UNIVERSITAT DE LLEIDA

DEPARTAMENT DE MATEMÀTICA

TESIS DOCTORAL

Contribuciones al estudio de cadenas de Markov finitas mediante computación natural

AUTOR:

M.Alba Zaragoza Ramírez

DIRECTOR:

M.Àngels Colomer Cugat

Agradecimientos

En primer lugar quisiera expresar mis agradecimientos a mi directora de tesis M.Àngels Colomer por su dedicación, interés, sugerencias, disponibilidad y perseverancia en el desarrollo de la tesis. He de agradecerle los ánimos recibidos por su parte en los momentos en que parecía que este trabajo no vería su fin.

Como no agradecer a Juan Fibla los momentos dedicados con la finalidad de clarificar mis lagunas biológicos sobre ADN. A los compañeros del Departament de Matemàtiques i Informàtica de la Universitat de les Illes Balears y en especial a Joe y a Cesc por animarnos a formar parte del ambicioso proyecto de la computación con ADN. A Mario por sus lecciones sobre computación celular con membranas, por sus pizarras y por haber hecho un hueco en su tan apretada agenda así como a todo el grupo de investigadores en Computación Natural del departamento de Ciencia Computacional e Inteligencia Artificial de la universidad de Sevilla por su hospitalidad.

Agradezco el respaldo presentado en todo momento por el personal del departamento de matemáticas especialmente a todos mis compañeros del área de estadística e investigación operativa.

Dejo para el final los agradecimientos por la inmensa ayuda tanto moral como científica aportada por mis compañeros de camino Mónica y Josep. Josep me ha ayudado a profundizar mis conocimientos matemáticos de cadenas de Markov presentándomelos de manera sencilla y entendedora. Con Mónica hemos iniciado este camino juntas que por motivos personales y principalmente de diferencia de edad he podido terminar antes que ella. Le doy ánimos y ya sabe que me tendrá siempre a su lado en esta labor.

El tiempo que le he absorbido a mi familia, en especial a Jaume y al pequeño Pau que se ha gestado junto a la elaboración final de la tesis y como no a mis padres por permitirme en su momento realizar los estudios de matemáticas que me han llevado a mi situación actual.

"No hay rama de la matemática, por abstracta que sea, que no pueda aplicarse algún día a los fenómenos del mundo real."(Nikolay Lobachevsky)

Índice

Estructura de la tesis

Esta tesis está estructurada en ocho capículos cuyos contenidos son:

Capítulo 1: Introducción

En este capítulo se contextualizan los trabajos realizados en el campo de la computación natural y se establecen los objetivos de la tesis. Seguidamente se describen los artículos que forman dicha memoria.

Capítulo 2: Teoría básica: Cadenas de Markov discretas

En este capítulo se introducen los conceptos, notaciones y resultados, presentados en diferentes textos, en los que se fundamentan los trabajos presentados en los distintos artículos que forman la tesis.

Capítulo 3: Computación natural

En este capítulo se pretende situar el campo de la computación natural desde el cual se abordarán los diferentes resultados de las cadenas finitas y homogéneas de Markov. Se presenta además una descripción de las dos áreas de este campo que han tenido más auge y que serán las utilizadas para llegar a los resultados de la tesis: la computación molecular con ADN y la computación celular con membranas.

Capítulo 4: Markov chains: Computing limit existence and approximations with DNA

En este capítulo se adjunta el artículo publicado en la revista BioSystems en el año 2005. En dicho artículo se aborda el problema de la convergencia de una cadena de Markov desde la computación con ADN. También se presenta, en el caso de que éste exista, un algoritmo que nos permite obtener una estimación de dicho límite mediante computación con ADN.

Capítulo 5: A step towards a DNA Computation model

En este capítulo se adjunta el artículo presentado en la revista *BioSystems*

durante el año 2006. Dicho artículo presenta una forma de clasificación de los estados de una cadena finita y homogénea de Markov desde la computación con ADN. El estudio de la periodicidad de las clases recurrentes se aborda a partir de una proposición que sustituye el cálculo matemático del período de un estado por un problema de existencia evitando de esta manera el problema de error de cálculo que presenta trabajar con ADN.

Capítulo 6: Handling Markov Chains with Membrane Computing

En este capítulo se adjunta el artículo publicado en Lecture Notes in Computer Science en el año 2006 en el cual se presenta la manera de calcular las potencias de la matriz de transición asociada a una cadena de Markov mediante computación celular con membranas.

Capítulo 7: Classifying States of a Finite Markov Chain with Membrane Computing

En este capítulo se adjunta el artículo publicado en Lecture Notes in Computer Science en el año 2006 correspondiente al diseño de un P sistema de transición cuya salida es la clasificación de los estados de una determinada cadena de Markov así como su periodo. En este caso se hace uso de la definición de período de un estado ya que la computación con membranas nos permite cálculos exactos.

Capítulo 8: Conclusiones

En este capítulo se exponen las aportaciones más relevantes de cada una de las cuatro contribuciones que forman la tesis.

Capítulo 1

Introducción

La tesis se sitúa en el campo de las cadenas finitas y homogéneas de Markov y en el campo de la computación natural. Más concretamente se abordan dos aspectos de las cadenas finitas y homogéneas de Markov presentándose dos resultados: el primer resultado nos permite calcular las potencias n-ésimas de la matriz de transición con la finalidad de poder obtener una aproximación del límite de la cadena de Markov en el caso que éste exista. El segundo resultado nos permite clasificar los estados de una cadena finita y homogénea de Markov. Ambos resultados son abordados desde dos de los campos que han tenido más auge dentro de la computación natural: la computación molecular con ADN y la computación celular con membranas. En este primer capítulo se contextualizan los trabajos realizados en el campo de computación natural y se presentan los objetivos de la tesis. A continuación se resumen los artículos con los que se defiende la tesis.

Contexto

Desde que fue propuesta por primera vez la idea de Feynman en 1960 de realizar computación a nivel molecular ha ido creciendo su interés. De esta forma, en 1994 Leonard Adleman [1] muestra como es posible resolver el *Hamiltonian Path Problem, (HPP)* mediante ADN. Adleman propone un algoritmo basado en operaciones con ADN que permite resolver en un espacio breve de tiempo el problema del camino Hamiltoniano. Propone la resolución para el caso de un grafo simple que, con el fin de demostrar su factibilidad, lleva al laboratorio. Desde este momento, se abren tres cuestiones:

- tipo de algoritmos que se pueden implementar con ADN
- capacidad del ADN en computación universal
- control de los errores de la manipulación con ADN.

La primera de las cuestiones ha encontrado respuesta con los trabajos teóricos presentados durante los dos años posteriores al trabajo de Adleman. En 1994, D. Boneh, C. Dunworth y R.J. Lipton muestran en [8] cómo las computadoras basadas en ADN se pueden utilizar para resolver el problema de satisfiabilidad para circuitos booleanos. Concretamente, un año más tarde, describen en [9] el potencial de la computación molecular para atacar el problema *United States Data Encryption Standard (DES)*. En este trabajo, se da una descripción detallada de una librería de operaciones que son de gran utilidad en computación molecular. En 1995 R.J. Lipton muestra en [29] como resolver el problema *SAT* con ADN. Otro trabajo que muestra el progreso de los algoritmos basados en ADN es el de F. Guarnieri, M. Fliss y C. Bancroft (1996) [20] en el que proponen un algoritmo para la suma basado en cadenas y propiedades del ADN con el cual demuestran la factibilidad del uso de experimentos basados en ADN para solucionar problemas combinatorios.

Estos trabajos abren la puerta a la idea de considerar la computación molecular como una de las mejores vías para la resolución de problemas NP completos, planteándose la necesidad de disponer de computadoras moleculares programables. P. Wilhelm y K. Rothermund [51] exponen en su trabajo la posibilidad de llevar a cabo máquinas de Turing con ADN. En 1996 D. Beaver [7] presenta la posibilidad de resolver mediante computadores moleculares cualquier problema que se pueda escribir mediante un algoritmo. Existen diversos trabajos presentados durante 1997 en los que se hace una revisión de los modelos presentados con ADN como son la tesis doctoral de M. Amos [2] y el trabajo de L. Kari [24] en el que además hace una excelente descripción de las operaciones biológicas posibles con ADN.

Siguiendo en la línea de los trabajos que pretenden dar respuesta sobre la capacidad del ADN en computación universal, en 1998 C. Maley [30] hace una revisión en su trabajo de la práctica y teoría en el campo de la computación molecular con la finalidad de responder a la cuestión sobre la utilidad de la computación con ADN en el futuro. También, N. Pisanti [46], ofrece una descripción de las discusiones y los resultados que han aparecido en la literatura de la computación molecular hasta entonces así como M.Amos (1999) [3] que hace

una revisión del experimento de Adleman en la que describe las operaciones utilizadas en dicho experimento y concluye definiendo la noción de algoritmo eficiente con ADN.

Con respecto a los errores cometidos en la manipulación de cadenas con ADN son muchos los trabajos que se encuentran en la bibliografía que se centran básicamente en la descripción de operaciones y la obtención de nuevas técnicas biológicas. Destacar el trabajo de M. Amos, Gh.Păun, G. Rozenberg y A. Saloma (2002) [4] en el que introducen la estructura básica del ADN así como las herramientas de procesamiento básicas de dichas moléculas.

Como ejemplo de aplicaciones más recientes con respecto a la computación con ADN, cabe destacar el trabajo de A. Leider [27] (2000) en el que muestra como usar cadenas binarias de ADN en esteganografia (técnica de encriptación mediante información oculta), el trabajo de R. Barau [6] propuesto en el 2003 en el cual propone un algoritmo recursivo con ADN para sumar dos números binarios en el cual se requieren $O(\log n)$ bio-pasos y solamente $O(n)$ tipos diferentes de cadenas de ADN siendo n la longitud de la cadena binaria que representa el más largo de los dos números, y el trabajo de S.Wang [49] del 2005 en el que da una discusión detallada de los algoritmos de ADN para solucionar el problema NP completo de *programación lineal con enteros*.

Mientras siguen los trabajos con computación molecular con ADN se introduce un nuevo modelo de computación natural: la computación celular con membranas. Es a finales de la década de los noventa (1998) cuando Gh. Păun introduce en [39] este nuevo modelo inspirado en la estructura y funcionamiento de las células de los organismos vivos. A diferencia de la computación con ADN, los P sistemas no se han implementado biológicamente si bien ha surgido alguna aplicación informática con la finalidad de simular P sistemas. En los trabajos [15] y [21] de F. Ciobano y D. Paraschiv (2002) y de M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez y A. Riscos (2006), respectivamente, se presentan simuladores como herramientas para trabajos basados en investigación en computación celular con membranas, herramientas con propósito pedagógico que dan soporte a la investigación.

Entre los muchos trabajos que encontramos de Gh. Păun caben destacar [40] y [43]. En el primero introduce las principales ideas de la computación con membranas. En este trabajo presenta algunos invariantes de P sistemas enfati-

zando dos tipos de resultados, la caracterización de los conjuntos de números recursivamente y la solución de problemas NP-completos en tiempo polinomial. El segundo de los trabajos citados de Gh. Păun introduce cuatro nuevas variantes de P sistemas; P sistemas que permiten mover reglas no objetos, P sistemas que permiten eliminar membranas repetidas, P sistemas que permiten acelerar reglas en las que el tiempo va disminuyendo, y P sistemas seguros donde todos los eventos posibles se dan lugar.

La computación con ADN ha inspirado a los investigadores en computación cecular en la utilización de cadenas como objetos. De esta forma, A. Păun considera en [37] tres tipos de P sistemas cuyos objetos son cadenas; los splicing P systems inspirados en la computación con ADN, los rewriting P systems, inspirados en la Teoría del Lenguaje Formal (las reglas son reglas de reescritura) y los rewriting P systems con paralelismo parcial que tienen el mismo número de reglas y objetos que el anterior con alguna restricción.

Con respecto a los problemas que pueden ser abordados con P sistemas encontramos en el 2005 el trabajo de V. Manca [31] en el que muestra como los sistemas con membranas pueden considerarse como entorno físico para resolver problemas de satisfiabilidad como el *SAT*. También, en el mismo año, en [28] A. Leparoti y C. Zandron proponen un algoritmo con membranas para resolver el problema NP-completo *3-SAT*. Se remarca que la manera de codificar los números binarios permite resolver otros problemas numéricos NP-completos. Otros trabajos que hablan de la eficiencia de los P sistemas para resolver problemas NP-completos son el trabajo de D. Díaz, M.A. Gutiérrez-Naranjo y M.J. Pérez-Jiménez [16] (2006) en el que se presenta una solución en tiempo lineal del problema NP completo, *3-COL* y el trabajo de M.A. Gutiérrez-Naranjo y M.J. Pérez-Jiménez [22] (2006) en el que se considera los P sistemas como una herramienta natural para tratar con fractales.

Por último, cabe remarcar en el 2002 el trabajo de Marcus [32] en el cual analiza las ventajas de las membranas respecto al ADN.

Objetivos

Las limitaciones en tiempo y espacio físico que presentan las máquinas actuales de computación para resolver determinados problemas nos lleva a pensar en modelos de computación alternativos. De esta manera se introducen las ideas biológicas en el marco de la computación pudiéndose llegar a pensar, en un futuro, en posibles máquinas biológicas. Aunque la idea principal sea resolver problemas computacionalmente difíciles, problemas NP, dichas máquinas también deberán permitir resolver cualquier tipo de problema. Por lo tanto, primeramente deberán presentarse algoritmos biológicos de resolución básica sin incrementar la complejidad que presentan los algoritmos actuales. En este contexto, en esta tesis se presenta cómo trabajar biológicamente con cadena finitas y homogéneas de Markov siendo los objetivos:

- La obtención del cálculo aproximado de las potencias n-ésimas de la matriz de transición de una cadena finita y homogénea de Markov mediante ADN computacional.
- La obtención del cálculo exacto de las potencias n-ésimas de la matriz de transición de una cadena finita y homogénea de Markov mediante membranas celulares.
- Clasificar los estados de una cadena finita y homogénea de Markov mediante ADN computacional.
- Clasificar los estados de una cadena finita y homogénea de Markov mediante membranas celulares.

Contribuciones

Se presentan cuatro contribuciones a modo de artículos que son la base para la defensa de la tesis.

En la *primera contribución* [11] se presentan dos algoritmos dentro del marco de las cadena finitas de Markov. El primero permite estudiar la convergencia de la sucesión de las potencias de la matriz de transición asociada a la cadena. En caso de que converja, el segundo algoritmo nos permite obtener una estimación de este límite, es decir, el cálculo de la potencia n-ésima de la matriz de transición. Ambos algoritmos son abordados desde la computación con ADN siendo

el tiempo y la cantidad de recursos polinomiales.

En la *segunda contribución* [14] se aborda el problema de la clasificación de los estados de una cadena de Markov finita, [19], así como el estudio de la periodicidad de las clases recurrentes. Para ello no se utiliza el cálculo del periodo de un estado sino que se utiliza un resultado que permite estudiar la existencia o no de clases recurrentes aperiódicas. Esta contribución pretende mostrar que para trabajar con computación con ADN muchas veces la traducción directa de algoritmos tradicionales a algoritmos biológicos para la realización de cómputos, no conduce a resultados óptimos, sino sólo, a meras estimaciones. Para ello, son necesarias nuevas formas de pensar adaptadas a las posibilidades que nos ofrece la computación biológica. En este trabajo se finaliza con un experimento biológico que muestra la factibilidad del algoritmo presentado, en particular, se implementa en el laboratorio el algoritmo para la generación de la trayectoria de un caso particular de cadena de Markov con dos estados.

En ambas aportaciones se han codificado los estados asociados a una cadena de Markov y sus probabilidades de transición mediante cadenas de ADN. A partir de la unión de dichas cadenas, se han generado las trayectorias de la cadena de Markov.

En la *tercera y cuarta contribución* se pretende llegar a los mismos resultados que las dos anteriores pero abordando dichos problemas mediante computación con membranas celulares. En la tercera [12] se presenta un P sistema de transición cuya salida corresponde a la potencia n-ésima de la matriz de transición asociada a una cadena finita de Markov. En este caso no se llega a una estimación como era el caso de la primera contribución, sino que se obtiene el cálculo exacto de los coeficientes de dicha matriz. La cantidad de recursos iniciales que se requieren en la construcción es polinomial en función del número de estados y de la potencia que se desea calcular mientras que el tiempo es lineal en la potencia y no depende del número de estados.

En la *cuarta y última contribución* [13] se presenta un P sistema de transición cuya salida, codificada en el entorno, corresponde a la clasificación de cada estado que forma la cadena de Markov con su período. En este caso, la cantidad de recursos iniciales que se requieren en la construcción es polinomial y el tiempo es lineal ambos en el número de estados.

Capítulo 2

Teoría básica: Cadenas de Markov discretas

2.1. Introducción

En la física clásica, el principio fundamental del determinismo científico desempeña un papel fundamental: a partir del estado de un sistema físico en el instante t_0 se puede deducir su estado en un instante posterior t. Como consecuencia de este principio se obtiene un método básico para analizar sistemas físicos: se puede deducir el estado de un sistema físico en un instante dado t_2 a partir del conocimiento de su estado en un instante cualquiera anterior t_1 y no depende del historial del sistema antes del instante t_1.

Para sistemas físicos que obedecen leyes probabilísticas en vez de leyes determinísticas, se puede enunciar un principio análogo: la probabilidad que el sistema físico esté en un estado dado en un instante t_2 se puede deducir del conocimiento de su estado en un instante anterior t_1 y no depende del historial del sistema antes del instante t_1. Los procesos estocásticos que representan observaciones de sistemas físicos que satisfacen esta condición son llamados procesos de Markov.[10]

Una clase especial de procesos de Markov es una cadena de Markov; se puede definir como un proceso estocástico cuyo desarrollo se puede considerar como

una serie de transiciones entre valores determinados (llamados estados del proceso) que tienen la propiedad de que la ley de probabilidad del desarrollo futuro del proceso, una vez que está en un estado dado, depende sólo del estado y no de cómo llegó el proceso a dicho estado. El número de estados posibles es finito o numéricamente finito.

A lo largo de este capítulo se exponen los conceptos básicos de las cadenas de Markov de parámetro discreto así como los principales resultados necesarios para el desarrollo de la tesis.

2.2. Definición

Un proceso discreto $X_0, \ldots, X_n, \ldots$ definido en el espacio de probabilidad $(\Omega, \Im, P)$ y con valores en un espacio mesurable discreto (E, ξ) es una *cadena de Markov discreta* si

$$P(X_{n+1} \in A \mid X_0, \ldots, X_n) = P(X_{n+1} \in A \mid X_n) \tag{2.1}$$

$\forall A \in \xi, \forall n \geq 0$. El espacio (E, ξ) recibe el nombre de espacio de estados y ξ es la σ-álgebra de todos los subconjuntos de E. [36]

2.3. Probabilidades de transición y la ecuación de Chapman - Kolmogorov

Para todos los instantes $n \geq m \geq 0$, y los estados e_i y e_j se establece la función de probabilidad:

$$p_i(n) = P[X_n = i] \tag{2.2}$$

y la función de probabilidad condicional:

$$p_{ij}(m, n) = P[X_n = j \mid X_m = i] \tag{2.3}$$

que recibe el nombre de *probabilidad de transición* de la cadena de Markov.
Se dice que una cadena de Markov discreta es homogénea en el tiempo o que tiene probabilidades de transición estacionarias si $p_{ij}(m, n)$ sólo depende de la diferencia $n - m$. Entonces se llama *función de probabilidad de transición en n pasos* de la cadena de Markov homogénea $\{X_n\}$ a:

$$p_{ij}(n) = P[X_n = j \mid X_{n-1} = i] = P[X_{n+t} = j \mid X_{n+t-1} = i] \tag{2.4}$$

para cualquiera entero $t \geq 0$ y $e_i, e_j \in E$, es decir, $p_{ij}(n)$ es la probabilidad condicional de que una cadena de Markov homogénea que se encuentra en el estado e_i al cabo de n pasos se encuentre en el estado e_j que, con la finalidad de unificar notaciones, escribiremos como $p_{ij}^{(n)}$. Las probabilidades de transición de una paso $p_{ij}(1)$ se escriben p_{ij}.

Una relación fundamental que satisface la función de probabilidad de transición de una cadena de Markov $\{X_n\}$ es la ecuación llamada *ecuación de Chapman-Kolmogorov*: para instantes cualesquiera $0 \leq m < u < n$ y los estados e_i y e_j,

$$p_{ij}(m,n) = \sum_{k \in E} p_{ik}(m,u) p_{kj}(u,n) \tag{2.5}$$

Las probabilidades de transición de una cadena de Markov $\{X_n\}$ con valores en el espacio de estados (E, ξ) se representan en forma de matriz, llamada matriz de probabilidades de transición:

$$P(m,n) = \begin{pmatrix} p_{11}(m,n) & p_{12}(m,n) & \dots & p_{1j}(m,n) & \dots \\ \vdots & \vdots & \dots & \vdots & \dots \\ p_{i1}(m,n) & p_{i2}(m,n) & \dots & p_{ij}(m,n) & \dots \\ \vdots & \vdots & \dots & \vdots & \dots \end{pmatrix} \tag{2.6}$$

donde sus coeficientes satisfacen las condiciones:

$$p_{ij}(m,n) \geq 0 \quad \text{para todos} \quad e_i, e_j \in E \tag{2.7}$$

$$\sum_{j \in E} p_{ij}(m,n) = 1 \quad \text{para todo} \quad e_i \in E. \tag{2.8}$$

es decir, es una matriz estocástica [26].
De esta manera, las ecuaciones de Chapman-Kolmogorov para todos los instantes $0 \leq m < u < n$ se pueden escribir:

$$P(m,n) = P(m,u) \cdot P(u,n) \tag{2.9}$$

2.4. Determinación de las probabilidades de transición

A partir de la ecuación de Chapman-Kolmogorov se pueden deducir varias relaciones recurrentes para las funciones de probabilidad de transición. Sea $\{X_n\}$

una cadena de Markov discreta con matriz de transición $P(n,m)$ de la ecuación 1.9 se deduce que:

$$\begin{array}{c} P(m,n) = P(m,n-1)\cdot P(n-1,n) = \ldots = \\ P(m,m+1)\cdot P(m+1,m+2)\ldots P(n-1,n) \end{array} \tag{2.10}$$

Por tanto, para conocer la matriz de probabilidades de transición basta conocer la sucesión de matrices de probabilidades de transición de un paso.

Si definimos el vector de probabilidad al cabo de n pasos (para $n = 0, 1, 2, \ldots$) como

$$q^{(n)} = \begin{pmatrix} q_0^{(n)} \\ q_1^{(n)} \\ \vdots \\ q_i^{(n)} \\ \vdots \end{pmatrix} \quad \text{donde} \quad q_i^{(n)} = P[X_n = i] \tag{2.11}$$

se cumple que $q^{(n)} = P(0,n)\cdot q^{(0)}$ siendo $q^{(0)} = \{q_i^{(0)}\}$ el vector de probabilidad inicial, con lo que se deduce que *la ley de probabilidad de una cadena de Markov* $\{X_n\}$ *se determina una vez se conocen la matriz de probabilidades de transición y el vector de probabilidad inicial.*

Si la cadena de Markov $\{X_n\}$ es homogénea, de la ecuación 2.10 se deduce que la matriz de probabilidades de transición de n pasos viene dada por $P(n) = P^n$ siendo $P = (p_{ij})_{1\leq i,j\leq k}$ la matriz de transición en un paso, con lo que $q^{(n)} = P^n \cdot q^{(0)}$. Por consiguiente, la ley de probabilidad queda completamente determinada a partir de la probabilidad de transición de un paso P y el vector de probabilidad inicial $q^{(0)}$.[23]

A lo largo de este trabajo nos centraremos en el estudio de *cadenas de Markov finitas* con k estados, es decir, el número de valores posibles de las variables aleatorias $\{X_n\}$ es finito, igual a k y homogéneas, por tanto, la matriz de probabilidades de transición será de la forma:

$$P = \begin{pmatrix} p_{11} & \cdots & p_{1j} & \cdots & p_{1k} \\ \vdots & \ddots & & & \vdots \\ p_{i1} & & p_{ij} & & p_{ik} \\ \vdots & & & \ddots & \vdots \\ p_{k1} & \cdots & p_{kj} & \cdots & p_{kk} \end{pmatrix} \quad \text{con} \quad e_1, \ldots, e_k \in E \tag{2.12}$$

A modo de ejemplo, supongamos que un ratón se mueve entre los cuatro compartimentos 1,2,3 y 4 del laberinto descrito en la Figura 2.1 de manera que en determinados instantes de tiempo $t = 0, 1, 2, \ldots$ el ratón elige una de las salidas o elige permanecer en el compartimiento (en un intervalo de tiempo dado) con la misma probabilidad admitiendo por hipótesis que "no usa su memoria", es decir, que el movimiento que va a hacer en cada instante es independiente de la sucesión de lugares que ha ido visitando hasta el momento. Así pues, si el

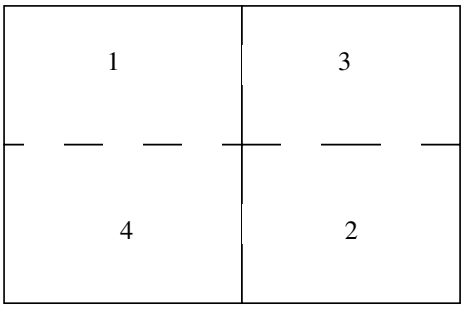

Figura 2.1: Laberinto

ratón está en 1 tiene ante sí cinco sucesos igualmente probables: elegir la salida 3, elegir la primera salida a 4, elegir la segunda salida hacia 4, elegir la tercera salida a 4, o quedarse en 1. Se trata pues de una cadena de Markov con 4 estados cuya matriz de transición viene dada por

$$P = \begin{pmatrix} 1/5 & 0 & 1/5 & 3/5 \\ 0 & 1/4 & 1/2 & 1/4 \\ 1/4 & 1/2 & 1/4 & 0 \\ 3/5 & 1/5 & 0 & 1/5 \end{pmatrix} \tag{2.13}$$

y que nos describe la posición en que se encuentra el ratón en cada instante de tiempo.

2.5. Digrafo

2.5.1. Definiciones básicas

Un **digrafo (grafo dirigido u orientado)** D es un par (V, E) donde V es un conjunto finito de elementos llamados *vértices o nodos* y E es un subconjunto de $V \times V$. Diremos que todo par ordenado $\alpha = (a, b) \in E$ determina un arco que en el caso de ser de la forma (a, a) se denomina lazo. Los vértices a y b de un arco $\alpha = (a, b)$ se llaman extremos del arco siendo a el *vértice inicial u origen* y b el *vértice final.*[55]

Un **subdigrafo** del digrafo $D = (V, E)$ es un digrafo $D' = (V', E')$ tal que $V' \subset V$ y $E' \subset E$. Se dice que el subdigrafo D' es un *subdigrafo generador* de D si $V' = V$.

Un **recorrido** de D es una secuencia finita de vértices y arcos $v_0, \alpha_0, v_1, \cdots \cdots, \alpha_{p-1}, v_p$ tal que $\alpha_i = (v_i, v_{i+1}) \in E$ para $i \in \{1, \cdots, p\}$ siendo p la *longitud* del recorrido que coincide con el número de arcos $(v_0, v_1), (v_1 v_2), \cdots (v_{p-1}, v_p)$ que lo forman. Un recorrido se llama *abierto* si $v_0 \neq v_p$ y cerrado si $v_0 = v_p$. Un *ciclo* es un recorrido cerrado.

