Universidad de Sevilla
Dpt. de Ciencias de la Computación
e Inteligencia Artificial

Modelización y simulación de fenómenos de la vida real en Computación Celular con Membranas

Una tesis presentada para el grado de
Doctor en Filosofía
Escuela Técnica Superior de Ingeniería Informática
Universidad of Sevilla

Manuel García–Quismondo Fernández

Aprobación de los Directores de la Tesis

Mario de Jesús Pérez Jiménez

Miguel Angel Martínez del Amor

11 de octubre de 2013

A mis padres y a mi abuela,
por ayudarme siempre y sacarme adelante.

Agradecimientos

Esta tesis es el culmen de seis años de trabajo. Un trabajo que, aunque firmado por sólo una persona, en absoluto es individual, y sin cuya ayuda y apoyo nunca hubiera podido realizar. Es por eso que quiero dedicar esta página a expresar my gratitud a todos los que hicieron posible este trabajo.

Me gustaría comenzar por mis padres, Juanjo y Margari, por su ayuda desde mi nacimiento, así como a mi abuela Antonia, mi hermano Juan, mis tíos Yeye y Pedro y mis primos Pedro e Irene, por acogerme como "hijo prestado" durante incontables periodos de tiempo.

Así mismo, quiero expresar my agradecimiento a todos mis compañeros en el Departamento de Ciencias de la Computación de la Universidad de Sevilla. Especialmente, mi agradecimiento va hacia Mario de Jesús Pérez por su acogida bajo su tutela a a hora de completar mi tesis doctoral. Su generosidad y capacidad de sacrificio por el prójimo ha sido siempre una fuente de inspiración para mí. Seguidamente, me gustaría agradecer a Ignacio Pérez y a Daniel Díaz por haberme dado la oportunidad de incorporarme al mundo de la investigación. A Fran Romero, por tutelarme durante mi primera etapa como investigador "remunerado", a Agustín Riscos y a Ana Ruiz, por su inestimable ayuda en temas docentes y burocráticos, a Miguel Angel Gutiérrez y a Fernando Sancho por sus anécdotas, a Carmen Graciani por su experiencia durante mi etapa como docente y a Alvaro Romero por su inestimable ayuda en temas de administración de sistemas en general. Asimismo, no podía faltar mi codirector de tesis Miguel Angel Martínez por enseñarme todo lo que sé sobre paralelismo, administración de servidores Linux, dispositivos GPU, lenguaje de programación CUDA y cocina con arroz bomba. Y, cómo no, mi más sincero agradecimiento a Luis Valencia y Luisfe Macías por compartir y animar tanísimas horas de despacho. Por último, nada más que extender mi agradecimiento a todo el Departamento de Ciencias de la Computación e Inteligencia Artificial.

También querría agradecer este trabajo a mis compañeros de carrera, en especial a José Manuel Pérez, Juan Sebastián Tinoco, David Fernández y Carlos Portero, por tantas horas de trabajos grupales compartidas. *I would like also to thank all people who fostered me during my research stays all over the world. Specifically, I would like to thank Martyn Amos, Angel Goñi, Javi Esquivel, Pete Harding, Matt Crossley and Andy Nisbet for taking me in and sharing with me their experience during my stay at the Novel Computation Group in Man-*

chester Metropolitan University, in Manchester (U.K.). In addition, I would like to convey my most sincere acknowledge to my companions at the Image Processing and Intelligent Control Key Laboratory in Wuhan, Hubei (China), especially to Linqiang Pan, Tao Song, Xueming Liu, Yansen Su, Yuan Kong and Qing Wei, and to Gexiang Zhang for his homely reception in Chengdu, Sichuan (China). Last but not least, I would like to extend my acknowledgement to all who fostered me as another member in the Department of Biology at Tufts University in Medford, Massachusetts (U.S.A.), with special regards to Francie Chew, Michael Reed, Coco Gómez, Kelly Boisvert (who is finishing off her poster beside me while I write this acknowledgement) and Anne Madden. Imi-as place asemenea sa da un mesaj de mulţimire la Ana Pavel pentru totul ajuta şi sprijini primita.

