT DE ESTADO
899   int *q=NULL;
900   q = calloc(N,sizeof(int));
901   int bin;
902   bin=SN;
903   for(i=0;(Node[I].V[i]!=-1)&&(i<N);i++){
904     if(bin==0 || bin==1){
905       q[Node[I].V[i]]=bin;
906       bin=bin/2;
907     }else{
908       q[Node[I].V[i]]=bin%2; //PONGO EL ESTADO SEG N EL ORDEN DE LAS C. INI.
909       bin=bin/2;
910     }
911   }
912   q[I]=yi;
```

```
  for(i=0;i<N;i++){
    printf("%d ", q[i]);
  }
  printf("\t");
  */
  for(xi=0;xi<NodeStates;xi++){
    for(SN=0;SN<Node[I].SNmax;SN++){
      sum+=Node[I].M[yi*Node[I].SNmax + SN1][xi*Node[I].SNmax + SN];
    }
    //printf("R(xi=%d)=%.15lf\t", i, (Node+I)->R[i*2*((Node+I)->SNmax) + yi*((Node+I
    )->SNmax) + SN]);
  }
  (Node+I)->S[yi*Node[I].SNmax+SN1] = sum;
  /*
  if((Node+I)->S[((Node+I)->SNmax)*yi+SN]>1.){
    //exit(999);
  }*/
}

double Acluster(int I, int xi, struct node *Node){
  int SN, SN1, yi;
  double sum=0;
  for(SN1=0;SN1<Node[I].SNmax;SN1++){
    for(SN=0;SN<((Node+I)->SNmax);SN++){
      for(yi=0;yi<NodeStates;yi++){
        sum+=Node[I].M[yi*((Node+I)->SNmax) + SN1][xi*((Node+I)->SNmax) + SN];
        //printf("%.15lf\n",Node[I].M[yi*((Node+I)->SNmax) + SN1][xi*((Node+I)->
    SNmax) + SN]);
      }
    }
  }
  return sum;
}
```

El primer archivo de texto necesario es el que contiene la información de los vecinos de cada nodo, "Vecinos.txt":

```
4 6 -1
4 5 6 -1
4 5 6 7 8 9 -1
4 5 6 7 8 9 -1
0 1 2 3 5 6 -1
1 2 3 4 6 -1
0 1 2 3 4 5 9 -1
2 3 8 9 -1
2 3 7 9 -1
2 3 6 7 8 -1
```

Además, se ha de aportar la condición inicial de todos los nodos en el archivo de texto ÇondicionesIniciales.txt":

```
CycD  1
CycE  0
CycA  0
CycB  0
Rb  1
E2F 0
p27 1
Cdc20 0
UbcH10  0
cdh1  1
```