En la Figura 2.2 se muestra el digrafo D=(V,E) con $V = \{1, 2, 3, 4, 5, 6\}$ y $E = \{(1, 2), (2, 1)), (2, 2)), (2, 3), (3, 5), (3, 6), (4, 2), (4, 4), (5, 2), (6, 3)\}$.

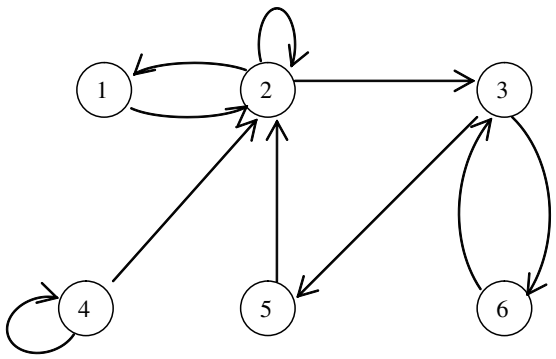

Figura 2.2: Digrafo D

Un subconjunto de vértices $F \subseteq V$ se llama **cerrado** si para todo $v_i \in F$ y para todo $v_j \in F$ entonces $(v_i, v_j) \in E$. Si ningún subconjunto propio es cerra-

do, F se llama **irreductible**.

Un vértice v_i es **accesible** desde el vértice v_j, $v_j \to v_i$, si hay un recorrido que va de v_j a v_i. Dos vértices $v_i, v_j \in V$ se **comunican**,$v_i \leftrightarrow v_j$ si existe un recorrido que va de v_i a v_j y un recorrido que va de v_j a v_i, es decir,$v_i \leftrightarrow v_j \Leftrightarrow v_i \to v_j \quad y \quad v_j \to v_i$. La relación *estar comunicado* es una relación de equivalencia en el conjunto V:

- $v_i \leftrightarrow v_i$
- $v_i \leftrightarrow v_j \Longrightarrow v_j \leftrightarrow v_i$
- $v_i \leftrightarrow v_j, v_j \leftrightarrow v_k \Rightarrow v_i \leftrightarrow v_k$.

Un digrafo $D = (V, E)$ se llama **fuertemente conexo** si para cualquier par $(v_i, v_j) \in V$ x V con $i \neq j$ existe un recorrido que conecta v_i con v_j. Los subdigrafos inducidos por las clases en que se divide el conjunto V a partir de la relación de equivalencia se llaman **componentes fuertemente conexas** del digrafo.

El digrafo D=(V,E) representado en la Figura 2.3 con $V = \{1, 2, 3\}$ y $E = \{(1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 3)\}$ se trata de un digrafo fuertemente conexo ya que siempre existe un recorrido que une dos vértices $(i, j) \in V$xV, con $i \neq j$ cualquiera.

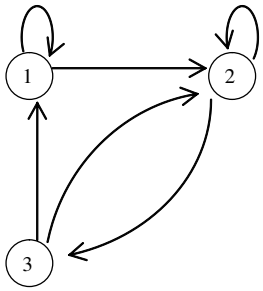

Figura 2.3: Digrafo fuertemente conexo

Un **digrafo ponderado** es una terna (V, E, p) en la que $D = (V, E)$ es un digrafo y $p : E \to \Re$ es una aplicación que asocia a cada arco (a, b) su peso $p((a, b))$.

2.5.2. Digrafo asociado a una cadena de Markov

Según la sección 2.4, una cadena de Markov finita y homogénea viene caracterizada por el vector de probabilidad inicial $q^{(0)} = (q_i^{(0)})_{e_i \in E}$ y por la matriz de probabilidad $P = \{p_{ij}\}_{e_i, e_j \in E}$. Esta matriz da lugar a un digrafo $D = (E, A)$ donde el conjunto de vértices está formado por los estados y el conjunto de arcos viene determinado por $A = \{(e_i, e_j) \mid p_{ij} > 0\}$ representando las transiciones posibles entre estados. De esta manera un estado e_i es accesible desde el estado e_j, $e_j \to e_i$ si y sólo si $i = j$ o existe un entero positivo n tal que $p_{ji}^{(n)} > 0$. Análogamente dos estados e_i y e_j se comunican, $e_i \leftrightarrow e_j$, si y solo si $i = j$ o existe un entero positivo n tal que $p_{ij}^{(n)} > 0$ y un entero positivo m tal que $p_{ji}^{(m)} > 0$.

Consideramos el ejemplo de la sección 2.4 del ratón que se mueve por un laberinto, Figura 2.1. En este caso el digrafo asociado a la cadena de Markov que describe el movimiento del ratón se trata de un digrafo ponderado cuya representación se muestra en la Figura 2.4.

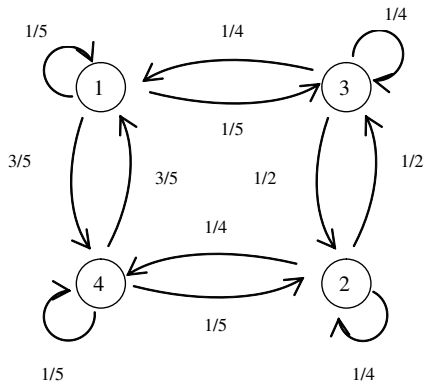

Figura 2.4: Digrafo asociado al ejemplo del laberinto

2.6. Clasificación de los estados

2.6.1. Clasificación de los estados de una cadena de Markov en términos de las propiedades aritméticas de las probabilidades de transición $p_{ij}^{(n)}$

En este apartado se tratará de clasificar los estados de una cadena de Markov según su digrafo asociado. [5], [25]

Un estado e_i es **no esencial** o **de paso** si existe un entero positivo m y un estado e_j tal que $p_{ij}^{(m)} > 0$ pero $p_{ji}^{(n)} = 0$ para todo n, en caso contrario, es **esencial** o **final**.

Consideramos el conjunto de los estados esenciales. Teniendo en cuenta las definiciones de la sección 2.5.2, desde un estado esencial solamente son accesibles los estados esenciales de la misma clase, consecuentemente, este conjunto se puede separar en $E_1, \cdots E_m$ conjuntos disjuntos o clases maximales por la relación de equivalencia $\leftrightarrow$. En el caso que una cadena de Markov esté formada por una sola clase, diremos que la cadena es **irreductible**, es decir, su digrafo asociado es fuertemente conexo. Si una clase maximal está formada por un solo estado, éste se llama **absorbente**.

Consideramos una cadena de Markov formada por 6 estados $E = \{e_1, e_2, e_3, e_4, e_5, e_6, \}$ y cuyo digrafo asociado se muestra en la Figura 2.5.

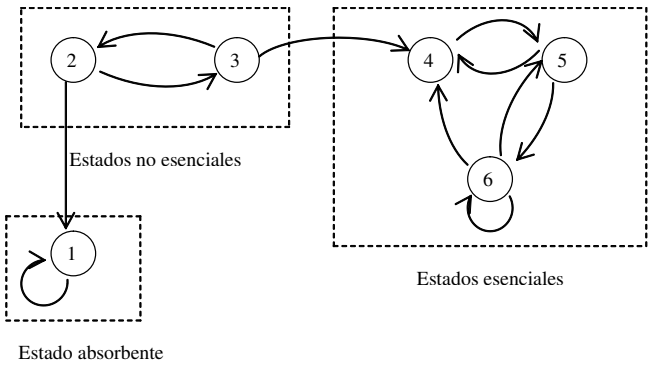

Figura 2.5: Digrafo con los estados clasificados

Según la clasificación definida en este apartado se trata de una cadena for-

mada por dos clases maximales E_1 y E_2 formadas por un estado absorbente $E_1 = \{e_1\}$ y tres estados esenciales $E_2 = \{e_4, e_5, e_6\}$ y por una clase de estados no esenciales $E_0 = \{e_2, e_3\}$.

Todas las clases maximales conjuntamente con el conjunto de estados de paso E_0 forman una partición de E de manera que la matriz de probabilidades de transición se puede escribir de la forma

$$P = \begin{pmatrix} Q_0 & Q_1 & \dots & \dots & Q_m \\ 0 & R_1 & 0 & \dots & 0 \\ \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \dots & 0 & R_m \end{pmatrix} \tag{2.14}$$

llamada forma canónica, donde la matriz Q_0 está formada por las probabilidades de transición entre los estados no esenciales de E_0, las matrices $R_1, \cdots, R_m$ están formadas por las probabilidades de transición entre los estados esenciales de cada clase maximal $E_1, \cdots E_m$ y las matrices $Q_1, \cdots, Q_m$ están formadas por las probabilidades entre los estados de paso y los estados finales de cada clase maximal.

Definición Dado un estado e_i definimos su periodo como

$$d(i) = m.c.d.\{n \quad \text{tal que } p_{ii}^{(n)} > 0\}. \tag{2.15}$$

Teorema 1.5.1 Dos estados pertenecientes a la misma clase de equivalencia tienen el mismo periodo llamado periodo de la clase.

En términos del digrafo asociado a la cadena de Markov, este teorema nos permite definir el período de una clase como el período de uno cualquiera de sus vértices.

Sea $e_i \in E$ con $d(i) = 1$, entonces la clase E se llama **aperiódica** y si $d(i) > 1$ se llama **periódica** de periodo $d(E) = d(i)$.[34]

Si consideramos la cadena de Markov cuyo digrafo se muestra en la Figura 2.5 observamos que la clase E_0 es una clase periódica con periodo $d(E_0) = 2$ mientras que la clase E_2 es una clase aperiódica con periodo $d(E_1) = 1$.

Lema 1.5.1 Sean $a_1, \cdots a_r \in N$ tales que $m.c.d.(a_1, \cdots, a_r) = 1$. Denotamos con A^+ al conjunto de todas las combinaciones lineales positivas

$$p_1a_1 + \cdots p_ra_r, \quad con \quad p_i \in Z^+, 1 \leq i \leq r$$

Entonces existe un número N tal que A^+ contiene a todos los enteros $n > N$.[18]

Demostración

Si $1 = m.c.d.(a_1, \cdots a_r)$ podemos expresar el 1 como $1 = a_1c_1 + \cdots + a_rc_r$ con $c_i \in Z, 1 \leq i \leq r$ según la Identidad de Bézout.
Sea $s = a_1 + \cdots a_r$. Cada entero positivo n admite una representación única $n = xs + y$ donde $x, y \in Z$ y $0 \leq y < s$ (teorema de la división entera). Entonces,

$$n = x(a_1+\cdots+a_r)+(a_1c_1+\cdots+a_rc_r)y = (x+yc_1)a_1+\cdots+(x+yc_r)a_r =$$

$$= \sum_{k=1}^{r}(x + c_ky)a_k$$

con $p_k = (x + c_ky)$ que serán positivos tan pronto como x exceda de y un número de veces igual al mayor de los números $\mid c_k \mid$.

$\diamondsuit$

Proposición 1.5.1 Una clase E maximal es aperiódica si y solo si existe un instante $m_0 \geq 1$ y un estado e_{k_0} perteneciente a la clase tal que para todo $e_j \in E$, $p_{jk_o}^{(m_0)} > 0$.

Demostración

$\Rightarrow$ Sea E una clase maximal aperiódica con k estados y sea e_{k_0} un estado de E. El conjunto $\{n_0 \in N\,; p_{k_0k_0}^{(n_0)} > 0\}$ está formado por $\{a_1, \ldots, a_n, \sum_{i=1}^{n} a_ir_i\}$ donde $a_i, r_i \in N$, $n \leq k$ y m.c.d.$\{a_1, \ldots, a_n\} = 1$ ya que por ser E aperiódica el periodo de sus estados es 1. Sea $m_0^{'}$ la primera de las combinaciones a partir de la cual las demás son consecutivas(que existe por el lema 1.5.1).
Si definimos,

$$n_i = min\{n; p_{ik_0}^{(n)} > 0\}$$
$$n = max\{n_i\}$$
$$m_0 = m_0^{'} + n,$$

entonces, $\forall e_i \in E$ se cumple que $p_{ik_0}^{(m_0)} \geq p_{ik_0}^{(n_i)} p_{k_0k_0}^{(m_0-n_i)} > 0$ ya que $m_0 - n_i = m_0^{'} + n - n_i$ existe para todo $n - n_i$ por el lema 1.5.1.

$\Leftarrow$ Supongamos que existe un instante $m_0 \geq 1$ y un estado $e_{k_o} \in E$ tal que $\forall\ e_j \in E$, $p_{jk_0}^{(m_0)} > 0$ y por tanto, $p_{k_0k_0}^{(m_0)} > 0 \Rightarrow m_0 = \widehat{\dot{d(k_0)}}$.
Sea $e_i \in E$ un estado tal que $p_{k_0i} > 0 \Rightarrow 0 < p_{ik_0}^{(m_0)} p_{k_0i} \leq p_{ii}^{(m_0+1)} \Rightarrow m_0 + 1 = \widehat{\dot{d(i)}}$.
Por el teorema 1.5.1 $d(i) = d(k_0)$ y por tanto, $d(E) = 1$, es decir, E es una clase aperiódica.

$\Diamond$

Sea E una clase de equivalencia de periodo d. Fijamos un estado $e_i \in E$ y definimos las siguientes subclases:

$$\begin{gathered} C_0 = \{e_j \in E | p_{ij}^{(n)} > 0 \Rightarrow n \equiv 0(\text{mod d})\}; \\ C_1 = \{e_j \in E | p_{ij}^{(n)} > 0 \Rightarrow n \equiv 1(\text{mod d})\}; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ C_d - 1 = \{e_j \in E | p_{ij}^{(n)} > 0 \Rightarrow n \equiv d - 1(\text{mod d})\} \end{gathered}$$

Estos conjuntos reciben el nombre de **subclases cíclicas** y forman una partición de E.

Teorema 1.5.2 Sea $e_j \in C_p$ y supongamos que $p_{jk} > 0$, entonces $e_k \in C_{p+1(\text{mod d})}$.

Según este teorema la submatriz de las probabilidades de transición de una clase cíclica E de periodo d tiene la estructura siguiente:

$$\begin{pmatrix} 0 & C_{0,1} & 0 & \cdots & 0 \\ \vdots & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ 0 & 0 & \ldots & 0 & C_{d-2,d-1} \\ C_{d-1,0} & 0 & \cdots & \cdots & 0 \end{pmatrix} \tag{2.16}$$

2.6.2. Clasificación de los estados de una cadena de Markov en términos de las propiedades asintóticas de las probabilidades de transición $p_{ij}^{(n)}$

Para todo $e_i, e_j \in E$ definimos:

$$f_{ii}^{(n)} = P(\{X_n = i, X_m \neq i, 1 \leq m \leq n-1\}/X_0 = i) \tag{2.17}$$

y para $i \neq j$

$$f_{ij}^{(n)} = P(\{X_n = j, X_m \neq j, 1 \leq m \leq n-1\}/X_0 = i) \tag{2.18}$$

que representan respectivamente la probabilidad que saliendo del estado e_i el proceso regrese por primera vez al estado e_i en el instante n y la probabilidad que saliendo del estado e_i el proceso pase por el estado e_j por primera vez en el instante n.

Para cada $e_i \in E$ la probabilidad que saliendo del estado e_i el proceso regrese al estado e_i alguna vez viene dada por

$$f_{ii} = \sum_{n=1}^{\infty} f_{ii}^{(n)} \tag{2.19}$$

Un estado $e_i \in E$ es **recurrente** si $f_{ii} = 1$ y es **transitorio** si $f_{ii} < 1$.

Para cada estado $e_i \in E$ recurrente se define el **tiempo medio de recurrencia** como:

$$\mu_i = \sum_{n=1}^{\infty} n f_{ii}^{(n)} \tag{2.20}$$

Un estado $e_i \in E$ recurrente diremos que es **recurrente positivo** si $\mu_i < \infty$ y que es **recurrente nulo** si $\mu_i = \infty$. En una cadena de Markov finita no existen estados nulos. [18]

Como el cálculo de las funciones $f_{ii}^{(n)}$ puede ser bastante complicado, los siguientes lemas para probar si un estado es o no recurrente van a ser de utilidad.

Lema 1.6.1

a) Un estado e_i es recurrente si i sólo si

$$\sum_{n=1}^{\infty} p_{ii}^{(n)} = \infty$$

b) Si un estado e_j es recurrente y $e_i \leftrightarrow e_j$ entonces el estado e_i es también recurrente.

Lema 1.6.2 Si e_j es un estado transitorio, entonces

$$\sum_{n=1}^{\infty} p_{ij}^{(n)} < \infty$$

para todo e_i y por tanto $p_{ij}^{(n)} \to 0, \quad n \to \infty$.

Teorema 1.6.1 Todos los estados de una cadena finita de Markov no pueden ser transitorios.

Teorema 1.6.2(*Teorema básico del límite*) Sea $f_1, f_2 \cdots$ una sucesión de números $f_n \geq 0$ tales que $\sum f_n = 1$ y 1 es el máximo común divisor de las n para las cuales $f_n > 0$. Sean $u_0 = 1$, y

$$u_n = f_1 u_{n-1} + f_2 u_{n-2} + \cdots + f_n u_0 = \sum_{n=1}^{n} f_k u_{n-k}, \quad n \geq 1. \tag{2.21}$$

Entonces,

$$u_n \to \mu^{-1} \quad donde \quad \mu = \sum_{n=1}^{\infty} n f_n \tag{2.22}$$

Lema 1.6.3 Sea e_j un estado recurrente con $d(j) = 1$.

a) Si $e_i \leftrightarrow e_j$, entonces

$$p_{ij}^{(n)} \to \frac{1}{\mu_j} > 0, \quad n \to \infty$$

b) Si e_i y e_j pertenecen a diferentes clases, entonces

$$p_{ij}^{(n)} \to \frac{f_{ij}}{\mu_j} \quad n \to \infty$$

Lema 1.6.4 Sea e_j un estado recurrente con $d(j) > 1$.

a) Si e_i y e_j pertenecen a la misma clase y si e_i pertenece a la subclase cíclica C_r y e_j a C_{r+a}, entonces

$$p_{ij}^{(nd+a)} \to \frac{d}{\mu_j}$$

.

b) Si e_i es arbitrario, entonces,

$$p_{ij}^{(nd+a)} \to (\sum_{r=0}^{\infty} f_{ij}^{((rd+a)}) \frac{d}{\mu_j}$$

siendo $a = 0, 1 \cdots, d-1$.

2.7. Estudio de la existencia de la distribución límite

Para finalizar este capítulo, daremos respuesta a la pregunta de en qué condición existe la distribución límite para una cadena finita de Markov. Empezaremos con algunas condiciones necesarias para la existencia de la distribución estacionaria π, $\pi = \pi P$ donde P es la matriz de transición de la cadena de Markov. Ver [33] o [47].

Teorema 1.7.1 Consideramos una cadena de Markov con un número finito de estados y con matriz de transición $P = (p_{ij})$ tal que el límite

$$\lim_{n\to\infty} p_{ij}^n = \pi_j$$

existe para todo e_i y e_j y no depende de e_i.
Entonces:

a) $\sum_i \pi_i \leq 1, \quad \sum_i \pi_i p_{ij} = \pi_j$

b) Para todo e_j, $\pi_j = 0$ o bien $\sum_j \pi_j = 1$

c) Si $\pi_j = 0$ para todo e_j entonces no existe la distribución estacionaria. Si $\sum_j \pi_j = 1$, entonces $\pi = (\pi_1, \cdots, \pi_k)$ es la única distribución estacionaria.

Teorema 1.7.2 Para una cadena de Markov con un número finito de estados existe una única distribución estacionaria si y solo si el conjunto de estados contiene exactamente una clase recurrente positiva.

Teorema 1.7.3 En una cadena finita de Markov existe la distribución límite si y solo si hay, en el conjunto E de estados de la cadena, exactamente una clase recurrente positiva aperiódica C tal que $f_{ij} = 1$ para todo $e_j \in C$ y $e_i \in E$.

Capítulo 3

Computación natural

3.1. Introducción

El afán del hombre para resolver situaciones con las que se ha ido encontrando a lo largo de la historia le ha llevado a la búsqueda de métodos de resolución o algoritmos. A finales del siglo XIX surge la necesidad por la rigorización de dichos métodos siendo los matemáticos quienes en la resolución de ciertos problemas usen el formalismo y la metodología lógica matemática. Siguiendo en la línea de la formalización es cuando a principios del siglo XX surge la idea de encontrar una axiomatización de las matemáticas que llevará a la definición de un nuevo modelo para máquinas, el modelo de la computación. Dicho modelo deberá resolver en tiempo y espacio todo tipo de problemas.

En la década de los sesenta surge la teoría de la Complejidad. Las máquinas computacionales de las que se dispone poseen una potencia muy limitada. La excesiva lentitud y la escasa memoria hacen necesario el estudio de la cantidad de recursos que un algoritmo necesita para su ejecución así como un estudio comparativo de distintas soluciones algorítmicas de un mismo problema. Con el desarrollo de máquinas computacionales teóricas se empieza a pensar en sus limitaciones en tiempo y espacio físico a la hora de la resolución de determinados problemas.

En la búsqueda de modelos alternativos de computación que tengan en cuenta la minimización en tiempo y espacio, aparece la utilización de ideas biológicas en el marco computacional: esfuerzo para entender los procesos que se dan en la naturaleza con la finalidad de obtener algoritmos más eficientes o incluso para

el diseño de nuevos tipos de ordenadores. De esta manera aparece la computación natural como una de las posibles alternativas a la computación clásica para proporcionar una solución efectiva a las limitaciones que estos modelos poseen. El hecho que en espacio reducido sea capaz de guardar mucha más información y la procese de manera rápida y eficiente sin errores de cálculo, hace pensar que la naturaleza sea un perfecto candidato. Este tipo de computación se caracteriza por la simulación e implementación de procesos dinámicos que se dan en la naturaleza y que son susceptibles de ser interpretados como procedimientos de cálculo.

Las dos áreas que han alcanzado mayor auge en la computación natural son la computación molecular con ADN y más recientemente, la computación celular con membranas. Una de las ventajas de estos modelos respecto los modelos clásicos es el paralelismo masivo que en ellos se implementa de manera natural permitiendo ejecutar simultáneamente muchas operaciones en una unidad de tiempo.

La computación molecular tiene como objetivo usar moléculas orgánicas como hardware biológico para realizar cómputos. Estas moléculas, en un entorno con determinadas propiedades, pueden evolucionar hacia estados estructuralmente más complejos modificando sus características. La computación molecular proporciona un modelo de computación orientado a programas siguiendo, por lo tanto, una estructura similar a la de los algoritmos clásicos en donde la realización de cada paso depende del resultado del paso anterior. El primer modelo teórico fue propuesto por T. Head a mediados de la década de los ochenta. Este modelo, *el modelo splicing*, se basa en la capacidad del ADN para almacenar y manipular información. En 1994 L.M. Adleman [1] realiza el primer experimento en el laboratorio que permite resolver un problema matemático computacionalmente intratable.

La computación celular con membranas o P sistemas, se introduce a finales de la década de los noventa con Gh. Păun [39] y está inspirado en la forma en que las células vivas procesan compuestos químicos provocando una transformación de las componentes químicas presentes en sus membranas. Estos procesos a nivel celular pueden ser interpretados como procedimientos de cálculo. El modelo de computación proporcionado por la computación celular está orientado a máquina, se empieza con una configuración inicial (estructura de membranas con ciertas componentes químicas en sus compartimientos) que va evolucionando

mediante un sistema de reglas (abstracción de las reacciones químicas). Ver[45].

3.2. Computación molecular

3.2.1. Introducción

El estudio de cómo las características genéticas pasan de generación en generación ha despertado el interés de muchos científicos que les ha incitado el afán por saber cual es la naturaleza del gen. A principios de los 40 se llega al descubrimiento que el material hereditario se encuentra en los cromosomas y que está formado por ácido desoxirribonucleico (ADN) y proteínas. Pero no es hasta principios de los 50 cuando se resuelve la duda de cuál de las dos sustancias es el material hereditario, llegándose a la conclusión que es el ADN quien lleva la información genética.

Basándose en la información procedente de diversos estudios sobre la estructura del ADN y el mecanismo hereditario, en 1953, los científicos James Watson y Francis Crick empiezan a descifrar la estructura del ADN, doble hélice entrelazada y de gran longitud formada únicamente por cuatro bases. Estos dos biólogos proponen un modelo de mecanismo hereditario mediante el cual la molécula de ADN se replica según un proceso semiconservativo, llegan de esta manera a mostrar como se puede almacenar la información en la molécula del ADN.

Desde ese momento el interés por descifrar la información que contiene el ADN conlleva a la realización de muchos proyectos entre los que se encuentra el proyecto del Genoma Humano, secuenciación de genomas, iniciado en 1990.

A partir de las investigaciones realizadas en el proyecto del genoma se obtienen nuevas técnicas que permiten manipular el ADN en el laboratorio simulando sus procesos de manera similar a su comportamiento en las células. El hombre es capaz de desarrollar en el laboratorio muchos de los procesos que tienen lugar en la célula en el orden que desea. Si se tiene presente la capacidad del ADN de almacenar tan solo mediante 4 letras una inmensa información con la cual, de acuerdo a las necesidades biológicas de los seres, el ADN efectúa diferentes operaciones en paralelo, parece lógico pensar que es un perfecto candidato para efectuar cálculos costosos en un breve espacio de tiempo. Estas nuevas técnicas pertenecientes al campo del ADN recombinante permiten realizar en el labora-

torio una serie de operaciones biológicas. Leonard Adleman en 1994 [1] utiliza alguna de estas técnicas para resolver el problema del camino Hamiltoniano. De esta forma se inicia un nuevo campo de investigación, la computación con ADN, cuyo objetivo consiste en usar esta molécula como hardware biológico que permite realizar computaciones.

3.2.2. Estructura del ADN

El ADN (ácido desoxirribonucleico) se encuentra en el núcleo de las células y es la molécula que codifica la información genética. Está formado por unas subunidades llamadas nucleótidos compuestos por un glúcido de desoxirribosa, un grupo fosfato y una base nitrogenada que es la que los diferencia, (figura3.1).

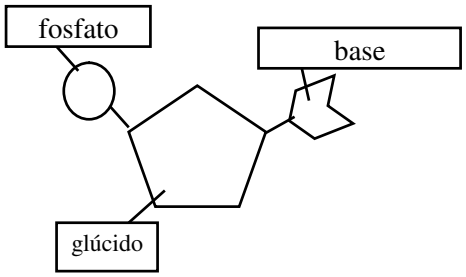

Figura 3.1: Elementos que forman un nucleótido.

Las cuatro bases son la adenina, la citosina, la guanina y la timina, comúnmente abreviadas por las letras A, C, G y T, respectivamente (figura 3.2).

Los nucleótidos se unen entre ellos por enlaces entre el grupo fosfato 5' de un nucleótido y el grupo hidróxilo 3' del nucleótido adyacente formando las cadenas simples de ADN. Una cadena simple corta, normalmente de menos de 30 nucleótidos, recibe el nombre de oligonucleótido.