Finalmente, agradezco profundamente todo el apoyo económico ofrecido por la beca de Formación de Profesorado Universitario (FPU) del Ministerio de Educación, la beca de Formación de Personal Investigador (FPI) del IV Plan Propio de la Universidad de Sevilla, de la Junta de Andalucía, y por los proyectos de investigación nacionales I+D+i del Ministerio de Economía y Competitividad TIN2006-13425, TIN2009-13192 y TIN2012-37434; todos ellos confinanciados por los fondos FEDER de la Unión Europea.

Índice general

1 Antecedentes

La Computación Natural es una terminología introducida para englobar tres tipos de métodos: (1) aquellos inspirados en la Naturaleza para el desarrollo de nuevas técnicas de resolución de problemas; (2) aquellos basados en el uso de ordenadores para sintetizar fenómenos de la Naturaleza; y (3) aquellos que utilizan materiales de la Naturaleza (como, por ejemplo, biomoléculas) para realizar cálculos [15]. Los paradigmas dentro de esta disciplina son las Redes Neuronales Artificiales [33], los Algoritmos Genéticos y la Computación Evolutiva [57], La Inteligencia Enjambre [58], los Sistemas Inmunes Artificiales [38] y la Computación basada en ADN [10], entre otras.

La Computación Celular con Membranas es una rama de la Computación Natural inspirada en la Naturaleza e iniciada por Gheorghe Păun que abstrae modelos computaciones de la estructura y el funcionamiento de las células vivas y de la organización de estas células en tejidos u otras estructuras de orden superior [41]. Esto se hace definiendo dispositivos teóricos conocidos como sistemas de membranas o *sistemas P*. Aunque el model fundacional (también conocido como *Sistemas P de Transición*) define una estructura de membranas que consiste en una organización jerárquica de membranas [44], varios marcos de modelización dentro de la Computación Celular con Membranas han sido presentadas desde su fundación. Entre estos modelos pueden citarse los *Sistemas P Neuronales de Impulsos* [40], los *Sistemas P de Tejidos* [39], *Los Sistemas P Numéricos* [45] y los *Sistemas P de Rescritura de Vectores* [11]. La mayoría

de los marcos de modelización basados en sistemas P definen un alfabeto de símbolos conocidos como *objectos*, una estructura de membranas compuesta de compartimentos separados o *membranas*, un *multiconjunto* asociado a cada membrana y un conjunto de reglas de reescritura por cada membrana que definen la evolución de un sistema mediante pasos de tiempo discretos.

El tiempo en los sistemas P es discreto y avanza en una serie de pasos finitos [44]. Además, se asume un reloj global que sincroniza todo el sistema [46]. En este sentido, la descripción instantánea en un instante t de un sistema P se conoce como *configuración*. Dada una configuración, el multiconjunto de reglas que define su transición hacia una nueva configuración se conoce como *paso de transición*. Del mismo modo, la secuencia de configuraciones de un sistema P para una posible secuencia de pasos de transición se conoce como *computación* [44].

La mayoría de los marcos basados en sistemas P son *no deterministas* y *maximalmente paralelos*. En términos informales, la propiedad de **Paralelismo Maximal** puede definirse del siguiente modo. Considérese un sistema P definido según un marco de Computación Celular con Membranas. Se dice que el marco es **Maximalmente Paralelo** si, por cada paso de transición del sistema P, cada regla que pueda ser aplicada *debe* set aplicada. Esto es, para cada paso de transición, si una regla es capaz de consumir un multiconjunto de objetos que no ha sido consumido por otras reglas, entonces esa regla debe consumirlo. Del mismo modo, en un paso de transición, es posible que en el sistema P existan diferentes maneras en las que los objectos de la membrana puedan ser distribuidas entre las reglas. Se dice que el marco es **no determinista** si y sólo si todas las posibles aplicaciones maximalmente paralelas son susceptibles de ser seleccionados de este rango de posibilidades [37].

La Computación Celular con Membranas ha sido principalmente aplicada como un enfoque alternativo para tratar problemas computacionalmente duros, sobretodo problemas NP completos. Los sistemas P son capaces de resolver problemas NP completos en tiempo polinomial cambiando tiempo por espacio. Es decir, cuando la Computación Celular con Membranas se usa como alternativa a enfoques tradicionales para la resolución de problemas de esta clase de computación, se crean un número exponencial de dispositivos computacionales de manera que cada uno trata una instancia o escenario del problema a resolver. Como ejemplo, pueden citarse problemas como el problema SAT [28], el problema 3–Col [20] y el problema de la mochila [43].

