
**TRABAJO DE FIN DE GRADO**

---

# Estudios de redes de regulación genéticas con modelos discretos

---

*Autor*:
Pablo Pérez Lázaro

*Directores*:
Dr. Pierpaolo Bruscolini

Dr. Joaquín Sanz Remón

UNIVERSIDAD DE ZARAGOZA

FACULTAD DE CIENCIAS

10 Septiembre 2021

# Índice

# 1. Resumen

Las distintas fases del ciclo celular están dirigidas por una red de regulación. La evolución de las moléculas involucradas en la red se puede modelizar como variables acopladas a través de interacciones. El modelo de ciclo celular que se utiliza es una red de corrientes de actividad, con variables cualitativas lógicas que representan las proteínas reguladoras del ciclo y los complejos cdk/ciclina de la célula. Se estudia la evolución de complejos cdk/ciclina porque se conocen sus interacciones con las proteínas reguladoras. Además, la periodicidad de estas moléculas sigue las distintas fases del ciclo celular.

Un método clásicamente utilizado en biología de sistemas para la resolución de la evolución de redes de regulación es el algoritmo de Gillespie. Una herramienta basada en este algoritmo en la que además ya se ha conseguido reproducir el modelo del ciclo celular es el entorno *MaBoSS* [1].

Por otra parte, se busca aplicar el método de Aproximación Variacional en Clúster, que hasta el momento no se ha utilizado nunca para redes de regulación genéticas. Este método se ha utilizado en tratamiento de sistemas como modelos epidémicos SIS o al estudio de transiciones en materiales con estructuras cristalinas. El objetivo es estudiar el modelo del ciclo celular con esta herramienta, comparándolo con *MaBoSS*, un método clásico en biología de sistemas.

Con motivo de estudiar cómo se comporta el método *CVM* en este nuevo ámbito, se evalúan ejemplos típicos de sistemas de redes de regulación de complejidad creciente.

En primer lugar, se trata un modelo de dos nodos que conforman un bucle entre cuatro estados del sistema. Después, se estudia el modelo resultante de añadir un tercer nodo, de forma que el sistema tiene un estado atractor alcanzado escapando del bucle de cuatro estados anterior o siguiendo un árbol de estados. Con estos dos modelos sencillos se identifican algunos de los objetos que suelen aparecer en redes de regulación: bucles, atractores o árboles. Finalmente se ha estudiado la evolución del modelo de ciclo celular en *CVM*, comparándola con los resultados de la herramienta *MaBoSS*.

Tras estudiar los tiempos de ejecución de los dos métodos y cómo de cercanos son los resultados del *CVM* a la teoría sobre el ciclo celular, se prueba que esta técnica es aplicable sin utilizar métodos clásicos como *MaBoSS* de forma auxiliar.

**Palabras clave** Ciclina, física biológica, entropía, proceso de Markov, clúster.

# 2. Introducción

## 2.1. Ciclo celular

En este Trabajo de Fin de Grado se va a tratar la física y modelización de las redes de regulación genética, enfocándonos en concreto en la red responsable del ciclo celular. Para ello, es necesario introducir ciertos conceptos básicos de la biología celular, ya que los sistemas vivos son distintos de los que se suelen estudiar en la física.[2]

### 2.1.1. La célula

Un ser vivo es modular, es decir, está formado por distintos niveles de organización (órganos, tejidos, células...) que tienen distintas escalas de tamaño. Gracias a esto y a otras cualidades como la redundancia hay resistencia a fallos, y en su funcionamiento, si una ruta no funciona, a menudo habrá otra que la sustituya. En concreto, la célula es la menor unidad de la vida. Ésta, a su vez, está formada por módulos subcelulares que realizan sus funciones. En toda célula de un ser vivo tiene lugar continuamente el ciclo de división celular, regulado por rutas a nivel subcelular.

**Ciclo celular** El ciclo celular es el proceso mediante el cual las células crecen y se duplican, dando lugar a dos nuevas células, con el mismo material genético que la madre. Uno de los aspectos más relevantes de esta función, que ha ocupado más tiempo a los biólogos es la duplicación del genoma de la célula.

El material genético que posee una célula contiene toda la información que necesita para la creación de proteínas, que regulan su funcionamiento. Y el genoma de una célula es todo su material genético, presente en forma de cromosomas (estructura compacta del ADN). Dada su relevancia para el funcionamiento de la célula, cobra mucha importancia su duplicación en el ciclo celular.

**Fases del ciclo celular** El ciclo celular tiene lugar en cinco fases, que se distinguen por los procesos que conllevan [3]:

- Fase $G0$: en ella se da la coordinación entre la célula y el crecimiento del organismo que la contiene. Se recibe una señal extra-celular que dicta si la célula ha de crecer y dividirse o mantenerse estancada.
- Fase $G1$: aparece una vez la $G0$ da paso a la división celular. En ella, la célula se prepara para la división, se da su crecimiento llegando a duplicarse en tamaño y masa. Suele durar de 6 a 20 horas, en las que se sintetizan las proteínas y ARN necesarias para fases posteriores.
- Fase $S$: es la fase más larga, en la que se produce la síntesis de todo el material genético. Dura de 10 a 12 horas según el organismo, hasta que se duplican todos los cromosomas iniciales.
- Fase $G2$: fase previa a la división celular, sirve de *cortafuegos* entre la duplicación del genoma y la división celular. Se realizan las comprobaciones pertinentes para evitar errores en la replicación de ADN o que haya quedado incompleta. Es un paso crítico previo a la mitosis, pues tales errores genéticos pasarían a una de las hijas, que a su vez extendería a las generaciones siguientes.
- Fase $M$: en la mitosis se da la división celular, se pueden identificar las etapas profase, metafase, anafase y telofase. Es la fase más corta, concluye cuando se tienen las dos células hijas idénticas. La mitosis se puede observar en células tintadas, ante microscopio óptico.

Es muy importante mantener el orden de estas fases para que el resultado sea el esperado. De esto se ocupa el motor del ciclo celular[4]: su red de regulación. Se van a describir algunos de sus componentes que sirven para construir el modelo de ciclo celular.

### 2.1.2. Ciclinas como indicadoras del transcurso del ciclo celular

Las ciclinas son una familia de proteínas que participan en la regulación del ciclo celular cuando cooperan con otras proteínas, las quinasas dependientes de ciclina (CDK). Estos complejos cdk/-ciclina marcan el paso entre las fases enunciadas en el punto anterior. Sin embargo, no solo nos interesa el funcionamiento de estos complejos, sino también su concentración.

La concentración de ciclinas presente en la célula varía a lo largo del ciclo celular, aumentando en cada cambio de fase (Figura 1). Entonces, si se consigue un modelo del avance de sus concentraciones, se pueden monitorizar las distintas fases del ciclo celular[5]. Se tienen los siguientes complejos:

- cdk4/6-Ciclina D ($CycD$): en vertebrados la ciclina D se asocia con las Cdk 4 y 6 [6]. La producción de este complejo representa la señal externa a la célula, marca el paso de la fase $G0$ a la $G1$. Está presente en concentraciones altas a lo largo de todas la división celular y se inhibe su producción tras la división.

- cdk2-Ciclina E ($CycE$): la producción de la ciclina E se activa en la fase G1. Este complejo $CycE$ impulsa el paso de la célula a la fase S [7].

- cdk2-Ciclina A ($CycA$): la ciclina A se empieza a producir en la fase S, reemplazando así a las ciclinas E en el complejo con cdk2 [8]. El nuevo complejo CycA se encarga de iniciar la replicación del ADN en la fase S, además de evitar que se replique más de una vez [9]. Está presente también durante la fase $G2$, posterior a la replicación.

- cdk1-Ciclina B ($CycB$): la producción de ciclinas B se activa con la entrada a la fase M [10]. Este complejo está presente durante toda la mitosis y es el que termina el ciclo celular tras la división. Sin embargo, si se detectan errores en el ADN replicado, la célula para la producción de ciclinas B, deteniendo así la mitosis.

### 2.1.3. Proteínas reguladoras

Si se pretende tener una imagen clara del motor del ciclo celular, se han de tener en cuenta los complejos cdk-ciclina y también las proteínas que interaccionan con ellos. Estas últimas regulan y son reguladas por los complejos anteriores.

Hay muchas proteínas que cooperan con los complejos, pero se van a describir solamente las que conforman nuestro modelo de ciclo celular:

- Proteína del retinoblastoma ($Rb$): es la proteína que evita el crecimiento de la célula cuando se encuentra en la fase $G0$, pues inhibe la producción de ciclinas A y E [12]. Los complejos $CycD$ causan el paso de $G0$ a $G1$ desactivando estas proteínas $Rb$ [13]. Es una proteína bastante estudiada, pues es muy relevante en el estudio de tumores cancerígenos.

- Proteína *p27* [14]: esta proteína se une al complejo $CycE$ para inhibir su funcionamiento y también se une al $CycA$, es esencial para desarrollar su función. Además, lo hacen de forma que el complejo $CycA$ se lleva las proteínas *p27* que se encuentran ligadas a los $CycE$. Es decir, la *p27* permite la progresión en el ciclo, pasando de un complejo a otro.

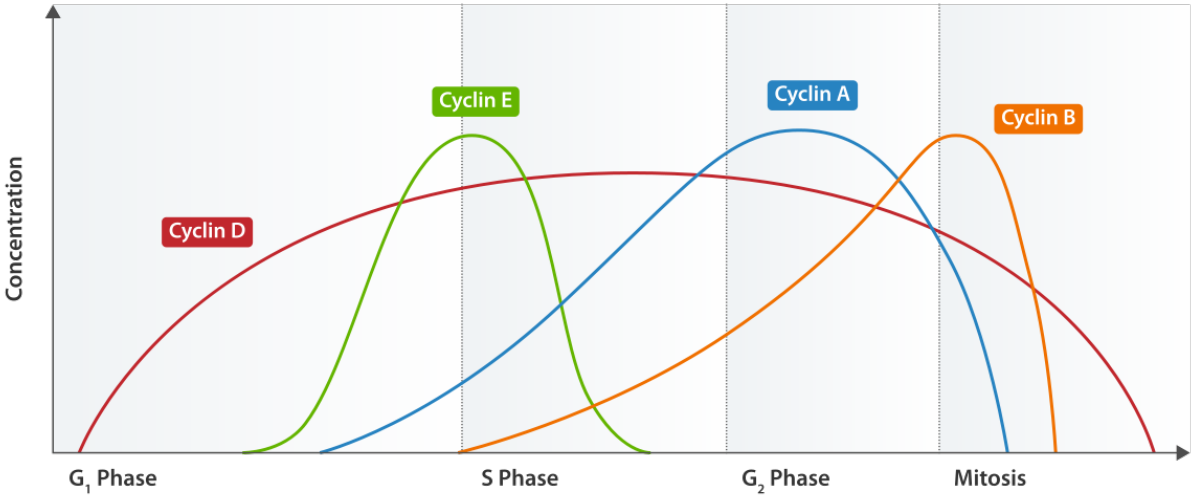


Figura 1: En esta imagen se presentan las concentraciones de las ciclinas en la célula. Podemos observar cómo la ciclina D está presente durante todo el ciclo, pues se activa por agentes externos e inicia el ciclo celular. También, la ciclina E inicia la fase S y la ciclina A se mantiene durante toda la replicación del ADN y su comprobación posterior. Finalmente, la ciclina B está presente durante la mitosis, hasta que la célula se divide [11].

- Proteínas *E2F* [5]: son una familia de factores de transcripción. Es decir, proteínas que regulan la expresión de la información genética del ADN en forma de creación de proteínas. En concreto, activan la producción de ciclinas E. Sin embargo, se pueden unir a proteínas *Rb*, formando un complejo sin la función de transcripción, pausando así la expresión genética.
- Complejo de Promoción de la Anafase (*APC*) [15]: complejo proteico que se activa cíclicamente. Sus activadoras son las proteínas *Cdc20* y *Cdh1*, pero lo hacen de distinta forma, dando lugar a dos funciones distintas. Cuando en la fase de mitosis se liga al complejo la *Cdc20*, permite el paso de la metafase a la anafase. Es decir, da lugar a la separación de los cromosomas, hacia puntos opuestos de la célula. Sin embargo, la *Cdh1* al ligarse al *APC*, controla el final de la mitosis y la próxima fase $G1$ tras la división.
- Proteína Cdc20 [15]: proteína activadora del complejo *APC*. A su vez, su producción es activada por el complejo *CycB* pero inhibida por la proteína *Cdh1*.
- Proteína *Cdh1* [15]: proteína activadora del complejo *APC* e inhibidora de las funciones del complejo *CycB* y *CycA*. Las *Cdc20* activan su producción.
- Proteína *UbcH10* [5]: proteína que permite la activación del complejo *CycA* en la fase S, incluso cuando la *Cdh1* se encuentra todavía activa.

## 2.2. Biología de sistemas

La biología de sistemas es uno de los campos que contribuye a la biología esclareciendo las propiedades y comportamientos de sistemas presentes en la ella [16]. En particular, la biología de sistemas a partir de las interacciones entre elementos sub-celulares como proteínas o complejos de ellas, obtiene como resultado la explicación del comportamiento celular.

Además, en las últimas décadas la capacidad de experimentación que se ha alcanzado en biología aporta una gran cantidad de datos. El procesado computacional o mediante modelos de estos datos complementa los experimentos, permitiendo obtener mucha más información a partir de ellos. De esta forma, las matemáticas y la física contribuyen de forma notable a la comprensión de la vida, convirtiendo la biología en un campo interdisciplinar.

## 2.3. Modelos de redes de regulación genética

A la hora de representar una red de regulación de procesos celulares, es esencial elegir el acercamiento adecuado a la modelización, ya que como cabe esperar, cada enfoque tiene sus ventajas e inconvenientes [17]. Es decir, los modelos se han de adaptar a los problemas a resolver. Se van a presentar los distintos enfoques que se suelen usar para modelizar redes de regulación genética, con pequeñas explicaciones que ayudarán a justificar la elección para el caso del ciclo celular.

**Representación de la red** En este trabajo se modeliza la evolución de la expresión genética de las moléculas involucradas en la regulación del ciclo celular como variables acopladas a través de interacciones. Su topología define una red de regulación genética. Se distinguen cuatro familias de redes de interacción de este tipo, que se describen en la siguiente figura 2.

**Información disponible para el modelo** Como es de esperar, la información de la que se dispone sobre el sistema a la hora de hacer el modelo condiciona el enfoque utilizado. Se distinguen dos casos:

- **Enfoque basado en información** (bottom-top): se parte de información obtenida de la literatura científica o bases de datos. En concreto, se suelen utilizar bases de datos con modelos previos como *Biomodels* [19].
- **Enfoque basado en datos** (top-bottom): en vez de partir de interacciones y parámetros conocidos, se obtiene la topología de la red a partir de resultados experimentales previos. Cobra relevancia para la estimación de parámetros.