Esta forma de unirse los nucleótidos induce una orientación en la cadena simple de ADN conocida como 5' 3', es decir, bajo el nombre del extremo inicial 5' y el extremo final 3' de la cadena. La estructura de doble hélice de ADN se forma cuando dos cadenas simples se unen mediante puentes de hidrógeno entre pares de bases, A con T y C con G, que reciben el nombre de pares de bases complementarias. Además, para que dos cadenas simples de ADN se unan entre

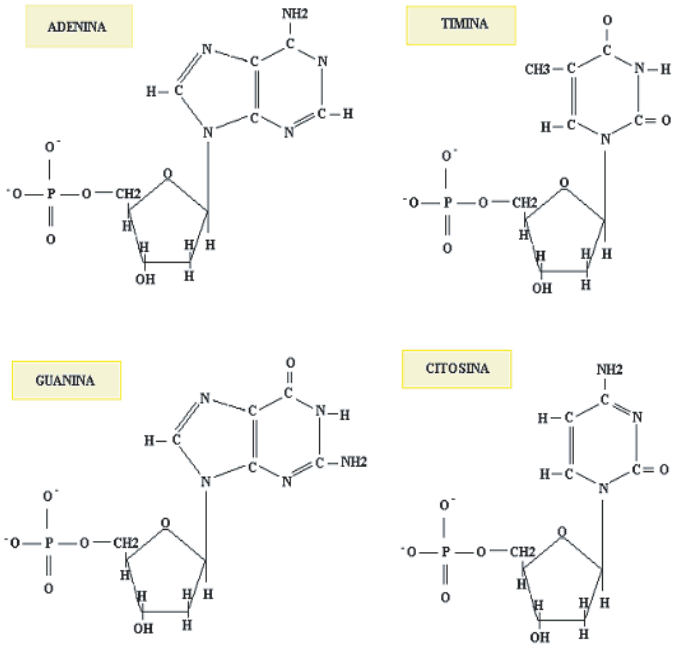

Figura 3.2: Estructura de la adenina, la citosina, la guanina y la timina.

ellas deben tener orientaciones opuestas: el extremo 3' de una de ellas ha de unirse con el extremo 5' de la otra (figura 3.3). Esta estructura fue descubierta por Watson y Crick [50]. Sin embargo, dos cadenas que no son completamente complementarias pueden unirse si las condiciones de temperatura y de medio son las adecuadas.

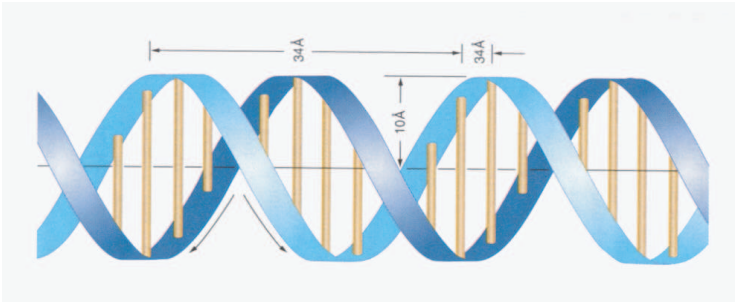

Figura 3.3: Estructura de la cadena doble de ADN

3.2.3. Operaciones biológicas

Los modelos de computación con ADN se basan en una secuencia de operaciones biológicas sobre un conjunto de cadenas [24]. A continuación presentamos una breve descripción de aquellas que serán necesarias para la obtención de algunos de los resultados que se presentan en este trabajo.

Síntesis

La síntesis de una cadena según una secuencia de bases fijada consiste en generar una cadena simple de ADN partiendo de los cuatro tipos de nucleótidos. Existen básicamente dos maneras de sintetizar cadenas de ADN, la química, realizada en los laboratorios mediante la cual se fabrican cadenas artificialmente de acuerdo a una secuencia deseada, y la natural, que es la que realizan las propias células (figura 3.4).

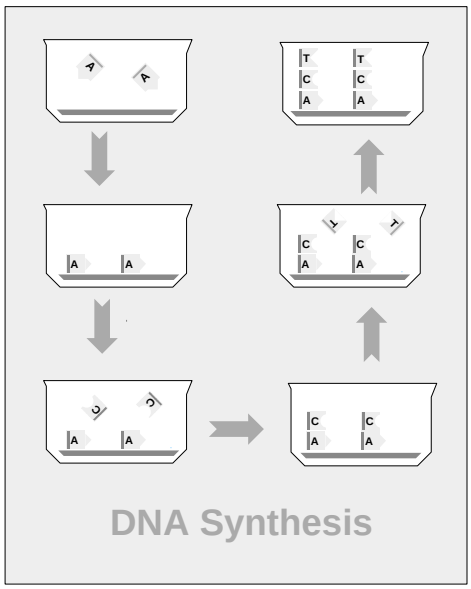

Figura 3.4: Síntesis. Fuente objeto [24].

Amplificación

Esta operación nos permite hacer crecer exponencialmente el contenido de cadenas dobles de ADN. La amplificación permite hacer copias de cadenas

de ADN. La técnica que hoy en día se utiliza es la conocida como PCR, Polymerasse Chain Reaction [48]. La PCR es un método in vitro que requiere del enzima ADN polimerasa para poder amplificar una secuencia específica de ADN en disolución.
Para la amplificación de una cadena de ADN se necesita una cadena simple de ADN (conocida como template) y un primer, oligonucleótido de unos 20 nucleótidos complementario al extremo del trozo de cadena que se quiere amplificar. La PCR supone una serie repetitiva de ciclos de temperatura cada uno de los cuales está formado por tres etapas:

- calentar la disolución para que se separen (melting) las cadenas que forman la doble hélice
- enfriar la disolución para que los primers se unan (annealing) a cada extremo de las dos cadenas simples (templates)
- extensión de los primers gracias al ADN polimerasa hasta completar cada una de las cadenas.

Cada ciclo dobla el número de cadenas de ADN a las que se les aplica la PCR (figura 3.5).

Renaturalización y Desnaturalización

Dos cadenas simples complementarias de ADN se unen formando una cadena doble. Esta operación, renaturalización, se produce al enfriar la solución. Recíprocamente, al aumentar la temperatura las cadenas dobles de ADN se separan en cadenas simples, desnaturalización.

Gel electroforesis

La técnica conocida como gel electroforesis permite separar las cadenas de ADN por longitud o peso. Se colocan las moléculas en la parte superior del gel húmedo (hecho normalmente de agarosa, policramida o una combinación de los dos) y se aplica un campo eléctrico que produce el movimiento de estas moléculas que están cargadas eléctricamente hacia la parte de abajo. El gel es una red densa con poros a través de los cuales las moléculas pueden moverse de manera que las moléculas largas se deslizan más lentamente. Después de un período de tiempo, las moléculas quedan distribuidas en distintas bandas según su desplazamiento, que depende de su longitud y su peso. Para poder visualizar los resultados se tiñe el ADN con bromuro fluorescente y se observa el gel mediante una luz ultravioleta (figura 3.6).

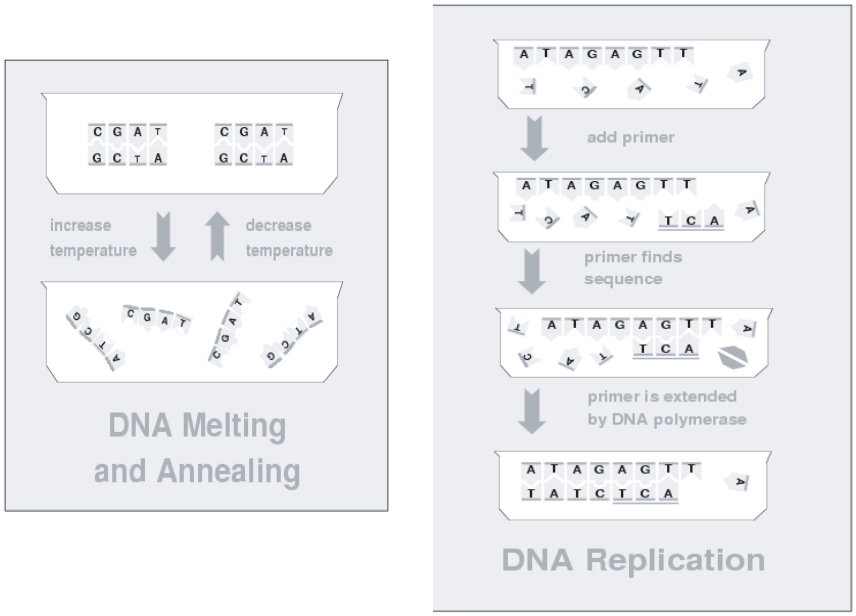

Figura 3.5: Etapas de la PCR. Fuente objeto [24].

Extracción

Este proceso, conocido también como purificación por afinidad o aislamiento, nos permite extraer de la disolución las cadenas simples de ADN que contienen una secuencia corta de bases fijada (figura 3.7).

Detección

Detecta la existencia de una o unas secuencias concretas de bases en una cadena de ADN. Para ello se necesitan unos cebadores (primers) que indiquen el principio y el final de la secuencia a copiar. Si encuentran la secuencia, se copia mediante la PCR. Otra forma de detección es marcando con fluorescencia la secuencia a detectar.

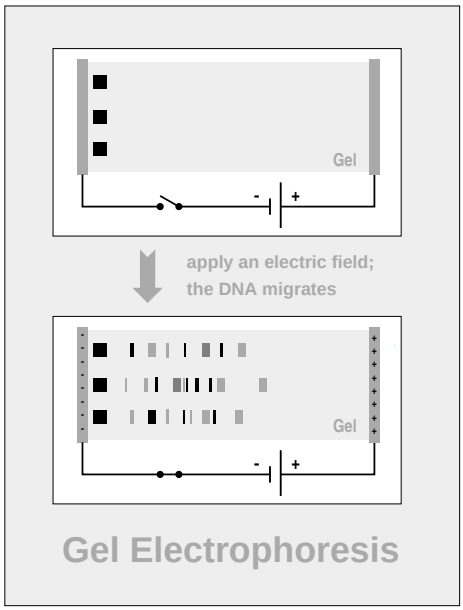

Figura 3.6: Gel electroforesis. Fuente objeto [24].

Ligación

Es la unión de cadenas dobles por uno de los extremos pegajosos gracias a la presencia en disolución del enzima DNA ligasa.

Cortar

Cadenas dobles de ADN se cortan por un lugar específico gracias a la presencia en disolución de enzimas de restricción. Estos enzimas en el momento que encuentran una secuencia concreta de bases cortan la cadena(figura 3.8).

3.2.4. Operaciones algebraicas

Las cadenas simples de ADN se pueden representar matemáticamente mediante una cadena x formada por las letras del alfabeto $\{A, G, C, T\}$ de manera que $\overline{x}$ representará su cadena complementaria según Watson-Crick. Las operaciones biológicas se aplican a un conjunto de cadenas en un tubo de ensayo. A continuación se presentan las operaciones necesarias para llevar a cabo los diferentes algoritmos propuestos en los artículos del capítulo 3 y 4.

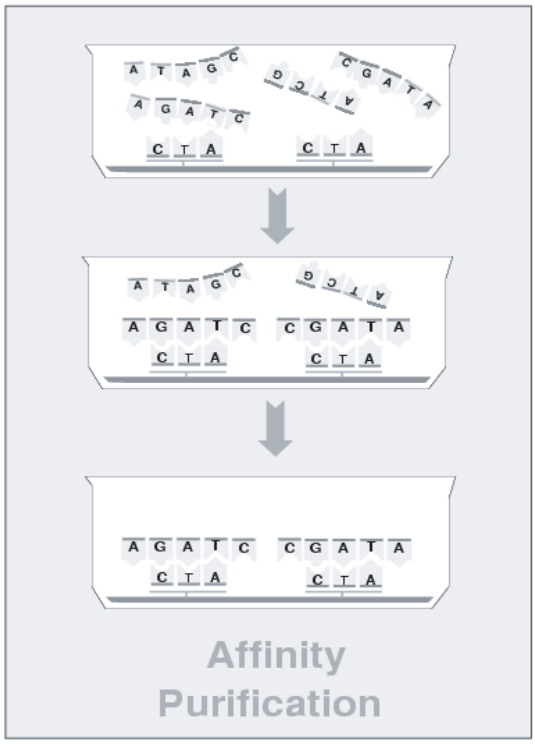

Figura 3.7: Purificación por afinidad. Fuente objeto [24].

Length–extract. Dado un tubo de ensayo U que contiene una solución de cadenas de ADN y dado un entero n se produce, mediante *gel electroforesis*, un nuevo tubo, $\texttt{lextract}(U, n)$. Este contiene en todas las cadenas de U cuya longitud es igual a n.

Ligate. Dados dos tubos de ensayo U_1 y U_2 de cadenas de ADN, mediante *ligación*, se obtiene otro tubo $\texttt{ligate}(U_1, U_2)$ que contiene cadenas formadas por la unión de cadenas de U_1 con cadenas de U_2.

String–extract. Dado un tubo de ensayo U de cadenas de ADN y dada una cadena x, mediante *detección* y *extracción*, se obtiene un nuevo tubo de ensayo, $\texttt{sextract}(U, x)$ formado por las cadenas de U que contienen la cadena x.

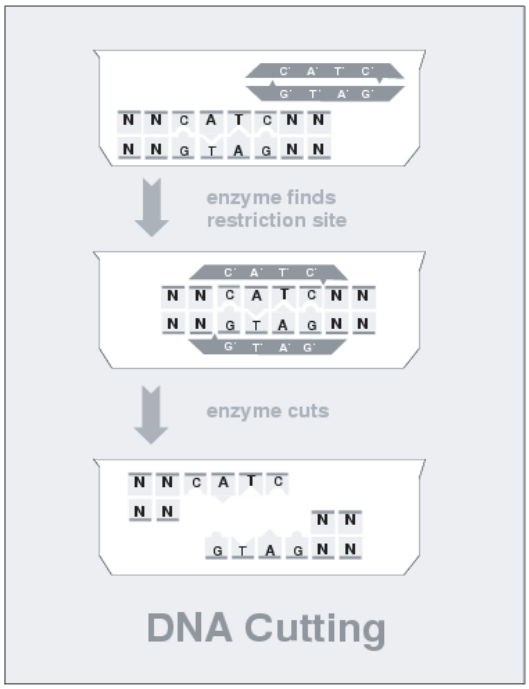

Figura 3.8: Cortar. Fuente objeto [24].

Copy. Dado un tubo de ensayo U de cadenas de ADN y dadas dos cadenas x y y, mediante *amplificación*, se produce un nuevo tubo de ensayo, $\texttt{copy}(U, x, y)$ formado por copias de piezas de las cadenas de U que empiezan por x y acaban por y de manera que solamente contienen una vez la cadena y.

3.3. Computación celular con membranas

3.3.1. Introducción

El estudio del funcionamiento de los organismos vivos ha llevado a la introducción de los modelos de computación natural. Hasta finales de la década de los noventa se simula el modo en que la naturaleza calcula a un nivel genético o a un nivel neuronal. No obstante, más recientemente se ha llegado a un nuevo nivel de computación en el cual se considera el comportamiento de las células como el de una máquina de cálculo: *el nivel celular*.

Uno de los objetivos de este modelo de computación es simular el comportamiento de las células con la finalidad de obtener soluciones alternativas a problemas que actualmente son computacionalmente intratables. Dicho comportamiento puede ser considerado como una máquina no trivial en la que por medio de una distribución jerárquica de membranas interiores se produce el paso y la alteración de las componentes químicas que la propia célula procesa.

La distintas partes del sistema biológico que componen las células se encuentran delimitadas por varios tipos de membranas, desde la membrana más externa, *la piel*, hasta las distintas membranas internas. Estas membranas permiten el paso de ciertos orgánulos de forma selectiva e incluso algunas veces en una sola dirección. Por ello, este modelo de computación está basado en una *estructura de membranas* en el cual distintas unidades de cálculo trabajan independientemente pero estructuradas en una jerarquía vertical. En estas membranas se introducirán unos elementos que pueden aparecer repetidos y sin orden de ubicación denominados *multiconjuntos* de *objetos* y se definen una serie de *reglas de evolución* (reacciones químicas). Si estos objetos son capaces de evolucionar, se obtiene un mecanismo de computación llamado *P sistema de transición*. Ver [39], [41] y [44].

3.3.2. La célula

Entre los muchísimos tipos de células que existen, se encuentran las células eucariotas que son las de todos los seres vivos con la excepción de las bacterias. A pesar de su diversidad, todas son muy similares. Cada célula es una unidad autónoma rodeada por una membrana que controla el paso de materiales hacia su interior y hacia su exterior. Esto hace posible que la célula difiera bioquímica y estructuralmente del medio circundante. La materia viva limitada por la

membrana consiste, en los eucariotas, en el núcleo y el citoplasma, que contiene los orgánulos (figura 3.9).

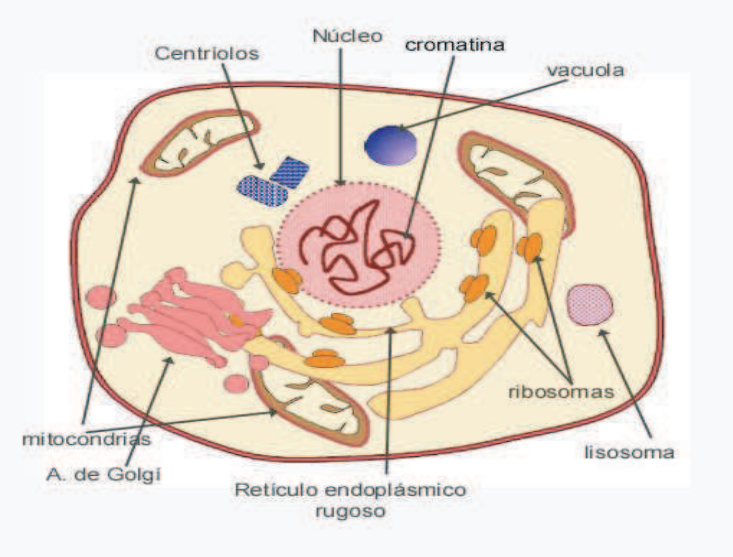

Figura 3.9: La célula eucariótica. Fuente objeto: [53]

La membrana celular, *membrana plasmática*, delimita el territorio de la célula y controla el contenido químico de la célula: representa el límite entre el medio extracelular y el intracelular. Es de gran importancia para los organismos ya que a través de ella se transmiten mensajes que permiten a las células realizar numerosas funciones. La célula de los eucariotas está formada por bicapas de fosfolípidos donde se encuentran moléculas de proteínas y de colesterol (figura 3.10).

El núcleo es un cuerpo grande rodeado por una membrana doble, la envoltura nuclear, que lo separa del citoplasma. Los poros de la envoltura nuclear suministran los canales a través de los cuales pasan las moléculas desde y hacia el citoplasma. El núcleo contiene el material hereditario, los cromosomas, que, cuando la célula no está dividiéndose, están en una forma extendida llamada cromatina. El nucleolo, visible dentro del núcleo, es responsable de la formación de ribosomas. Al actuar juntamente con el citoplasma, el núcleo ayuda a regular las actividades de la célula.

El citoplasma se encuentra dentro de la membrana y contiene los orgánulos

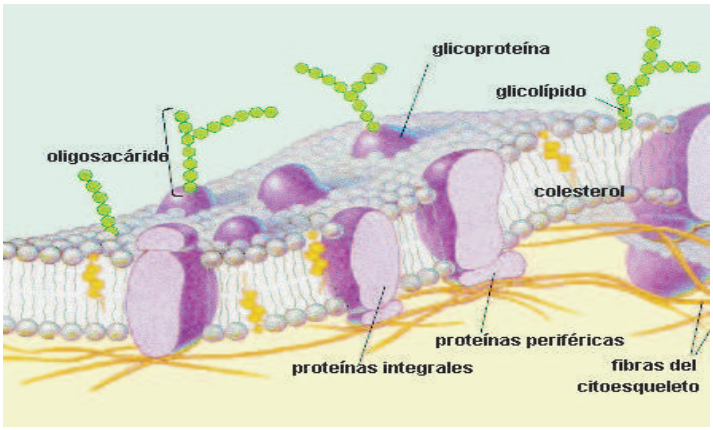

Figura 3.10: La membrana plasmática. Fuente objeto: [53]

(vacuolas y vesículas (lisosomas y peroxisomas), ribosomas, complejos de Golgi, mitocondrios, centríolos). Está atravesado y subdividido por un complejo sistema de membranas, el retículo endoplasmático, que sirven como superficie de trabajo para muchas de sus actividades bioquímicas. En el citoplasma es dónde se ejecutan prácticamente todas las funciones.

El funcionamiento de la célula ha sido implantado en el modelo de computación celular con membranas, P sistemas. Los objetos de este modelo corresponden a los orgánulos que hay en el citoplasma. Estos, mediante las señales que reciben mediante reacciones químicas, pueden transformase dentro del citoplasma. Estas transformaciones internas corresponden a las reglas internas que en este modelo se describen. Finalmente, el paso selectivo hacia dentro y hacia fuera que permite la membrana a algunos orgánulos, corresponde a las reglas externas de los P sistemas.

3.3.3. Estructura de membranas

La estructura de membranas de un P sistema consiste en una disposición jerárquica de membranas rodeadas de una membrana externa, la piel, que las separa del entorno. Cada membrana, etiquetada con un nivel, define una región que contiene un multiconjunto de objetos y un conjunto de reglas de evolución. Gráficamente se representa una estructura de membrana mediante un diagrama de Euler-Venn (figura 3.11). Matemáticamente, se representa como un árbol

enraizado en el que los nodos se denominan membranas, la raíz se denomina piel, y las hojas se denominan membranas elementales (figura 3.12) o bien mediante una cadena de paréntesis.

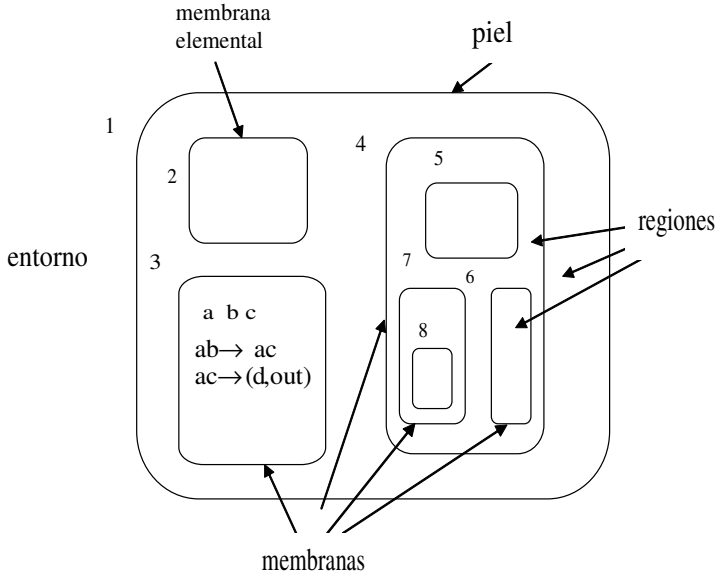

Figura 3.11: Estructura de membrana.

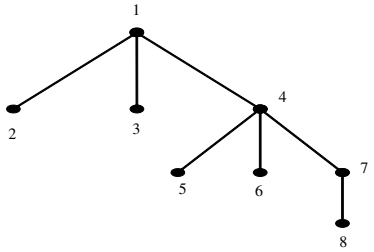

Figura 3.12: Árbol asociado a la estructura $[_1\ [_2\ ]_2\ [_3\ ]_3\ [_4\ [_5\ ]_5\ [_6\ ]_6\ [_7\ [_8\ ]_8\ ]_7\ ]_4\ ]_1$ de la figura 2.11.

Los objetos son representados con las letras de un alfabeto Γ cuyos elementos se denominan símbolos con los que se pueden formar cadenas finitas y ordenadas a modo de *palabras*. El número de elementos que tiene una palabra se denomina *longitud* de manera que la palabra a^n indicará que es una palabra de longitud n

formada por n repeticiones del símbolo a. La palabra de longitud 0 se denomina *palabra vacía* y se denota por λ. Al conjunto de todas las palabras que se pueden formar sobre un alfabeto, Γ, se denota por Γ^*.

Una regla de evolución es un par ordenado (u, v) representado mediante la forma $u \rightarrow v$ siendo $u \in \Gamma^*$ y v un multiconjunto sobre el alfabeto (Γ x $\{here, out\}) \cup (\Gamma$ x $\{in_j : 1 \leq j \leq n\})$ siendo n el grado del P sistema. La longitud de u se denomina *radio* de la regla (u, v). De esta manera, la regla $ab \rightarrow ac$ $(ab \rightarrow (ac, here))$ es de radio 2 e indica que una copia del objeto a y del b se convierten en una copia del objeto a y una copia del objeto c quedándose en la misma membrana, la regla $ab \rightarrow (ac, out)$ indicará que la nueva copia saldrá de la membrana pasando a la membrana inmediatamente superior y la regla $ab \rightarrow (ac, in_i)$ que indicará que la nueva copia pasará a la membrana i siempre y cuando ésta sea inmediatamente interior a la membrana donde se aplica dicha regla. Si aparece el símbolo δ, $ab \rightarrow (ac, \delta)$ la membrana desaparece (se disuelve) y la nueva copia se añade a la región de la membrana inmediatamente superior. Si en el P sistema de transición hay reglas de radio mayor o igual a 2, el sistema se dice *cooperativo*; en caso contrario, *no cooperativo*. La ejecución de las reglas puede considerarse en un cierto orden a través de una *relación de prioridad*. En caso que existan varias reglas en una misma membrana que puedan ejecutarse en un mismo instante, el sistema actuará de manera *no determinista*. Además, en cada paso, las reglas se aplicarán hasta que no quede ningún objeto, es decir, de forma *maximal*.

Los P sistemas se caracterizan por el *paralelismo* que implementan. Dentro de cada membrana las reglas se aplicarán de forma simultánea y todas las membranas trabajarán a la vez y de forma sincronizada. Las evoluciones que se producen en la realización de un P sistema reciben el nombre de *configuraciones*. Diremos que un P sistema es *de parada* si llegamos a una configuración, *la configuración de parada*, en la cual ya no es posible la realización de ninguna de las reglas que se tienen definidas en todas las membranas. La membrana que contiene la solución del P sistema recibe el nombre de *membrana de salida*.

3.3.4. Formalización de los P sistemas de transición con membrana de salida

Un P sistema de transición de grado n, $n \geq 1$, es una construcción $\Pi = (\Gamma, \mu, \mathcal{M}_1, \cdots \mathcal{M}_m, (R, \rho))$ donde:

- Γ es un alfabeto. Sus elementos se llaman objetos.
- $\mathcal{M}$ es una estructura de membranas formada por m membranas, con las membranas etiquetadas inyectivamente por los elementos de un conjunto dado $H = \{1, \ldots, m\}$, donde m es el grado de Γ.
- $\mathcal{M}_i, 1 \leq i \leq m$ son cadenas que representan multiconjuntos sobre Γ asociados a las regiones $1, \ldots, m$ de μ .
- R son conjuntos finitos de reglas de evolución sobre Γ y ρ es una relación de orden parcial sobre R, llamada relación de prioridad.

  Una regla de evolución es un par (u, v), normalmente escrito de la forma $u \rightarrow v$, donde u es una cadena sobre Γ y v es una cadena sobre $\{a_{here}, a_{out}, a_{in_j} | a \in \Gamma, 1 \leq j \leq m\}$. La longitud de u se llama radio de la regla $u \rightarrow v$.