Otra faceta de los sistemas P es su utilidad como un marco de modelización computacional para fenómenos de la vida real. Cuando se estudian estos pro-

blemas, construir modelos suele ser útil. El grado de abstracción es un aspecto crucial para el diseño de un modelo capaz de capturar la dinámica de un proceso. Es decir, los modelos deben mantenerse lo más simple posible, eliminando toda la complejidad innecesaria y definiendo criterios para seleccionar qué información se integra en el modelo [24]. En este sentido, los sistemas P se revelan como abstracciones formales del fenómeno bajo estudio, filtrando aspectos de menor importancia e incluyendo aquellos que demuestran ser relevantes mediante el proceso iterativo de modelización, simulación y validación de manera modular, es decir, de tal modo que el cambio, eliminación e inclusión de pequeñas porciones de conocimiento en el sistema implica cambios en el modelo proporcionalmente pequeños [47]. Los fenómenos modelizados son bastante diversos, abarcando áreas como la bioquímica [19, 47, 48] y el procesamiento de imágenes digitales [54] e incluyendo robótica [52], economía [45], software [20], generación automática de música [18] y ecología [13].

Dado un proceso de la vida real a estudiar y su correspondiente modelo, para verificar que el comportamiento del modelo corresponde al del sistema suele ser necesario *simular* el sistema. Un enfoque alternativo consistiría en *implementar* los modelos. Sin embargo, en el caso de la Computación Celular con Membranas esto no siempre es posible, principalmente debido a la creación exponencial de dispositivos computacionales que, por el momento, no ha sido implementado con éxito. Tradicionalmente, los simuladores en Computación Celular con Membranas han sido completamente aplicaciones *ad–hoc* para el modelo a simular, los parámetros del simulador se introducían a mano en el código, que no era reutilizable porque se diseñaba para *ex–profeso* para el sistema P a simular [49] [14]. En este sentido, P–Lingua fue una aplicación software pionera en la estandarización de sistemas P implementando un software libre y basado en plugins dispuesta para su extensión mediante desarrolladores externos. P–Lingua es una herramienta software que proporciona un lenguaje de especificación en el que diseñadores pueden definir sistemas P. Este lenguaje puede extenderse fácilmente cuando sea necesario. Además, también incporpora un conjunto de simuladores en lenguaje Java [1], de manera que los usuarios pueden seleccionar qué simulador de entre aquellos incluidos en la herramienta funciona mejor para su caso. Además, P–Lingua implementa un mecanismo de extensión en el que nuevos formatos y simuladores pueden incluirse en el software [17].

Originalmente, los simuladores desarrollados para modelos de Computación Celular con Membranas fueron implementados exclusivamente en arquitecturas secuenciales, tales como ordenadores personales, ocasionalmente usando lenguajes declarativos como Prolog [14]. Sin embargo, las limitaciones inhe-

rentes a estos dispositivos no concuerdan bien con la naturaleza paralela de los sistemas P, de manera que la búsqueda de nuevas plataformas cobra importancia. En este sentido, es natural recurrir a arquitecturas paralelas como placas FPGA boards [53] [34] [37], granjas de computadores [12], microcontroladores [27] [26] y tarjetas gráficas [9] [35] [5] [30].

El principal objetivo de esta tesis es el desarrollo de modelos de Computación Celular con Membranas y de simuladores para fenómenos de la vida real. Específicamente, se tratan tres aplicaciones distintas: la Robótica Bioinspirada, las Redes Reguladoras de Genes y los Ecosistemas. La variedad de las aplicaciones demuestra la versatilidad de la Computación Celular con Membranas como un marco de modelización computacional. También se han desarrollado en el transcurso de esta tesis simuladores secuenciales y paralelos para simular los modelos definidos en estos marcos. Para ello, se ha extendido el lenguaje P–Lingua language [17], incluyendo nuevas características y, en el caso de los ecosistemas, nuevos modelos desde cero. Los simuladores secuenciales han sido desarrollados en C++ y Java, siendo estos últimos incorporadas en la API P–Lingua. Los simuladores paralelos han sido implementados utilizando CUDA [2], un lenguaje de programación para computación paralela en GPUs que ya ha sido previamente aplicado con éxito para simular sistemas P [9] [8] [6] [7, 16].