**Modelos cuantitativos y cualitativos** Distinguimos los modelos cuantitativos de los lógicos [18]:

- **Modelos cuantitativos cinéticos**: cada nodo almacena un valor, que puede ser cantidad de moléculas de un tipo o actividad de un gen. En ellos, cada interacción viene modulada por parámetros presentes en las ecuaciones diferenciales que describen el modelo o mediante simulación estocástica. Se pueden desarrollar con distintos grados de profundización en la mecánica del sistema:

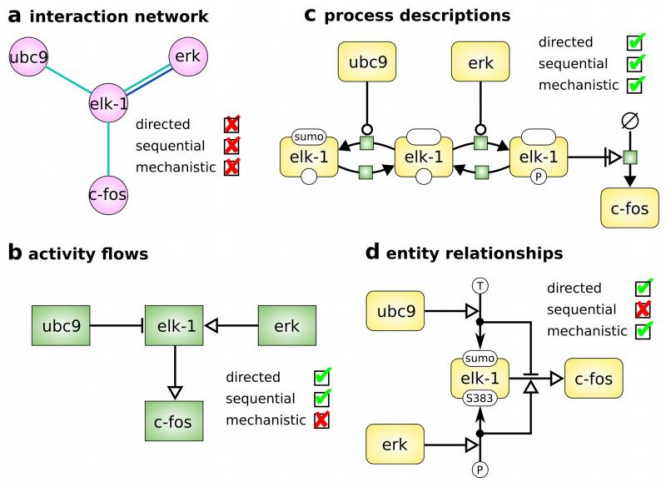


Figura 2: **a. Redes de interacción**: representan una lista de interacciones entre los objetos del sistema, que no son direccionales. Se suelen usar para la comprensión de la regulación genética, cuando se conocen las interacciones entre los genes y las proteínas. Pero no son útiles para modelos dinámicos (por ejemplo, modelización de polímeros). **b. Corrientes de actividad**: este tipo de red es necesaria cuando no se conocen los mecanismos moleculares detrás del sistema. Solo requieren la información de que una actividad $i$ estimula otra actividad $j$, siendo así redes sujetas a modelos cualitativos por naturaleza. Estas interacciones son dirigidas y secuenciales. **c. Proceso descriptivo**: trata interacciones dirigidas y secuenciales, al igual que las corrientes de actividad. Sin embargo, diferencia dos tipos de nodos: variables de las que interesa su evolución y procesos que influyen sobre el valor de estas variables. Son los adecuados para la química cinética. **d. Relaciones entre entidades**: considera que las entidades o nodos pueden encontrarse en estados determinados además de interaccionando con otros nodos [18].

  - *Química cinética*: se hace uso de la ley de acción de masas.
  - *Funciones de Hill*: es una forma genérica de representar modulaciones que suele ser válida para regulación de genes. Aparecen en fenómenos cooperativos.
  - *Linealización de las ecuaciones diferenciales del sistema*: se pueden aproximar las funciones de Hill por funciones escalón.

  La principal limitación de los modelos cuantitativos es la falta de datos en cuanto a parámetros cinéticos, ya que son difíciles de obtener.

- **Modelos lógicos**: en ellos, los nodos suelen representar variables cualitativas, que pueden tomar valores de los enteros según el estado en que se encuentren. Aunque de este tipo de modelos no se logre obtener información tan detallada como en los cuantitativos, son versátiles y flexibles, permitiendo así estudiar la estructura del sistema biológico. Además, no requieren tantos datos para poder extraer resultados de ellos.

  Sin embargo, el inconveniente principal es que suele ser difícil obtener conclusiones a partir de sus resultados.

- **Modelos híbridos**: dada la heterogeneidad en la naturaleza de los datos experimentales en biología, una mezcla de los dos tipos de modelo suele ser un enfoque bastante acertado.

Para el caso del ciclo celular que se va a tratar, dada la naturaleza de los datos de los que se dispone y dado también el tipo de interacciones presentes entre las proteínas en el ciclo celular, se va a utilizar un modelo lógico de una red de corrientes de actividad en base a datos de un modelo previo [5] disponible en el repositorio *GINsim*.

Un método clásicamente utilizado en biología de sistemas para la resolución de la evolución de redes de regulación es el algoritmo de Gillespie. Éste define trayectorias del sistema, la evolución temporal de las variables booleanas mediante una simulación del estilo Monte-Carlo, partiendo de una condición inicial determinada. Estas trayectorias son procesos de Markov de tiempo continuo. La herramienta MaBoSS extiende el formalismo de Monte Carlo cinético de Gillespie al estudio de modelos booleanos, y proporciona la evolución del sistema definiendo una discretización temporal de las trayectorias, y promediando un número arbitrario de trayectorias, obteniendo la probabilidad de activación de las variables de la red en cada ventana temporal. Cuando el número de trayectorias es lo suficientemente grande, las probabilidades convergen a soluciones de la ecuación maestra de Markov, que definen la evolución del sistema. Además, el modelo de ciclo celular escogido [5] ya ha sido tratado previamente en el entorno *MaBoSS* [1].

El *Cluster Variational Method* es una técnica basada en restringir a una elección de clústeres la correlación entre nodos del sistema en una aproximación de campo medio. Según la topología de dichos clústeres elegidos como maximales se toma una aproximación u otra. Este método se ha utilizado en tratamiento de sistemas como modelos epidémicos SIS o al estudio de transiciones en materiales con estructuras cristalinas. Se busca entonces aplicar por primera vez la aproximación variacional en clúster a la cinética de redes de regulación, obteniendo así un método alternativo a técnicas clásicas como el algoritmo de Gillespie. Además, se esperan tiempos mucho menores de ejecución para resultados comparables con la herramienta *MaBoSS* y capacidad para redes de regulación con un número mucho mayor de variables.

## 3. Objetivos

Para tratar con redes de regulación, primero es necesario conocer la literatura actual sobre biología de sistemas aplicada a la modelización de estos sistemas. Así se puede identificar cómo de capaz es el método *CVM* de reproducir el modelo del ciclo celular[1]. También se ha de comprender el ciclo celular para poder entender bien el problema.

El *CVM* se han utilizado hasta el momento para otro tipo de sistema. Como su aplicación a redes de regulación es novedosa, se empieza reproduciendo modelos de juguete sencillos con la ayuda de *MaBoSS*. Se busca que los resultados coincidan entre los dos métodos y también observar qué limitaciones y qué ventajas tiene *CVM* con respecto a *MaBoSS*. Este proceso sirve para estudiar la implementación del método de Aproximación Variacional en Clúster. Además, en estas redes pequeñas de dos o tres nodos se puede obtener la evolución exacta, que es el mejor punto de referencia posible con el que comparar los dos métodos.

Finalmente, el objetivo del trabajo es reproducir una red de regulación genética como la del ciclo celular utilizando *CVM*. Como punto de referencia para los resultados se va a utilizar la herramienta

[1]Un objetivo inicial era partir de un modelo de ciclo celular continuo y transformarlo en un problema discreto para resolverlo con *CVM*. Se descartó por la dificultad en el tratamiento de las ecuaciones diferenciales y parámetros.

*MaBoSS*, aunque siempre teniendo presente la literatura sobre el ciclo celular, pues se trata de una aproximación que no da resultados exactos.

Además, se busca comparar los tiempos de ejecución de los dos métodos y cómo de acertados son los resultados para comprobar que el uso del *CVM* aporta ventajas con respecto a métodos clásicos de la biología de sistemas.

# 4. Métodos

A la hora de aplicar un nuevo algoritmo a un modelo, conviene disponer de otro algoritmo comprobado previamente en el que se consiga también reproducir dicho modelo. De esta forma, se pueden comparar los resultados obtenidos en ambos, asegurando así su correcto funcionamiento. Entonces, es necesario conocer a fondo los dos algoritmos.

## 4.1. MaBoSS 2.0: un entorno para modelización booleana estocástica

MaBoSS [20] es un entorno de modelización booleana de tiempo continuo orientado a redes biológicas. Tiene un acercamiento cualitativo, pero al ser de tiempo continuo, puede hacer de unión entre modelos cuantitativos y cualitativos. Esto lo consigue aplicando el algoritmo de Monte-Carlo Cinético (Gillespie).

### 4.1.1. Algoritmo de Monte Carlo Cinético (BKMC)

El algoritmo se basa en considerar una red de nodos. El estado de esta red viene dado por el conjunto de estados de cada nodo, que puede ser 0 o 1, un número Booleano. La evolución del sistema es estocástica, considerando el tiempo continuo y la actualización de cada nodo como consecuencia de su interacción con el resto de la red.

**Procesos de Markov** Para entender el algoritmo *BKMC*, antes es necesario entender la matriz de transición de Markov [1].

Sea una red de $N$ nodos, se define un estado de la red como:

$$\vec{S} = \{S_i = 0, 1 | i = 1, ..., N\} \subset \Gamma$$

Donde $\Gamma$ es el espacio de los estados de la red. Un proceso de Markov es un proceso estocástico, cuya evolución viene descrita por $s : t \to s(t)$ sobre $\Gamma$ y $\forall t \in I$ intervalo, tal que $s(t)$ representa una variable aleatoria en $\Gamma$, con probabilidades:

$$P(s(t) = \vec{S}) \in [0, 1] \, \forall \vec{S} \in \Gamma \quad t.q. \quad \sum_{\vec{S} \in \Gamma} P(s(t) = \vec{S}) = 1$$

Además, este proceso cumple la propiedad de Markov, que considerando el caso de tiempo discreto $I = \{t_1, t_2, ...\}$, se enuncia:

$$P\left(s(t_{i+1}) = \vec{S}^{(i+1)} \middle| s(t_1) = \vec{S}^{(1)}, ..., s(t_i) = \vec{S}^{(i)}\right) = P\left(s(t_{i+1}) = \vec{S}^{(i+1)} \middle| \, s(t_i) = \vec{S}^{(i)}\right) \qquad (1)$$

Es decir, las probabilidades de variables aleatorias en el futuro condicionadas a otras en el tiempo presente y pasado, solo dependen del tiempo presente. Esto se entiende como la falta de memoria del sistema. De aquí se deduce que un proceso de Markov está completamente determinado por las probabilidades de transición $P\left(s(t_{i+1}) = \vec{S} \middle| s(t_i) = \vec{S}'\right)$ entre los $2^n$ estados de $\Gamma$ y las condiciones iniciales $P\left(s(t_1) = \vec{S}\right)\ \forall \vec{S} \in \Gamma$.

Sin embargo, *BKMC* surge del caso con tiempo continuo, en el que aparece también la definición de matriz de transición. La definición de proceso de Markov dada se generaliza al continuo tomando el intervalo temporal $I = [t_{min}, t_{max}]$, tal que entonces el proceso de Markov está completamente determinado por el conjunto de todos los ritmos de transición $\rho_{\vec{S}\to\vec{S}'}$ y las condiciones iniciales $P\left(s(t_{min}) = \vec{S}\right)\ \forall \vec{S} \in \Gamma$.

Ahora bien, la propiedad de Markov en el tiempo continuo se expresa como la ecuación maestra, ecuación diferencial que determina la evolución de las probabilidades $P\left(s(t) = \vec{S}\right)$ en $t \in I$:

$$\frac{\partial}{\partial t}\vec{P}(t) \equiv \frac{\partial}{\partial t}P\left(s(t) = \vec{S}\right) = \sum_{\vec{S}'}\{\rho_{\vec{S}'\to\vec{S}}P\left(s(t) = \vec{S}'\right) - \rho_{\vec{S}\to\vec{S}'}P\left(s(t) = \vec{S}\right)\} \equiv M\vec{P}(t)$$

Con $(M)_{ij} \equiv \rho_{\vec{S}^j\to\vec{S}^i} - \sum_k \rho_{\vec{S}^j\to\vec{S}^k}\delta_{ij}$ la matriz de transición. De esta forma, calculando la exponencial de dicha matriz se resuelve la ecuación maestra, obteniendo así las probabilidades:

$$\vec{P}(t) = \vec{P}(0)e^{Mt} \quad \forall t \in I$$

Se observa la equivalencia con el caso discreto al utilizar las probabilidades de transición:

$$P\left(s(t_{i+1}) = \vec{S}' \middle| s(t_i) = \vec{S}\right) = \frac{\rho_{\vec{S}\to\vec{S}'}}{\sum_{\vec{S}''}\rho_{\vec{S}\to\vec{S}''}} \tag{2}$$

**Algoritmo de Gillespie** Este algoritmo que permite explorar el espacio de probabilidades $\Gamma$ de un proceso de Markov, definido por el conjunto de ritmos de transición $\rho_{\vec{S}\to\vec{S}'}$ y las condiciones iniciales $P\left(s(t_{min}) = \vec{S}\right)\ \forall \vec{S} \in \Gamma$ [21].

Esto se consigue con una simulación del estilo Monte-Carlo, generando un conjunto de trayectorias de un proceso de Markov de tiempo continuo. A partir de estas trayectorias se pueden calcular las probabilidades de los distintos estados, pues cuando el número de trayectorias tiende a infinito, el conjunto converge a la solución de la ecuación maestra de Markov $\vec{P}(t)\ \forall t \in I = [0, t_{max}]$.

Sea pues una trayectoria una función que a cada tiempo $t$ le asocia un estado de $\Gamma$, $\hat{S} : I = [0, t_{max}] \to \Gamma$. Entonces, para calcular la evolución de dicha trayectoria en un proceso de Markov dado se sigue el siguiente algoritmo:

0. En t=0, se calcula el estado inicial de la trayectoria $\hat{S}(0) = \vec{S}_0 \in \Gamma$ a partir de las probabilidades $P\left(s(0) = \vec{S}\right)\ \forall \vec{S} \in \Gamma$, que son condición inicial.

1. Sea $\hat{S}(t_0) = \vec{S} \in \Gamma$ el estado de partida, se calcula el ritmo total de transiciones desde tal estado, $\rho_{total} = \sum_{\vec{S}'}\rho_{\vec{S}\to\vec{S}'}$, sumando sobre todos los posibles estados siguientes $\vec{S}'$.

2. Se generan dos números aleatorios $z_1, z_2 \in [0, 1]$ según la distribución uniforme.

3. Se calcula el tiempo de transición $\delta t = -\frac{log(z_1)}{\rho_{total}}$

4. De entre los procesos futuros posibles $\vec{S}'(j)$ ordenados con $j = 1, 2, ....$ Se calcula que $\vec{S}'(k)$ cumple:

$$\sum_{j=1}^{k-1} \rho_{\vec{S}\to\vec{S}'(j)} < z_2\rho_{total} \leq \sum_{j=1}^{k} \rho_{\vec{S}\to\vec{S}'(j)}$$

   Se toma entonces que $\hat{S}(t_0 + \delta t) = \vec{S}'(k)$.

5. Repetir pasos 1-4 hasta que $t \geq t_{max}$.

De esta forma, seleccionando un ancho de ventana temporal para probabilidades $\triangle t$, *MaBoSS* toma para cada trayectoria $\hat{S}_i$ como probabilidad resultante en cada ventana:

$$P\left(s(t) = \vec{S}' \,|n \triangle t \leq t < (n+1) \triangle t\right) = \frac{t_{\vec{S}'}}{\triangle t}$$
$$\forall n \in \mathbb{N}\, t.\, q.\, (n \triangle t, (n+1) \triangle t) \subset (0, t_{max})$$

Con $t_{\vec{S}'}$ el tiempo que pasa la trayectoria $\hat{S}$ en el estado $\vec{S}'$ para el intervalo de tiempo $(n \triangle t, (n+1) \triangle t)$.