Dadas dos configuraciones $C_1 = (\mu^{'}, \mathcal{M}_1^{'}, \ldots, \mathcal{M}_m^{'})$ y $C_2 = (\mu^{''}, \mathcal{M}_1^{''}, \ldots, \mathcal{M}_m^{''})$ de Γ, $C_1 \Rightarrow C_2$ nos indica que tenemos una transición de la forma C_1 a C_2 , si podemos pasar de C_1 a C_2 mediante las reglas de evolución de R.

Una secuencia de transiciones entre configuraciones de un P sistema Π recibe el nombre de computación. Se dice que una computación es exitosa si y solo si es de parada, es decir, no hay ninguna regla aplicable a los objetos que contiene la última configuración. La salida de una computación exitosa es $\Psi_\Gamma(\mathcal{M})$, donde $\mathcal{M}$ describe el multiconjunto de objetos de Γ enviados fuera del sistema durante la computación. El conjunto de vectores $\Psi_\Gamma(\mathcal{M})$ se denotan por $P_S(\Pi)$ y está generado por Π.

Capítulo 4

Markov chains: Computing limit existence and approximations with DNA

Available online at www.sciencedirect.com
SCIENCE DIRECT®
BioSystems 81 (2005) 261–266
BioSystems
www.elsevier.com/locate/biosystems

Markov chains: Computing limit existence and approximations with DNA

M. Cardona[a], M.A. Colomer[a,*], J. Conde[a], J.M. Miret[a], J. Miró[b], A. Zaragoza[a]

[a] *Universitat de Lleida, 25001 Lleida, Spain*
[b] *Universitat de les Illes Balears, 07122 Palma de Mallorca, Spain*

Received 17 November 2004; received in revised form 10 May 2005; accepted 10 May 2005

Abstract

We present two algorithms to perform computations over Markov chains. The first one determines whether the sequence of powers of the transition matrix of a Markov chain converges or not to a limit matrix. If it does converge, the second algorithm enables us to estimate this limit. The combination of these algorithms allows the computation of a limit using DNA computing. In this sense, we have encoded the states and the transition probabilities using strands of DNA for generating paths of the Markov chain.

Keywords: DNA computing; Markov chains

1. Introduction

In the last decade, biochemical techniques began to be used to solve some computationally intractable problems. The first work in this field was done by Adleman (1994), concerning the Hamiltonian Path Problem. His experiment showed the potential benefits of using DNA molecules as a computing device. In this paper, we use DNA computation techniques as an alternative method to simulate paths of a Markov chain. More precisely, we present an algorithm determining whether or not the powers of the transition matrix converge to a limit matrix, and another algorithm to obtain the powers of this transition matrix.

The first algorithm is quite straightforward. A Markov chain can be represented by a graph where the vertexes are the states of the chain, and the edges the transitions between states. We represent the vertexes and edges by single DNA strands in the usual manner and obtain all the paths through the graph. Then, by checking the ends of the strands we can calculate the period of each state. We determine the cyclic classes

* Corresponding author. +34 973 702526; fax: +34 973 238264.
E-mail addresses: mcardona@matematica.udl.es (M. Cardona), colomer@matematica.udl.es (M.A. Colomer), jconde@matematica.udl.es (J. Conde), miret@matematica.udl.es (J.M. Miret), joe.miro@uib.es (J. Miró), alba@matematica.udl.es (A. Zaragoza).

0303-2647/$ – see front matter © 2005 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.biosystems.2005.05.003

by checking for each pair of states with the same period d whether a path exists from the first to the second state. Finally, we determine from the cyclic classes the recurrent ones selecting a representative of each cyclic class and checking if there is a path from each state in the class to another one. If there is no class that is both recurrent and cyclic, the sequence of powers of the transition matrix is convergent. Otherwise, it is not.

For the second algorithm we propose a novel technique. We code the paths as before, but now we introduce for each edge a number of strands that is proportional to the probability of the transition taking place. We again construct all the paths through the graph, but now the amount of strands in the solution that represent a given path is proportional to the probability of existence of that particular state sequence in the chain. From the test tube V_j containing the strands that begin with a given state s_j and represent paths of length n of the Markov chain, we determine the proportions of those that end by $s_1, \ldots, s_k$. To do this, we add strands with fluorescence $s_1^*, \ldots, s_k^*$ that encode the final states of the paths. By means of the fluorescence of the ending strands, we can determine the proportions among the paths of the Markov chain that begin by s_j and end by each of the possible states. Consequently, the coefficients of the powers of the transition matrix can be obtained.

We are aware that this technique must be validated through experimentation, which will be done in the future. We have written and executed simple computer simulations that support so far the correctness of these algorithms.

2. Overview of biomolecular computation

DNA is the polymeric molecule living in the nucleus cells that carries the genetic information of cellular organisms. It is comprised of subunits called nucleotides that are joined into *polymer chains*, commonly referred to as DNA *strands*. There are four sorts of nucleotides in DNA that depend on the base attached to it. These bases are *adenine*, *cytosine*, *guanine* and *thymine* denoted by `A`, `C`, `G` and `T`.

Nucleotides link among them, via reaction between the 5′ phosphate of one nucleotide and the 3′ hydroxyl of another, forming single strands of DNA. Thus, any single strand has a natural orientation. Moreover, the base of one nucleotide can link with the base of another forming hydrogen bonds between `A`, `T` and `C`, `G` (which are called *complementary bases*) and so forming double strands with a helix structure discovered by Watson and Crick (1953). Notice that we can mathematically represent a simple strand of DNA by means of a string x over the alphabet $\{$`A`, `C`, `G`, `T`$\}$. Furthemore, we will denote by $\overline{x}$ the Watson–Crick complementary string of x. Now, we recollect several biological operations, which will be applied to a set of strands in a test tube. For more details about these operations, we refer to Păun et al. (1998) or Pisanti (1998).

Length-extract: Given a test tube U containing a solution of DNA strands and an integer n, a new test tube `lextract`(U, n) is produced by means of gel electrophoresis consisting of all strands in U with length equal to n.

Ligate: Given two test tubes U_1 and U_2 of DNA strands, another test tube `ligate`(U_1, U_2) is obtained, containing strands we get linking strands of U_1 with strands of U_2 using enzymes called *ligases*.

String-extract: Given a test tube U of DNA strands and a string x, by means of DNA complementarity, a new test tube `sextract`(U, x) is produced, consisting of strands of U which contained the string x.

Copy: Given a test tube U of DNA strands and two strings x and y, by means of *amplification* a new test tube `copy`(U, x, y) is produced, consisting of copies of pieces of strands of U which start by x and end by y and so that they only contain once the string y.

3. Finite Markov chains

A finite Markov chain is given by a sequence of events of an experiment with $e_1, \ldots, e_k$ possible results or states, where the result of any event depends only on the result of the preceding event. Moreover, the finite Markov chain is said to be a homogenous chain if the dependence between consecutive events does not change.

A finite and homogenous Markov chain is characterized by the initial probabilities q_{0j} to get the state e_j in the first event and the transition probabilities p_{ij} to

get the state e_j in an event if in the previous state were e_i. So the probability to get the states $(e_{j_0}, \ldots, e_{j_n})$ is given by:

$$P(e_{j_0}, \ldots, e_{j_n}) = q_{0j_0} p_{j_0 j_1} \cdots p_{j_{n-1} j_n}.$$

Furthermore, the transition probabilities form a stocastic matrix:

$$\mathbf{P} = \begin{pmatrix} p_{11} & \cdots & p_{1k} \\ \vdots & \ddots & \vdots \\ p_{k1} & \cdots & p_{kk} \end{pmatrix} \quad \text{with } \sum_{j=1}^{k} p_{ij} = 1,$$

$\forall i \in \{1, \ldots, k\}$,

and the Kolmogorov–Chapman equation holds:

$$p_{ij}^{(1)} = p_{ij}, \qquad p_{ij}^{(2)} = \sum_{l=1}^{k} p_{il} p_{lj}, \ldots,$$

$$p_{ij}^{(n)} = \sum_{l=1}^{k} p_{il} p_{lj}^{(n-1)}.$$

Notice that if we denote the initial probabilities by means of the vector $\mathbf{q}_0 = (q_{01}, \ldots, q_{0k})$, the distribution in the nth event is given by $\mathbf{q}_n = \mathbf{q}_0 \mathbf{P}^n$. So, in order to determine the distribution $\mathbf{q}_n$ it is enough to study the matrix $\mathbf{P}^n$ and, besides, the limit of the sequence of these matrices allows us to obtain the distribution limit.

On the other hand, we will consider the digraph associated with the Markov chain whose vertexes are the states and the edges are given by the pairs (e_i, e_j) such that $p_{ij} > 0$. This will be useful to implement the algorithm to determine the existence of the matrix limit of the sequence $\{\mathbf{P}^n\}_{n \geq 1}$,

3.1. Classification of the states

A state e_i is an *accessible state* from the state e_j, $e_j \to e_i$ if $e_j = e_i$ or there is an integer $n > 0$ such that $p_{ji}^{(n)} > 0$. Likewise, two states e_i, e_j are *communicating states*, $e_i \leftrightarrow e_j$ if $e_j \to e_i$ and $e_i \to e_j$. In the set of states $\{e_1, \ldots, e_k\}$, the relation $\leftrightarrow$ is an equivalence relation. Moreover, the relation $\to$ induces an order relation in the quotient set of classes $\{e_1, \ldots, e_k\}/\leftrightarrow$. These relations can be rewritten in terms of the associated digraph. Indeed, since the definition $e_i \to e_j$ does not depend on the values of the transition probabilities but on their positivity, it is equivalent to the existence of a path from e_i to e_j on the associated digraph.

On the other hand, for all e_i, we denote by $f_{ii}^{(n)}$ the probability of leaving e_i to get the state e_i for the first time in n steps. Moreover, we denote by f_{ii} the probability of leaving from state e_i to return, sooner or later, to the state e_i, namely $f_{ii} = \sum_{n=1}^{\infty} f_{ii}^{(n)}$. With these notations, e_i is *recurrent* if $f_{ii} = 1$ and *transient* if $f_{ii} < 1$.

Notice that the states in nonmaximal classes by the order relation $\to$ are transient states and, if the set of states is finite (which is our case), the states in maximal classes (which always exist) are recurrent states. These classes will be called *recurrent* classes.

We introduce the notion of *period* of a class. If e_i is a state for which there exists an integer $n \geq 1$ with $p_{ii}^{(n)} > 0$, we can define the period of e_i as follows:

$$d_i = \text{g.c.d.}\{n \geq 1 | p_{ii}^{(n)} > 0\}.$$

All states that belong to the same class have the same period, which is therefore naturally called the period of the class. If the value of the period is 1, the class is said to be *aperiodic*; otherwise, we refer to it as a *periodic* or *cyclic* class.

In this context there is a well known result (Shiryayev, 1984) which provides us with information about the existence of the limit of the sequence of the matrix powers of a finite Markov chain. Concretely, given a finite Markov chain with transition matrix $\mathbf{P}$, the sequence $\{\mathbf{P}^n\}_{n \geq 1}$ has limit if there are no recurrent nor cyclic classes. Otherwise, the limit does not exist.

4. Simulating finite Markov chains using DNA

In this section, we present a method based on DNA computation to simulate paths of the digraph associated with a Markov chain in order to study some of its properties in a more efficient way. More precisely, we give an algorithm to determine the existence of the limit matrix and another to obtain the powers of the transition matrix.

In order to implement these algorithms, we need the corresponding solutions formed by DNA strands that encode the vertexes and the edges of the digraph of a Markov chain. We start by presenting a way to code them.

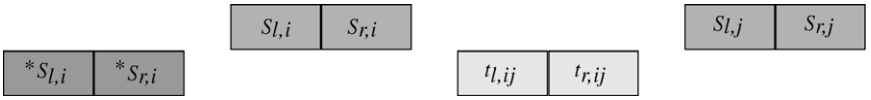

Fig. 1. Representation of the states and the transition probabilities.

4.1. Representing states and transition probabilities of a Markov chain

We will encode the elements we need to carry out our work by means of simple strands (x or $\overline{x}$), where x is a string over the alphabet $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. More precisely, we describe below their encodings and we represent them in Fig. 1.

- *Strands* s_i: Encode the states of the Markov chain. They are formed by single strands consisting of two distinguished parts $s_{l,i}s_{r,i}$.
- *Strands* $^*s_i$: Encode the initial states of the paths. They are formed by single strands consisting of two distinguished parts $^*s_{l,i}{}^*s_{r,i}$, being $^*s_{r,i} = \overline{s}_{l,i}$.
- *Strands* t_{ij}: Encode the positive transition probabilities of the Markov chain. They are formed by single strands consisting of two distinguished parts $t_{l,ij}t_{r,ij}$, being $t_{l,ij} = \overline{s}_{r,i}$ and $t_{r,ij} = \overline{s}_{l,j}$.

In order to carry out the experiment, the strands $^*s_i$, s_i and t_{ij} will be taken with the same number of nucleotids, which will contain at least 20-mers in each one of the two distinguished parts (Feldkamp et al., 2001).

4.2. Algorithm to determine the existence of the limit matrix

The encoding presented in Section 4.1 allows us to use biological operations with DNA strands to perform certain operations on strings that represent the states and transition probabilities of a Markov chain.

In this subsection, we present, given a finite Markov chain of k states e_i and transition probabilities p_{ij}, an algorithm to determine the existence of the limit matrix of the sequence of powers of the transition matrix. This algorithm can be described as follows:

Step 1. First we introduce in a test tube U the strands $^*s_i$ that represent the initial states e_i of paths, the strands s_i and t_{ij} that encode, respectively, the elements e_i and the positive transition probabilities p_{ij}, which react to each other (using enzymes ligases) becoming double DNA strands, which represent the paths of the digraph associated with the Markov chain (see Fig. 2).

Step 2. Once the paths of the digraph of a length large enough have been generated, we calculate the period of each state as follows:

Input: U
  $U_1 \longleftarrow U, \ldots, U_k \longleftarrow U$
  for $i = 1$ **to** k **do**
    $U_{i,i} \longleftarrow \mathtt{copy}(U_i, {}^*s_i, s_i)$
  end for
Output: $U_{1,1}, \ldots, U_{k,k}$.

Each test tube $U_{i,i}$ contains all paths of length $\leq km + \frac{m}{2}$ (m being the length of strands that encode the states) which start by $^*s_i$ and end by state s_i. Next, using gel eletrophoresis we read, for each tube, the length of their strands. So the period of each state e_i can be obtained taking the greatest common divisor d_i of the lengths of strands of the tube $U_{i,i}$.

Step 3. We determine the cyclic classes. For each ordered pair of states with the same period d (greater than 1), we check whether there exists a path from the first state to the second one. More precisely, for each period $d > 1$ we consider the set of states $e_{i_1}, \ldots, e_{i_r}$. Notice

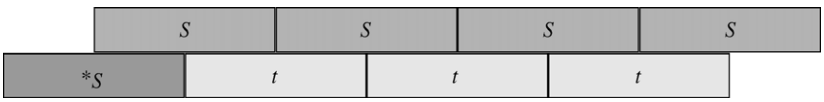

Fig. 2. Representation of the paths of the digraph.

that if $r = d$ there exists a unique cyclic class consisting of all these states. Otherwise, we proceed as follows:

Input: U
  for $j = i_1$ **to** i_r **do**
    $V_j \longleftarrow \texttt{sextract}(U, {}^*s_j)$
    $V_{j,i_1} \longleftarrow V_j, \ldots, V_{j,i_r} \longleftarrow V_j$
    for $h = i_1$ **and** $h \neq j$ **to** i_r **do**
      $V'_{j,h} \longleftarrow \texttt{copy}(V_{j,h}, {}^*s_j, s_h)$
    end for
  end for
Output: $V'_{j,h}, i_1 \leq j, h \leq i_r,\ j \neq h.$

Now the cyclic classes of the period d can be obtained by checking which of the sets $V'_{j,h}$ are empty. Indeed, if $V'_{j,h} \neq \emptyset$ and $V'_{h,j} \neq \emptyset$ then the states e_j and e_h belong to the same cyclic class.

Step 4. We determine from the cyclic classes the recurrent ones, that is to say, those for which there is no state in the class with accessibility to some other state out of the class. To do this, we select a representative of each cyclic class and we check whether or not a path exists from each state in the class to another one or to an aperiodic state.

More precisely, we consider states $e_{i_1}, \ldots, e_{i_s}$ which are representative of all diferent cyclic classes and states $e_{i_{s+1}}, \ldots, e_{i_t}$ which have period 1. Then, in order to know if the cyclic class with representative $e_\ell, i_1 \leq \ell \leq i_s$, is a recurrent cyclic class, we proceed as follows:

Input: V_ℓ
  for $j = i_1,\ j \neq \ell$ **to** i_t **do**
    $V_{\ell,j} \longleftarrow \texttt{copy}(V_\ell, {}^*s_\ell, s_j)$
  end for
Output: $V_{\ell,j}, i_1 \leq j \leq i_t,\ j \neq \ell.$

Then, if all the sets $V_{\ell,j}$ are empty, the cyclic class with representative e_ℓ is a recurrent class. Otherwise, this class will be a nonrecurrent class.

Therefore, given a Markov chain, if there is no recurrent and cyclic class, the sequence of powers of the transition matrix is convergent. Otherwise, the sequence is not convergent.

4.3. Algorithm to determine powers of a transition matrix

Now we give an algorithm to estimate the coefficients of a power of the transition matrix $\mathbf{P} = (p_{ij})$ of a Markov chain with k states.

Step 1. In order to compute a good approximation of the nth power of $\mathbf{P}$, first we generate strands representing paths of the digraph of the Markov chain, similarly to the previous algorithm. In this case we need to assure the amounts of the different generated paths have suitable proportions according to the transition matrix.

In order to achieve this, we put in a test tube U the strands ${}^*s_i$, the strands s_i and the strands t_{ij} with ratios p_{ij}. Furthermore, to maintain these ratios, the total amount of strands t_{ij} will be large enough with respect to the total amount of strands ${}^*s_i$ and s_i.

Step 2. From the previous solution we select the paths which begin with the states ${}^*s_1, \ldots, {}^*s_k$ and we put them in different test tubes. Next, we extract from these solutions those paths with length $nm + \frac{m}{2}$ (m being the length of the strands that encode the states).

Input: U
  for $j = 1$ **to** k **do**
    $U_j \longleftarrow \texttt{sextract}(U, {}^*s_j)$
    $V_j \longleftarrow \texttt{lextract}(U_j, nm + \frac{m}{2})$
  end for
Output: $V_1, \ldots, V_k.$

Step 3. From the test tube V_j containing the strands which begin with a given state ${}^*s_j$ and which represent paths of length n of the digraph of the Markov chain, we determine the proportions of those that end by $s_1, \ldots, s_k$. To do this, we add strands with fluorescence $s_1^*, \ldots, s_k^*$ that encode the final states of the paths.

Input: V_j
  for $i = 1$ **to** k **do**
    $V_{ji} \longleftarrow V_j$
    $U_{ji} \longleftarrow \texttt{ligate}(V_{ji}, s_i^*)$
  end for
Output: $U_{j1}, \ldots, U_{jk}.$

Table 1
Complexity cost

Algorithm	Biosteps	Strands
`Determining_existence_limit_matrix`	$\mathcal{O}(k^2)$	k^k
`Determining_powers_transition_matrix`	$\mathcal{O}(k^2)$	k^n

By means of the fluorescence of the ending strands in the test tubes $U_{j1}, \ldots, U_{jk}$, we can determine the proportions among the paths of the digraph of the Markov chain which begin by e_j and end by each of the possible states. Consequently, the coefficients of the powers of the transition matrix can be obtained.

4.4. Biocomplexity

We end this section giving a brief analysis of the complexity of these algorithms within a commonly employed model of DNA computation. According to Boneh et al. (1996) and Pisanti (1998) we quantify the computational complexity by counting the number of required biological operations (biosteps) and the quantity of different strands that may appear in the same test tube during the computation. The cost of the algorithms described in the previous section is given in Table 1 in terms of k and n, k being the number of states and n the power of the transition matrix which will be computed.

Finally, notice that chosen a codification for the k states of the Markov chain, any other Markov chain with the same number k of states can be described using the same codification. Indeed, we can take advantage of the fact that for each transition probability $p_{ij} > 0$, its encoding t_{ij} is formed by means of the last half basis of state s_i together the first half bases of state s_j.

5. Conclusions

We have produced two new DNA computing algorithms. The main novelty of the first one is that it computes the *existence of a limit* of the sequence of powers of the transition matrix of a Markov chain. This algorithm uses DNA operations that are well understood and have all been profusely discussed in the literature. The second algorithm computes an approximation of the limit matrix if it exists. We use in this algorithm a novel operation, consisting of introducing for each edge a number of strands that is proportional to the probability that the corresponding transition will take place. After the computation the amount of strands in the solution that represent a given path is proportional to the probability of existence of that particular state sequence in the chain, and can be measured through fluorescence.

It is, to our knowledge, the first time a limit is computed using DNA techniques and so our work extends the scope of problems DNA computing is suited for. Our work is still theoretical, although we have tested the algorithms through program simulation. Much work needs to be done to finally validate the algorithms and techniques shown here. In the future, we will start actual experiments toward that goal.

Acknowledgement

Partially supported by Spanish DGES Grants BFM2000-1113-C02 and BFM-2003-0771.

References

Adleman, L., 1994. Molecular computation of solutions to combinatorial problems. Science 266, 1021–1024.

Boneh, C., Dunworth, D., Lipton, R.J., 1996. On the computational power of DNA. Discrete Appl. Math. 71 (1–3), 74–79.

Feldkamp, W., Banzhaf, U., Raube, H., 2001. A DNA sequence compiler. In: DNA6, Sixth International Meeting on DNA Based Computers. Lecture Notes in Computer Science, vol. 2054. Springer-Verlag.

Pisanti, N., 1998. A survey on DNA computating. Bulletin of the EATCS 64, pp. 171–187.

Păun, G., Rozenberg, G., Saloma, A., 1998. DNA Computation. EATCS Series. Springer-Verlag.

Shiryayev, A. N., 1984. Probability. GTM, vol. 95. Springer.

Watson, J., Crick, F., 1953. Molecular structure of nucleic acids: a structure for deoxrribose nuclei acid. Nature 171, 737–738.

Capítulo 5

A step towards a DNA Computation model

A STEP TOWARDS A DNA COMPUTATION MODEL

Alba Zaragoza, M. Àngels Colomer, Mònica Cardona
Dept. de Matemàtiques. Universitat de Lleida. 25001 Lleida. SPAIN
alba@matematica.udo.es, colomer@matematica.udl.es, cardona@matematica.udl.es

Jose Miro-Julia
Dept. de Matemàtiques i Informàtica. Universitat de les Illes Balears. 07122 Palma de Mallorca. SPAIN
joe.miro@uib.es

New computational forms require new thinking processes. In this paper we present a result needed to detect efficiently the aperiodic classes of a finite Markov chain. The importance of this result is that it is specifically to produce an algorithm suited for DNA computing. These types of results will be needed if this and other methods of Natural Computation are to take off.

Keywords: DNA computing; Markov chains; Modelling

1. The need of a new model

New computational forms require new thinking processes. This is the main difficulty. Compared to this it is relatively easy to master the new technology, extend its limits, make it faster and more reliable. But without the new thought processes the new computational form demands, we will only use a small fraction of its potential. Technology is easy, algorithms are hard.

One of the reasons of the spectacular success of electronic computers is that they were built to take advantage of logic and reasoning forms that have been used for centuries. Our programs of today are the great-grandchildren of aristotelian reasoning, brought and refined by Ramn Llull, Pascal, Leibniz, Boole, Turing. Babbage and von Neummann specified the architecture needed to convert the mathematical reasoning and algorithms into computation.

Even within electronic computers we see a case where the lack of a powerful new paradigm has hindered the advance of computation. Parallelism has not lived up to its potential. A clear case is that of massive parallel computers such as the Connection Machine, that did not last long despite the initial hype. Parallelism has only succeeded where a new thinking model was not needed. This is the case of vector processors, where powerful vector compilers automatically created efficient parallel code from the so called *dusty decks:* old programs created with the old paradigm.

For new DNA computing to succeed, or any other type of new computing for that matter, we must develop a new way of thinking, a new algorithmic paradigm. This is a very long process as changing our minds is much harder than changing from silicon and electricity to DNA strands and enzymes, but without this change the new forms of computation has very few chances of surviving. That is what the brief but intense history of computers have taught us.

DNA computing has taken the first logical steps. It has used the known algorithms that are best suited to its characteristics to gauge its potential. This is what Adleman [1] did in his famous first experiment or Ogihara and Ray [5], using DNA to build boolean circuits. This has shown that DNA computation is feasible and given us a taste of its potential. But much more is needed.

Not all research in DNA computation has followed the known algorithmic path. The work of Erik Winfree [10] in creating self-assembly tiles as the basis of computation is worth mentioning. And the work we present here is another step away from the tread path. We prove and use a mathematical proposition that is useless in the electronic computing framework, but that allows us to create a very efficient algorithm to classify the states of a finite Markov chain using DNA computation.

We are aware that this is only a modest example, a first step. But it is important as it shows the type of mathematical results that will be useful to construct DNA computer algorithms.

2. Technical overview

Although what is important is the new thinking process, to understand it it is necessary to know the technical concepts both of DNA Computation and finite Markov chains. A brief overview, that can be skipped by those familiar to this material, is presented here.

2.1. *DNA computation*

DNA is the polymeric molecule living in the cell's nucleus that carries the genetic information. It is formed by subunits called nucleotides that are joined into *polymer chains,* commonly referred to as DNA *strands.* There are four sorts of nucleotides in DNA that depend on the base attached to it. These bases are *adenine, cytosine, guanine* and *thymine* denoted by `A`, `C`, `G` and `T`.

Nucleotides link among them, via reaction between the 5'phosphate of one nucleotide and the 3'hydroxyl of another, forming single strands of DNA. Thus any single strand has a natural orientation. Moreover, the base of one nucleotide can link with the base of another forming hydrogen bonds between `A`, `T` and `C`, `G` (which are called *complementary bases*) and so forming double strands. We can represent a simple strand of DNA by means of a string x over the alphabet $\{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$. Furthermore, we will denote by $\overline{x}$ the Watson-Crick complementary string of x.

Through biochemical processes we can modify the strands, thus performing a computation. The power of DNA computations is that these biochemical processes

perform operations over the billions of strands simultaneously. The parallelism achieved is many orders of magnitude greater than what the most powerful silicon computer can do. There are many biochemical operations that can be applied [6,8]. The ones we will use are:

Length–extract. Given a test tube U containing a solution of DNA strands and an integer n, we obtain a new test tube $\texttt{lextract}(U, n)$ consisting of all strands in U with length equal to n.

Ligate. Given two test tubes U_1 and U_2 of simple DNA strands, another test tube $\texttt{ligate}(U_1, U_2)$ is obtained, containing the double strands we get by linking strands of U_1 with complementary strands of U_2.

Copy. Given a test tube U of DNA strands and two strings x and y, by means of *amplification* a new test tube $\texttt{copy}(U, x, y)$ is produced, consisting of copies of pieces of strands of U which start by x and end by y and so that they only contain once the string y.