2

Sistemas P Enzimáticos Numéricos

En este capítulo se introduce un simulador basado en tecnología GPU para sistemas P Enzimáticos Numéricos (ENPSs), un marco de modelización en Computación Celular con Membranas diseñado para reproducir el comportamiento de sistemas robóticos que 1) se componen de elementos modulares, cada uno con una función específica 2) son capaces con interactuar entre ellos y con el entorno para conseguir objetivos.

2.1. Formalización de los Sistemas P Enzimáticos Numéricos

Un sistema P Enzimático Numérico de grado $m \geq 1$ es una tupla

$$\Pi = (H, \mu, (Var_1, Pr_1, Var_1(0)) \dots (Var_m, Pr_m, Var_m(0)))$$

donde:

- H, μ y $(Var_1, Var_1(0)) \dots (Var_m, Var_m(0))$
- H es un **alfabeto** con m símbolos usados como etiquetas de las m membranas del sistema. Los elementos de H son las etiquetas de las membranas de Π.

- μ es una **estructura de membranas**, un árbol enraizado con m membranas.
- $Var_i = \{x_{1,i} \dots x_{k_i,i}\}$ es el conjunto finito de **variables** asociado al compartimento i, $1 \leq i \leq m$.
- $Var_i(0) = (\lambda_{1,i} \dots \lambda_{k_i,i})$ son valores numéricos (*números reales*) para las variables de Var_i. Estos valores son los valores iniciales de las variables; en el instante 0 de la evolución del sistema tenemos $x_{j,i} = \lambda_{j,i},\ 1 \leq i \leq m, 1 \leq j \leq k_i$.
- Pr_i es el conjunto de programas asociados a la membrana i. Cada l-ésimo programa en el conjunto Pr_i puede tener una de las siguientes formas:
  - $Pr_{l,i} = (F_{l,i}(x_{1,i}, \dots, x_{k_i,i}), c_{l,1}|v_1 + \dots + c_{l,n_i}|v_{n_i})$
  - $Pr_{l,i} = (F_{l,i}(x_{1,i}, \dots, x_{k_i,i}), (e_{l,i} \rightarrow), c_{l,1}|v_1 + \dots + c_{l,n_i}|v_{n_i})$

  En ambas formas, todos los valores tienen el mismo significado, siendo $e_{l,i}$ una variable de Var_i. Esta variable se conoce como la variable enzimática asociada con $Pr_{l,i}$ y su valor no puede ser consumido por el programa.

Para este modelo, se presenta un simulador paralelo basado en tecnología GPU.

2.2. Simulación paralela de ENPSs

Antes de describir el simulador, se introducen las estructuras de datos que se utilizan en el mismo. Como es habitual en computación GPU [9], los datos que representan el modelo a simular se almacenan en vectores, los cuales se describen a continuación:

V **(Variables):** un vector de números reales de dimensión n, donde $n \geq 1$ es el número total de variables del sistema. Cada elemento $V_i,\ 1 \leq i \leq n$, representa una variable del sistema.

PNT **(Tipos de Nodos de Producción):** un vector de caracteres de dimensión o, donde $o \geq 1$ es el número total de constantes, variables y operadores en todas las funciones de producción del sistema. Cada elemento $PNT_j,\ 1 \leq j \leq o$, codifica uno de los siguientes valores constantes: **CONSTANT**, **VARIABLE** o uno de los siguientes: +, **-**, *****, / y ^.

PNV **(Production Node Values):** un vector de números reales de dimensión o. Cada elemento $PNV_j,\ 1 \leq j \leq o$, codifica uno de los siguientes valores, en función del valor de PNV_j:

- Si $PNT_j = CONSTANT$, entonces PNV_j codifica una constante real.
- Si $PNT_j = VARIABLE$, entonces $PNV_j \leq n$ representa una variable en el vector *variables*.

En otro caso, PNV_j no tiene ningún significado particular.

$PNLO$ **(Production Node Left Operands):** un vector de números reales de dimensión o. Si PNT_j es **CONSTANT** o **VARIABLE**, entonces $PNLO_j,\ 1 \leq j \leq o$, no tiene ningún significado en particular. En otro caso, $PNLO_j$ almacena el operando izquierdo de la operación representada por el índice j.