Promediando estas probabilidades para todas las trayectorias $\hat{S}_i$ $\forall i = 1, ..., N$, convergen a la solución de la ecuación maestra de Markov $\vec{P}(t)$ cuando $N \to \infty$.

**Cálculo de la entropía** Una vez calculadas las probabilidades de cada estado $P\left(s(t) = \vec{S}\right)$ $\forall \vec{S} \in \Gamma$ en una ventana temporal $[n \triangle t, (n+1) \triangle t]$. Los autores de *MaBoSS* estiman la entropía en esa ventana como:

$$H(t) = -\sum_{\vec{S}\in\Gamma} P\left(s(t) = \vec{S}\right) \log_2(P\left(s(t) = \vec{S}\right))$$

De forma que $2^{H(t)}$ es una estimación del número de estados que tienen probabilidad no nula en la ventana temporal $[n \triangle t, (n+1) \triangle t]$.

**Cálculo de la entropía de transición** Medida que usan los autores de *MaBoSS* para caracterizar cómo de determinista es la dinámica del sistema, pues cuando $TH = 0$ el sistema solo puede transicionar a un estado dado. Sean las probabilidades de transición entre estados $P_{\vec{S}\to\vec{S}'}$ definidas en la ecuación (2), se define la entropía de transición para cada estado $\vec{S}$ como:

$$TH(\vec{S}) = -\sum_{\vec{S}'} P_{\vec{S}\to\vec{S}'} \log_2(P_{\vec{S}\to\vec{S}'})$$

De forma que en una ventana de tiempo $\tau$, se define como su valor esperado para el sistema en ese intervalo de tiempo:

$$TH(\tau) = \sum_{\vec{S}} P[s(\tau) = \vec{S}]TH(\vec{S})$$

Dadas las definiciones de $H$ y $TH$, cuando el sistema tiende a una distribución estacionaria se pueden caracterizar dos casos:

- Se trata de un punto fijo cuando $H = 0 = TH$.
- Se trata de un ciclo cuando $TH = 0$ pero $H \neq 0$

### 4.1.2. Finalidad de MaBoSS

Si se introduce en *MaBoSS* la siguiente información sobre el modelo a tratar:

- Condición inicial para determinar los estados iniciales de cada trayectoria.
- Reglas lógicas que determinan la posible activación o desactivación de un nodo en función del estado del resto.
- Ritmos de transición para cada nodo. Pueden ser parámetros experimentales como los utilizados en modelos de ecuaciones diferenciales o simplemente estimaciones de ellos. Para cada nodo $i$, estos ritmos son $R_i^{up/down}(\vec{S}) \in [0, \infty)$, que dependen de las reglas lógicas que cumple el estado de la red de partida $\vec{S}$. A partir de ellos, los ritmos de transición de Markov hacia un estado $\vec{S'}$ que solo se diferencia de $\vec{S}$ en el nodo $i$, vienen dados por:

$$\rho_{\vec{S}\to\vec{S'}} = R_i^{up}(\vec{S}\, con\, S_i = 0) \quad \text{ó} \quad \rho_{\vec{S}\to\vec{S'}} = R_i^{down}(\vec{S}\, con\, S_i = 1)$$

Se consigue observar cualitativamente la dinámica del sistema biológico, a pesar de partir de un modelo Booleano.

## 4.2. Variational Cluster Approximation

La Aproximación Variacional en Clúster a la cinética es la extensión del *Cluster Variational Method* (CVM)[22] a la evolución temporal de un sistema, utilizando el tiempo como dimensión extra. El interés que ha despertado en los campos de física estadística y optimización ha dado lugar a una formulación moderna y diversos algoritmos para este problema. La evolución markoviana del sistema en el tiempo se describe en lenguaje variacional [23].

### 4.2.1. Energía libre variacional

Consideremos un modelo que consiste en una red de $N$ nodos. Considerando el caso binario, qe es el más sencillo, cada nodo es un grado de libertad que puede tomar dos valores $\xi_i^{(t)} = \{0, 1\}\ \forall i \in 1, ..., N$ en cualquier momento de una escala temporal discreta $t \in \{0, 1, ..., T\}$. Entonces, el estado del sistema en un tiempo $t$ se denota como $\xi^{(t)} \equiv [\xi_1^{(t)}, ..., \xi_N^{(t)}]$. Así, se puede definir una trayectoria que puede tomar el sistema y la probabilidad de que la siga como:

$$\left(x^{(0)}, ..., x^{(T)}\right) \quad t.q. \quad p\left(x^{(0)}, ..., x^{(T)}\right) = p\left(\xi^{(0)} = x^{(0)}, ..., \xi^{(T)} = x^{(T)}\right) \tag{3}$$

Para una trayectoria se definen también la condición inicial $P^{(0)}\left(x^{(0)}\right)$ (probabilidad de la configuración inicial de la trayectoria $x^{(0)}$) y las probabilidades de transición entre estados $\omega^{(t)}\left(x^{(t+1)}|x^{(t)}\right)$.

Sea $\mathcal{H}(\xi)$ el hamiltoniano que describe el sistema, se expresan la probabilidad en el equilibrio térmico y la función canónica como:

$$P(\xi) = \frac{1}{Z} e^{-\mathcal{H}(\xi)} \qquad y \qquad e^{-F} = Z = \sum_{\xi} e^{-\mathcal{H}(\xi)} \tag{4}$$

donde $F$ es la energía libre. Tal y como se vio en la ecuación (**??**), en el equilibrio la energía libre del sistema es la mínima, de forma que:

$$F = \min_{p} \mathcal{F}(p) = \min_{p} (U - TS) = \min_{p} \sum_{\xi} (p(\xi)\mathcal{H}(\xi) + p(\xi) \ln p(\xi)) \quad con \quad \sum_{\xi} p(\xi) = 1$$

con $\mathcal{F}$ la energía libre variacional. Es fácil de ver que la probabilidad que lo minimiza es la dada por la distribución de Boltzmann, vista en la ecuación (4):

$$\hat{p}(\xi) = \frac{1}{Z} e^{-\mathcal{H}(\xi)} \qquad tal \quad que \qquad F = \mathcal{F}(\hat{p})$$

De esta forma, el variacional de energía libre que se está minimizando se puede expresar en la forma de *Kullback-Leiber*:

$$\mathcal{F}(p) = F + \sum_{\xi} p(\xi) \ln \frac{p(\xi)}{\hat{p}(\xi)} \tag{5}$$

En esta expresión, el término de la energía no da problemas en la factorización, ya que las interacciones suelen ser entre primeros vecinos (corto alcance), lo que permite factorizar la probabilidad. El segundo término representa la entropía, es más dificil de tratar, ya que es una suma de probabilidades sobre todos los $2^N$ posibles estados. Por ello, se hace una aproximación para este segundo término. Bajo la asunción de proceso de Markov, se puede considerar el variacional de energía libre como la divergencia de *Kullback-Leiber*:

$$\mathcal{F}(p) = \sum_{\xi} p(\xi) \ln \frac{p(\xi)}{p\left(\xi^{(0)} = x^{(0)}\right) \prod_{t=0}^{T-1} \omega^{(t)} \left(\xi^{(t+1)} = x^{(t+1)} | \xi^{(t)} = x^{(t)}\right)}$$

Este variacional tiene un mínimo absoluto cuando se satisface la identidad:

$$p(\xi) = p\left(\xi^{(0)} = x^{(0)}\right) \prod_{t=0}^{T-1} \omega^{(t)} \left(\xi^{(t+1)} = x^{(t+1)} | \xi^{(t)} = x^{(t)}\right) \ \forall \xi$$

**Clústeres** Ahora, se define *clúster* $\alpha$ como el subconjunto del grafo de factores (apéndice A) tal que si un nodo factor está en el clúster, entonces todas sus variables también lo están. Es decir, si $a_j \in \alpha$, entonces $\Xi_j \subset \alpha$ con $j \in \mathcal{J}$. También se puede definir como un conjunto de nodos agrupados según un criterio, en este caso según los arcos que relacionan los nodos. Entonces, para un clúster $\alpha$ dado, se define su energía y distribución de probabilidad como:

$$\mathcal{H}_\alpha(\xi_\alpha) = \sum_{a_j \in \alpha} \mathcal{H}_j \left(\{\xi_i\}_{\xi_i \in \Xi_j \subset \{\xi_1, \ldots, \xi_N\}}\right) \qquad p_\alpha(\xi_\alpha) = \sum_{\xi \setminus \xi_\alpha} p(\xi) \tag{6}$$

donde $\xi_\alpha \in \alpha$. De esta forma, la entropía del cluster es $S_\alpha = -\sum_{\xi_\alpha} p_\alpha(\xi_\alpha) \ln p_\alpha(\xi_\alpha)$ por la definición de entropía.

Se definen los cumulantes de la entropía en subclústeres $\beta$ de $\alpha$ como $S_\alpha = \sum_{\beta \subset \alpha} \tilde{S}_\beta$. Entonces, por la definición de la inversión de Möbius [24], se pueden despejar estos cumulantes obteniendo su expresión:

$$\tilde{S}_\beta = \sum_{\alpha \subset \beta} (-1)^{|\alpha \setminus \beta|} S_\alpha = \sum_{\alpha \subset \beta} (-1)^{n_\alpha - n_\beta} S_\alpha \tag{7}$$

con $n_\alpha$ el número de variables en un cluster $\alpha$. Así, a partir de la forma de *Kullback-Leiber* vista en la ecuación (5), se llega al variacional de energía libre en función de cumulantes y como suma sobre todos los clústeres posibles:

$$\mathcal{F}(p) = \sum_{\xi} p(\xi) \mathcal{H}(\xi) - \sum_{\beta} \tilde{S}_\beta \tag{8}$$

Ahora bien, si en vez de sumar sobre todos los clústeres posibles se suma sobre $\Lambda$ un conjunto de clústeres que se considera como maximal, se puede aproximar $\sum_\beta \tilde{S}_\beta \simeq \sum_{\beta \in \Lambda} \tilde{S}_\beta = \sum_{\alpha \in \Lambda} a_\alpha S_\alpha$. Donde $a_\alpha$ son los números de Möbius, tales que aseguran que cada subcluster $\alpha$ se cuente solo una única vez.

Finalmente, con todo lo enunciado, se puede expresar el funcional de energía libre en función de las energías libres de cada clúster:

$$\mathcal{F}(\{p_\alpha | \alpha \in \Lambda\}) = \sum_{\alpha \in \Lambda} a_\alpha \mathcal{F}_\alpha(p_\alpha) = \sum_{\alpha \in \Lambda} a_\alpha \left( \sum_{\xi_\alpha} [p_\alpha(\xi_\alpha) \mathcal{H}_\alpha(\xi_\alpha) + p_\alpha(\xi_\alpha) \ln p_\alpha(\xi_\alpha)] \right) \tag{9}$$

Cuyas distribuciones de probabilidad cumplen condiciones de normalización y ligaduras de compatibilidad:

$$\sum_{\xi_\alpha} p_\alpha(\xi_\alpha) = 1 \qquad p_\beta(\xi_\beta) = \sum_{\xi_{\alpha \setminus \beta}} p_\alpha(\xi_\alpha) \qquad \forall \beta \subset \alpha \in \Lambda \tag{10}$$

#### 4.2.2. Aproximación PQR [25]

La elección del cluster maximal $\Lambda$ da lugar a aproximaciones más o menos bondadosas. Para modelizar el ciclo celular, se ha escogido $PQR$ como cluster maximal, pues el algoritmo da una precisión similar a tomar el conjunto de todos los posibles clústeres.

Según notación utilizada anteriormente, los clústeres P, Q y R se definen para cada nodo $i$ como:

$$\begin{aligned} &\{\xi_i^t, \{\xi_j^t\}_{j \in \partial i}, \xi_i^{t+1}\} \subset \Xi_{P_i} \\ &\{\xi_i^t, \xi_j^t, \xi_i^{t+1}, \xi_j^{t+1}\} \subset \Xi_{Q_i j} \qquad \forall j \in \partial i \\ &\{\xi_i^t, \xi_i^{t+1}, \{\xi_j^{t+1}\}_{j \in \partial i}\} \subset \Xi_{R_i} \end{aligned}$$

con $\partial i$ el conjunto de vecinos del nodo $i$. Así, las distribuciones de probabilidad de cada clúster definidas en la ecuación (6) para cada nodo, dado un paso de una trayectoria $\left(\xi^{(t)} = x^{(t)}, \xi^{(t+1)} = x^{(t+1)}\right)$,

queda:

$$
\begin{aligned}
P_i^{(t)}\left(x_i^{(t+1)}, x_i^{(t)}, \{x_j^{(t)}\}_{j\in\partial i}\right) &= p_{P_i}\left(\xi_i^t = x_i^{(t)}, \{\xi_j^t\}_{j\in\partial i} = \{x_j^{(t)}\}_{j\in\partial i}, \xi_i^{t+1} = x_i^{(t+1)}\right)\\
Q_{ij}^{(t)}\left(x_i^{(t+1)}, x_j^{(t+1)}, x_i^{(t)}, x_j^{(t)}\right) &= p_{Q_{ij}}\left(\xi_i^t = x_i^{(t)}, \xi_j^t = x_j^{(t)}, \xi_i^{t+1} = x_i^{(t+1)}, \xi_j^{t+1} = x_j^{(t+1)}\right)\\
R_i^{(t)}\left(x_i^{(t+1)}, \{x_j^{(t+1)}\}_{j\in\partial i}, x_i^{(t)},\right) &= p_{R_i}\left(\xi_i^t = x_i^{(t)}, \xi_i^{t+1} = x_i^{(t+1)}, \{\xi_j^{t+1}\}_{j\in\partial i} = \{x_j^{(t+1)}\}_{j\in\partial i}\right)
\end{aligned}
$$

Además, la entropía del cluster $\alpha \equiv PQR$ viene dada por los números de Möbius $a_\alpha \equiv a_{PQR}$ y entropía de los subclústeres $\{P, Q, R, S, T, U, V, Z\} \equiv \beta \subset PQR$ :

$$
\begin{aligned}
\mathcal{S}_{PQR} = \textstyle\sum_{t=0}^{T-1}\Big(\sum_i\Big[\mathcal{S}[P_i^{(t)}] + \mathcal{S}[R_i^{(t)}] + (|\partial i|-1)\,\mathcal{S}[V_i^{(t)}]\Big] + \sum_{i,j\in\partial i}\mathcal{S}[Q_{ij}^{(t)}]\\
- \textstyle\sum_{i,j\in\partial i}\{\mathcal{S}[T_{i,ij}^{(t)}] + \mathcal{S}[U_{ij,i}^{(t)}]\}\Big) - \sum_{t=1}^{T-1}\Big(\sum_i \mathcal{S}[S_i^{(t)}] - \sum_{ij}\mathcal{S}[Z_{ij}^{(t)}]\Big)
\end{aligned}
\tag{11}
$$