2.2. *Finite Markov chains*

A finite Markov chain is a random process $X_0, X_1 \ldots, X_t, \ldots$ which is produced by a change of results or states $e_1, \ldots, e_k$ at discreet times $t = 1, 2, 3 \ldots$ where the result of each event only depends on the result of the previous event, that is to say,

$$P(X_{t+1} = e_j / X_t = e_i, X_{t-1} = e_{i_{t-1}}, \ldots, X_0 = e_{i_0}) = P(X_{t+1} = e_j / X_t = e_i)$$

The value

$$p_{ij}(t) = P(X_t = e_j / X_{t-1} = e_i)$$

is the probability that the chain reaches the state e_j at time t if at time, $t-1$ it was in the state e_i. We call this the *transition probability.* The transition probability matrix

$$P(t) = (p_{ij}(t))$$

is non negative for all t and their rows add up to 1, so it is a stochastic matrix.

In the case that the transition probabilities do not depend on time, $p_{ij}(t) = p_{ij}$ for all t, the Markov chain is called *stationary* or *homogeneous*, and the transition matrix $P = (p_{ij})$ is constant, and verifies the Kolmogorov-Chapman equation

$$p_{ij}^{(1)} = p_{ij}, \quad p_{ij}^{(2)} = \sum_{l=1}^{k} p_{il} p_{lj}, \quad \ldots, \quad p_{ij}^{(n)} = \sum_{l=1}^{k} p_{il} p_{lj}^{(n-1)}.$$

where $p_{ij}^{(n)}$ is the transition probability of leaving e_i to reach the states e_j in n steps.

A finite and homogeneous Markov chain is characterized by the initial probabilities q_{0j} that e_j is the first event and the transition probabilities p_{ij}:

$$P(e_{j_0}, \ldots, e_{j_n}) = q_{0j_0} p_{j_0 j_1} \cdots p_{j_{n-1} j_n}.$$

A state e_j is an *accessible state* from the state e_i, denoted by $e_i \to e_j$, if there is an integer $n > 0$ such that $p_{ij}^{(n)} > 0$. Two states e_i, e_j are *communicating states,* denoted by $e_i \leftrightarrow e_j$, if e_i is accessible from e_j and e_j is accessible from e_i. The relation $\to$ induces an order relation in the quotient set of classes $\{e_1, \ldots, e_k\}/\leftrightarrow$.

State e_i is called a *recurrent state* if for all e_j such that $e_i \to e_j$, then $e_j \to e_i$. On the other hand, the state e_i is a *transition state.*

Given a state e_i such that $p_{ii}^{(n)} > 0$ for any integer $n > 0$, we define their *period* as:

$$d(i) = \text{m.c.d.}\{n \geq 1 \mid p_{ii}^{(n)} > 0\}.$$

All states that belong to the same class have the same period. If the period is 1, the class is said to be *aperiodic*, otherwise it is said to be *periodic*. For more details see [7] or [9].

3. Classification of finite Markov chain states

The algorithm presented here is the second one developed by the authors concerning finite Markov chains. The first one [3] computed the existence and limit of the powers of the transition matrix. To do so we started from known mathematics and algorithms and transformed them into a form that was suitable to the powers of DNA computation. The main point of this work was to show that certain types of computation, the existence of a limit, could be done using DNA computers.

In this paper we have proven a proposition that has no computational value in the 'classic computer algorithmic' sense, but that yields a nice DNA computer algorithm. We noted that DNA computation is very good at detecting existence, so we have established mathematical properties based on the existence instead of the numerical quantification to classify the states of a finite Markov chain.

In this work we create a classification algorithm of the states of a finite and homogenous Markov chain in which the step corresponding to the study of the periodicity of the recurrent classes uses the following proposition instead of the periodicity definition.

Proposition 1. A recurrent class α is an aperiodic class if and only if there is an instant $m_0 \geq 1$ and a state e_{k_0} belongs to the class such that for all $e_j \in \alpha$, $p_{jk_o}^{(m_0)} > 0$.

Proof.

$\Rightarrow$ Let α be an aperiodic recurrent class with k states and let $e_{k_0} \in \alpha$ be such that the set $\{n_0 \in N \ ; \ p_{k_0k_0}(n_0) > 0\}$ is equal to $\{a_1, \ldots, a_n, \sum_{i=1}^{n} a_i r_i\}$ where $a_i, r_i \in N$ with $n \leq k$ and m.c.d.$\{a_1, \ldots, a_n\} = 1$ because of α is aperiodic and then, the period of their states is 1. Then exist a positive combination $m_0^{'}$ of $a_1 \cdots a_n$ such that the set of all the positive combinations

of $a_1 \cdots a_n$ contains all the integers $m_0 > m_0^{'}$. (For a proof, see [4], 11 of Chapter XIII).

If we now define,

$$n_i = min\{n; p_{ik_0}^{(n)} > 0\}$$
$$n = max\{n_i\}$$
$$m_0 = m_0^{'} + n,$$

then, $\forall E_i \in E$, $p_{ik_0}^{(m_0)} \geq p_{ik_0}^{(n_i)} \cdot p_{k_0k_0}^{(m_0-n_i)} > 0$ where $m_0 - n_i = m_0^{'} + n - n_i$. $\Leftarrow$ We suppose that $m_0 \geq 1$ exits and so does a state $e_{k_o} \in \alpha$ such that $\forall$ $e_j \in \alpha$, $p_{jk_0}^{(m_0)} > 0$. In particular $p_{k_0k_0}^{(m_0)} > 0 \Rightarrow m_0 = \widehat{\dot{d(k_0)}}$.

Given $e_i \in \alpha$ such that $p_{k_0i}^{(1)} > 0 \Rightarrow 0 < p_{ik_0}^{(m_0)} p_{k_0i}^{(1)} \leq p_{ii}^{(m_0+1)} \Rightarrow m_0 + 1 = \widehat{\dot{d(i)}}$.

Because e_i and e_{k_0} belong to the same class, $d(i) = d(k_0) = 1$ and as a consequence α is an aperiodic class. □

The implementation of this proposition by means of classic computational methods is unthinkable as its cost would be $O(k_\alpha^{m_0})$ whereas, as we shall soon see, the implementation with DNA is simple and has constant cost.

4. The classification algorithm

In this section we present the DNA algorithm based that will classify the k states e_i of a finite and homogeneous Markov chain. There are four classes of strands we will use to encode the data. As mentioned in Section 2.1 we will encode these elements as simple DNA strands (x or $\overline{x}$), where x is a string over the alphabet {A, C, G, T}. The four classes of strands are

- Strands s_i: they encode the states of the Markov chain. They are formed by single strands consisting of two distinguished parts $s_{l,i}s_{r,i}$.
- Strands ${}^*s_i$: they encode the initial states of the paths. They are formed by single strands consisting of two distinguished parts ${}^*s_{l,i}{}^*s_{r,i}$, being ${}^*s_{r,i} = \overline{s}_{l,i}$.
- Strands t_{ij}: they encode the positive transition probabilities of the Markov chain. They are formed by single strands consisting of two distinguished parts $t_{l,ij}t_{r,ij}$, being $t_{l,ij} = \overline{s}_{r,i}$ and $t_{r,ij} = \overline{s}_{l,j}$.
- Strands s_i^*: they encode the final states of the paths. They are formed by single strands consisting of two distinguished parts $s_{l,i}^*s_{r,i}^*$, being $s_{r,i}^* = \overline{s}_{l,i}$.

The first step is to ligate these strands to compute the paths of the Markov chain. This step, step 1, is straightforward and presented in [3].

Step 1 We put in a test tube U the strands ${}^*s_i$ that represent the initial states e_i of paths, the strands s_i and t_{ij} that encode, respectively, the elements e_i and

the positive transition probabilities p_{ij}, which react with each other (using enzymes ligases) becoming double DNA strands, which represent the paths of the Markov chain.

The second step is to select the double strands of DNA with a long enough length and we read only the final states of these strands. The states that we are reading are all the recurrent states of the Markov chain.

Step 2

Input: U
$\Omega \leftarrow \texttt{lextract}(U, mn)$
for $i = 1$ **to** k **do**
$U_i \leftarrow \texttt{sextract}(\Omega, \bar{s}_i)$
end for
Output: $U_1, \ldots, U_k$

If the test tube $U_i \neq \emptyset$ then state e_i is a recurrent state, otherwise it is a transition state.

In the third step, for each recurrent state we build their class, such that two states belong to the same class only if they are communicating states.

Step 3

Input: U, recurrent states
$C = \emptyset$
$Class = \emptyset$
$i = 1$
if s_i recurrent and $s_i \notin Class$ **do**
$C(i) = \emptyset$
$j = 1$
if s_j recurrent
$U_{ij} \leftarrow \texttt{copy}(U, s_i, s_j)$
if $U_{ij} \neq \emptyset$ **do**
$C(i) = C(i) \cup \{s_j\}$
$Class = Class \cup \{s_j\}$
end if
$j = j + 1$
end if
$C = C \cup C(i)$
$i = i + 1$
end if
Output: C

At this stage set C is composed of the different recurrent classes of the Markov chain.

Step 4 Finally, it is necessary to know if the recurrent classes are periodic or aperiodic. Here is where we use Proposition 1. With it, all that is necessary is to look for a state e_i and an integer m_0 such that, for all states $e_j \in C(i)$ there is almost a strands of DNA with length equal to mm_0 that starts with e_j and ends with e_i. In order to do that we amplify, by PCR, all the strands which begin with e_j and end with e_i for all $e_j \in C(i)$. After that, we verify by means of electrophoresis if there is for all $e_j \in C(i)$ a string with length equal to mm_0. In this case, the recurrent class $C(i)$ is aperiodic. In the other case, we repeat the same process changing the final state for another state of the class. If after this process we do not find any strands with the same length, the class is periodic. Note that only two biochemical operations are needed: a PCR amplification and a length classification by gel electrophoresis.

We use the method proposed by Boneh [2] and Pisanti [8] to quantify the computational complexity by the number of biological operations and the amount of strands that use this algorithm. It is easy to see that this algorithm needs $\mathcal{O}(k^2)$ biosteps and k^k strands.

5. Biological feasibility

To verify the feasibility of the algorithm we have conducted part of the biological computations. The experience has consisted of computing a trajectory formed by two states, e_1 and e_2 where the only possible way is the one that goes from e_1 to e_2. To perform the computation we have needed:

- Strands s_1, s_2: encodes the states of the path.
- Strand ${}^*s_1$: encodes the initial state of the path.
- Strand t_{12}: encodes the positive transition probability.
- Strand s_2^*: encodes the final state of the path.
- Strands $s_{1primer}, s_{2primer}$: that are complementary to s_1 and s_2 respectively and are needed to identify if two states are accessible states. These primers allow us to copy the fragments of strings that start at e_1 and end at e_2.

The codes of the stands are shown in Table 1.
The lab protocol has been the following:

- Oligonucleotides were 5' phosphorylated by T4 polynucleotide kinase (T4 PNK, Invitrogen).
- Each oligo was incubated at a concentration of 15,4 pmol/micL in T4 PNK buffer containing 10 mM ATP and 1 unit/micL of T4 PNK enzyme.
- Reaction was maintained during 30 min at 37°C and the stopped by incubating for 5 minutes at 70°C.

s_1	TTTTGCTGGTTGCGACAGACGGTGTGTAAATGAGCCAGAG
s_2	TCCACTTTGACCCCATCCATGTTCTGCCATACTCGGGATA
${}^*s_1$	GTCTGTCGCAACCAGCAAAACTATGTTGGTAATGGTCGGC
t_{12}	ATGGATGGGGTCAAAGTGGACTCTGGCTCATTTACACACC
s_2^*	TTGCCCAGGTTTCACATCAGTATCCCGAGTATGGCAGAAC
$s_{1primer}$	AAAACGACCAACGCTGTCTGCCACACATTTACTCGGTCTC
$s_{2primer}$	AGGTGAAACTGGGGTAGGTACAAGACGGTATGAGCCCTAT

Table 1. Codifications

- Annealing was performed mixing 50 and 150 pmol of each 5' phosphorylated oligo in 1X annealing buffer (10 mM Tris pH 8.0, 50 mM NaCl and 1 mM EDTA).
- Mixed oligos were placed in a heat block at 95°C for 5 min and then the heat block was removed and allows to cool at room temperature.
- Four micL of annealed oligo solution were mixed in 1X ligation buffer (6,6 mM Tris-HCl pH 7,5; 5 mM MgCL2; 1 mM dithioerytritol and 1 mM ATP) and 1 unit of T4 DNA ligase (Roche) in 10 micL final volume.
- Ligation reaction was extended for 16h at 4°C. Ligated products were detected by acrylamide gel electrophoresis.

The final result obtained is shown in Figure 1.

6. Conclusions

We have proven a new result to classify states of a finite Markov chains. This result directly yields an efficient DNA computing algorithm. We have also shown the feasibility of the algorithm in the lab. To take full advantage of the potential of DNA computing, as well as other natural computing methods, it is necessary to develop these new results and new ways of thinking and not just adjust known algorithms and procedures to make them suitable for DNA computation. This is only a first step and much more research is needed.

References

1. L. Adleman. Molecular computation of solutions to combinatorial problems. *Science* 266, 1021–1024, 1994.
2. D. Boneh, C. Dunworth, and R.J. Lipton. On the computational power of DNA. *Discrete Appl. Math.* 71 (1-3), 79–94, 1996. *Science* 266, 1021–1024, 1994.
3. M.Cardona, M.A. Colomer, J. Conde, J.M. Miret, J. Mir and A. Zaragoza *Markov chains: computing limit existence and approximations with DNA*, *Biosystems*, 81, 3 (2005) 261–266.
4. W.Feller. *Introduccin a la teora de las probabilidades y sus aplicaciones*, Limusa, Mexico, 1998.

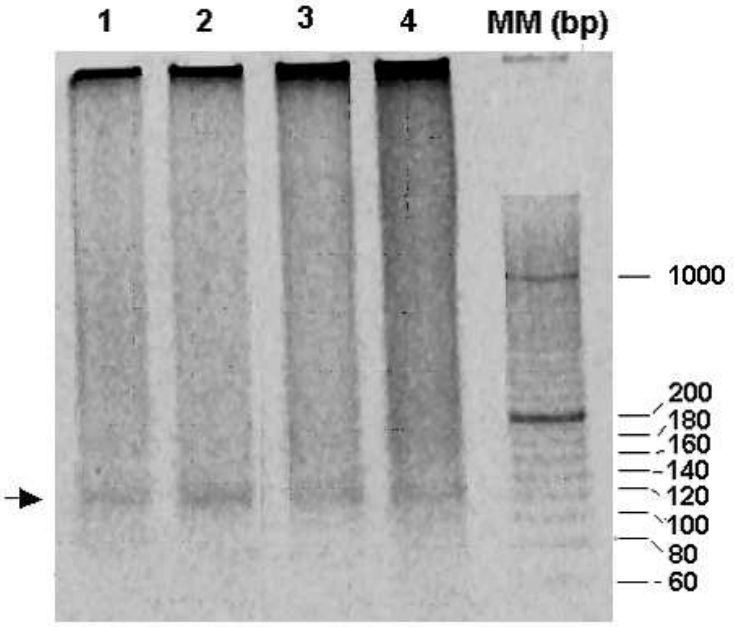

Fig. 1. Resolution of ligated producs by acrylamide gel electrophoresis. A 12% acrylamide gel was used to resolve fragments obtained after oligonucleotide annealing and ligation. 50 (1 and 2) and 150 (3 and 4) pmol of 5' phosphorylated oligos were mixed and heated at 95°C and cooled in the heat block (1 and 3) or standing at room temperature (2 and 4). Annealed products were then ligated as described. DNA fragments were detected by ethidium bromide staining. The arrow indicates the expected 120 bp DNA fragment. MM, molecular weight marker (20 pb ladder).

5. M. Ogihara and A. Ray, Executing parallel logical operations with DNA. *Proceedings of the Congress on Evolutionary Computation*, pages 972 – 979, IEEE Computer Society Press, Piscataway, NJ, 1999.
6. G. Păun, G. Rozenberg, and A. Saloma. *DNA Computation*. EATCS Series Springer Verlang, 1998.
7. J.R. Norris. Markov Chains. Cambridge university press.
8. N. Pisanti. A survey on DNA computing. *Bulletin of the EATCS* 64, 171–187, 1998.
9. A. N. Shiryayev. *Probability*. GTM 95, Springer, 1984.
10. Erik Winfree. *Design and self-assembly of two-dimensional DNA crystals*. Ph.D. Thesis. California Institute of Technology, June 1998

Capítulo 6

Handling Markov Chains with Membrane Computing

Handling Markov Chains with Membrane Computing

Mónica Cardona[1], M. Angels Colomer[1],
Mario J. Pérez-Jiménez[2], and Alba Zaragoza[1]

[1] Department of Mathematics, University of Lleida
Av. Alcalde Rovira Roure, 191. 25198 LLeida, Spain
{colomer, alba, mcardona}@matematica.udl.es
[2] Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
marper@us.es

Abstract. In this paper we approach the problem of computing the n–th power of the transition matrix of an arbitrary Markov chain through membrane computing. The proposed solution is described in a semi–uniform way in the framework of P systems with external output. The amount of resources required in the construction is polynomial in the number of states of the Markov chain and in the power. The time of execution is linear in the power and is independent of the number of states involved in the Markov chain.

1 Introduction

In the field of the Natural Computing, two areas that have attracted a great interest are the *molecular computing based on DNA* and, more recently, the *cellular computing with membranes*. One of the advantages of these models with respect to the classic ones is the massive parallelism that in these models is implemented in a natural way and allowing the simultaneous execution of many operations in an unit of the time.

The molecular computing provides a model of computation oriented to program and so, the computing devices proposed follow a structure similar to the classic algorithms, in which the operations realized in each step depend on the result obtained in the previous step. However, the cellular computing with membranes provides a model of computation oriented to machines. In this model, the computing devices, similar to a Turing machine, start from an initial configuration (a structure of membranes with certain chemical compounds in its compartments) which evolves by means of rules of the system (abstraction of the different chemical reactions which are allowed in membranes). The rules are applied according to a specific semantic, that is to say, the execution of such devices modifies the content of their components until arriving in a halting state in which the machine does not evolve any longer.

C.S. Calude et al. (Eds.): UC 2006, LNCS 4135, pp. 72–85, 2006.

The calculation of the natural powers of the transition matrix of a finite and homogenous Markov chain is important, because it allows us to estimate its limit in the case that it is convergent and so, we can know the stationary distribution of the process. This subject have been treated in [1], where two algorithms based on DNA are described that only allow us to obtain an *estimation* of the powers. These algorithms run in polynomial time and require a polynomial amount of resources.

In this work this problem is approached within the framework of cellular computing with membranes, and an exact solution is provided in time which is linear in the order of the power and is independent of the number of states of the Markov chain. The amount of used resources is polynomial in the power and the number of states.

The paper is structured as follows. In the next section, basic concepts concerning Markov chains and P systems that are necessary for the development of the work are introduced. In Section 3, a P system with external output solving (in a semi–uniform way) the problem to find the n–th power of the transition matrix, and a formal verification of the system is presented; the run time and the resources required in the description of the system are analyzed.

2 Preliminaries

2.1 Markov Chains

Roughly speaking, a (discrete–time) Markov chain is a discrete–time stochastic process such that the past is irrelevant for predicting the future given knowledge of the present, i.e., the conditional distribution of what happens in the future given everything up to now depends only on the present state, and not otherwise on the past.

More formally, a *finite Markov chain* is a sequence $\{X_t : t \in \mathbf{N}\}$ of random variables verifying the following (Markov) property:

$$P(X_{t+1} = j/X_0 = i_0, X_1 = i_1, \ldots, X_t = i_t) = P(X_{t+1} = j/X_t = i_t).$$

That is, given the present, the future does not depend of the past: the result of each event only depends on the result of the previous event.

The range of the random variables is called the *state space* of the Markov chain, and the value of X_t is interpreted as the state of the process at time t. We suppose that the state space is finite, that is, the random variables only take the discrete values $e_1, \ldots, e_k$, called *states* or *results*.

Hence, a Markov chain $\{X_t : t \in \mathbf{N}\}$ provides a random process by a change of states or results $e_1, \ldots, e_k$ in certain instants of discrete times $t \in \mathbf{N}$, and where the result of each event only depends on the result of the previous event. So, such a Markov chain is characterized by the conditional distribution

$$p_{ij}(t) = P(X_t = e_j/X_{t-1} = e_i), \text{ for all } t \geq 1,$$

which is called the *transition probability* of the process, providing one–step transition probability.

The matrix $P(t) = (p_{ij}(t))_{1\leq i,j\leq k}$ is called the *transition matrix* associated with the Markov chain $\{X_t : t \in \mathbf{N}\}$. The term (i, j) of the transition matrix is the probability of a transition from the state e_i to the state e_j. For that, every element of the transition probability matrix is positive, and the sum of each row is 1 because for all i $(1 \leq i \leq k)$ we have

$$\sum_{j=1}^{k} p_{ij}(t) = \sum_{j=1}^{k} P(X_t = e_j / X_{t-1} = e_i) = 1.$$

Hence, the matrix of transition probabilities associated with a Markov chain is stochastic. Moreover, every stochastic matrix can be viewed as the matrix of transition probabilities of some Markov chain.

We say that a finite Markov chain $\{X_t : t \in \mathbf{N}\}$ with k states is *stationary* or *homogeneous* if the transition probabilities do not depend on time, that is, $\forall t\ \forall i, j\ (1 \leq i, j \leq k \rightarrow p_{ij}(t) = p_{ij}(t + 1))$. In this case, we denote $p_{ij}(t) = p_{ij}$, for all $t \in \mathbf{N}$, and $P = (p_{ij})_{1\leq i,j\leq k} = P(t) = (p_{ij}(t))_{1\leq i,j\leq k}$.

The probability of a transition in two, three or more steps is derived in a natural way from the one–step transition probability and the Markov property. From the law of total probability, for all $t \geq 2$ we have:

$$p_{ij}^{(2)}(t) = \sum_{r=1}^{k} P(X_t = e_j / X_{t-1} = e_r) \cdot P(X_{t-1} = e_r / X_{t-2} = e_i).$$

That is, $p_{ij}^{(2)}(t) = \sum_{r=1}^{k} p_{rj}^{(1)}(t) \cdot p_{ir}^{(1)}(t-1)$, where $p_{ij}^{(1)}(t) = p_{ij}(t)$, for all $t \geq 1$.

Then, if the Markov chain is homogeneous, the transition matrix for the two–steps transition is:

$$(p_{ij}^{(2)})_{1\leq i,j\leq k} = (\sum_{r=1}^{k} p_{rj}^{(1)} \cdot p_{ir}^{(1)})_{1\leq i,j\leq k} = P \cdot P = P^2.$$

In general, for each $n \geq 2$ we have

$$p_{ij}^{(n)}(t) = \sum_{r=1}^{k} p_{rj}^{(1)}(t) \cdot p_{ir}^{(n-1)}(t-1)$$

If the Markov chain is homogeneous, then the transition matrix for the n–steps transition is:

$$(p_{ij}^{(n)})_{1\leq i,j\leq k} = (\sum_{r=1}^{k} p_{rj}^{(1)} \cdot p_{ir}^{(n-1)})_{1\leq i,j\leq k} = P^{n-1} \cdot P = P^n.$$

The conditions

$$\begin{cases} p_{ij}^{(1)} = p_{ij} \\ p_{ij}^{(n)} = \sum_{r=1}^{k} p_{rj}^{(1)} \cdot p_{ir}^{(n-1)}, \text{ for all } n \geq 2 \end{cases}$$

are called the *Kolmogorov–Chapmann equations* associated with the homogeneous Markov chain, whose transition matrix is $(p_{ij})_{1 \leq i,j \leq k}$.

A finite and homogeneous Markov chain $\{X_t : t \in \mathbf{N}\}$ where the set of states is $\{e_1, \ldots, e_k\}$, is characterized by the initial probabilities $q_0^j = P(X_0 = e_j)$ $(1 \leq j \leq k)$ to get the state e_j in the first event, and the transition probability matrix $P = (p_{ij})_{1 \leq i,j \leq k}$.

We denote the initial probabilities by means of the vector $q_0 = (q_0^1, \ldots, q_0^k)$, and for each $n \geq 1$, we consider the vector $q_n = (q_n^1, \ldots, q_n^k)$, where q_n^j $(1 \leq j \leq k)$ the probability to reach the state e_j after n–steps of the random process.

Notice that we have $q_n = q_0 \cdot P^n$, for each $n \geq 1$. So, in order to determine the distribution q_n it is enough to study the matrix P^n. Moreover, the limit of the sequence $\{P^n : n \in \mathbf{N}\}$ of these matrices allows us to obtain the distribution limit in the case it exits. For more details see [2] and [3].

Markov chains have many applications. For example, they are used in chemical engineering (modelling the probabilities in chemical reactions and in flow systems), in biology (to model processes that are analogous to biological populations), in bioinformatics (for coding region/gene prediction), in physics (for simulation of particle systems and spatial statistics), in telecommunications (using Markov models for queues), and in geostatistics (in two or three dimensional stochastic simulations of discrete variables conditional on observed data).

2.2 Membrane Systems

Membrane computing is an emergent branch of Natural Computing initiated in the fall of 1998 by Gh. Paun by a paper circulated at that time on web and published in 2000 [4]. Since then, the area has simply flourished and it has received important attention from the scientific community. In February 2003, the Institute for Scientific Information, ISI, has mentioned the foundational paper [4] as a *fast breaking paper* in Computer Science, and in October 2003, the domain itself was qualified by ISI as *fast emerging research front in computer science*.

This new model of computation has been introduced with the aim of defining a computing device, called P system, which abstracts from the structure and functioning of living cells, as well as from the organization of cell in tissues, organs, and other higher order structures.

The main *syntactic* ingredients of a P system are the following:

– A cell–like *membrane structure* consisting of several membranes arranged hierarchically inside a main membrane, the *skin*, and delimiting compartments or *regions*.

- *Multisets* of symbol–objects corresponding to chemical substances present in the compartments of a cell.
- *Evolution rules* corresponding to chemical reactions that can take place inside the cell, and that permit evolve the objects in a synchronous maximally parallel manner.

The *semantics* of P systems is defined through a non–deterministic and synchronous model (a global clock is assumed). A *configuration* of a P system consists of a membrane structure and a family of multisets of objects associated with each region of the structure. At the beginning, there is a configuration called the *initial configuration* of the system. We get *transitions* from one configuration of the system to the next one by applying the evolution rules to the objects placed inside the regions, in a non–deterministic, maximally parallel manner (in each region all objects that can evolve must do it). A *computation* of the system is a (finite or infinite) sequence of configurations such that each configuration –except the initial one– is obtained from the previous one by a transition. A computation which reaches a configuration where no more rules can be applied to the existing objects and membranes, is called a *halting computation*. The result of a halting computation is usually defined through the multiset associated with a specific output membrane (or the environment, as it is the case in this paper) in the final configuration.

More formally, a P *system with external output* of degree m is a tuple $\Pi = (\Gamma, \mu, \mathcal{M}_1, \cdots \mathcal{M}_m, (R_1, \rho_1), \ldots, (R_m, \rho_m))$ where:

- Π is an alphabet. Its elements are called objects.
- μ is a membrane structure consisting of m membranes, with the membranes injectively labeled with $1, 2, \ldots, m$.
- $\mathcal{M}_i, 1 \leq i \leq m$, are strings which represent multisets over Γ associated with the regions $1, \ldots, m$ of μ .
- $R_i, 1 \leq i \leq m$, are finite sets of evolution rules over Γ and ρ_i, $1 \leq i \leq m$, are partial orders over R. The sets R_i and ρ_i are associated with the region i of μ. An evolution rule is a pair (u, v), which usually write in the form $u \to v$, where u is a string over Γ and v is a string over $\Gamma \times (\{here, out\} \cup \{in_j : 1 \leq j \leq m\})$.