$PNRO$ **(Production Node Right Operands):** un vector de números reales de dimensión o. Si PNT_j es **CONSTANT** o **VARIABLE**, entonces $PNRO_j,\ 1 \leq j \leq o$, no tiene ningún significado en particular. En otro caso, $PNRO_j$ almacena el operando izquierdo de la operación representada por el índice j.

RPV **(Repartition Protocol Variables):** una matriz de números enteros de dimensión $s \times q$, donde $s \geq 1$ es el número total de programas del sistema y $q \geq 1$ es el máximo número de pares $(constant, variable)$ en todos los protocolos de repartición del sistema. Cada elemento $RPV_{l,k} \leq n,\ 1 \leq l \leq s, 1 \leq k \leq q$, representa una variable del sistema. Si el protocolo de repartición del programa l de la membrana k tiene menos de q pares, entonces $RPV_{l,k} = -1$

RPC **(Repartition Protocol Constants):** una matriz de números enteros de dimensión $s \times q$. Cada elemento $RPC_{l,k},\ 1 \leq l \leq s, 1 \leq k \leq q$, representa una variable del sistema. Si el protocolo de repartición del programa l de la membrana k tiene menos de q pares, entonces $RPC_{l,k} = 0$

E **(Enzymes):** un vector de números enteros de dimensión s. Cada elemento $E_l,\ 1 \leq l \leq s$ representa la enzima del programa l. Si F_l es de la forma $(F_l(x_{1,u}, \ldots, x_{k_u,u}), c_{l,1}|v_1 + \ldots + c_{l,n_u}|v_{n_u})$, donde u es la membrana asociada con la función de producción del programa l, entonces $e_l = -1$.

Además de las estructuras usadas para representar el sistema, otros vectores se usan para almacenar datos temporales necesarios para las simulaciones. Estos vectores son:

PFR **(Production Function Results):** un vector de números reales de dimensión s que almacena los resultados de los cálculos de las funciones de producción calculadas.

AP **(Applicable Program):** un vector de caracteres de dimensión s que almacena los marcadores que indican si un programa se aplicará en el paso actual de computación. Estos marcadores pueden estar *Activos* (ciertos) o *Inactivos* (falso).

El algoritmo 2.2.1 describe cómo simular sistemas P numéricos enzimáticos en líneas generales.

Algoritmo 2.2.1 Algoritmo de simulación paralela de modelos ENPS

Entrada:

- T: un número entero $T \geq 1$ indicando las iteraciones de la simulación.
- Las estructuras de datos indicadas en la subsección 2.2.

1: $s \leftarrow$ número total de programas en todas las membranas
2: $q \leftarrow$ número de pares $(variable, constant)$ en todos los protocolos de repartición
3: Normalizar las constantes de los protocolos de repartición en $s \times k$ hilos
4: **para** $t \leftarrow 0$ **hasta** T **hacer**
5: Chequear la aplicablilidad de los programas en s hilos
6: Calcular las funciones de producción de los programas aplicables en s hilos
7: Poner a 0 las variables consumidas por los programas aplicables en s hilos
8: Distribuir los resultados de las funciones de producción de los programas aplicables en q hilos
9: **fin para**

2.3. Resultados

El simulador se probó en dos modelos, un modelo *dummy* diseñado para probar el rendimiento del simulador y un modelo para aproximar la función exponencial e^x. El máximo factor de aceleración conseguido para el modelo dummy es de 6.5x para modelos con 15000 programas. Para el modelo de aproximación de funciones, la aceleración máxima conseguida es de 10x para 100000 membranas.

ENPSCUDA y ENPSC++ están disponibles en [3], y han dado lugar a las siguientes publicaciones:

- M. García–Quismondo, L.F. Macías–Ramos, M.J. Pérez–Jiménez. Implementing enzymatic numerical P systems for AI applications by means of graphic processing units. In J. Kelemen, J. Romportl and E. Zackova (eds.) *Beyond Artificial Intelligence. Contemplations, Expectations, Applications, Springer, Berlin–Heidelberg, Series: Topics in Intelligent Engineering and Informatics*, Volume 4, 2013, chapter XIV, pp. 137–159.