Finalmente, de la ecuación (9) se deduce el variacional de energía libre:

$$
\begin{aligned}
\mathcal{F}_{PQR} = &\sum_{t=0}^{T-1}\sum_i \sum_{x_i^{t+1}, x_{i,\partial i}^t} P_i^{(t)}(x_i^{t+1}, x_{i,\partial i}^t)\ln\frac{P_i^{(t)}(x_i^{t+1}, x_{i,\partial i}^t)}{\omega_i^{(t)}\left(x_i^{t+1}|x_{i,\partial i}^t\right)S_i^{(t)}(x_{i,\partial i}^t)}\\
&+\sum_{t=0}^{T-1}\sum_{ij}\sum_{x_{i,j\in\partial i}^{t+1}, x_{i,j\in\partial i}^t} Q_{ij}^{(t)}(x_{i,j}^{t+1}, x_{i,j}^t)\ln\frac{Q_{ij}^{(t)}(x_{i,j}^{t+1}, x_{i,j}^t)Z_{ij}^{(t)}(x_{i,j}^t)}{T_{i,ij}^{(t)}(x_i^{t+1}, x_{i,j}^t)T_{j,ji}^{(t)}(x_j^{t+1}, x_{j,i}^t)}\\
&+\sum_{t=0}^{T-1}\sum_i\sum_{x_{i,\partial i}^{t+1}, x_i^t} R_i^{(t)}(x_{i,\partial i}^{t+1}, x_i^t)\ln\frac{R_i^{(t)}(x_{i,\partial i}^{t+1}, x_i^t)V_i^{(t)}(x_i^{t+1}, x_i^t)^{|\partial i|-1}}{\prod_{j\in\partial i}U_{ij,i}^{(t)}(x_{i,j}^{t+1}, x_i^t)}
\end{aligned}
$$

Entonces, la condición de mínimo aparece cuando todos los argumentos de los logaritmos se anulan. Es decir, se llega a que $\forall t, i, j \in \partial i$ se cumplen las ecuaciones:

$$
P_i^{(t)}(x_i^{t+1}, x_{i,\partial i}^t) = \omega_i^{(t)}\left(x_i^{t+1}|x_{i,\partial i}^t\right)S_i^{(t)}(x_{i,\partial i}^t) \tag{12}
$$

$$
Q_{ij}^{(t)}(x_{i,j}^{t+1}, x_{i,j}^t) = \frac{T_{i,ij}^{(t)}(x_i^{t+1}, x_{i,j}^t)T_{j,ji}^{(t)}(x_j^{t+1}, x_{j,i}^t)}{Z_{ij}^{(t)}(x_{i,j}^t)} \tag{13}
$$

$$
R_i^{(t)}(x_{i,\partial i}^{t+1}, x_i^t) = \frac{\prod_{j\in\partial i}U_{ij,i}^{(t)}(x_{i,j}^{t+1}, x_i^t)}{V_i^{(t)}(x_i^{t+1}, x_i^t)^{|\partial i|-1}} \tag{14}
$$

#### 4.2.3. Algoritmo

Para resolver este problema planteado, es necesario un proceso iterativo a lo largo de los pasos temporales $t = 0, ..., T-1$. Se va a utilizar el siguiente:

0. En primer lugar, se evalúan las condiciones iniciales:, obteniendo $S^{(0)}$ y $Z^{(0)}$ para todos los posibles estados de las variables de cada cluster S y Z:

$$
S_i^{(0)}(x_{i,\partial i}) = p_i^{(0)}(x_i)\prod_{j\in\partial i}p_j^{(0)}(x_j) \qquad Z_{ij}^{(t)}(x_{i,j}^t) = p_i^{(0)}(x_i)p_j^{(0)}(x_j) \qquad \forall i, j \in \partial i
$$

1. Conocidas las probabilidades de transición entre todos los estados, tal y como se han definido antes de la ecuación (4), se calculan las distribuciones de probabilidad del cluster $P$ según la ecuación (12). Además, se calculan las distribuciones de los subclústeres $T$ y $V$ a partir de las siguientes ligaduras de compatibilidad, que se deducen a partir de las ecuaciones (10):

$$\sum_{x_{\partial i \setminus j}} P_i^{(t)}(x_i^{t+1}, x_{i,\partial i}^t) = T_{i,ij}^{(t)}(x_i^{t+1}, x_{i,j}^t) \quad \sum_{x_{\partial i}} P_i^{(t)}(x_i^{t+1}, x_{i,\partial i}^t) = V_i^{(t)}(x_i^{t+1}, x_i^t) \quad \forall i, j \in \partial i$$

2. Ahora, se calculan las distribuciones de probabilidad del clúster $Q$ en el tiempo $t$ según la ecuación (13). Con ellas, se pueden calcular las distribuciones de los subclústeres $U$ y $Z$ (en este caso, en el tiempo $t+1$), a partir de las ligaduras de compatibilidad:

$$\sum_{x_j} Q_{ij}^{(t)}(x_{i,j}^{t+1}, x_{i,j}^t) = U_{ij,i}^{(t)}(x_{i,j}^{t+1}, x_i^t) \qquad \sum_{x_{i,j}} Q_{ij}^{(t)}(x_{i,j}^{t+1}, x_{i,j}^t) = Z_{ij}^{(t+1)}(x_{i,j}^{t+1}) \qquad \forall i, j \in \partial i$$

3. Finalmente, se calculan las distribuciones de probabilidad del clúster $R$ en el tiempo $t$ según la ecuación (14). Con ellas, se pueden calcular las distribuciones de probabilidad del subclúster $S$ en el tiempo $t+1$, a partir de las ligaduras de compatibilidad:

$$\sum_{x_i} R_i^{(t)}(x_{i,\partial i}^{t+1}, x_i^t) = S_i^{(t+1)}(x_{i,\partial i}^{t+1}) \qquad \forall i$$

4. Se repiten los pasos 1-3 en el paso de tiempo $t+1$.

De esta forma, se obtienen todas las distribuciones de probabilidad de cada uno de los clústeres a tener en cuenta en la aproximación $PQR$.

#### 4.2.4. Relación entre ritmos de *MaBoSS* y *CVM*

En el trabajo se trata con modelos que han sido implementados en primer lugar para *MaBoSS*. Entonces, se han obtenido los distintos $\omega_i^{(t)}\left(x_i^{t+1}|x_{i,\partial i}^t\right)$ a partir de los rates $\rho_{\vec{S}\to\vec{S}'}$ definidos para el algoritmo de Gillespie. Se obtienen mediante la expresión:

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

Donde $\delta_{x_i,y_i}$ es la delta de Kronecker. El parámetro $\tau$ 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 $\tau \to 0$.

# 5. Resultados

El objetivo de este trabajo ha sido reproducir el comportamiento de las ciclinas del ciclo celular a partir del *Variational Cluster Approximation*, una aproximación determinista para dinámicas estocásticas en redes. Para ello, se ha utilizado la aproximación PQR ya presentada.

Para evaluar la validez de los resultados, se han comparado con el entorno *MaBoSS 2.0*, que es una aplicación de un algoritmo de Monte-Carlo cinético (Gillespie) a dinámicas booleanas asíncronas de tiempo continuo. Antes de nada se han transformado los ritmos de *MaBoSS* tal y como se expone en el apartado 4.2.4.

Antes de aplicar cada uno de los métodos, se ha estudiado cómo relacionar sus resultados para poder realizar comparaciones posteriores. A partir de MaBoSS, se han obtenido las probabilidades de activación de los nodos, además de la entropía del sistema en cada ventana temporal. En el *Cluster Variational Method*, estos resultados vienen representados como:

- Las probabilidades de activación de las ciclinas se han obtenido a partir del clúster A, que se define para cada nodo $i$ como:

  $$\xi_i^t \in \Xi_{A_i}$$

  Entonces por definición, su distribución de probabilidad para el nodo $i$ activado se ha calculado como:

  $$A_i^{(t)}(x_i \equiv 1) = \sum_{x_{\partial i}} \sum_{y_i} P_i^{(t)}(y_i, x_{\partial i}, x_i \equiv 1) \; \forall i$$

- La entropía para cada ventana de tiempo que calcula MaBoSS se ha interpretado como los términos entrópicos de la ecuación (11) de la entropía de caminos que dependen únicamente del estado de nodos en una ventana de tiempo dada, sin tener en cuenta el resto de tiempos.

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

## 5.1. Modelo de juguete

Antes de aplicar un modelo del ciclo celular a los métodos, se ha utilizado un modelo sencillo de juguete [1]. Así se han conseguido corregir errores en la aplicación del algoritmo de la aproximación PQR y se ha comprobado que las relaciones entre *MaBoSS* y *CVM* eran las esperadas.

El modelo está formado por tres nodos: $A$, $B$ y $C$. Éstos se relacionan entre sí mediante unas reglas booleanas que rigen su activación, estas se encuentran en el apéndice B (Figura 3).

### 5.1.1. Dos nodos: ciclo

En lugar de reproducir directamente el juguete de 3 nodos, se ha comenzado por una simplificación de este modelo. Se ha tomado el modelo de juguete forzando el nodo $C$ a 1, que es equivalente a eliminar el enlace entre los estados $ABC \equiv 001$ y $ABC \equiv 000$. De esta forma, los estados del sistema describen un ciclo en el que $11 \to 01 \to 00 \to 10$, como se puede deducir de la figura 3.

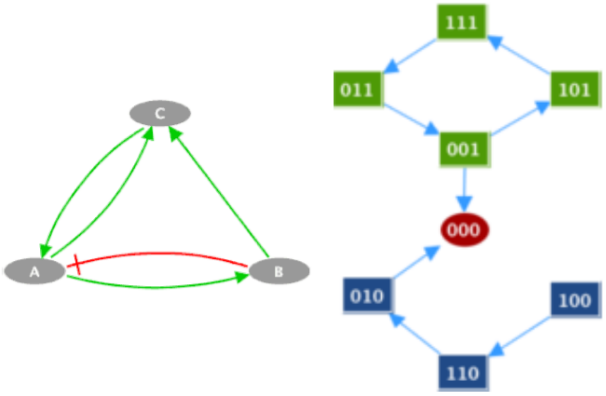


Figura 3: Modelo de juguete de tres nodos [1]: mientras el nodo $C$ se encuentre activo el estado del sistema entra en un ciclo con la activación de los nodos de $A$ y $B$, con una cierta probabilidad de escape al estado $ABC \equiv 000$. Sin embargo, si se parte del estado $ABC \equiv 100$ el sistema sigue una trayectoria hasta el estado $ABC \equiv 000$. De esta forma, el estado $ABC \equiv 000$ es un atractor, al que se llega desde el ciclo con $C$ activo o directamente partiendo de un estado con $C$ desactivado.

**MaBoSS** Se ha ejecutado el entorno MaBoSS con el modelo de 2 nodos, tomando 500.000 trayectorias y una ventana temporal de 0,01. Los resultados se presentan en la figura 4.

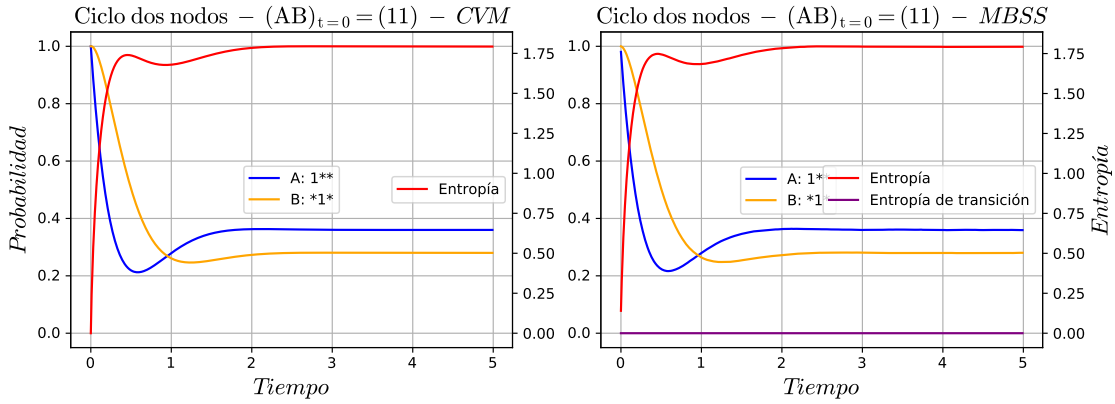


Figura 4: A la derecha, evolución temporal del modelo de juguete de dos nodos en MaBoSS. A la izquierda, en CVM. Se han dibujado las probabilidades de activación de los nodos $A$ y $B$ siguiendo el eje de ordenadas de la izquierda y las entropías siguiendo el eje de la derecha. Partiendo de la condición inicial $AB \equiv 11$, se observa cómo la primera probabilidad en disminuir es la del nodo $A$, tal y como se esperaba la primera transición es $11 \to 01$. Después del descenso de las probabilidades producido por las transiciones $11 \to 01$ y $01 \to 00$, se aprecia cómo las distribuciones de probabilidad de ambos nodos se vuelven estacionarias. Es decir, como MaBoSS obtiene las probabilidades del promedio de distintas trayectorias y al tratarse de un bucle, con el tiempo la probabilidad de encontrarse en un estado del bucle se hace constante, se pierde la información de la condición inicial. En cuanto a las entropías, cuando las probabilidades de los nodos son constantes (sistema en estado estacionario), la entropía de transición $TH$ se anula y la entropía $H$ es distinta de cero, lo que coincide con la descripción de ciclo del apartado 4.1.1.

**Variational Cluster Approximation** Se ha aplicado la aproximación PQR del *Cluster Variational Method* al modelo de juguete de dos nodos. Para ello se ha tomado $\tau = 0{,}01$, pues de esta forma se pueden relacionar los resultados de MaBoSS con los del CVM (Figura 4). Se observa cómo tanto las probabilidades de los nodos $A$ y $B$ como la entropía coinciden con los resultados de MaBoSS en forma y magnitud. Se han conseguido entonces reproducir los resultados de MaBoSS pero

el modelo es demasiado sencillo, puede que no se manifiesten errores por tratarse simplemente de dos nodos que son vecinos entre sí.

### 5.1.2. Tres nodos: ciclo con escape

Una vez tratado el modelo sencillo con dos nodos, se han hecho las mismas comparaciones entre MaBoSS y CVM para el modelo de juguete de tres nodos completo. Para explotar al máximo este modelo, se han probado tres configuraciones de condiciones iniciales y parámetros:

- *Ciclo escape rápido*: tomando $ABC \equiv 111$ como condición inicial y un ritmo de inhibición del nodo $C$ muy alto se debería observar inicialmente el ciclo ya estudiado en el modelo de dos nodos. Sin embargo, el sistema escapa con un ritmo rápido al estado atractor $ABC \equiv 000$ tal y como se observa en la figura 3, pues la inhibición alta del nodo $C$ causa una alta probabilidad de escapar en cada vuelta del ciclo.
- *Escape lento desde el bucle*: tomando $ABC \equiv 111$ como condición inicial y un ritmo de inhibición del nodo C bajo se debería observar lo mismo que en la configuración anterior, pero en este caso se tarda más tiempo en observar el escape al atractor $ABC \equiv 000$.
- *Trayectoria* $C \equiv 0$: tomando $ABC \equiv 100$ como condición inicial y un ritmo de inhibición del nodo $C$ cualquiera, se observa en la figura 3 como el sistema describe una trayectoria desde el estado inicial hasta el atractor $ABC \equiv 000$.