In this way, a comprehensive and systematic interdisciplinary research area was developed, a high generality and versatility, where models can be devised for a large range of processes where compartmentalization and multiset processing are natural ingredients. Thus, although the initial goal of membrane computing was only to learn new ideas, tools, and techniques from cell biology to the help of standard computers, much in the same way as, e.g., evolutionary computing suggests algorithms to be implemented on electronic computer, membrane computing became a new framework for building models for a large variety of processes, especially from biology (cell biology, tissues, populations of bacteria, controlling networks of complex phenomena, tumor growth, etc.), but also from linguistics, management, with several applications to computer science (computer graphics,

approximative solutions to computationally hard problems, modelling parallel architectures, cryptography).

Most of these models were proven to be computationally universal, able to compute whatever a Turing machine can compute. In the case when an enhanced parallelism is available, by means of membrane division, string-object replication, or membrane creation, polynomial (often linear) time solutions to **NP**-complete problems were found.

In many variants, P systems are seen as devices of a *generative* nature, that is, from a given initial configuration several distinct computations may be developed, in a non–deterministic manner, producing different outputs.

In this paper we work with P systems with external output and that perform *computing* tasks. For example, if a certain natural number, n, is encoded by the multiplicity of a special object in the initial configuration and we consider the cardinality of the multiset contained in the environment of a halting configuration as the result of a successful computation, then we can say that the system *computes* a partial function from natural numbers onto the set of natural numbers.

In the following, we assume that the reader is familiar with the basic notions of P systems, and we refer to [5] for details.

3 Computing the Natural Power of a Markov Chain

The calculation of the natural powers of the transition matrix of a finite and homogenous Markov chain allows us to estimate its limit in the case that it is convergent and so, we can know the stationary distribution of the process.

In this section, the natural powers of such Markov chains within the framework of the cellular computing with membranes are computed. The solution provided is linear in the order of the power and is independent of the number of states of the Markov chain. The amount of used resources is polynomial in the power and the number of states. Also, a formal verification of the solution is presented.

3.1 Designing a P System

For each Markov chain and each natural number, $n \geq 2$, we construct a P system with external output computing the n–th power of the matrix P_k associated with the Markov chain. Therefore, we provide a *semi–uniform* solution to this problem, that is, we give a family $\mathbf{\Pi} = \{\Pi(P_k, n) : \ n \in \mathbf{N}\}$ such that:

- There exists a deterministic Turing machine working in polynomial time which constructs the system $\Pi(P_k, n)$ from $n \in \mathbf{N}$.
- The output of the P system $\Pi(P_k, n)$ encodes the n–th power of the matrix P_k.

Let $P_k = (p_{ij})_{1 \leq i,j \leq k}$ be the matrix of the transition probabilities associated with a finite and homogeneous Markov chain of order k. Having in mind that

p_{ij} are real numbers in $[0, 1]$ and P systems only work with natural numbers, we have to prefix the approximation to be used in order to represent those numbers in our system. In this paper, as an example, we will work with an approximation to one decimal digit, reason why several objects appear with a factor of 10 in their multiplicities in the description of our system (similarly, if we want to work with m decimal digits, then we must use a 10^m factor).

Let $n \geq 2$ be a natural number. We define a P system of degree 3 with external output,

$$\Pi(P_k, n) = (\Gamma(P_k, n), \mu(P_k, n), \mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, R)$$

associated with the matrix P_k and the natural number n, computing the n–th power of P_k, as follows:

– Working alphabet:

$$\begin{aligned}\Gamma(P_k, n) = &\{s_{ij}^{(r)} : \ 1 \leq i, j \leq k, \ 0 \leq r \leq n\} \ \cup \{s_{ij} : \ 1 \leq i, j \leq k\} \cup \\ &\{t_{ij} : \ 1 \leq i, j \leq k\} \ \cup \ \{t_{iju}^{(r)} : \ 1 \leq i, j, u \leq k, \ 0 \leq r \leq n-1\} \cup \\ &\{p_{ij} : \ 1 \leq i, j \leq k\} \ \cup \ \{c_i : \ 1 \leq i \leq n-1\}.\end{aligned}$$

– Membrane structure: $\mu(P_k, n) = [_1 \ [_2 \ [_3 \ ]_3 \ ]_2 \ ]_1$.

– Initial multisets:

$$\mathcal{M}_1 = \mathcal{M}_2 = \emptyset;$$

$$\mathcal{M}_3 = \{t_{ij}^{k \cdot 10 \cdot p_{ij}} : \ 1 \leq i, j \leq k\} \ \cup \ \{s_{ii}^{(0)10} : \ 1 \leq i \leq k\} \ \cup \ \{c_1\}.$$

– The set R of evolution rules consists of the following rules:

- Rules in the skin membrane labeled by 1:

$$\{t_{iju}^{(r)10p_{ij}} \longrightarrow (t_{ij}^{10p_{ij}} s_{uj}^{(r+1)10p_{ij}}, in_2) : \ 1 \leq i, j, u \leq k, \ 0 \leq r \leq n-1\} \ \cup$$
$$\{s_{ij} \longrightarrow (s_{ij}, out) : \ 1 \leq i, j \leq k\}.$$

- Rules in the membrane labeled by 2:

$$\{s_{ij}^{(r)10} t_{j1}^{10p_{j1}} \ldots t_{jk}^{10p_{jk}} \longrightarrow (t_{j1i}^{(r)10p_{j1}} \ldots t_{jki}^{(r)10p_{jk}}, out) : \ 1 \leq i, j \leq k, \ 0 \leq r \leq n-1\} \ \cup \ \{s_{ij}^{(n)} \longrightarrow (s_{ij}, out) : \ 1 \leq i, j \leq k\}.$$

- Rules in the membrane labeled by 3:

$$\{s_{ii}^{(0)} \longrightarrow s_{ii}^{(0)10} : \ 1 \leq i \leq k\} \ \cup \ \{t_{ij} \longrightarrow t_{ij}^{10} : \ 1 \leq i, j \leq k\} \ \cup \ \{c_i \longrightarrow c_{i+1} : \ 1 \leq i \leq n-2\} \ \cup \ \{c_{n-1} \longrightarrow \delta\}.$$

3.2 An Overview of Computations

The P system $\Pi(P_k, n)$ works in the following way. At the beginning, the skin membrane and the membrane labeled by 2 are empty and the membrane labeled by 3 has: (a) objects t_{ij} $(1 \leq i, j \leq k)$ encoding the elements p_{ij} of the trasition matrix of the Markov chain; (b) objects s_{ii} $(1 \leq i \leq k)$ encoding the states e_i of the chain; and (c) objects c_i $(1 \leq i \leq n-1)$ interpreted as counters used to know when a suitable number of objects t_{ij} and s_{ii} have been produced.

In the $n-2$ first steps only rules in the internal membrane labeled by 3 are applied. During this (so called) first stage, each object $s_{ii}^{(0)}$ and each t_{ij} is replicated 10 times in each transition step. Furthermore, in membrane 3 a counter c_i appears, initialized to c_1, whose subindex increase by one unit during each step. For that reason, after those $n-2$ steps, membrane 3 contains the multiset of objects $s_{ii}^{(0)\cdot 10^{n-1}}\ t_{ij}^{k\cdot 10^{n-1}\cdot p_{ij}}$, and the counter c_{n-1}. In the $(n-1)$–th step, each object $s_{ii}^{(0)}$ and each object t_{ij} are replicated 10 times and, moreover, membrane 3 is dissolved, and its content pass to the internal membrane labeled by 2.

Therefore, when the system is going to execute the n–th step, the skin membrane continues being empty and the content of the internal membrane labeled by 2 is $s_{ii}^{(0)\cdot 10^{n}}\ t_{ij}^{k\cdot 10^{n}\cdot p_{ij}}$.

In a second stage, in each $(n-1+2m+1)$–th step, with $m \in \mathbf{N}$, only rules of membrane 2 will be applied; they will consume all the objects $s_{ij}^{(m)}$ and some objects t_{ij}, sending to the skin certain objects t_{jui}. In each $(n-1+2m)$–th step, with $m \in \mathbf{N}-\{0\}$, only rules in the skin are applied (because there do not exist objects s_{ij} in membrane 2), sending new objects t_{ij} and objects $s_{ij}^{(m)}$ to membrane 2. This second phase finalizes when $m = n-1$.

A third stage begins with the execution of the $(3n-1)$–th step, after which the skin membrane will be empty and in membrane 2 objects $s_{ij}^{(n)}$ appear. Then, the execution of the rules $s_{ij}^{(n)} \longrightarrow (s_{ij}, out)$ sends these objects to the skin membrane, and in the following step, they sent to the environment by means of the rules skin $s_{ij} \longrightarrow (s_{ij}, out)$. In this moment, no rule of the system will be applicable and so, the configuration obtained after the $(3n+1)$–th step will be a halting one. Moreover, in the last step, the content of the environment will be $s_{ij}^{w_{ij}^{(n)}}$.

Finally, it will remain to show that the multiplicity $w_{ij}^{(n)}$ of the object s_{ij} is equal to the (i,j)–term of the matrix $10^n \cdot P_k^n$.

3.3 Formal Verification

Throughout this section we are going to justify that the system $\Pi(P_k, n)$ is deterministic and that the computation of the system codifies in the environment of the halting configuration the n–th power of the transition probability matrix, P_k, associated with a finite and homogeneous Markov chain.

First of all, let us list the necessary resources to build the system $\Pi(P_k, n)$ from the matrix P_k and the natural number $n \geq 2$:

- Size of the alphabet: $(n+1)k^2 + k^2 + k^2 + nk^3 + k^2 + (n-1) \in \Theta(nk^3)$.
- Sum of the sizes of initial multisets: $\leq 10k^3 + 10k + 1 \in \Theta(k^3)$.
- Maximum of rules' lengths: $20k + 10 \in \Theta(k)$.
- Number of rules: $nk^3 + k^2 + nk^2 + k^2 + k + k^2 + (n-1) \in \Theta(nk^3)$.

Bearing in mind the recursive description of the rules and that the amount of resources is polynomial in $n \cdot k$, it is possible to construct the system $\Pi(P_k, n)$ from the matrix P_k and the natural number $n \geq 2$, by means of a deterministic Turing machine working in polynomial time.

Next, we are going to define in a recursive manner in r the expression $w_{ij}^{(r)}$, for each i, j such that $1 \leq i, j \leq k$, that it is necessary in the formal verification of the system that will follow.

Definition 1. *Let $w_0 = 10^n$. For each i, j such that $1 \leq i, j \leq k$, we define:*

$$w_{ij}^{(0)} = \begin{cases} w_0 & \text{if } i = j, \\ 0 & \text{if } i \neq j. \end{cases}$$

$$w_{ij}^{(r+1)} = \sum_{u=1}^{k} w_{iu}^{(r)} p_{uj}, \text{ for } r < n.$$

Remark: In the case of a finite and homogeneous Markov chain, with states $e_1, \ldots, e_k$ and transition matrix P_k, the definition of the values w_{ij}^{r+1} can be interpreted as an *abstraction* of the equation of Kolmogorov–Chapmann: the transition of the state e_i to the state e_j in $(r+1)$ steps (that is, the value $w_{ij}^{(r+1)}$) is obtained from all the transitions in r steps of the state e_i to any state e_u (that is, the value $w_{iu}^{(r)}$), with $1 \leq u \leq k$, multiplied by the transitions of e_u to e_j in only one step (that is, the value p_{uj}).

Next, we establish the relation that exists between the elements $w_{ij}^{(r)}$ and the term (i, j) of the matrix $10^n \cdot P_k^r$, for each $1 \leq r \leq n$.

Proposition 1. *Let $n \geq 2$. Let us denote $B(r, n) = 10^n \cdot P_k^r$, for each r such that $r \geq 1, r \leq n$. If $B(r, n) = (b_{ij}^{(r,n)})_{1 \leq i,j \leq k}$, then*

$\forall r \geq 1 \ (r \leq n \longrightarrow \forall i, j \ (1 \leq i, j \leq k \longrightarrow b_{ij}^{(r,n)} = w_{ij}^{(r)}))$.

Proof. By induction on r.
The base case, $r = 1$, follows from the following remark:

$$w_{ij}^{(1)} = \sum_{u=1}^{k} w_{iu}^{(0)} p_{sj} = w_0 p_{ij} = 10^n p_{ij} = b_{ij}^{(1,n)}.$$

Let $r \geq 1$ be such that $r < n$ and suppose that $b_{ij}^{(r,n)} = w_{ij}^{(r)}$ is verified. Bearing in mind that $10^n \cdot P_k^{r+1} = 10^n P_k^r \cdot P_k = B(r, n) \cdot P_k$, we deduce that for each i, j such that $1 \leq i, j \leq k$:

$$b_{ij}^{(r+1,n)} = \sum_{u=1}^{k} b_{iu}^{(r,n)} p_{uj} \overset{h.i.}{=} \sum_{u=1}^{k} w_{iu}^{(r)} p_{uj} = w_{ij}^{(r+1)}. \qquad \square$$

For each $m \in \mathbf{N}$, we denote by $\mathcal{C}_m$ the configuration of the system obtained after the execution of m steps. For each label $l \in \{1, 2, 3\}$, we denote by $C_m(l)$ the multiset of objects contained in the membrane labeled by l in the configuration

$\mathcal{C}_m$. Also, we denote by $C_m(env)$ the content of the environment of the system in the configuration $\mathcal{C}_m$.

From the definition of the system $\Pi(P_k, n)$, the following holds: $C_0(1) = C_0(2) = \emptyset$, and $C_0(3) = \{t_{ij}^{k\cdot 10\cdot p_{ij}} : 1 \leq i, j \leq k\} \cup \{s_{ii}^{(0)10} : 1 \leq i \leq k\} \cup \{c_1\}$.

Next, we are going to determine the content of the different membranes of the system along the execution, and will show that after $3n+1$ steps the system reaches a halting configuration, in which the content of the environment is the multiset of objects $\{s_{ij}^{w_{ij}^{(n)}} : 1 \leq i, j \leq k\}$.

In the proof of the following result, it can be checked that there exists only one multiset of rules applicable to a non halting configuration in each transition step, and, consequently, the membrane system $\Pi(P_k, n)$ is deterministic.

Theorem 1. *Let $n \geq 2$.*

(a) For each $r \in \mathbf{N}$ such that $1 \leq r \leq n-2$ we have:

$$\begin{cases} C_r(1) = \emptyset \\ C_r(2) = \emptyset \\ C_r(3) = \{t_{ij}^{k\cdot 10^{r+1}\cdot p_{ij}} : 1 \leq i, j \leq k\} \cup \{s_{ii}^{(0)10^{r+1}} : 1 \leq i \leq k\} \cup \{c_{r+1}\} \end{cases}$$

(b) Moreover, we have:

$$\begin{cases} C_{n-1}(1) = \emptyset, \\ C_{n-1}(2) = \{t_{ij}^{k\cdot 10^{n}\cdot p_{ij}} : 1 \leq i, j \leq k\} \cup \{s_{ii}^{(0)10^{n}} : 1 \leq i \leq k\} \end{cases}$$

(c) For each $m \in \mathbf{N}$ such that $m < n$ we have:

$$\begin{cases} C_{(n-1)+2m}(1) = \emptyset, \\ C_{(n-1)+2m}(2) = \{t_{ij}^{kw_0p_{ij}} : 1 \leq i, j \leq k\} \cup \{s_{ij}^{(m)w_{ij}^{(m)}} : 1 \leq i, j \leq k\}, \\ C_{(n-1)+2m+1}(1) = \{t_{jui}^{(m)w_{ij}^{(m)}p_{ju}} : 1 \leq i, j, u \leq k\}, \\ C_{(n-1)+2m+1}(2) = \{t_{ij}^{kw_0p_{ij}-\sum_{u=1}^{k} w_{ui}^{(m)}p_{ij}} : 1 \leq i, j \leq k\}. \end{cases}$$

(d) The configuration $\mathcal{C}_{3n+1}$ is a halting one, and

$$C_{3n+1}(env) = \{s_{ij}^{w_{ij}^{(n)}} : 1 \leq i, j \leq k\}.$$

Proof.

(a) If $n = 2$, then the result is obvious. Let us assume now that $n > 2$. We will prove the result by induction on r.

In order to prove the base case $r = 1$, let us observe that from the initial configuration of the system we have: $C_0(1) = C_0(2) = \emptyset$, and $C_0(3) = \{t_{ij}^{k\cdot 10\cdot p_{ij}} : 1 \leq i, j \leq k\} \cup \{s_{ii}^{(0)10} : 1 \leq i \leq k\} \cup \{c_1\}$.

Then, only in the membrane 3 of configuration $\mathcal{C}_0$ there are applicable rules and, concretely, the rules: $s_{ii}^{(0)} \longrightarrow s_{ii}^{(0)10}$, for $1 \leq i \leq k$; $t_{ij} \longrightarrow t_{ij}^{10}$, for $1 \leq i, j \leq k$, and $c_1 \longrightarrow c_2$.

Therefore, $C_1(1) = C_1(2) = \emptyset$, and $C_1(3) = \{t_{ij}^{k\cdot 10^2\cdot p_{ij}} : 1 \leq i, j \leq k\} \cup \{s_{ii}^{(0)10^2} : 1 \leq i \leq k\} \cup \{c_2\}$.

Let r be a natural number such that $1 \leq r < n-2$. Let us suppose that $C_r(1) = C_r(2) = \emptyset$, and $C_r(3) = \{t_{ij}^{k \cdot 10^{r+1} \cdot p_{ij}} : 1 \leq i,j \leq k\} \cup \{s_{ii}^{(0)10^{r+1}} : 1 \leq i \leq k\} \cup \{c_{r+1}\}$.

Let us note that only in membrane 3 there are applicable rules to configuration $\mathcal{C}_r$; specifically, this is the case for the rules: $s_{ii}^{(0)} \longrightarrow s_{ii}^{(0)10}$, for $1 \leq i \leq k$; $t_{ij} \longrightarrow t_{ij}^{10}$, for $1 \leq i,j \leq k$; and $c_{r+1} \longrightarrow c_{r+2}$ (recall that $r < n-2$ and then $r+1 < n-1$).

Therefore, $C_{r+1}(1) = C_{r+1}(2) = \emptyset$, and $C_{r+1}(3) = \{t_{ij}^{k \cdot 10^{r+2} \cdot p_{ij}} : 1 \leq i,j \leq k\} \cup \{s_{ii}^{(0)10^{r+2}} : 1 \leq i \leq k\} \cup \{c_{r+2}\}$.

(b) Directly from (a) it follows that: $C_{n-2}(1) = C_{n-2}(2) = \emptyset$, and $C_{n-2}(3) = \{t_{ij}^{k \cdot 10^{n-1} \cdot p_{ij}} : 1 \leq i,j \leq k\} \cup \{s_{ii}^{(0)10^{n-1}} : 1 \leq i \leq k\} \cup \{c_{n-1}\}$.

In order to obtain the configuration $\mathcal{C}_{n-1}$, let us note that only in membrane 3 there are applicable rules, namely the rules: $s_{ii}^{(0)} \longrightarrow s_{ii}^{(0)10}$, for $1 \leq i \leq k$; $t_{ij} \longrightarrow t_{ij}^{10}$, for $1 \leq i,j \leq k$; and $c_{n-1} \longrightarrow \delta$.

Then, membrane 3 will be dissolved, its content goes to membrane 2, and the counters c_i disappear. Thus, $C_{n-1}(1) = \emptyset$, and $C_{n-1}(2) = \{t_{ij}^{k \cdot 10^n \cdot p_{ij}} : 1 \leq i,j \leq k\} \cup \{s_{ii}^{(0)10^n} : 1 \leq i \leq k\}$.

(c) We prove the result by induction on m. From (b) we deduce that $C_{n-1}(1) = \emptyset$, and $C_{n-1}(2) = \{t_{ij}^{kw_0 p_{ij}} : 1 \leq i,j \leq k\} \cup \{s_{ij}^{(0)w_{ij}^{(0)}} : 1 \leq i,j \leq k\}$.

From Definition 1 we have

$$\{s_{ii}^{(0)10^n} : 1 \leq i \leq k\} = \{s_{ii}^{(0)w_0} : 1 \leq i \leq k\} = \{s_{ij}^{(0)w_{ij}^{(0)}} : 1 \leq i,j \leq k\}.$$

Let us note that only in the internal membrane (labeled by 2) there are applicable rules to configuration $\mathcal{C}_{n-1}$, namely the rules:

$$s_{ij}^{(0)10} t_{j1}^{10p_{j1}} \ldots t_{jk}^{10p_{jk}} \longrightarrow (t_{j1i}^{(0)10p_{j1}} \ldots t_{jki}^{(0)10p_{jk}}, out), \text{ for } 1 \leq i,j \leq k.$$

By the condition of maximal parallelism, each one of these rules will be applied $\frac{w_{ij}^{(0)}}{10}$ times. Consequently, we have:

$$\begin{cases} C_n(1) = \{t_{jui}^{(0)w_{ij}^{(0)}p_{ju}} : 1 \leq i,j,u \leq k\}, \\ C_n(2) = \{t_{ij}^{kw_0p_{ij} - w_0p_{ij}} : 1 \leq i,j \leq k\} = \{t_{ij}^{kw_0p_{ij} - \sum_{u=1}^{k} w_{ui}^{(0)}p_{ij}} : 1 \leq i,j \leq k\}. \end{cases}$$

Hence, the result is true for $m = 0$. Let $m < n-1$. Let us suppose that the result holds for m, that is,

$$\begin{cases} C_{(n-1)+2m}(1) = \emptyset, \\ C_{(n-1)+2m}(2) = \{t_{ij}^{kw_0p_{ij}} : 1 \leq i,j \leq k\} \cup \{s_{ij}^{(m)w_{ij}^{(m)}} : 1 \leq i,j \leq k\}, \\ C_{(n-1)+2m+1}(1) = \{t_{jui}^{(m)w_{ij}^{(m)}p_{ju}} : 1 \leq i,j,u \leq k\}, \\ C_{(n-1)+2m+1}(2) = \{t_{ij}^{kw_0p_{ij} - \sum_{u=1}^{k} w_{ui}^{(m)}p_{ij}} : 1 \leq i,j \leq k\}. \end{cases}$$

In order to obtain the configuration $\mathcal{C}_{(n-1)+2m+2}$, let us note that there are applicable rules only in the skin membrane of the configuration $\mathcal{C}_{(n-1)+2m+1}$. Specifically, this is the case with the rules:

$$t_{jui}^{(m)10p_{ju}} \longrightarrow (t_{ju}^{10p_{ju}} s_{iu}^{(m+1)10p_{ju}}, in_2), \text{ for } 1 \leq i, j, u \leq k.$$

By the condition of maximal parallelism, each one of these rules will be applied $\frac{w_{ij}^{(m)} p_{js}}{10p_{js}} = \frac{w_{ij}^{(m)}}{10}$ times. Hence,

$$\begin{cases} C_{(n-1)+2m+2}(1) = \emptyset, \\ C_{(n-1)+2m+2}(2) = \{t_{ij}^{kw_0 p_{ij} - \sum_{u=1}^{k} w_{ui}^{(m)} p_{ij}} : 1 \leq i, j \leq k\} \cup \\ \qquad \{t_{ij}^{w_{1i}^{(m)} p_{ij} + \cdots + w_{ki}^{(m)} p_{ij}} : 1 \leq i, j \leq k\} \cup \\ \qquad \{s_{ij}^{(m+1) w_{i1}^{(m)} p_{1j} + \cdots + w_{ik}^{(m)} p_{kj}} : 1 \leq i, j \leq k\} \\ \qquad = \{t_{ij}^{kw_0 p_{ij}} : 1 \leq i, j \leq k\} \cup \{s_{ij}^{(m+1) w_{ij}^{(m+1)}} : 1 \leq i, j \leq k\}. \end{cases}$$

In order to obtain the configuration $\mathcal{C}_{(n-1)+2m+3}$, let us note that there are applicable rules only in the internal membrane of the configuration $\mathcal{C}_{(n-1)+2m+2}$ (let us recall that $m < n-1$, and then $m+1 < n$). Specifically the following rules can be applied:

$$s_{ij}^{(m+1)10} t_{j1}^{10p_{j1}} \ldots t_{jk}^{10p_{jk}} \longrightarrow (t_{j1i}^{(m+1)10p_{j1}} \ldots p_{jki}^{(m+1)10p_{jk}}, out), \ 1 \leq i, j \leq k.$$

By the condition of maximal parallelism, each one of these rules will be applied $\frac{w_{ij}^{(m+1)}}{10}$ times. Thus,

$$\begin{cases} C_{(n-1)+2m+3}(1) = \{t_{jui}^{(m+1) w_{ij}^{(m+1)} p_{ju}} : 1 \leq i, j, u \leq k\}, \\ C_{(n-1)+2m+3}(2) = \{t_{ij}^{kw_0 p_{ij} - \sum_{u=1}^{k} w_{ui}^{(m+1)} p_{ij}} : 1 \leq i, j \leq k\}. \end{cases}$$

Hence, the result is true for $m+1$, concluding the proof of (c).

(d) From (c) we deduce that

$$\begin{cases} C_{3n-2}(1) = \{t_{jui}^{(n-1) w_{ij}^{(n-1)} p_{ju}} : 1 \leq i, j, u \leq k\}, \\ C_{3n-2}(2) = \{t_{ij}^{kw_0 p_{ij} - \sum_{u=1}^{k} w_{ui}^{(n-1)} p_{ij}} : 1 \leq i, j \leq k\}. \end{cases}$$

Let us observe that there are applicable rules only in the skin membrane of the configuration $\mathcal{C}_{3n-2}$. Specifically, the following rules can be used:

$$t_{jui}^{(n-1)10p_{ju}} \longrightarrow (t_{ju}^{10p_{ju}} s_{iu}^{(n)10p_{ju}}, in_2), \text{ for } 1 \leq i, j, u \leq k.$$

By the condition of maximal parallelism, each one of these rules will be applied $\frac{w_{ij}^{(n-1)} p_{js}}{10p_{js}} = \frac{w_{ij}^{(n-1)}}{10}$ times. Consequently, we have:

$$\begin{cases} C_{3n-1}(1) = \emptyset, \\ C_{3n-1}(2) = \{t_{ij}^{kw_0p_{ij} - \sum_{u=1}^{k} w_{ui}^{(n-1)} p_{ij}} : 1 \leq i,j \leq k\} \cup \\ \qquad \{t_{ij}^{w_{1i}^{(n-1)} p_{ij} + \cdots + w_{ki}^{(n-1)} p_{ij}} : 1 \leq i,j \leq k\} \cup \\ \qquad \{s_{ij}^{(n) w_{i1}^{(n-1)} p_{1j} + \cdots + w_{ik}^{(n-1)} p_{kj}} : 1 \leq i,j \leq k\} \\ \qquad = \{t_{ij}^{kw_0p_{ij}} : 1 \leq i,j \leq k\} \cup \{s_{ij}^{(n) w_{ij}^{(n)}} : 1 \leq i,j \leq k\}. \end{cases}$$

Then, to obtain the configuration $\mathcal{C}_{3n}$, it is possible to apply rules only in the internal membrane of the configuration $\mathcal{C}_{3n-1}$, namely, the following rules:

$$s_{ij}^{(n)} \longrightarrow (s_{ij}, out), \text{ for } 1 \leq i,j \leq k.$$

Thus, $C_{3n}(1) = \{s_{ij}^{w_{ij}^{(n)}} : 1 \leq i,j \leq k\}$, and $C_{3n}(2) = \{t_{ij}^{kw_0p_{ij}} : 1 \leq i,j \leq k\}$.