Se ha reproducido este modelo con los dos métodos tratados:

**MaBoSS** Se ha ejecutado el entorno MaBoSS con el juguete de 3 nodos, tomando 500.000 trayectorias y una ventana temporal de 0.01 (Figuras 5, 6 y 7).

**Variational Cluster Approximation** Se ha aplicado la aproximación PQR del *Cluster Variational Method* al modelo de juguete de tres nodos. Para ello se ha tomado $\tau = 0{,}01$, pues de esta forma se pueden relacionar los resultados de MaBoSS con los del CVM (Figuras 5, 6 y 7). Las probabilidades de los tres nodos y la entropía coinciden con los resultados de MaBoSS en forma y magnitud. Se han conseguido reproducir los resultados de MaBoSS bajo estas condiciones.

En este modelo de juguete, los tres nodos son vecinos entre sí, entonces el clúster de los vecinos coincide con la totalidad del grafo. Es decir, el conjunto de clústeres que se consideran maximales para la aproximación PQR comprende todos los clústeres posibles del grafo. De esta forma, en este caso el CVM proporciona resultados exactos, pues no se está realizando la aproximación. Se ha comprobado esta afirmación calculando la entropía exacta.

Para el cálculo de la entropía exacta hay que recordar que para dinámicas de Markov la probabilidad de una trayectoria viene dada en función de las probabilidades de transición por la expresión general:

$$p\left(x^{(0)},...,x^{(T)}\right) = p^{(0)}\left(x^{(0)}\right)\prod_{t=0}^{T-1}\omega^{(t)}\left(x^{t+1}|x^{t}\right) = p^{(0)}\left(x^{(0)}\right)\prod_{t=0}^{T-1}\prod_{i}\omega_{i}^{(t)}\left(x_{i}^{t+1}|x_{i,\partial i}^{t}\right)$$

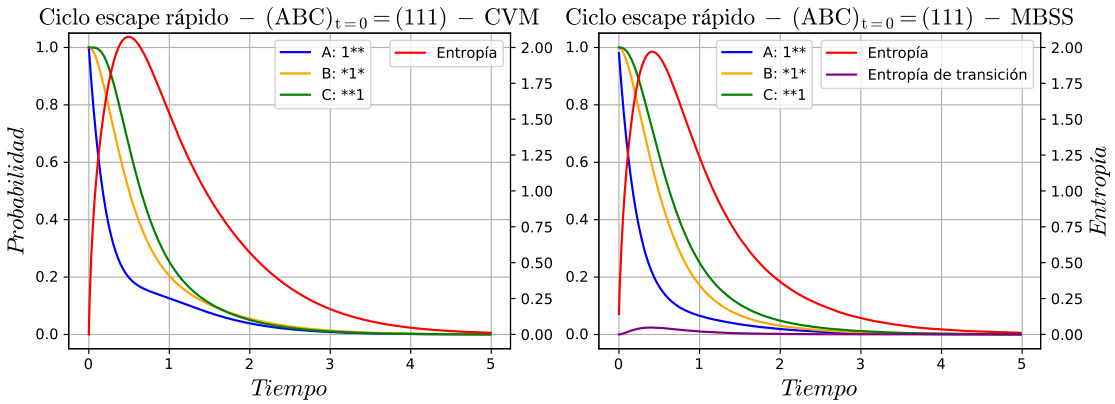


Figura 5: A la derecha, evolución temporal del modelo de juguete de tres nodos en MaBoSS para la configuración de escape rápido. A la izquierda, en CVM. Partiendo de la condición inicial $ABC \equiv 111$, se observa como la primera probabilidad en disminuir es la del nodo $A$, tal y como ocurría con el juguete de dos nodos. Pero además, esta vez la actividad del nodo $C$ desciende prácticamente a la par que la del nodo $B$. Es decir, siempre que el sistema llega al estado $ABC \equiv 001$, pasa al atractor $ABC \equiv 000$. Finalmente, las probabilidades de todos los nodos convergen a 0, tal que el sistema acaba en el atractor en todas las trayectorias. Este suceso también se observa en las entropías, cuando el sistema se encuentra en estado estacionario, las entropías $TH$ y $H$ se anulan, lo que coincide con la descripción de punto fijo del apartado 4.1.1.

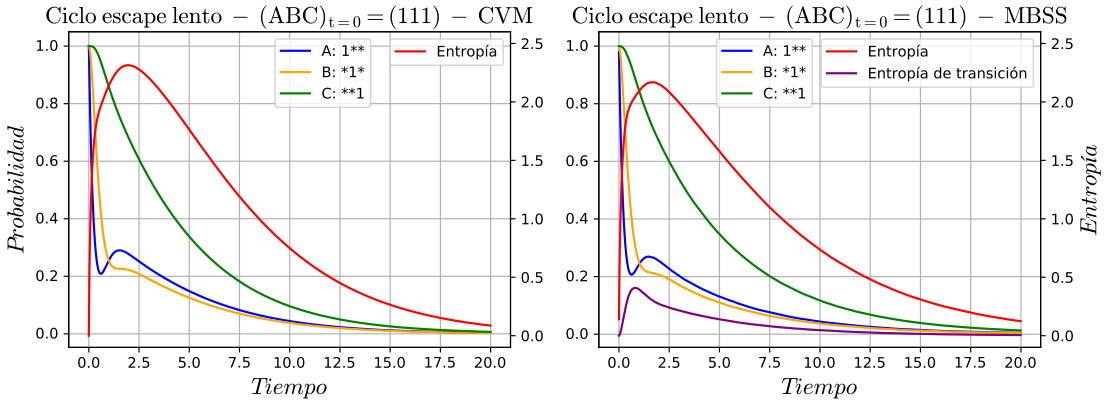


Figura 6: A la derecha, evolución temporal del modelo de juguete de tres nodos en MaBoSS para la configuración de escape lento. A la izquierda, en CVM. Partiendo de la condición inicial $ABC \equiv 111$, parece un inicio similar al escape rápido, con una disminución de los nodos $A$ y $B$. Sin embargo, como el ritmo de inhibición de $C$ es lento la probabilidad de este nodo tarda mucho más en disminuir. Además, parte de las trayectorias del sistema se quedan en el ciclo de los nodos $A$ y $B$ hasta que escapan en una de las iteraciones. Las probabilidades de todos los nodos convergen a 0 pero más lentamente, el sistema acaba también en el atractor en todas las trayectorias. Este suceso también se observa en las entropías, cuando el sistema se encuentra en estado estacionario las entropías $TH$ y $H$ se anulan, lo que coincide con la descripción de punto fijo del apartado 4.1.1.

Entonces, la probabilidad en un tiempo $t$ de un estado $x^{(t)} = \xi^{(t)}$ vendrá dada por las probabilidades de todas las trayectorias que acaban en ese estado. Es decir, se puede calcular según la

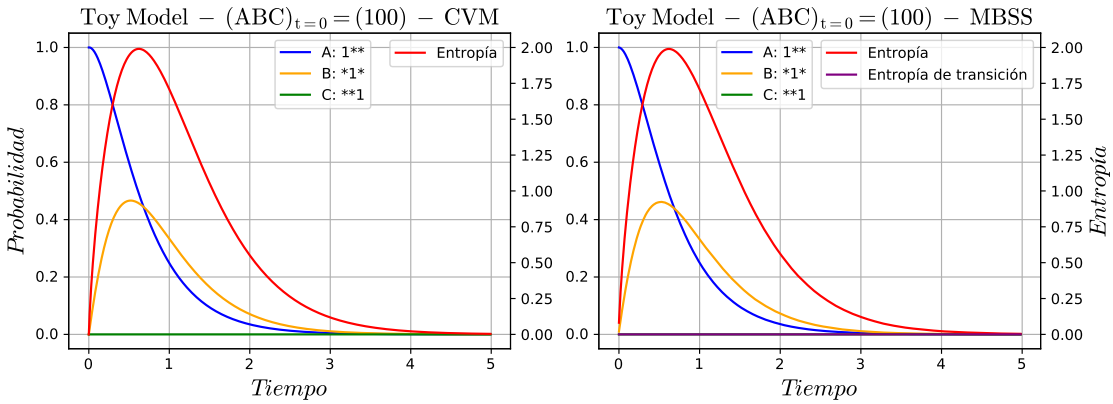


Figura 7: A la derecha, evolución temporal del modelo de juguete de tres nodos en MaBoSS para la configuración de trayectoria con $C \equiv 0$. A la izquierda, en CVM. Las probabilidades que se observan coinciden con lo esperado, partiendo de la condición inicial $ABC \equiv 100$ la probabilidad de B sube y la de A baja, pues todas las trayectorias llegan al estado $ABC \equiv 010$ pasando por el $ABC \equiv 110$. Finalmente, todas las trayectorias van a parar al atractor $ABC \equiv 000$, que es un punto fijo. Este suceso también se observa en las entropías, cuando el sistema se encuentra en estado estacionario las entropías $TH$ y $H$ se anulan, lo que coincide con la descripción de punto fijo del apartado 4.1.1.

| Método | $(ABC)_{t=0} \equiv 111$ | $(ABC)_{t=0} \equiv 100$ |
|---|---|---|
| MBSS | 1.6587 | 1.309100 |
| CVM | 1.5969 | 1.28988560 |
| Exacto | 1.6422 | 1.28988557 |

Cuadro 1: Entropía exacta del sistema en el tiempo t=20. Se ha comparado con *CVM* y *MaBoSS*, observando que cuando se toma $(ABC)_{t=0} \equiv 111$ como condición inicial el resultado de obtenido con el *CVM* no coincide con el exacto. Esto podría estar relacionado con que en la trayectoria hay un bucle. Se ha tomado también la condición inicial $(ABC)_{t=0} \equiv 100$ para la que las trayectorias siguen un árbol, observando como la entropía resultante del *CVM* coincide con la exacta, mientras que la resultante de *MaBoSS* se aleja de ella.

expresión matricial:

$$p\left(x^{(t)} = \xi^{(t)}\right) = \mathcal{W}^t p\left(x^{(0)} = \xi^{(0)}\right)$$

Donde $\xi^{(0)}$ es el estado inicial del sistema y se define la matriz $\left(\omega^{(t)}\left(\xi^{t+1}_{(j)}|\xi^{t}_{(i)}\right)_{ij}\right) = \mathcal{W}^{(t)}$. Una vez calculadas las probabilidades de todos los estados posibles del sistema en el tiempo $t$, la entropía exacta se calcula fácilmente con la suma a todos los estados $H_{exacta}(t) = -\sum_i p(\xi^{(t)}_{(i)}) \log_2(p(\xi^{(t)}_{(i)}))$. La entropía exacta calculada en el cuadro 1 es el mejor punto de referencia posible y se observa claramente que la herramienta *MaBoSS* no es exacta. Es decir, esta herramienta sirve como referencia solo hasta cierto punto, en ocasiones *CVM* aportará resultados más exactos.

## 5.2. Ciclo celular

Finalmente se han aplicado los métodos *CVM* y *MaBoSS* al ciclo celular. El modelo utilizado [5] está formado por las ciclinas y proteínas reguladoras presentadas en el apartado 2.1, relacionadas

entre sí mediante unas reglas booleanas que rigen su producción. Se enuncian en el apéndice C.

**MaBoSS** Se ha ejecutado el entorno MaBoSS con el modelo del ciclo celular, tomando 1,000,000 trayectorias y una ventana temporal de 0,2. Los resultados se exponen en la figura 8.

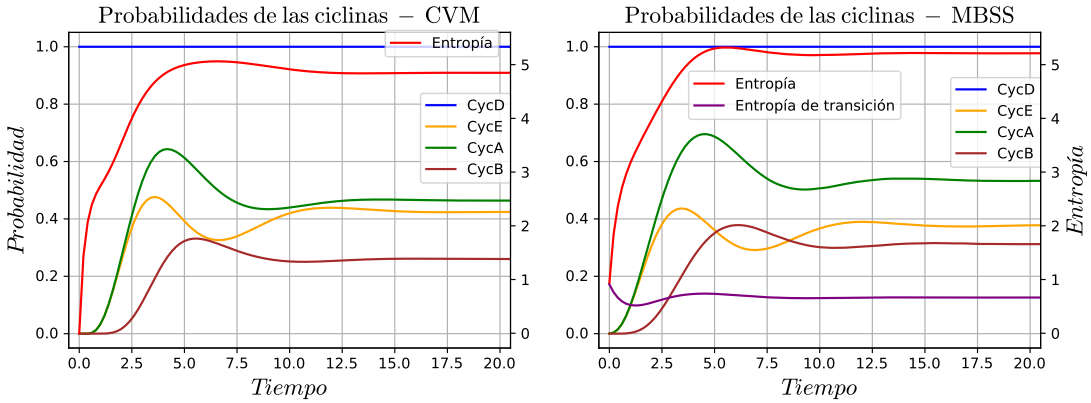


Figura 8: A la derecha, resultados del ciclo celular en MaBoSS. A la izquierda, en CVM. Los primeros picos de concentración de las ciclinas coinciden con las fases del ciclo celular presentadas en la figura 1: el primer pico en observarse es el de *CycE* que coincide con la fase *S*, el segundo es de la ciclina *CycA* que coincide con el inicio de la fase *G2* y finalmente el de *CycB* al comienzo de la mitosis. La magnitud a la que llegan los picos de las distribuciones de probabilidad y los valores asintóticos no coinciden pues se trata de métodos basados en aproximaciones distintas. Cabe destacar de todas formas que hay acuerdo en el orden de los eventos y los valores de las probabilidades no difieren mucho, es suficiente para extraer información sobre el comportamiento biológico del modelo. Tras los primeros picos se observa una oscilación amortiguada, que corresponde a los ciclos celulares sucesivos de la población celular, donde la desincronización entre células da lugar a esta amortiguación. Finalmente las entropías tienen carácter asintótico, que corresponde a un atractor cíclico. Esta interpretación coincide con el modelo utilizado, un ciclo celular en el que las ciclinas *CycD* están siempre presentes.

**Variational Cluster Approximation** Se ha aplicado la aproximación PQR del *Cluster Variational Method* al modelo del ciclo celular, tomando $\tau = 0{,}2$ para poder relacionar los resultados con los de *MaBoSS*, pues así coincide con el valor de la ventana temporal en *MaBoSS* (Figura 8). Se han obtenido distribuciones de probabilidad de forma y picos parecidos para las ciclinas. Sin embargo las magnitudes no se acercan, posiblemente por tratarse de la aproximación PQR.

Se ha estudiado el uso de distintas $\tau$ para este modelo del ciclo celular. El parámetro $\tau$ relaciona los ritmos de activación y desactivación de MaBoSS con los del CVM. Además, hace el papel de ventana temporal para la aproximación PQR del *Cluster Variational Method*, de forma que en el límite $\tau \to 0$ se pasa a tratar con procesos de Markov de tiempo continuo. En este caso, la aproximación PQR se aleja de aproximaciones menores como tomar el clúster $\{\xi_i^t, \{\xi_j^t\}_{j\in\partial i}, \xi_i^{t+1}, \{\xi_j^{t+1}\}_{j\in\partial i}\} \subset \Xi_{M_i}$ y se acerca a aproximaciones peores como la PQ [25]. Entonces, no existe un único $\tau$ para hacer el paso de tiempo continuo a discreto. En la figura 9, de todo el espectro de valores de $\tau$, se han buscado los resultados más cercanos al ciclo celular real para el *CVM*.

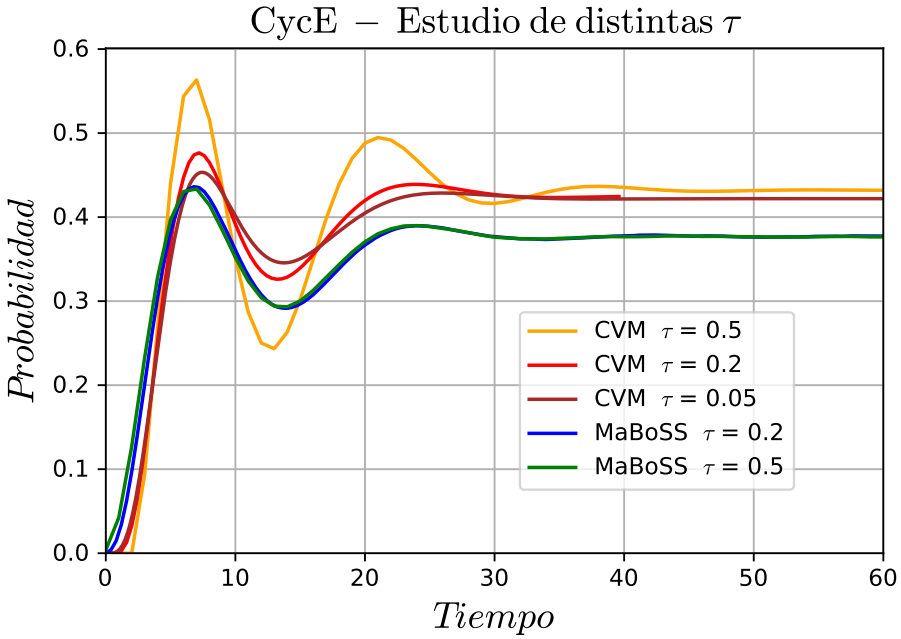


Figura 9: Evolución temporal de la distribución de probabilidad para la ciclina *CycE*. En primer lugar, se observa cómo para *MaBoSS* al variar el ancho de ventana temporal no se aprecian cambios en los picos de probabilidad. Para el *CVM* además de la diferencia en magnitud al tratarse de la aproximación PQR, al disminuir $\tau$ la distribución tiene picos menos marcados, alejándose así de los resultados de *MaBoSS* y de lo esperado del ciclo celular. Esto último ocurre porque al disminuir $\tau$ la aproximación PQR se acerca a la PQ, una aproximación menos precisa. Además, al aumentar el parámetro a $\tau = 0{,}5$ se observan más marcadas las oscilaciones de los ciclos celulares sucesivos, lo que se puede considerar más cercano al ciclo celular real. A su vez, no se aleja demasiado de la forma de los resultados de *MaBoSS* ni de las probabilidades estacionarias para *CVM* con $\tau = 0{,}2$. Todo esto indica que se puede tomar el método *CVM* para otros problemas sin tener en cuenta *MaBoSS*, obteniendo resultados más acertados.

### 5.2.1. Estudio sobre el tiempo de ejecución del programa

Una motivación inicial para estudiar la aplicación del método *CVM* a redes de regulación era que se esperaba encontrar una herramienta mucho más eficiente que métodos clásicos como por ejemplo el algoritmo de Gillespie, plasmado en *MaBoSS*. Para justificar este trabajo, es necesario hacer un pequeño estudio sobre tiempos de ejecución bajo resultados de calidad comparable, por lo que se han tomado las ejecuciones que han resultaron en las figuras presentadas.

Se ha programado el *Cluster Variational Method* en C, por lo que se ha tenido en cuenta el tiempo de de ejecución de programa. Además, el método *MaBoSS* se ha utilizado a través de *Bash*, así que el tiempo que se ha tenido en cuenta va desde que se pulsa *Enter* en la consola hasta que acaba la ejecución.

Como se aprecia en el cuadro 2, el tiempo de ejecución es mucho menor y parece que escala mucho menos con el número de nodos del sistema. En la práctica, hay redes de regulación mucho mayores que las tratadas en el trabajo, para las cuales no es factible el uso de métodos clásicos como

| | CVM | MBSS |
|---|---|---|
| Ciclo dos nodos | 0.2152 s | 13.987 s |
| Ciclo escape rápido | 0.4022 s | 10.584 s |
| Ciclo escape lento | 0.6129 s | 453.708 s |
| Ciclo celular | 0.4264 s | 149.642 s |

Cuadro 2: Se puede observar claramente que el *CVM* ha sido mucho más rápido a la hora de obtener los datos para las gráficas presentadas en el trabajo.

*MaBoSS* pero sí con el *CVM*.

# 6. Conclusiones

Se ha conseguido aplicar por primera vez la técnica de aproximación variacional en clúster a al cinética de redes de regulación genética, introduciendo en la biología de sistemas de este tipo una nueva herramienta. Además se han cumplido los objetivos propuestos para el estudio del *CVM*: obtener resultados coherentes con la literatura sobre el ciclo celular, observar que *CVM* es una herramienta más eficiente que otros enfoques más clásicos y preparar el camino hacia la aplicación de este método a redes mayores y más complejas en el futuro.

El resultado del trabajo ha sido la reproducción de redes de regulación de complejidad ascendente, comprobando que a partir de la aproximación PQR se obtienen resultados que coinciden perfectamente con los de la herramienta *MaBoSS*. También inesperadamente, al tratar con sistemas más complejos como el modelo del ciclo celular, para ciertos valores del parámetro $\tau$ los resultados del *CVM* se pueden considerar como más representativos de la literatura que los de *MaBoSS*. Sin embargo, al utilizar un modelo sin parámetros experimentales los resultados son cualitativos, esto no ha permitido realizar comparaciones de resultados tan consistentes como para resultados cuantitativos.

A partir de los resultados presentados en este trabajo, se ha llegado a la conclusión de que el método *CVM* permite estudiar la evolución de redes de regulación genética. Esto implica que se dispone de una herramienta con un potencial muy importante en el estudio de este tipo de sistemas, pues aporta enfoque distinto de las herramientas más clásicas y se prevé incluso un mejor funcionamiento.

Queda abierto el estudio más exhaustivo de esta herramienta, utilizando distintos modelos discretos más complejos y de mayor tamaño. Esto comprende un estudio del tiempo de ejecución utilizando un sistema conocido como el Delta-Notch o también comparando con aproximaciones sistemáticamente mejores que la PQR dentro del método *CVM*, como por ejemplo la aproximación del clúster maximal M [25].

## Referencias


[1] 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.

[2] Kim Sneppen y Giovanni Zocchi. "What is special about living matter?" En: *Physics in Molecular Biology*. Cambridge University Press, 2005, págs. 4-7.

[3] NIH. *Ciclo Celular*. URL: `https://www.genome.gov/es/genetics-glossary/Ciclo-celular`.

[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] Adrien Fauré y col. "Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle". En: *Bioinformatics* 22.14 (jul. de 2006), e124-e131. ISSN: 1367-4803.

[6] Lodish et al. *Biología celular y molecular*. Buenos Aires: Médica Panamericana. ISBN: 950-06-1974-3.

[7] P. W. Hinds y col. "Regulation of retinoblastoma protein functions by ectopic expression of human cyclins". En: *Cell* 70.6 (sep. de 1992), págs. 993-1006.

[8] B. Henglein y col. "Structure and cell cycle-regulated transcription of the human cyclin A gene". En: *Proc Natl Acad Sci U S A* 91.12 (jun. de 1994), págs. 5490-5494.

[9] D. Coverley, H. Laman y R. A. Laskey. "Distinct roles for cyclins E and A during DNA replication complex assembly and activation". En: *Nat Cell Biol* 4.7 (jul. de 2002), págs. 523-528.

[10] H. L. Ford y A. B. Pardee. "Cancer and the cell cycle". En: *J Cell Biochem* Suppl 32-33 (1999), págs. 166-172.

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

[12] Adrien Fauré y col. "Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle". En: *Bioinformatics* 22.14 (jul. de 2006), e124-e131. ISSN: 1367-4803.

[13] D. W. Goodrich y col. "The retinoblastoma gene product regulates progression through the G1 phase of the cell cycle". En: *Cell* 67.2 (oct. de 1991), págs. 293-302.

[14] Sara Molatore y Natalia S. Pellegata. "Chapter 13 - The MENX Syndrome and p27: Relationships with Multiple Endocrine Neoplasia". En: *Neuroendocrinology*. Ed. por Luciano Martini. Vol. 182. Progress in Brain Research. Elsevier, 2010, págs. 295-320.

[15] M.A. Morgan DO Keaton. "The Cell Cycle: Principles of Control (Primers in Biology)". En: *Cell Div* 2.27 (2007), págs. 891-921.

[16] R. Albert y J. Thakar. "Boolean modeling: a logic-based dynamic approach for understanding signaling and regulatory networks and for making useful predictions". En: *Wiley Interdiscip Rev Syst Biol Med* 6.5 (2014), págs. 353-369.

[17] A. Naldi y col. “Cooperative development of logical modelling standards and tools with CoLoMoTo”. En: *Bioinformatics* 31.7 (abr. de 2015), págs. 1154-1159.

[18] N. Le Novère. “Quantitative and logic modelling of molecular and gene networks”. En: *Nat Rev Genet* 16.3 (mar. de 2015), págs. 146-158.

[19] Rahuman S Malik-Sheriff y col. “BioModels — 15 years of sharing computational models in life science”. En: *Nucleic Acids Research* 48.D1 (ene. de 2020), págs. D407-D415.

[20] Gautier Stoll y col. “MaBoSS 2.0: an environment for stochastic Boolean modeling”. En: *Bioinformatics* 33.14 (mar. de 2017), págs. 2226-2228. ISSN: 1367-4803.

[21] D. T. Gillespie. “Stochastic simulation of chemical kinetics”. En: *Annu Rev Phys Chem* 58 (2007), págs. 35-55.

[22] Ryoichi Kikuchi. “A Theory of Cooperative Phenomena”. En: *Physical Review* 81.6 (mar. de 1951), págs. 988-1003.

[23] 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. ISSN: 1361-6447.

[24] Steven Roman. *Field theory*. New York: Springer, 2006, pág. 324. ISBN: 0387276777.

[25] 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. ISSN: 1742-5468.

[26] F.R. Kschischang, B.J. Frey y H.-A. Loeliger. “Factor graphs and the sum-product algorithm”. En: *IEEE Transactions on Information Theory* 47.2 (2001), págs. 498-519.

[27] Wikimedia Commons. *Simple Bipartite Graph*. 2007. URL: `https://commons.wikimedia.org/wiki/File:Simple-bipartite-graph.svg`.

# Apéndice A Grafos de factores

Para dar una definición acertada de clúster, hay que hablar antes sobre grafos de factores [26]. Sea $g(x_1, ..., x_N)$ una función que se factoriza de forma:

$$g(x_1, ..., x_N) = \prod_{j \in \mathcal{J}} f_j(X_j) \qquad con \qquad X_j \subset \{x_1, ..., x_N\} \qquad y \qquad |\mathcal{J}| < \mathfrak{c} \tag{15}$$

Entonces, el grafo de factores de $g$ es el grafo bipartito que asocia las variables $x_i \in X_j$ a los nodos factores $f_j$. De esta forma, con cada $f_j$ se relacionan solamente los $x_i$ de los que depende como función.

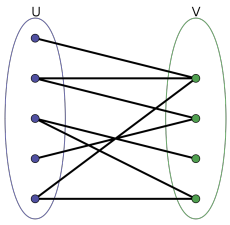


Figura 10: En esta figura se representa un grafo bipartito cualquiera. Un grafo bipartito es aquel que se separa en dos conjuntos disjuntos $U$ y $V$. El grafo de factores se definiría tomando $U = \{x_1, ..., x_N\}$ y $V = \{f_j | j \in \mathcal{J}\}$ [27].

En el caso concreto de que el sistema que se está tratando, definido en la ecuación (3), sea un proceso de Markov, se tiene que la probabilidad de una trayectoria es factorizable de la siguiente forma:

$$\begin{aligned} p\left(x^{(0)}, ..., x^{(T)}\right) &= p^{(0)}\left(x^{(0)}\right) \prod_{t=1}^{T} \omega^{(t)}\left(x^{(i)} | x^{(0)}, ..., x^{(i-1)}\right) \overset{Ec.\,(1)}{=} p^{(0)}\left(x^{(0)}\right) \prod_{t=1}^{T} \omega^{(t)}\left(x^{(i)} | x^{(i-1)}\right) \\ &= \prod_{j=1}^{N} p_j^{(0)}\left(x_j^{(0)}\right) \prod_{t=1}^{T} \left[ \prod_{k=1}^{N} \omega_k^{(t)}\left(x_k^{(i)} | x_k^{(i-1)}, x_{\partial k}^{(i-1)}\right) \right] \end{aligned}$$

Donde $\{x_i\}_{i \in \partial i}$ son los primeros vecinos del nodo $i$ en el sistema. Entonces, como tiene la forma vista en la ecuación (15), tendrá un grafo de factores asociados. Observando que la probabilidad también se puede escribir según la distribución de Boltzmann definida en la ecuación 4, el Hamiltoniano se puede escribir como:

$$\mathcal{H} = \sum_{j \in \mathcal{J}} H_j\left(\{\xi_i\}_{\xi_i \in \Xi_j \subset \{\xi_1, ..., \xi_N\}}\right)$$

donde $f_j(\Xi_j)$ factor de la probabilidad de Boltzmann.

## Apéndice B 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 activa
rate_up=(C AND (NOT B))  ? $Au : 0.0;
//Este segundo operador ternario resulta en el ritmo de inhibici n 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 C Modelo del ciclo celular

Las variables e interacciones del modelo están representadas en la figura 11 A continuación se

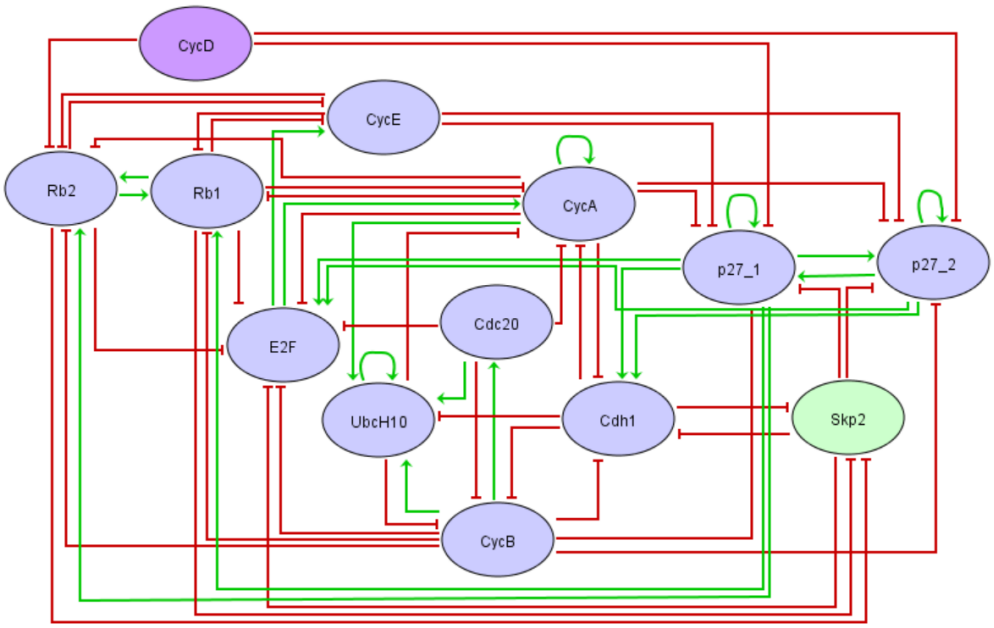


Figura 11: Representación del modelo del ciclo celular.