Then, there are applicable rules only in the skin membrane of the configuration $\mathcal{C}_{3n}$. Specifically, the following rules can be applied:

$$s_{ij} \longrightarrow (s_{ij}, out), \text{ for } 1 \leq i,j \leq k.$$

Hence, we have: $C_{3n+1}(1) = \emptyset; C_{3n+1}(2) = \{t_{ij}^{kw_0p_{ij}} : 1 \leq i,j \leq k\}$, and $C_{3n+1}(env) = \{s_{ij}^{w_{ij}^{(n)}} : 1 \leq i,j \leq k\}$.

Then, there is no other applicable rule to configuration $\mathcal{C}_{3n+1}$. Consequently, this configuration is a halting one. □

Theorem 2. *Let $k \geq 1, n \geq 2$. Let $P_k = (p_{ij})_{1 \leq i,j \leq k}$ be the transition matrix associated with a finite and homogeneous Markov chain. Let $\Pi(P_k, n)$ be the P system defined in Section 3.1. The output of the only computation of this system (that is, the content of the environment in the halting configuration) codifies the matrix $B(n,n) = 10^n \cdot P_k^n$.*

Proof. From (d) in Theorem 1 we deduce that the configuration $\mathcal{C}_{3n+1}$ of the system $\Pi(P_k, n)$ is a halting one and, moreover, the multiset associated with the environment is $C_{3n+1}(env) = \{s_{ij}^{w_{ij}^{(n)}} : 1 \leq i,j \leq k\}$.

Directly from Proposition 1, with $r = n$, it follows that $\forall i \; \forall j \; (1 \leq i,j \leq k \rightarrow w_{ij}^{(n)} = b_{ij}^{(n,n)})$. That is, the multiplicity $w_{ij}^{(n)}$ of the object s_{ij} in the environment of the halting configuration $\mathcal{C}_{3n+1}$ coincides with $b_{ij}^{(n,n)}$, the (i,j)–term of the matrix $B(n,n) = 10^n \cdot P_k^n$. □

4 Conclusions

In this paper, a deterministic P system with external output associated with a natural number, $n \geq 2$, and to a finite and homogeneous Markov chain, is described. This P system provides the n–th power of the transition matrix associated with the Markov chain, encoding the power in the environment of a halting configuration of the system.

In [1] this problem has been addressed by means of a molecular DNA based algorithms, giving an *estimation* of this power in polynomial time, and providing a new approach to the problem of computing the limit of a Markov chain.

The solution presented in this work is placed in the scope of the cellular computing with membranes. It is a *semi–uniform* solution, because for each Markov chain and each power, a specific P system is designed. The solution is efficient, because it is linear in the power and independent of the number of states of the Markov chain. Furthermore, the amount of resources initially required to construct the system is polynomial in the power and in the order of the Markov chain.

The paper also provides a new example of formal verification of P systems designed to solve a problem, following a specific methodology valid in some cases like those considered in the paper. These examples are always interesting, for instance, in order to find systematic processes of formal verification in a model of computation oriented to machines, like the cellular model, in where it is well known that the mechanisms of verification are often a very hard task.

Acknowledgement

The third author wishes to acknowledge the support of the Project TIN2005-09345-C04-01 of the Ministerio de Educación y Ciencia of Spain, cofinanced by FEDER funds, and the support of the Project of Excelence TIC-581 of the Junta de Andalucía.

References

1. M. Cardona, M.A. Colomer, J. Conde, J.M. Miret, J. Miró, A. Zaragoza, Markov chains: computing limit existence and approximations with DNA, *Biosystems*, 81, 3 (2005) 261–266.
2. O. Häggström, *Finite Markov chains and algorithmic applications*, London Mathematical Society, Cambridge University Press, 2002.
3. R. Nelson, *Probability, stochastic processes, and queueing theory:the mathematics of computer performance modeling*, Springer-Verlag, New York, 1995.
4. Gh. Păun, Computing with membranes, *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143, and *Turku Center for Computer Science-TUCS Report* Nr. 208, 1998.
5. Gh. Păun, *Membrane Computing. An Introduction*, Springer-Verlag, Berlin, 2002.

Capítulo 7

Classifying States of a Finite Markov Chain with Membrane Computing

Classifying States of a Finite Markov Chain with Membrane Computing

Mónica Cardona[1], M. Angels Colomer[1],
Mario J. Pérez-Jiménez[2], and Alba Zaragoza[1]

[1] Department of Mathematics, University of Lleida
Avda. Alcalde Rovira Roure, 191. 25198 LLeida, Spain
{mcardona,colomer,alba}@matematica.udl.es

[2] Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
marper@us.es

Abstract. In this paper we present a method to classify the states of a finite Markov chain through membrane computing. A specific P system with external output is designed for each boolean matrix associated with a finite Markov chain. The computation of the system allows us to decide the convergence of the process because it determines in the environment the classification of the states (recurrent, absorbent, and transient) as well as the periods of states. The amount of resources required in the construction is polynomial in the number of states of the Markov chain.

1 Introduction

Markov chains constitute an important type of stochastic processes characterized by their evolution along determinate values (called states of the process) over time. These chains represent observations of physic systems whose evolution at a future time, conditioned on their present and past values, depends only on their present value. Thus, the Markov chain loses the *memory* of its starting state.

In order to study the evolution in time of a Markov chain as well as the existence of the stationary distribution it is necessary to classify its states. This classification depends on the path structure of the chain.

In this work this problem is approached within the framework of the cellular computing with membranes. The amount of resources that we use is polynomial in the number of states. This subject has been also treated in terms of DNA computing ([1]), based on a mathematical proposition of existence rather than on the classical definition of the period of a state. This is due to the fact that DNA computing is good in detecting the existence, but it has difficulties in obtaining numerical quantifications.

The paper is structured as follows. In the next section, basic concepts concerning Markov chains and P systems are introduced. In Section 3 a semi–uniform

H.J. Hoogeboom et al. (Eds.): WMC 7, LNCS 4361, pp. 266–278, 2006.

solution to the problem of classifying the states of a Markov chain in the framework of membrane computing is presented. Moreover, a formal verification of the system is given, and the run time and the resources required in the description of the system are analyzed.

2 Preliminaries

2.1 Markov Chains

Markov chains are a class of random processes exhibiting a certain *memoryless property* and providing a fundamental ingredient in the study of randomized algorithms. Their study is one of the main areas in modern probability theory.

A Markov process is a stochastic process that has a limited form of historical dependency. Let $\{X(t) : t \in \tau\}$ be a stochastic process defined on the parameter τ. We will think of τ in terms of time and the values that $X(t)$ can assume are called *states* which are elements of a *state space* S. In the case when the set τ is *discrete* and the set S is *finite*, the Markov process is called a *discrete–time finite Markov chain*. We consider this kind of Markov chains because computer programs work in discrete steps and computers work with a finite amount of resources and have a finite number of states.

More formally, a *finite Markov chain* is a sequence $\{X_t : t \in \mathbf{N}\}$ of random variables verifying the following (Markov) property:

$$P(X_{t+1} = j/X_0 = i_0, X_1 = i_1, \ldots, X_t = i_t) = P(X_{t+1} = j/X_t = i_t).$$

That is, the value of X_{t+1} conditioned on the value of X_t, is independent of the values of random variables X_m for $m < t$.

We suppose that the state space of the chain, S, is the (finite) set of nonnegative integers $\{e_1, \ldots, e_k\}$ (whose elements are called *states* or *results*), and the chain is characterized by its evolution among these states over time.

Hence, a finite Markov chain $\{X_t : t \in \mathbf{N}\}$ provides a random process by a change of states or results $e_1, \ldots, e_k$ in certain instants of discrete times $t \in \mathbf{N}$, and where the result of each event only depends on the result of the previous event. So, such a Markov chain is characterized by the conditional distribution

$$p_{ij}(t) = P(X_t = e_j/X_{t-1} = e_i), \text{ for all } t \geq 1,$$

which is called the *transition probability* of the process, providing one–step transition probability.

We say that a finite Markov chain is *time homogeneous* or it has *stationary transition probabilities* if the dependence between consecutive states does not change, that is, $P(X_n = e_j/X_{n-1} = e_i) = P(X_{n+m} = e_j/X_{n+m-1} = e_i)$, for all $n, m \in \mathbf{N}$, $e_i, e_j \in S$. In this case, we write the transition probability as $p_{ij} = P(X_n = e_j/X_{n-1} = e_i)$. These probabilities form a stochastic matrix $P = (p_{ij})$ with $\sum_{j=1}^{k} p_{ij} = 1, \forall i \in \{1, \ldots, k\}$, called *transition matrix*. If $p_{ij} \neq 0$ then we say that the transition from the state e_i to state e_j is possible.

The Markov property allows us to write an expression for the probability of a transition in one, two, three or more steps. For $n = 1$ this probability is simply p_{ij} and is given by the position (i, j) of the transition matrix P. For $n = 2$ the probability that the chain is in state e_j at step 2 is $p_{ij}^{(2)} = \sum_{r=1}^{k} p_{ir} p_{rj}$, and is the position (i, j) of the matrix P^2. In general, the probability that the process is in state e_j n steps after being in state e_i is given by $p_{ij}^{(n)} = \sum_{r=1}^{k} p_{ir}^{(m)} p_{rj}^{(n-m)}$, $0 \leq m \leq n$, and is the position (i, j) of the matrix $P^n = P^m P^{n-m}$ (by the rules for matrix multiplication).

The conditions

$$\begin{cases} p_{ij}^{(1)} = p_{ij}, \\ p_{ij}^{(n)} = \sum\limits_{r=1}^{k} p_{rj}^{(1)} \cdot p_{ir}^{(n-1)}, \text{ for all } n \geq 2, \end{cases}$$

are called the *Kolmogorov–Chapmann equations* associated with the homogeneous Markov chain whose transition matrix is $P = (p_{ij})_{1 \leq i,j \leq k}$ ([4]).

We denote the initial probabilities by means of the vector $q_0 = (q_0^1, \ldots, q_0^k)$, and for each $n \geq 1$ we consider the vector $q_n = (q_n^1, \ldots, q_n^k)$, where q_n^j $(1 \leq j \leq k)$ is the probability to reach the state e_j after n steps of the random process.

Notice that we have $q_n = q_0 P^n$, for each $n \geq 1$. So, in order to determine the distribution q_n it is enough to study the matrix P^n. In [2] the natural powers of the transition matrix of a finite and homogeneous Markov chain within the framework of membrane computing are computed. Moreover, the limit of the sequence $\{P^n : n \in \mathbf{N}\}$ of these matrices allows us to obtain the distribution limit in the case that it exits, and to know the stationary distribution of the process. For more details see [3] and [4].

There is a well known result [5] relating the existence of the limit of the sequence $\{P^n : n \in \mathbf{N}\}$ with the classification of the states of the Markov chain. So, we give now a classification of the states of a Markov chain and the condition by the existence of the limit.

A state e_j is *accessible* from the state e_i, denoted by $e_i \to e_j$, if there is a natural number $n > 0$ such that $p_{ij}^{(n)} > 0$. Two states e_i, e_j are *communicating states,* denoted by $e_i \leftrightarrow e_j$, if e_i is accessible from e_j and e_j is accessible from e_i. The relation of communication is an equivalence, so we can consider the equivalence classes associated with it.

The following result shows that in a finite Markov chain with k states, if we know all the paths of length $k - 1$, then we can know all the communicating states.

Proposition 1. *Let e_i, e_j be states of a finite Markov chain with k states such that e_j is accessible from e_i. Then there exists a path with length smaller than k from e_i to e_j.*

This result can be proved by substituting the nodes repeated in the path by only one copy of each of them.

A state e_i is called *recurrent* if for all e_j such that $e_i \to e_j$, then $e_j \to e_i$. On the contrary, if there exist j such that $e_i \to e_j$ but $e_j \not\to e_i$ (that is, there is a natural number m and a state e_j such that $p_{ij}^{(m)} > 0$ but $p_{ji}^{(n)} = 0$ for all $n \in \mathbf{N}$), the state e_i is called *transient.* If in an equivalence class there exists a recurrent (resp. transient) state, then every state of the class is recurrent (respectively) transient. If the class of a recurrent state e_i is formed only by this state, we say that e_i is an *absorbent state* ($p_{ij}^{(n)} = 0$ for all $e_j \neq e_i$ and for all $n \in \mathbf{N}$).

Given a state e_i such that there exists $n > 0$ and $p_{ii}^{(n)} > 0$, we define its *period* as

$$d(i) = \text{g.c.d.}\{n \geq 1 \mid p_{ii}^{(n)} > 0\}.$$

All states that belong to the same class have the same period. If the period is 1, the class is said to be *aperiodic*, otherwise we refer to it as a *periodic* class.

The problem of classification is an important one in the mathematical study of Markov chains and related stochastic processes because it allows us to study their asymptotic behavior. If we think a Markov chain as a system evolving along the time, then we are interested in analyzing how that evolution is carried out. For that, we study the existence and the uniqueness of the stationary distributions, and the convergence to stationarity starting from any initial distribution. That study is related with the number of recurrent classes of a finite Markov chain.

There are some necessary conditions for the existence of stationary distributions, that is to say, there are some results which provide us with information about the existence of the limit of the sequence of the matrix powers of a finite Markov chain ([5]).

Theorem 1. *For any Markov chain with finite states, there exists a unique stationary distribution if and only if the set of states contains precisely one recurrent class.*

Theorem 2. *A necessary and sufficient condition for the existence of a limit distribution is that there is, in the set of states of the chain, exactly one aperiodic recurrent class.*

2.2 Membrane Systems

Membrane computing is a branch of Natural Computing, considered in October 2003 by Thomson Institute for Scientific Information (**ISI**) as a *Fast Emerging Research Front* in Computer Science [9]. It was initiated at the end of 1998 by Gh. Păun (by a paper circulated at that time on web and published in 2000 [6]). Since then it has received important attention from the scientific community. Details can be found at the web page `http://psystems.disco.unimib.it`, maintained in Milano under the auspices of the European Molecular Computing Consortium, EMCC.

In short, one abstracts computing models from the structure and the functioning of living cells, as well as from the organization of cell in tissues, organs,

and other higher order structures. The main components of such a model are a cell-like *membrane structure*, in the *compartments* of which one places *multisets of symbol-objects* which evolve in a synchronous maximally parallel manner according to given *evolution rules*, also associated with the membranes. The objects can also be described by strings, they can pass through membranes, can exit the system; in turn, membranes can be divided, dissolved, created.

A large variety of computing models, called P systems, were considered in this framework, based on the fundamental concept of biological membrane; the respective models are distributed (compartmentalized) parallel computing devices, processing multisets of abstract objects by means of various types of evolution rules. Parallelism, communication, non-determinism, synchronization, dynamic architecture of the model, etc. are central concepts of the theory, with biological, mathematical, and computer science sources of inspiration.

In this way, a comprehensive and systematic interdisciplinary research area was developed, of a high generality and versatility, where models can be devised for a large range of processes where compartmentalization and multiset processing are natural ingredients. Thus, although the initial goal of membrane computing was only to learn new ideas, tools, techniques from cell biology to the help of standard computers, much in the same way as, e.g., evolutionary computing suggests algorithms to be implemented on the electronic computer, the membrane computing became a new framework for building models for a large variety of processes, especially from biology (cell biology, tissues, populations of bacteria, controlling networks of complex phenomena, tumor growth, etc.), but also from linguistics, management, with several applications to computer science (computer graphics, approximative solutions to computationally hard problems, modeling parallel architectures, cryptography).

Most of these models were proven to be computationally universal, able to compute whatever a Turing machine can compute. In the case when an enhanced parallelism is available, by means of membrane division, string-object replication, or membrane creation, polynomial (often linear) time solutions to **NP**-complete problems were found.

In many variants, P systems are seen as devices of a *generative* nature, that is, from a given initial configuration several distinct computations may be developed, in a non–deterministic manner, producing different outputs.

In this paper we work with P systems with external output and performing *computing* tasks. For example, if a certain natural number, n, is encoded by the multiplicity of a special object in the initial configuration and we consider the cardinality of the multiset contained in the environment of a halting configuration as the result of a successful computation, then we can interpret that to mean that the system *computes* a partial function from natural numbers onto sets of natural numbers.

In the following, we assume that the reader is familiar with the basic notions of P systems, and we refer, for details, to [7].

3 Computing the Classification of the Steps of a Finite Markov Chain

3.1 Designing a P System

The goal of this paper is to obtain the classification of the states of a finite and homogeneous Markov chain within the framework of the cellular computing with membranes.

Let $P_k = (p_{ij})_{1\leq i,j\leq k}$ be a boolean matrix associated with a finite and homogeneous Markov chain of order k such that $p_{ij} = 0$ if the transition from e_i to e_j is not possible, and $p_{ij} = 1$ if the transition from e_i to e_j is possible; that is, P_k is the adjacency matrix of the directed graph associated with the Markov chain.

The solution presented in this paper is a *semi–uniform* solution to the problem of classification, in the following sense: we give a family $\mathbf{\Pi} = \{\Pi(P_k) : k \in \mathbf{N}\}$, associating with P_k a P system with external output, such that:

- There exists a deterministic Turing machine working in polynomial time which constructs the system $\Pi(P_k)$ from P_k.
- The output of the P system $\Pi(P_k)$ provides the classification of the k states of the Markov chain as well as the period of the states.

We associate with the matrix P_k a P system of degree 4 with external output,

$$\Pi(P_k) = (\Gamma(P_k), \mu(P_k), \mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4, R, \rho)$$

defined as follows:

- Working alphabet:

$$\begin{aligned}\Gamma(P_k) = &\{a_{ij},\ b_{ij},\ d_{ij},\ t_{ij} :\ 1 \leq i,j \leq k,\} \cup \{c_r :\ 0 \leq r \leq 2k+2\} \cup \\ &\{t_{ijur} :\ 1 \leq i,j,u \leq k,\ 0 \leq r \leq k\} \cup \{\beta_i :\ 0 \leq i \leq \alpha+1\} \cup \\ &\{s_{ijr} :\ 1 \leq i,j \leq k,\ 0 \leq r \leq k\} \cup \{A_{i1},\ \gamma_i :\ 1 \leq i \leq k\} \cup \\ &\{T_{ij},\ R_{ij} :\ 1 \leq i,j \leq k\}\end{aligned}$$

  where $\alpha = 2k + 4 + \lceil \lg_2 k \rceil + \frac{(k-1)(k+2)}{2}$.
- Membrane structure: $\mu(P_k) = [_1\ [_2\ [_3\ [_4\ ]_4\ ]_3\ ]_2\ ]_1$.
- Initial multisets:

$$\mathcal{M}_1 = \emptyset;\ \mathcal{M}_2 = \{\beta_0\};\ \mathcal{M}_3 = \{c_0\};$$
$$\mathcal{M}_4 = \{s_{ii0} \quad t_{ij}^{p_{ij}(k-1)} :\ 1 \leq i,j \leq k\}.$$

- The set R of evolution rules consists of the following rules:
  - Rules in the skin membrane labeled by 1:

$$r_1 = \{b_{ij}b_{ji} \to a_{ij}a_{ji} :\ 1 \leq i < j \leq k\}$$
$$r_2 = \{b_{ij} \to \gamma_i;\quad \gamma_i a_{ij} d_{ip} d_{jp} \to (T_{ip}T_{jp}, out) :\ 1 \leq i,j,p \leq k\}$$
$$r_3 = \{\gamma_i d_{ip} \to (T_{ip}, out) :\ 1 \leq i,j,p \leq k\}$$
$$r_4 = \{a_{ij}d_{ip} \to (R_{ip}, out) :\ 1 \leq i,j,p \leq k\}$$
$$r_5 = \{d_{i1} \to (A_{i1}, out) :\ 1 \leq i \leq k\}$$

- Rules in the membrane labeled by 2:
  $r_6 = \{b_{ij}^2 \to b_{ij} : \ 1 \leq i, j \leq k\} \cup \{\beta_i \to \beta_{i+1} : \ 0 \leq i \leq \alpha\} \cup \{\beta_{\alpha+1} \to \delta\}.$
  $r_7 = \{d_{ij}^2 \to d_{ij} : \ 1 \leq i, j \leq k\}$
  $r_8 = \{d_{ij} d_{i(j+l)} \to d_{ij} d_{il} : \ 1 \leq i \leq k, \ 2 \leq j + l \leq k\}$

- Rules in the membrane labeled by 3:
  $r_9 = \{t_{ijur} \to (t_{ij} s_{uj(r+1)}, in_4) \ b_{uj} : \ p_{ij} = 1, u \neq j, 1 \leq i, j, u \leq k, 0 \leq r < k\}$
  $r_{10} = \{t_{ijuk} \to (t_{ij}, in_4) \ b_{uj} : \ p_{ij} = 1, u \neq j, 1 \leq i, j, u \leq k\}$
  $r_{11} = \{t_{ijjr} \to (t_{ij}, in_4) \ d_{j(r+1)} : \ p_{ij} = 1, 1 \leq i, j \leq k, 0 \leq r < k\}$
  $r_{12} = \{t_{ijjk} \to (t_{ij}, in_4) : \ p_{ij} = 1, 1 \leq i, j \leq k\}$
  $r_{13} = \{c_r \to c_{r+1} : \ 0 \leq r \leq 2k + 1\} \ \cup \{c_{2k+2} \to \delta\}$
- Rules in the membrane labeled by 4:
  $r_{14} = \{s_{uir} t_{i1}^{p_{i1}} \ldots t_{ik}^{pik} \to (t_{i1ur}^{p_{i1}} \ldots t_{ikur}^{pik}, out) : \ 1 \leq u, i \leq k, \ 0 \leq r \leq k\}.$

– The partial order relation ρ over R consists of the following relations on the rules of R:
  - Priority relation in the skin membrane: $\{r_1 > r_2 > r_3 > r_4 > r_5\}$.
  - Priority relation in the membrane labeled by 2: $\{r_7 > r_8\}$.
  - Priority relation in the membranes labeled by 3: $\emptyset$.
  - Priority relation in the membranes 4: $\emptyset$.

Remark 1. Let us observe that the resources initially required for constructing the P system P_k are the following:

– Size of the alphabet: $\theta(k^4)$.
– Intitial number of membranes: 4.
– Number of rules: $O(k^4)$.
– Maximal length of a rule: $O(k)$.
– Number of priority relations: $O(k^6)$.

3.2 An Overview of Computations

At the beginning, the skin membrane is empty. The membrane labeled by 2 only contains the object β_0 which is a counter used to dissolve that membrane in the $(\alpha+2)$–th step, where $\alpha = 2k+4+\lceil lg_2 k \rceil + (k-1)(k+2)/2$. The membrane labeled by 3 contains the object c_0 which is a counter used to dissolve the membrane 2 in the $(2k + 3)$-th step. Initially, the membrane labeled by 4 contains: (a) objects s_{ii0} $(1 \leq i \leq k)$ encoding the states e_i of the chain; and (b) objects t_{ij} $(1 \leq i, j \leq k)$ encoding the elements p_{ij} of the boolean matrix associated to the transition matrix of the Markov chain.

In the first $2k + 3$ steps one applies rules only in the internal membranes labeled by 2, 3, and 4. During this (so called) first stage, we determine the accessibility between states (encoded by the objects b_{ij} meaning that we can

reach e_j from e_i) as well as the recurrent time of each state (encoded by the objects d_{ij} meaning that there exists a path from e_i to e_i with length j). In the even steps, the rules of membrane 4 will consume all the objects s_{uir} and some objects t_{ij}, sending to membrane 3 some objects t_{ijur}. In the odd steps, only rules in membrane 3 are applied (but not in membrane 4, because there do not exist objects s_{uir} in that membrane), sending new objects t_{ij} and objects $s_{uj(r+1)}$ (with $u \neq j$) to that membrane and producing objects b_{uj} and d_{jr} in membrane 3. The first stage finalizes in the configuration C_{2k+3} when the rule $c_{2k+3} \to \delta$ dissolves membrane 3. In this moment we have some objects t_{ij} (with $1 \leq i, j \leq k$) in membrane 4, objects d_{jr}, b_{uj} (with $1 \leq j, u, r \leq k$) and the object β_{2k+3} in membrane 2 (notice that in each step of this first stage the rule $\beta_i \to \beta_{i+1}$ of membrane 2 has been carried out). The skin region is empty.

The second stage begins with the execution of the $(2k+4)$–th step. During this stage we eliminate repeated copies of objects b_{ij} and d_{ij} in membrane 2, and we compute the period of each state (encoded in the second subscript of the objects d). The rules of membrane 2 permit transforming two copies of the object b_{ij} and d_{ij} into one copy, and the period of each state e_i is calculated by means of the rules of type r_8. For that, we need at most $\alpha = 2k+4+\lceil lg_2 k \rceil + (k-1)(k+2)/2$. steps. This stage finalizes when the rule $\beta_{\alpha+1} \to \delta$ dissolves membrane 2 in the $(\alpha+2)$–th step. This stage is a non-deterministic one.

Finally, the third stage is the output phase, and begins with the execution of the $(\alpha+3)$–step. In this stage the objects $b_{ij}b_{ji}$ are transformed into the objects $a_{ij}a_{ji}$ by means of the rule r_1 (meaning that the states e_i and e_j belongs to the same equivalence class). When this rule cannot be applied, then the transient objects are expelled to the environment applying the rules of types r_2 and r_3. After that, the rule r_4 sends the recurrent states and their period to the external environment. The process finalized when the rule r_5 sends the absorbent states.

3.3 Formal Verification

Given a computation $\mathcal{C}$ of the P system $\Pi(P_k)$, for each $m \in \mathbf{N}$ we denote by $\mathcal{C}_m$ the configuration of the system obtained after the execution of m steps. For each label $l \in \{1, 2, 3, 4\}$, we denote by $C_m(l)$ the multiset of objects contained in the membrane labeled by l in the configuration $\mathcal{C}_m$. Also, we denote by $C_m(env)$ the content of the environment of the system in the configuration $\mathcal{C}_m$.

First of all, we show that during the first stage the objects s_{ijr} codify the existence of a path from e_i to e_j with length r, and the objects t_{ijur} codify the existence of a path from e_u to e_j with length r and with e_i next to last node.

Lemma 1. *For each r such that $1 \leq r \leq k$ we have the following:*

(a) *If $r = 1$, then for each i, j such that $1 \leq i, j \leq k$, the object t_{iji0} belongs to $C_1(3)$ if and only if there exists a path from e_i to e_j with length 1 and with e_i being next to last node.*
If $r > 1$, then for each i, j, u such that $1 \leq i, j, u \leq k$, the object $t_{iju(r-1)}$ belongs to $C_{(2r-1)}(3)$ if and only if there exists a path from e_u to e_j with length r and with e_i being next to last node.

(*b*) *For each* i, j *such that* $1 \leq i, j \leq k$, $i \neq j$, *the object* s_{ijr} *belongs to* $C_{2r}(4)$ *if and only if there exists a path from* e_i *to* e_j *with length* r.

Proof. We prove the lemma by induction on r.