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 &
  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 D Código *CVM* Ciclo Celular

La aplicación del *Cluster Variation Method* al ciclo celular se ha programado en lenguaje C:

```
#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 fast 1
#define slow 1
#define CycD_del 0
#define Cdc20_del 0
#define NodeStates 2
#define tmax 200
#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 *R;//Primero xi, segundo yi, tercero vecinosyi
double *****Q;//j, yi, yj, xi, xj el n mero de j's es d_i
double ***Z;//j, xi, xj
double ****T;//j, yi, xi, xj
double ****U;//j, yi, yj, xi
double **Vc;//yi, xi
int d_i;//#vecinos
};

struct entropy{
double P;
double R;
double V;
double Q;
double T;
double U;
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 Vcluster(int I, int yi, int xi, struct node *Node);
void Qcluster(int I, int J, int yi, int yj, int xi, int xj, struct node *Node);
void Ucluster(int I, int J, int yi, int yj, int xi, struct node *Node);
void Zcluster(int I, int J, int yi, int yj, struct node *Node);
void Rcluster(int I, int yi, int SN, int xi, struct node *Node);
void Scluster(int I, int yi, int SN, struct node *Node);
double Acluster(int I, int xi, struct node *Node);

int main(){

//ALLOC STRUCT DE NODOS
struct node *system;
system = (struct node*) malloc(N * sizeof(struct node));
//CONDICIONES INICIALES
int *xini=NULL;
xini = calloc(N,sizeof(int));
int i,j,t,yi,yj,xi,xj,SN;
LecturaCondicionesIniciales(xini);
LecturaVecinos(system);
//VOY A ALLOCAR LA MEMORIA EN MAIN, COMO DEBE SER
for(i=0;i<N;i++){
(system+i)->S = calloc(2*((system+i)->SNmax), sizeof(double));
(system+i)->P = calloc(4*((system+i)->SNmax), sizeof(double));
(system+i)->R = calloc(4*((system+i)->SNmax), sizeof(double));
(system+i)->Q = calloc((system+i)->d_i, sizeof(double****));
(system+i)->Z = calloc((system+i)->d_i, sizeof(double**));
(system+i)->T = calloc((system+i)->d_i, sizeof(double***));
(system+i)->U = calloc((system+i)->d_i, sizeof(double***));
(system+i)->Vc = calloc(NodeStates, sizeof(double*));
for(yi=0;yi<NodeStates;yi++){
(system+i)->Vc[yi] = calloc(NodeStates, sizeof(double));
```

```
}
for(j=0;j<system[i].d_i;j++){
(system+i)->Q[j] = calloc(NodeStates, sizeof(double***));
(system+i)->Z[j] = calloc(NodeStates, sizeof(double*));
(system+i)->T[j] = calloc(NodeStates, sizeof(double**));
(system+i)->U[j] = calloc(NodeStates, sizeof(double**));
for(yi=0;yi<NodeStates;yi++){
(system+i)->Q[j][yi] = calloc(NodeStates, sizeof(double**));
(system+i)->Z[j][yi] = calloc(NodeStates, sizeof(double));
(system+i)->T[j][yi] = calloc(NodeStates, sizeof(double*));
(system+i)->U[j][yi] = calloc(NodeStates, sizeof(double*));
for(yj=0;yj<NodeStates;yj++){
(system+i)->Q[j][yi][yj] = calloc(NodeStates, sizeof(double*));
(system+i)->T[j][yi][yj] = calloc(NodeStates, sizeof(double));
(system+i)->U[j][yi][yj] = calloc(NodeStates, sizeof(double));
for(xi=0;xi<NodeStates;xi++){
(system+i)->Q[j][yi][yj][xi] = calloc(NodeStates, sizeof(double));
}
}
}
}
}
iniScluster(system,xini);//HAGO EL S INICIAL
iniZcluster(system,xini);//Z INICIAL
//LIBERO MEMORIA DE C.I.
free(xini);

//COSAS PARA SACAR
FILE *g;
g=fopen("Results.txt","w");
if(g==NULL){
printf("No existe el fichero de salida\n");
}
fprintf(g, "t\tCycD\tCycE\tCycA\tCycB\tRb\tE2F\tp27\tCdc20\tUbcH10\tcdh1\tH\tTH\n")
double A[N];
double norm=0;
struct entropy S;
S.H=0;//INICIO AQU S TOTAL, tengo que ir sumando en el tiempo
/*
//Para sacar transiciones
for(i=0;i<N;i++){
for(SN=0;SN<system[i].SNmax;SN++){
```

```
printf("Nodo %d\t", i);
printf("%.15lf", TransitionProb(i,0,0,system[i].V,SN));
printf("\n");
}
}
*/
//Hasta aqu

//Para logic
/*
for(i=0;i<N;i++){
for(xi=0; xi<NodeStates;xi++){
for(SN=0;SN<system[i].SNmax;SN++){
printf("Nodo %d ", i);
if(xi==0){
printf("up\t");
}else{
printf("down\t");
}
printf("%d", logic(i, xi, system[i].V, SN));
printf("\n");
}
}
}
*/
for(t=0;t<tmax;t++){
printf("Paso %d\n", t);
//INICIO ENTROPAS AUXILIARES
S.P=0;
S.Q=0;
S.R=0;
S.S=0;
S.V=0;
S.Z=0;
S.T=0;
S.U=0;

//SUMA AQU PARA ENTROPA S Y Z, AL FINAL SER N DE t+1
for(i=0;i<N;i++){
for(xi=0;xi<NodeStates;xi++){
for(SN=0;SN<((system+i)->SNmax);SN++){
if((system+i)->S[((system+i)->SNmax)*xi+SN]==0){//HE PRINTEADO Y SIEMPRE ES 0, WTF
```

```
S.S-=0;
}else{
S.S-=(system+i)->S[((system+i)->SNmax)*xi+SN]*log2((system+i)->S[((system+i)->SNmax
}
}
for(xj=0;xj<NodeStates;xj++){
for(j=0;j<(system[i].d_i);j++){
if(((system+i)->Z[j][xi][xj]==0)||(i>system[i].V[j])){
S.Z-=0;
}else{
S.Z-=(system+i)->Z[j][xi][xj]*log2((system+i)->Z[j][xi][xj]);
}
}
}
}
}
//STEP 2
for(i=0;i<N;i++){//Paso c lculo P
for(yi=0;yi<NodeStates;yi++){
for(xi=0;xi<NodeStates;xi++){
for(SN=0;SN<((system+i)->SNmax);SN++){
//PARA PRINT
//printf("Nodo %d\t", i);
Pcluster(i,yi,xi,SN,system);
//printf("%.15lf\n", (system + i)->P[2 * yi * ((system + i)->SNmax) + xi * ((system
if((system+i)->P[2*yi*((system+i)->SNmax) + xi*((system+i)->SNmax) + SN]==0){
S.P-=0;
}else{
S.P-=(system+i)->P[2*yi*((system+i)->SNmax) + xi*((system+i)->SNmax) + SN]*log2((sy
}
}
}
}
for(yi=0;yi<NodeStates;yi++){//Paso c lculo V y T
for(xi=0;xi<NodeStates;xi++){
//printf("Nodo %d\t", i);
Vcluster(i, yi, xi, system);
if((system+i)->Vc[yi][xi]==0){
S.V-=0;
}else{
S.V-=((system+i)->d_i-1)*(system+i)->Vc[yi][xi]*log2((system+i)->Vc[yi][xi]);
}
```

```
for(j=0;j<(system[i].d_i);j++){
for(xj=0;xj<NodeStates;xj++){
//printf("Nodo %d\t", i);
Tcluster(i, system[i].V[j], yi, xi, xj, system);
if((system+i)->T[j][yi][xi][xj]==0){
S.T-=0;
}else{
S.T-=(system+i)->T[j][yi][xi][xj]*log2((system+i)->T[j][yi][xi][xj]);//He quitado l
}
}
}
}
}
A[i]=Acluster(i,1,system);
}
//STEP 3
for(i=0;i<N;i++){//Paso c lculo Q
for(j=0;j<system[i].d_i;j++){
for(yi=0;yi<NodeStates;yi++){
for(yj=0;yj<NodeStates;yj++){
for(xi=0;xi<NodeStates;xi++){
for(xj=0;xj<NodeStates;xj++){
//printf("Nodo %d\t", i);
Qcluster(i,j,yi,yj,xi,xj,system);
if((system+i)->Q[j][yi][yj][xi][xj]==0){
S.Q-=0;
}else{
S.Q-=(system+i)->Q[j][yi][yj][xi][xj]*log2((system+i)->Q[j][yi][yj][xi][xj]);
}
}
}
}
}
}
for(yi=0;yi<NodeStates;yi++){//Paso c lculo U y Z
for(yj=0;yj<NodeStates;yj++){
Zcluster(i, j, yi, yj, system);
for(xi=0;xi<NodeStates;xi++){
Ucluster(i, j, yi, yj, xi, system);
if((system+i)->U[j][yi][yj][xi]==0){
S.U-=0;
}else{
S.U-=(system+i)->U[j][yi][yj][xi]*log2((system+i)->U[j][yi][yj][xi]);//Aqu tambi
```

```
}
}
}
}
norm = system[i].Z[j][0][0] + system[i].Z[j][0][1] + system[i].Z[j][1][0] + system[
system[i].Z[j][0][0] = system[i].Z[j][0][0] / norm;
system[i].Z[j][0][1] = system[i].Z[j][0][1] / norm;
system[i].Z[j][1][0] = system[i].Z[j][1][0] / norm;
system[i].Z[j][1][1] = system[i].Z[j][1][1] / norm;
}

}//Puede que sea posible meterlo dentro de bucle i, pues distintos nodos no afectan

//STEP 4
for(i=0;i<N;i++){
norm=0;
for(yi=0;yi<NodeStates;yi++){//Paso calculo R
for(xi=0;xi<NodeStates;xi++){
for(SN=0;SN<((system+i)->SNmax);SN++){
//printf("Nodo %d\t", i);
Rcluster(i,yi,SN,xi,system);
if((system+i)->R[xi*2*((system+i)->SNmax) + yi*((system+i)->SNmax) + SN]==0){
S.R-=0;
}else{
S.R-=(system+i)->R[xi*2*((system+i)->SNmax) + yi*((system+i)->SNmax) + SN]*log2((sy
}
}
}
}
for(yi=0;yi<NodeStates;yi++){//Paso calculo S
for(SN=0;SN<((system+i)->SNmax);SN++){
//printf("Nodo %d\t", i);
Scluster(i,yi,SN,system);
//printf("%.15lf\n", (system+i)->S[((system+i)->SNmax)*yi+SN]);
norm += system[i].S[(system[i].SNmax)*yi + SN];
}
}
for(yi=0;yi<NodeStates;yi++){
for(SN=0;SN<((system+i)->SNmax);SN++){
system[i].S[(system[i].SNmax)*yi + SN] = system[i].S[(system[i].SNmax)*yi + SN] / n
}
}
```

```
}
S.H = S.P + S.R + S.V + S.Q - S.T - S.U - S.S + S.Z;//+= Porque suma en el tiempo

//IMPRIMO AL TXT
fprintf(g, "%d\t", t);
for(i=0;i<N;i++){
fprintf(g, "%.15lf\t", A[i]);
}
fprintf(g, "%.15lf\t%.15lf\n", S.S-S.Z, S.H);
}
fclose(g);

//LIBERO TODA LA MEMORIA DINMICA
for(i=0;i<N;i++){
free((system+i)->S);
free((system+i)->P);
free((system+i)->R);
for(yi=0;yi<NodeStates;yi++){
free((system+i)->Vc[yi]);
}
free((system+i)->Vc);
for(j=0;j<system[i].d_i;j++){
for(yi=0;yi<NodeStates;yi++){
for(yj=0;yj<NodeStates;yj++){
for(xi=0;xi<NodeStates;xi++){
free((system+i)->Q[j][yi][yj][xi]);
}
free((system+i)->Q[j][yi][yj]);
free((system+i)->T[j][yi][yj]);
free((system+i)->U[j][yi][yj]);
}
free((system+i)->Q[j][yi]);
free((system+i)->Z[j][yi]);
free((system+i)->T[j][yi]);
free((system+i)->U[j][yi]);
}
free((system+i)->Q[j]);
free((system+i)->Z[j]);
free((system+i)->T[j]);
free((system+i)->U[j]);
}
free((system+i)->Q);
```

```
free((system+i)->Z);
free((system+i)->T);
free((system+i)->U);
}
free(system);

return 0;
}

void LecturaCondicionesIniciales(int *p){//REVISADO
char xbuffer[10];
FILE *f;
f=fopen("CondicionesIniciales.txt","r");
int i;
if(f==NULL){
printf("No existe el fichero de condiciones iniciales\n");
}
else{
for(i=0;!feof(f);i++){
fscanf(f,"%s %d",&xbuffer,p+i);
}
}
fclose(f);
}

void LecturaVecinos(struct node *p){//REVISADO
int i,j,k;//Recuerda que no leo cuando el vecino es s  mismo
FILE *f;
f=fopen("Vecinos.txt","r");
if(f==NULL){
printf("No existe el fichero de vecinos\n");
}else{
for(j=0;j<N;j++){
for(i=0;(i<N)&&((p+j)->V[i-1]!=-1)&&!feof(f);i++){
fscanf(f,"%d",&(p+j)->V[i]);
}
(p+j)->SNmax=1;
(p+j)->d_i=i-1;
for(k=0;k<i-1;k++){//AQU  HAGO 2^i
(p+j)->SNmax*=2;
}
}
```

```
}
fclose(f);
}

void iniScluster(struct node *system,int *xini){//REVISADO
int i, j;
int SN[N];
//Aqu  paso de xini a SN en cada nodo
for(j=0;j<N;j++){
SN[j]=0;
for(i=0;(((system+j)->V[i])!=-1)&&(i<N);i++){//ojo, para m  los SN son binarios al
if(xini[(system+j)->V[i]]==1){
SN[j]+=(int)(pow(2,i)+0.5);//potencia en int
}
}
}

//Aqu  hago S
for(i=0;i<N;i++){
(system+i)->S[xini[i]*((system+i)->SNmax)+SN[i]]=1.;
}
}

void iniZcluster(struct node *system,int *xini){//REVISADO
int i,j;
for(i=0;i<N;i++){
for(j=0;j<(system+i)->d_i;j++){
(system+i)->Z[j][xini[i]][xini[(system[i].V[j])]] = 1.;
}
}
}