- Let us suppose that $r = 1$.
  (*a*) Let i, j be such that $1 \leq i, j \leq k$.
  If $t_{iju0} \in C_1(3)$, having in mind the composition of the initial configuration, there exists objects s_{ii0} and t_{ij} in $C_0(4)$. So, $p_{ij} = 1$ and (e_i, e_j) is an arc of the graph associated with the Markov chain. Hence, there exists a path from e_i to e_j with length 1 and with e_i next to last node.
  Conversely, if there exists a path from e_i to e_j with length 1 and with e_i next to last node, then $p_{ij} = 1$. So, the object t_{ij} belongs to $C_0(4)$. Having in mind that $s_{ii0} \in C_0(4)$, and applying the rules of type r_{14} we have $t_{iji0} \in C_1(3)$.
  (*b*) Let i, j be such that $1 \leq i, j \leq k$, $i \neq j$.
  Let us suppose that $s_{ij1} \in C_2(4)$. Then that object has been produced by an object t_{ijio} belongs to $C_1(3)$ and applying the rules of type r_9. From *a*) we deduce that there exists a path from e_i to e_j with length 1 (and with e_i next to last node).
  If there exists a path from e_i to e_j with length 1, then from *a*) we deduce that the object t_{iji0} belongs to $C_1(3)$. Applying the rule of type r_9 we obtain that $s_{ij1} \in C_2(4)$.
- Let $r \geq 1$ and $r < k$ and let us suppose that conditions (*a*) and (*b*) hold for r. Let us show that these conditions hold for $r + 1$.
  (*a*) Let i, j, u be such that $1 \leq i, j, u \leq k$.
  If the object t_{ijur} belongs to $C_{2r+1}(3)$, then in the $(2r + 1)$–th step the rules of type r_{14} has been applied in membrane 4, in order to produce the object t_{ijur}. Then, the objects s_{uir} and t_{ij} must belongs to $C_{2r}(4)$. By the induction hypothesis there exists a path from e_u to e_i of length r. Having in mind that $t_{ij} \in C_{2r}(4)$, it follows that (e_i, e_j) is an arc of the graph associated. Consequently there exists a path from e_u to e_j with length $r + 1$ with e_i next to last node.
  Let us suppose that there exists a path from e_u to e_j with length $r+1$ with e_i next to last node. Then there is a path from e_u to e_i of length r. By the induction hypothesis, the object s_{uir} belongs to $C_{2r}(4)$. Moreover, $p_{ij} = 1$ because (e_i, e_j) is an arc of the graph associated, so t_{ij} belongs to $C_{2r}(4)$. Applying the rules of type r_{14}, we have $t_{ijur} \in C_{2r+1}(3)$.
  (*b*) Let i, j be such that $1 \leq i, j \leq k$, $i \neq j$.
  If the object $s_{ij(r+1)}$ belongs to $C_{2r+2}(4)$, then there exists u $(1 \leq u \leq k)$ such that the object t_{ujir} belongs to $C_{2r+1}(3)$. By the induction hypothesis, there exists a path form e_i to e_j of length $r + 1$ with e_i next to last node. Then, there exists a path from e_i to e_j of length $r + 1$.
  Conversely, let us suppose that there exists a path from e_i to e_j of length $r + 1$. Let u be such that e_u is the next to last node of this path. By induction hypothesis, we have $t_{ujir} \in C_{2r+1}(3)$. Applying the rules of type r_9 we obtain that the object $s_{ij(r+1)}$ belongs to $C_{2r+2}(4)$. □

Lemma 2. *For each r such that $1 \leq r \leq k$ we have the following:*

(a) There are i, j, u such that $1 \leq i, j, u \leq k$, $t_{iju(r-1)} \in C_{2r-1}(3)$, $s_{ijr} \in C_{2r}(4)$.
(b) For all i, j, u, z such that $1 \leq i, j, u, z \leq k$, we have:

$$t_{ijuz} \notin C_{2r}(3),\ s_{ijz} \notin C_{2r-1}(4), t_{ij}^{p_{ij}(k-1)} \in C_{2r}(4)$$

Proof. By induction on r. First of all, recall that

$$C_0(4) = \{s_{ii0} t_{ij}^{p_{ij}(k-1)} :\ 1 \leq i, j \leq k\},\ C_0(3) = \{c_0\}.$$

Let i, j be such that $1 \leq i, j \leq k$ and $p_{ij} = 1$. Applying the rules of type r_{14} at the initial configuration we have $t_{iji0} \in C_1(3)$. Then, applying the rules of type r_9 in the second step we have $s_{ij1} \in C_2(4)$. Moreover, each object t_{ij} that has evolved in the first step, returns to membrane 4 in the next step. So, $t_{ij}^{p_{ij}(k-1)} \in C_2(4)$, for all i, j $(1 \leq i, j \leq k)$.

Having in mind that in the first step all objects s_{ii0} are consumed, we have $s_{ijz} \notin C_1(4)$, for all i, j, z $(1 \leq i, j, z \leq k)$ Hence, $t_{ijuz} \notin C_2(3)$, for all i, j, u, z $(1 \leq i, j, u, z \leq k)$.

Assuming the result holds for $r < k$ $(r \geq 1)$, we prove the result holds for $r+1$.

By the induction hypothesis, there exist i, j $(1 \leq i, j \leq k)$ such that $s_{ijr} \in C_{2r}(4)$. But there is u $(1 \leq u \leq k)$ such that $t_{uj} \in C_{2r}(4)$; applying the rules of type r_{14} we have we have $t_{ijur} \in C_{2r+1}(3)$. Then, applying the rules of type r_9 in the next step we have $s_{uj(r+1)} \in C_{2r+2}(4)$. Moreover, each object t_{ij} that has evolved in the r–th step, returns to membrane 4 in the next step. So, $t_{ij}^{p_{ij}(k-1)} \in C_{2r+2}(4)$, for all i, j $(1 \leq i, j \leq k)$.

Having in mind that in the r–th step all objects s_{ijr} which belong to $C_{2r}(4)$ have evolved, we have $s_{ijz} \notin C_{2r+1}(4)$, for all i, j, z $(1 \leq i, j, z \leq k)$ Hence, $t_{ijuz} \notin C_{2r+2}(3)$, for all i, j, u, z $(1 \leq i, j, u, z \leq k)$. □

Proposition 2. *For each i, j such that $1 \leq i, j \leq k$ we have the following:*

(1) If $i \neq j$, then the following assertions are equivalent:
  (a) There exists a path from e_i to e_j.
  (b) The object b_{ij} belongs to $C_{2k+2}(3)$.
  (c) The object b_{ij} belongs to $C_{2k+3}(2)$.
(2) The following conditions are equivalent
  (a) There exists a path from e_i to e_i with length j.
  (b) The object d_{ij} belongs to $C_{2k+2}(3)$.
  (c) The object d_{ij} belongs to $C_{2k+3}(2)$.

Proof. Let i, j be such that $1 \leq i, j \leq k$.

(1) Let $i \neq j$ and let us suppose that there exists a path from e_i to e_j. Let $r \geq 1$ be the length of that path r, and let e_u be the next to last node of that path. From Lemma 1, we have $t_{uji(r-1)} \in C_{2r-1}(3)$. Applying the rules of type r_9 or r_{10} we obtain that $b_{ij} \in C_{2r}(3)$. Hence $b_{ij} \in C_{2k+2}(3)$.

Conversely, let us suppose that $b_{ij} \in C_{2k+2}(3)$. Then, from Lemma 2 there exists r $(1 \geq r \leq k)$ such that $t_{uji(r-1)} \in C_{2r-1}(3)$. From Lemma 1 we deduce that there exists a path from e_i to e_j.
Obviously, $b_{ij} \in C_{2k+2}(3) \Longleftrightarrow b_{ij} \in C_{2k+3}(2)$.

(2) Let us suppose that there exists a path from e_i to e_i of length j. Then, there exists a state e_u and a path from e_i to e_u of length $j-1$, and with (e_u, e_i) being an arc of the associated graph. From Lemma 1, the object $t_{uii(j-1)}$ belongs to $C_{2j-1}(3)$. Applying the rules of type r_{11} or r_{12} we have $d_{ij} \in C_{2j}(3)$. Hence, $d_{ij} \in C_{2k+2}(3)$.

Conversely, let us suppose that $d_{ij} \in C_{2k+2}(3)$. Then, from Lemma 2 there exists r $(1 \geq r \leq k)$ such that $t_{uii(j-1)} \in C_{2r-1}(3)$. From Lemma 1 we deduce that there exists a path from e_i to e_i with length j.

Obviously, $d_{ij} \in C_{2k+2}(3)$ if and only if $d_{ij} \in C_{2k+3}(2)$. □

Proposition 3. *If* $\alpha = 2k + 4 + \lceil lg_2 k \rceil + (k-1)(k+2)/2$, *then:*

$$C_{\alpha+1}(2) = \{b_{ij} : 1 \leq i, j \leq k, i \neq j, \text{ there is a path from } e_i \text{ to } e_j\} \cup \{d_{ip} : 1 \leq i, p \leq k, \ p \text{ is the period of the state } e_i\} \cup \{\beta_{\alpha+1}\}.$$

$$C_{\alpha+2}(1) = \{b_{ij} : 1 \leq i, j \leq k, i \neq j, \text{ there is a path from } e_i \text{ to } e_j\} \cup \{d_{ip} : 1 \leq i, p \leq k, \ p \text{ is the period of the state } e_i\}.$$

Proof. Applying repeatedly the rules $\beta_i \longrightarrow \beta_{i+1}$ $(0 \leq i \leq \alpha)$ starting from the initial configuration, we have $\beta_{\alpha+1} \in C_{\alpha+1}(2)$.

From Proposition 2 we deduce that in membrane 2 of the configuration C_{2k+3} we have objects b_{ij}, with different multiplicities, such that there is a path from the state e_i to state e_j, and objects d_{ij}, with multiplicity 1, such that there is a path from the state e_i to state e_i with length j. Then, applying the rules of type r_6 in, at most, $\lceil \lg_2 k \rceil$ steps, we get that the multiplicity of each object is 1. Simultaneously, applying the rules of type r_7 and (8) in at most $\lceil \lg_2 k \rceil + (k-1)(k+2)/2$ steps we produce the objects d_{ip}, where p is the greatest common divisor of $\{d_{ij} : d_{ij} \in C_{2k+3}(2)\}$.

In the step $\alpha + 2$, membrane 2 is dissolved by executing the rule $\beta_{\alpha+1} \to \delta$. □

Theorem 3. *Let* C_f *be the final configuration of the computation* $\mathcal{C}$ *of the system* $\Pi(P_k)$. *Then:*

(*a*) *The state* e_i *is transient with period* p *if and only if* $T_{ip} \in C_f(env)$.
(*b*) *The state* e_i *is recurrent (and not absorbent) with period* p *if and only if* $R_{ip} \in C_f(env)$.
(*c*) *The state* e_i *is absorbent (with period* 1*) if and only if* $A_{i1} \in C_f(env)$.

Proof. (a) Let us suppose that e_i is a transient state. If the equivalence class of e_i has more than one element, then we can apply the rules of type r_1 in membrane 1 of the configuration $C_{\alpha+2}$ producing objects a_{ij} and a_{ji}. In this case, there exists r $(1 \leq r \leq k)$ such that the object b_{ir} belongs to $C_{\alpha+2}(1)$ but $b_{ri} \notin C_{\alpha+2}(1)$. So, in the $(\alpha+4)$–th step the object γ_i is produced applying the rules of type r_2. Then in the next step (and using the object d_{ip}) we obtain that $T_{ip} \in C_{\alpha+5}(env)$, where p is the period of e_i (from Proposition 3).

If the equivalence class of e_i is a singleton, then the rules of type r_1 cannot be applied in the configuration $C_{\alpha+2}$. So, we apply the rules of type r_2 producing

the object γ_i that in the next step produces (together with the object d_{ip}) the object T_{ip} in the environment (that is, $T_{ip} \in C_{\alpha+4}(env)$).

Reciprocally, let us suppose that $T_{ip} \in C_{\alpha+4}(env)$). Then, the object γ_i must be generated in order to can apply the rules of types r_2 and/or r_3. If only the rules of type r_2 are applied, then there are j, j $(1 \leq j, j \leq k)$ such that e_j is accessible from e_i and e_i is accessible from e_j, and e_i, $e_{j'}$ are communicating states. Hence, e_i is a transient state whose equivalence class has more than one object. If the rules of type r_3 are applied, then e_i is a transient state whose equivalence class is a singleton.

(b) Let us suppose that the state e_i is recurrent (and not absorbent) with period p. Then, the equivalence class of e_i has more than one object and there is no transient state belongs to that class. So, the rules of type r_1 will be applied in the configuration $C_{\alpha+2}$ and the object γ_i cannot be produced. Hence, applying the rules of type r_4 in the next configuration we have $R_{ip} \in C_{\alpha+4}(env)$.

Reciprocally, if $R_{ip} \in C_f(env)$ then the rule $a_{ij}d_{ip} \rightarrow (R_{ip}, out)$ has been applied (for some j, p with $1 \leq j, p \leq k$). For that, the objects a_{ij} and a_{ji} have been produced and the object γ_i has not been generated. Consequently, the state e_i is recurrent and its equivalence class has more than one object (that is, it is not an absorbent state).

(c) Let us suppose that the object e_i is absorbent (consequently its period is 1). In this case, it equivalence class is a singleton. So, there is no j $(1 \leq j \leq k)$ such that b_{ij} and b_{ji} belongs to $C_{\alpha+2}(1)$. Then the rules of type r_1 are not applicable for i. Applying the rule $d_{i1} \rightarrow (A_{i1}, out)$ we obtain that $A_{i1} \in C_{\alpha+3}(env)$.

Reciprocally, if $A_{i1} \in C_f(env)$, then the object d_{i1} belongs to membrane 1 in the next to last configuration. So, the objects a_{ij} has not been produced. Then, the state is recurrent and its equivalence class has only one object. □

Remark 2. Let us note that the number of steps of the computation of the P system P_k is either $\alpha + 3$ or $\alpha + 4$. That is, the number of steps is quadratic in the number of states of the Markov chain.

4 Conclusions

One of the central issues in Markov chain theory is the asymptotic long–term behavior of Markov chains.

Due to different results concerning the existence (and the uniqueness) of a stationary distribution, the problem of classification of states is an important one in the mathematical study of Markov chains and related stochastic processes.

In this paper we give an efficient (*semi–uniform*) solution of the problem of classification in the framework of the cellular computing with membranes. The solution is semi–uniform because for each adjacency matrix of the directed graph associated with a Markov chain, a specific P system with external output is designed. The solution is efficient, because it is quadratic in the number of states of the Markov chain. Furthermore, the amount of resources initially required to construct the system is polynomial in the order of the Markov chain.

The paper also provides a new example of formal verification of P systems designed to solve a problem (in this case a problem of classification, not a decision problem), following a specific methodology. These examples are always interesting, for instance, in order to find systematic processes of formal verification in a model of computation oriented to machines, like the cellular model, a case where it is well known that the mechanisms of verification are often a very hard task.

Acknowledgement

The third author wishes to acknowledge the support of the project TIN2005-09345-C04-01 of the Ministerio de Educación y Ciencia of Spain, co–financed by FEDER funds, and of the Project of Excelence TIC 581 of the Junta de Andalucia.

References

1. M. Cardona, M.A. Colomer, J. Miró, A. Zaragoza, A step towards DNA computation model. Submitted, 2006.
2. M. Cardona, M.A. Colomer, M.J. Pérez–Jiménez, A. Zaragoza, Handling Markov chains with membrane computing. *Lecture Notes in Computer Science*, 4135 (2006), 72–85.
3. O. Häggström, *Finite Markov Chains and Algorithmic Applications*. London Mathematical Society, Cambridge University Press, 2003.
4. R. Nelson, *Probability, Stochastic Processes, and Queueing Theory: The Mathematics of Computer Performance Modeling*. Springer-Verlag, New York, 1995.
5. A.N. Shiryayev. *Probability*. GTM 95, Springer, 1984.
6. Gh. Păun, Computing with membranes. *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143, and *Turku Center for Computer Science-TUCS Report* Nr. 208, 1998.
7. Gh. Păun, *Membrane Computing. An Introduction*. Springer-Verlag, Berlin, 2002.
8. Gh. Păun, G. Rozenberg, A guide to membrane computing. *Theoretical Computer Science*, 287 (2002), 73–100.
9. ISI web page: `http://esi-topics.com/erf/october2003.html`

Capítulo 8

Conclusiones

Los artículos con los que se defiende esta tesis se sitúan en el campo de la computación natural y más concretamente se centran en dos de las áreas que están teniendo más auge: la computación molecular con ADN y la computación celular con membranas. Estos nuevos modelos de computación aparecen con la finalidad de resolver las limitaciones que presentan los ordenadores convencionales tanto en tiempo como en espacio. Una de las ventajas de estos modelos con respecto los clásicos es el paralelismo masivo que se implementa de forma natural y que permite ejecutar de manera simultánea una gran cantidad de operaciones en una unidad de tiempo.

Esta tesis pretende ser una contribución al estudio de un tipo de procesos estocásticos caracterizados por su evolución a través de unos determinados valores, los estados del proceso y cuya evolución en el futuro solamente depende del presente, siendo irrelevante el pasado: las cadenas de Markov. Más concretamente, y debido a que los programas máquina trabajan en unidades discretas, se consideran las cadenas finitas de Markov.

El primer trabajo [11] ofrece un método alternativo para el cálculo de la potencia n-ésima de la matriz de transición de una cadena finita de Markov basado en computación con ADN sin incrementar la complejidad de los algoritmos convencionales ya existentes. Este tipo de computación permite trabajar en paralelo reduciendo de manera considerable el tiempo de cálculo si bien la cuestión planteada actualmente es la factibilidad. Esta factibilidad queda demostrada para el caso concreto de cadenas de Markov por la parte experimental presentada en [14] que corresponde al segundo artículo presentado en la tesis.

Esta segunda contribución plantea la necesidad de adecuar la forma de pensar a la computación con ADN. Se debe evitar la traducción de los algoritmos clásicos al lenguaje del ADN al igual que intentar buscar nuevas propiedades matemáticas basadas en existencia para resolver los problemas. En el segundo trabajo se propone un algoritmo de clasificación de los estados de una cadena de Markov eficiente en la computación molecular con ADN y cuya aplicación en computación convencional no tiene sentido por su complejidad.

En la tercera y cuarta contribución se abordan los mismos problemas que en las primeras pero desde otra perspectiva, la computación celular con membranas, P systemas. La computación con membranas ofrece soluciones deterministas de los problemas mientras que en computación con ADN las soluciones son aleatorias. En [12] se tiene el valor exacto de la potencia n-ésima de la matriz de transición de una cadena finita de Markov mientras que en la contribución [11] se obtenía una aproximación.

En las aportaciones hechas con ADN se acaba calculando el período de los estados de la cadena de Markov utilizando un cálculo matemático. En el cuarto de los artículos presentados, [13], se diseña un P systema para calcular dicho periodo.

De las dos últimas contribuciones cabe destacar que proporcionan una nueva manera de verificación formal en modelos de computación orientados a máquina siguiendo una metodología específica.

Para finalizar, resaltar que en esta tesis no se ha resuelto ningún problema computacionalmente difícil, siendo esto uno de los objetivos principales de la computación natural, si bien se ha realizado una importante aportación a la resolución de problemas tradicionales desde otra perspectiva como es la computación natural. En particular se han presentado cuatro trabajos en los que se muestra como modelizar el comportamiento asintótico de una cadena finita de Markov utilizando computación natural.

Bibliografía

[1] L. Adleman. *Molecular computation of solutions to combinatorial problems.* Science 266, 1021–1024, 1994.

[2] M. Amos. *DNA Computation.* Ph.D.Thesis. University of Warwick, September 1997.

[3] M. Amos. *Theoretical and Experimental DNA Computation.* The Natural Computing Column, 125-138. 20, 1999.

[4] M. Amos, Gh. Păun, G.Rozenberg, A. Salomaa. *Topics in theory of DNA computing.* Theoretical Computer Science. Elsevier Science B.V., 2002.

[5] T.W. Anderson and L.A. Goodman. *Statistical inference about Markov Chains.* Ann. Math. Stat. 28, 89-110, 1957.

[6] R. Barau, J. Misra. *Binay Arithmetic for DNA Computers.* DNA8, LNCS 2568, 124-132, Springer-Verlag, 2003.

[7] D. Beaver. *A Universal Molecular Computer.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 27, 29-36, 1996.

[8] D. Boneh, C. Dunworth, and R.J. Lipton. *On the computational power of DNA.* Discrete Appl. Math. 71 (1-3), 79–94, 1996. *Science* 266, 1021–1024, 1994.

[9] D. Boneh, C. Dunworth, and R.J. Lipton. *Breaking DES using a molecular computer.* Princeton CS Tech-Report CS-TR-489, 1995.

[10] N. Bouleau. *Processus stochastiques et applications.* Herman, 1988.

[11] M. Cardona, M.A. Colomer, J. Conde, J.M. Miret, J. Miró, A. Zaragoza. *Markov chains: Computing limit existence and approximations with DNA.* BioSystems 81 , 261-266, 2005.

[12] M. Cardona, M.A. Colomer, M.J. Pérez-Jiménez, A. Zaragoza. *Handling Markov Chains with Membrane Computing*. UC 2006, LNCS 4135 72-85. Springer-Verlag 2006.

[13] M. Cardona, M.A. Colomer, M.J. Pérez-Jiménez, A. Zaragoza. *Classifying States of a Finite Markov Chains with Membrane Computing*. WMC 2006, LNCS 4361 266-278 Springer-Verlag 2006.

[14] M. Cardona, M.A. Colomer, J. Miró, A. Zaragoza. *A step towards DNA computation model*. Submitted, 2006.

[15] G. Ciobano, D Paraschiv. *P-System Software Simulator*. Fundamenta Informaticae 49(1-3), 67-80, 2002.

[16] D. Díaz-Pernil, M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez. *Solving 3-COL with Tissue P Systems*. Fourth Brainstorming Week on Membrane Computing, Sevilla, vol II, 18-29, 2006.

[17] W. Feldkamp, U. Bonzhaf, H. Raube. *A DNA sequence computer*. In: DNA6, Sixth International Meeting on DNA Based Computers. LNCS, vol 2054. Springer-Verlag, 2001.

[18] W.Feller. *Introducción a la teoría de las probabilidades y sus aplicaciones*. Limusa, Mexico, 1998.

[19] B.L. Fox and D. M. Landi. *An Algorithm for Identifying the Ergodic Subchains and Transient States of a Stochastic Matrix*. Comunications of the ACM 11, 619-621, 1968.

[20] F. Guarnieri, M. Fliss, C. Bancroft. *Making DNA Add*. Science, vol. 273, 1996.

[21] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez. *Available Membrane Computing Programs*. Applications of Membrane Computing-Berlin, Alemania. Springer-Verlag, 411-436, 2006.

[22] M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez. *Fractals and P Systems*. Fourth Brainstorming Week on Membrana Computing, Sevilla, vo.II, 65-86, 2006.

[23] O. Häggström, *Finite Markov chains and algorithmic applications*, London Mathematical Society, Cambridge University Press, 2002.

[24] L. Kari. *DNA computing: arrival of biological mathematics*. The Matematical Intelligencer, vol.19, n.2, Spring 9-22, 1997.

[25] J.G. Kemeny and J.L. Snell. *Finite Markov Chains* Springer-Verlag, 1960.

[26] P. Lancaster and K. Tismenetsky. *The Theory of Matrices with Applications*. Academic Press, second edition 9,11,12,13,34, 1985.

[27] A. Leier, C. Richter, W. Banzhaf, H. Rauhe. *Cryptography with DNA binary strands*. BioSystems 57, 13-22, 2000.

[28] A. Leparoti, C. Zandron. *A Family of P Systems Which Solve 3-SAT*. Cellular Computing. ESF PESC Exploratory Workshop, Sevilla, 247-256, 2005.

[29] R. Lipton. *DNA solution of hard computational problems*. Science 268,542-545, 1995.

[30] C. Maley. *DNA Computation: Theory, Practice, and Prospects*. Evolutionay Computation, vol.6, n.3, 201-229, 1998.

[31] V. Manca. *DNA and Membrane Algorithms for SAT*. Fundamenta Informaticae XX, 1-17, 2005.

[32] S. Marcus. *Membrane versus DNA*. Fundamenta Informaticae 49 (1), 223-227, 2002.

[33] R. Nelson. *Probability, stochastic processes, and queueing theory: the mathematics of computer performance modeling*. Springer-Verlag, New York, 1995.

[34] J.R. Norris. *Markov Cahins*. Cambridge University press. United Kingdom, 1997.

[35] M. Ogihara and A. Ray. Executing parallel logical operations with DNA. *Proceedings of the Congress on Evolutionary Computation*, pages 972 – 979, IEEE Computer Society Press, Piscataway, NJ, 1999.

[36] E. Parzen. *Procesos Estocásticos*. Paraninfo, Madrid, 41, 1972.

[37] A. Păun. *P Systems with String-Objects: Universality Results*. Pre-Proceedings of WMC01, 2001.

[38] Gh. Păun, G. Rozenberg and A. Salomaa. *DNA Computation*. EATCS Series, Springer-Verlag, 1998.

[39] Gh. Păun. *Computing with Membranes*. Journal of Computer and System Sciences, 61,1,108-143, 2000 and Turku Center for Computer Science-TUCS Report N°208, 1998.

[40] Gh. Păun. *From cells to computers: computing with membranes (P systems)* Biosystems 59, 139-158, 2001.

[41] Gh. Păun, G. Rozenberg. *A guide to membrane computing*. Theoretical Computer Science, 287, 73-100, 2002.

[42] Gh. Păun. *Membrane Computing. An Introduction*. Springer-Verlag, Berlin, 2002.

[43] Gh. Păun. *Membrane Computing: Some Non-Standard Ideas*. LNCS 2950, Aspects of Molecular Computing, 322-337, 2004.

[44] M. J. Pérez Jiménez, F. Sancho Caparrini. *Computación celular con membranas: Un modelo no convencional*. Ed. KRONOS, 2002.

[45] M. J. Pérez Jiménez, A. Riscos Núñez, D. Díaz Pernil, J.A. Nepomuceno Chamorro, A.j. Ramos Espina. *Modelos de Computación Molecular, Celular y Cuántica*. Sevilla, España. Fénix Editora, 2004.

[46] N. Pisanti. *A survey on DNA computing*. Bulletin of the EATCS 64, 171-187, 1998.

[47] A. N. Shiryayev. *Probability*. GTM 95, Springer, 1984.

[48] S. E. Tsirka, M. A. Frohman. *Polymerase Chain Reaction (PCR): Specialized Reactions*. Encyclopedia of life sciences, 2001.

[49] S. Wang, A. Yang. *DNA Solution of Integer Linear Programming.*Applied mathematics and computation (Appl. math. comput.), vol 170, n^{a}1, 626-632, 2005.

[50] J.D. Watson, F.H.C. Crick. *Molecular structure of nucleic acids: A structure for deoxyribose nucleic acid*. Nature 171, 737-738, 1953.

[51] P. Wilhelm, K. Rothermund. *A DNA and restriction enzyme implementation of Turing Machines*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 27, 75-119,1996.

[52] E. Winfree. *Design and self-assembly of two-dimensional DNA crystals.* Ph.D. Thesis. California Institute of Technology, June 1998.

[53] http://www.arrakis.es/ lluengo/membrana.html.

[54] ISI web page: http:// esi-topics.com/erf/october 2003. html.

[55] http://www.vc.ehu.es/campus/centros/farmacia/deptos.f/depme/gracia 1.html.