double TransitionProb(int I, int yi, int xi, int* V, int SN){//PARECE REVISADO
int delta;// DELTA DE KRONECKER
double w[2];
if(xi==yi){
delta=1;
}else {delta=0;}

/*
//Esto est  para ver la salida de transiciones
int *q=NULL;
```

```
q = calloc(N,sizeof(int));
int i;
int bin;
bin=SN;
for(i=0;(V[i]!=-1)&&(i<N);i++){
if(bin==0 || bin==1){
q[V[i]]=bin;
bin=bin/2;
}else{
q[V[i]]=bin%2; //PONGO EL ESTADO SEGN EL ORDEN DE LAS C. INI.
bin=bin/2;
}
}
q[I]=xi;
for(i=0;i<N;i++){
printf("%d ", q[i]);
}
printf("\t");
q[I]=yi;
for(i=0;i<N;i++){
printf("%d ", q[i]);
}
printf("\t");
//Hasta aqu
*/

switch(I){
case 0://OJO CON ESTE, DEPENDE SOLO DE CTE.
if(!CycD_del){
w[0]=0;
w[1]=0;
}else{
w[0]=0;
w[1]=fast;
}
return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
break;
case 1://CycE
if(logic(I, xi, V, SN)){
w[0]=slow;
w[1]=0;
}else{
```

```
w[0]=0;
w[1]=fast;
}
return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
break;
case 2://CycA
if(logic(I, xi, V, SN)){
w[0]=slow;
w[1]=0;
}else{
w[0]=0;
w[1]=fast;
}
return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
break;
case 3://CycB
if(logic(I, xi, V, SN)){
w[0]=slow;
w[1]=0;
}else{
w[0]=0;
w[1]=fast;
}
return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
break;
case 4://Rb
if(logic(I, xi, V, SN)){
w[0]=fast;
w[1]=0;
}else{
w[0]=0;
w[1]=fast;
}
return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
break;
case 5://E2F
if(logic(I, xi, V, SN)){
w[0]=slow;
w[1]=0;
}else{
w[0]=0;
w[1]=fast;
```

```
}
return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
break;
case 6://p27
if(logic(I, xi, V, SN)){
w[0]=fast;
w[1]=0;
}else{
w[0]=0;
w[1]=fast;
}
return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
break;
case 7://Cdc20
if(logic(I, xi, V, SN)&&!Cdc20_del){
w[0]=slow;
w[1]=0;
}else{
w[0]=0;
w[1]=fast;
}
return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
break;
case 8://UbcH10
if(logic(I, xi, V, SN)){
w[0]=slow;
w[1]=0;
}else{
w[0]=0;
w[1]=fast;
}
return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
break;
case 9://cdh1
if(logic(I, xi, V, SN)){
w[0]=fast;
w[1]=0;
}else{
w[0]=0;
w[1]=fast;
}
return w[xi]*tao*(1-delta)+(1-w[xi]*tao)*delta;
```

```
break;
default:
printf("No metiste nodo al TransitionProb\n");
}
}

int logic(int I,int xi, int* V, int SN){//REVISADO Y CORREGIDO
int *q=NULL;
q = calloc(N,sizeof(int));
int i;
int bin;
bin=SN;
for(i=0;(V[i]!=-1)&&(i<N);i++){
if(bin==0 || bin==1){
q[V[i]]=bin;
bin=bin/2;
}else{
q[V[i]]=bin%2; //PONGO EL ESTADO SEG N EL ORDEN DE LAS C. INI.
bin=bin/2;
}
}
q[I]=xi;//METO ESTO PORQUE EN V NO VOY A METER A SU MISMO NODO (por problemas luego

//PARA MOSTRAR ESTADO
/*
for(i=0;i<N;i++){
printf("%d", q[i]);
}
printf("\t");
*/

switch(I){
case 0:
if(*q){
return 1;
}else{return 0;}
break;
case 1:
if(!*(q+4) && *(q+5)){
return 1;
}else{return 0;}
break;
```

```
case 2:
if(!*(q+4) && !*(q+7) && !(*(q+8) && *(q+9)) && (*(q+2) || *(q+5))){
return 1;
}else{return 0;}
break;
case 3:
if(!*(q+7) && !*(q+9)){
return 1;
}else{return 0;}
break;
case 4:
if(!*q && !*(q+3) && (*(q+6) || !(*(q+2) || *(q+1)))){
return 1;
}else{return 0;}
break;
case 5:
if(!*(q+4) && !*(q+3) && (!*(q+2) || *(q+6))){
return 1;
}else{return 0;}
break;
case 6:
if(!*q && !*(q+3) && (!(*(q+2) || *(q+1)) || (*(q+6) && !(*(q+1) && *(q+2))))){
return 1;
}else{return 0;}
break;
case 7:
if(*(q+3)){
return 1;
}else{return 0;}
break;
case 8:
if((!(*(q+9) && !*(q+8)) && (*(q+2) || *(q+3))) || (!*(q+2) && !*(q+3) && (!*(q+9)
return 1;
}else{return 0;}
break;
case 9:
if(*(q+7) || (!*(q+3) && (!*(q+2) || *(q+6)))){
return 1;
}else{return 0;}
break;
default:
printf("Fallo en Logic\n");
```

```
return -1;
break;
}
}

void Pcluster(int I, int yi, int xi, int SN, struct node *Node){//OJO, ESTOY PENSAN
(Node+I)->P[2*yi*((Node+I)->SNmax) + xi*((Node+I)->SNmax) + SN] = TransitionProb(I,
//PRINT DE ESTADO
/*
int *q = NULL;
q = calloc(N, sizeof(int));
int i;
int bin;
bin = SN;
for (i = 0; (Node[I].V[i] != -1) && (i < N); i++)
{
if (bin == 0 || bin == 1)
{
q[Node[I].V[i]] = bin;
bin = bin / 2;
}
else
{
q[Node[I].V[i]] = bin % 2; //PONGO EL ESTADO SEGN EL ORDEN DE LAS C. INI.
bin = bin / 2;
}
}
q[I] = xi;
for (i = 0; i < N; i++)
{
printf("%d ", q[i]);
}
printf("\t");
q[I] = yi;
for (i = 0; i < N; i++)
{
printf("%d ", q[i]);
}
printf("\t");
printf("w=%.15lf\tS=%.15lf\t", TransitionProb(I, yi, xi, (Node + I)->V, SN), (Node

if((Node+I)->P[2*yi*((Node+I)->SNmax) + xi*((Node+I)->SNmax) + SN]>1.){
```

```
//exit(999);
}
*/
}

void Tcluster(int I, int J, int yi, int xi, int xj, struct node *Node){//PASALE SYS
double sum;
sum=0;
int i, j, SN, bin;
int *q=NULL;
q = calloc(N,sizeof(int));
//printf("I=%d J=%d yi=%d xi=%d xj=%d ", I,J,yi,xi,xj);
for(SN=0;SN<((Node+I)->SNmax);SN++){
bin=SN;
for(i=0;(((Node+I)->V[i])!=-1)&&(i<N);i++){//SACO EL ESTADO AL COMPLETO PRIMERO
if(bin==0 || bin==1){
q[(Node+I)->V[i]]=bin;
bin=bin/2;
}else{
q[(Node+I)->V[i]]=bin%2;
bin=bin/2;
}
}
q[I] = xi;//Esto lo traje del Toymodel, creo que solo es  til  a la hora de printea
if(q[J]!=xj){//DESPUS COMPRUEBO QUE XJ NO HA CAMBIADO EN EL ESTADO
sum+=0;
}else{
sum+=(Node+I)->P[2*yi*((Node+I)->SNmax) + xi*((Node+I)->SNmax) + SN];
//printf("P(x=%d%d%d%d%d%d%d%d)=%.15lf ",q[0],q[1],q[2], q[3], q[4], q[5], q[6]
}
}
for(j=0;j<N;j++){//COMPROBADO EXTERNAMENTE
if(J==Node[I].V[j]){
(Node+I)->T[j][yi][xi][xj] = sum;//ESTO ES NUEVO, PARA QUE sea el j en el orden de
break;
}
}
/*
if((Node+I)->T[j][yi][xi][xj] != sum){
printf("ERROR con T(j)");
}
printf("T=%.15lf\n", (Node+I)->T[j][yi][xi][xj]);
```

```
if((Node+I)->T[j][yi][xi][xj]>1.){
//exit(999);
}
*/
}

void Vcluster(int I, int yi, int xi, struct node *Node){//LE HE PASADO EL COMPLETO
//printf("I=%d  yi=%d  xi=%d  ", I,yi,xi);
double sum;
sum=0;
int i, SN, bin;
int *q=NULL;
q = calloc(N,sizeof(int));
for(SN=0;SN<(Node+I)->SNmax;SN++){
sum+=(Node+I)->P[2*yi*((Node+I)->SNmax) + xi*((Node+I)->SNmax) + SN];
//LO QUE VIENE AHORA ES PARA EL PRINT DEL ESTADO
/*
bin=SN;
for(i=0;(((Node+I)->V[i])!=-1)&&(i<N);i++){//SACO EL ESTADO AL COMPLETO PRIMERO
if(bin==0 || bin==1){
q[(Node+I)->V[i]]=bin;
bin=bin/2;
}else{
q[(Node+I)->V[i]]=bin%2;
bin=bin/2;
}
}
q[I] = xi;
//printf("P(x=%d%d%d%d%d%d%d%d)=%.15lf  ",q[0],q[1],q[2], q[3], q[4], q[5], q[6]
*/
}
(Node+I)->Vc[yi][xi] = sum;
/*printf("V=%.15lf\n", (Node+I)->Vc[yi][xi]);
if((Node+I)->Vc[yi][xi]>1.){
//exit(999);
}*/
}

void Qcluster(int I, int J, int yi, int yj, int xi, int xj, struct node *Node){
//printf("I=%d  J=%d  yi=%d  yj=%d  xi=%d  xj=%d  ", I,J,yi,yj,xi,xj);
int i,j;
j=Node[I].V[J];
```

```
for(i=0;i<N;i++){//COMPROBADO EXTERNAMENTE
if(I==Node[j].V[i]){
break;//ESTO ES NUEVO, PARA QUE sea el j en el orden de vecinos, no el del orden gl
}
}
if(i==N){
//printf("ERROR con Q(i)");
}
if((Node+I)->Z[J][xi][xj]<=eps){
(Node+I)->Q[J][yi][yj][xi][xj] = 0;
//printf("Q=0\n");
}else{
(Node+I)->Q[J][yi][yj][xi][xj] = (Node+I)->T[J][yi][xi][xj]*(Node+j)->T[i][yj][xj][
//printf("T1=%.15lf  T2=%.15lf  Z=%.15lf  Q=%.15lf\n", (Node+I)->T[J][yi][xi][xj], (No
}
/*
if((Node+I)->Q[J][yi][yj][xi][xj]>1.){
//exit(999);
}*/
}

void Ucluster(int I, int J, int yi, int yj, int xi, struct node *Node){
int i;
double sum;
sum=0;
//printf("I=%d  J=%d  yi=%d  yj=%d  xi=%d  ", I, J, yi, yj, xi);
for(i=0; i<NodeStates; i++){
sum+=(Node+I)->Q[J][yi][yj][xi][i];
//printf("Q(xj=%d)  %.15lf\t", i, (Node+I)->Q[J][yi][yj][xi][i]);
}
(Node+I)->U[J][yi][yj][xi] = sum;
/*
printf("U  %.15lf\n", (Node+I)->U[J][yi][yj][xi]);
if((Node+I)->U[J][yi][yj][xi]>1.){
//exit(999);
}*/
}

void Zcluster(int I, int J, int yi, int yj, struct node *Node){
int i, j;
double sum=0;
//printf("I=%d  J=%d  yi=%d  yj=%d  ", I, J, yi, yj);
```

```
for(i=0;i<NodeStates;i++){
for(j=0;j<NodeStates;j++){
sum+=(Node+I)->Q[J][yi][yj][i][j];
//printf("Q(xi=%d, xj=%d) %.15lf\t", i, j, (Node+I)->Q[J][yi][yj][i][j]);
}
}
(Node+I)->Z[J][yi][yj]=sum;
/*
printf("Z %.15lf\n", (Node+I)->Z[J][yi][yj]);
if((Node+I)->Z[J][yi][yj]>1.){
//exit(999);
}*/
}

void Rcluster(int I, int yi, int SN, int xi, struct node *Node){
int j, bin;
double V;
V=(Node+I)->Vc[yi][xi];
//printf("V %.15lf\t", V);
if(V<=eps){
(Node+I)->R[xi*2*((Node+I)->SNmax) + yi*((Node+I)->SNmax) + SN] = 0;
}else{
double prod;
prod=1;
bin=SN;
for(j=0;(((Node+I)->V[j])!=-1)&&(j<N);j++){//SACO EL ESTADO AL COMPLETO PRIMERO
if(bin==0 || bin==1){
prod*=((Node+I)->U[j][yi][bin][xi]/V);
//printf("U_%d(yj=%d) %.15lf\t", (Node+I)->V[j], bin, (Node+I)->U[j][yi][bin][xi]);
bin=bin/2;
}else{
prod*=((Node+I)->U[j][yi][bin%2][xi]/V);
//printf("U_%d(yj=%d) %.15lf\t", (Node+I)->V[j], bin%2, (Node+I)->U[j][yi][bin%2][x
bin=bin/2;
}
}
(Node+I)->R[xi*2*((Node+I)->SNmax) + yi*((Node+I)->SNmax) + SN] = prod*V;//Igual e
//printf("di %d\tR %.15lf", j-1, (Node+I)->R[xi*2*((Node+I)->SNmax) + yi*((Node+I)-
}
/*
printf("\n");
if((Node+I)->R[xi*2*((Node+I)->SNmax) + yi*((Node+I)->SNmax) + SN]>1.){
```

```
//exit(999);
}*/
}

void Scluster(int I, int yi, int SN, struct node *Node){
int i;
double sum=0;
/*
//PRINT DE ESTADO
int *q=NULL;
q = calloc(N,sizeof(int));
int bin;
bin=SN;
for(i=0;(Node[I].V[i]!=-1)&&(i<N);i++){
if(bin==0 || bin==1){
q[Node[I].V[i]]=bin;
bin=bin/2;
}else{
q[Node[I].V[i]]=bin%2; //PONGO EL ESTADO SEGN EL ORDEN DE LAS C. INI.
bin=bin/2;
}
}
q[I]=yi;
for(i=0;i<N;i++){
printf("%d ", q[i]);
}
printf("\t");
*/
for(i=0;i<NodeStates;i++){
sum+=(Node+I)->R[i*2*((Node+I)->SNmax) + yi*((Node+I)->SNmax) + SN];
//printf("R(xi=%d)=%.15lf\t", i, (Node+I)->R[i*2*((Node+I)->SNmax) + yi*((Node+I)->
}
(Node+I)->S[((Node+I)->SNmax)*yi+SN] = sum;
/*
if((Node+I)->S[((Node+I)->SNmax)*yi+SN]>1.){
//exit(999);
}*/
}

double Acluster(int I, int xi, struct node *Node){
int SN, yi;
double sum=0;
```

```
for (SN=0;SN<((Node+I)->SNmax);SN++){
for (yi=0;yi<NodeStates;yi++){
sum+=(Node+I)->P[2*yi*((Node+I)->SNmax) + 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 Çondicionesesiniciales.txt":

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