- M. García–Quismondo, A.B. Pavel, M.J. Pérez-Jiménez. Simulating large-scale ENPS models by means of GPU. In M.A. Martínez del Amor, Gh. Paun, I. Pérez Hurtado, F.J. Romero (eds.) *Proceedings of the Tenth Brainstorming Week on Membrane Computing*, Volume I, Seville, Spain, January 30–February 3, 2012, Report RGNC 01/2012, Fénix Editora, 2012, pp. 137–152.

3

Sistemas P de Dinámica de Redes Lógicas

En este capítulo se introduce un modelo de Computación Celular con Membranas para Redes Reguladoras de Genes conocido como sistemas P de *Dinámica de Redes Lógicas* (LNDP). Este marco se compone de elementos modulares (genes y operaciones sobre genes) que se comunican entre ellos. La consecuencia de esta comunicación es la emergencia de la dinámica de la red y su transición entre estados.

3.1. Formalización de las Redes Lógicas

Antes que nada, es necesario formalizar el concepto de Red Lógica.

Definición 3.1. *Una* red lógica *de tamaño* $n \geq 1$ *sobre un alfabeto* Γ *de modo que* $|\Gamma| \geq n$*, es una tupla* (Γ, f_1, f_2) *donde:*

1. $|\Gamma| = n$ *(*Γ *es el conjunto de genes de la red).*
2. $f_1 = (f_1^j, \ldots, f_1^{\alpha_1})$ *de modo que para cada* $j, 1 \leq j \leq \alpha_1$, $f_1^j = (g_1^{j,1}, g_1^{j,2}, \omega_1^j, op_1^j)$, *donde:*
   - $g_1^{j,1}, g_1^{j,2} \in \Gamma$.

- ω_1^j *es un número real in* $[0,1]$ *que representa la certidumbre de la operación unaria* f_1^j *(véase [56], teniendo en cuenta que* ω_1^j *equivale a* $U(B|A)$*, con* $B = g_1^{j,2}$ *y* $A = g_1^{j,1}$*).*
- op_1^j *es una aplicación de* $\mathbb{N}$ *en* $\{-1,0,1\}$ *que puede ser de uno de estos tipos:* sp^j, si^j, wp^j, wi^j*. Estas funciones se definen como sigue: para cada* $t \in \mathbb{N}$*:*
  - $sp^j(t) = \varphi_{g_1^{j,1}}(t) - \varphi_{\bar{g}_1^{j,1}(t)}$ *(promoción fuerte)*
  - $si^j(t) = -sp_1^j(t)$ *(inhibición fuerte)*
  - $wp^j(t) = \varphi_{g_1^{j,1}}(t)$ *(promoción débil)*
  - $wi^j(t) = -wp_1^j(t)$ *(inhibición debil)*

  Estas operaciones definen la contribución de f_1^j *al gen* $g_1^{j,2}$ *junto con* ω_1^j *para conocer su estado en el instante* $t+1$*.*

3. $f_2 = (f_2^j, \ldots, f_2^{\alpha_2})$ *de modo que para cada* $j, 1 \leq j \leq \alpha_2$, $f_2^j = (g_2^{j,1}, g_2^{j,2}, g_2^{j,3}, \omega_2^j, op_2^j)$*, donde:*

- $g_2^{j,1}, g_2^{j,2}, g_2^{j,3} \in \Gamma \cup \bar{\Gamma}$.
- ω_2^j *es un número real en* $[0,1]$ *que representa la certidumbre de la operación* f_2^j *(véase [56], considerando que* ω_2^j *equivale a* $U(C|f(A,B))$*, con* $C = g_2^{j,3}$*,* $A = g_2^{j,1}$*,* $B = g_2^{j,2}$ *y* $f = f_2^j$*).*
- op_2^j *es una aplicación de* $\mathbb{N}$ *en* $\{-1,1\}$ *que puede ser de uno de los siguientes tipos:* and^j, or^j, xor^j*. Estas funciones se definen como sigue, para cada instante* $t \in \mathbb{N}$*.*

$$
\begin{aligned}
and^j(t) &= [\varphi_{g_2^{j,1}}(t) \cdot \varphi_{g_2^{j,2}}(t) - \overline{\varphi_{g_2^{j,1}}(t) \cdot \varphi_{g_2^{j,2}}(t)}] \cdot (2 \cdot l_\Sigma(g_2^{j,3}) - 1) \\
or^j(t) &= [\varphi_{g_2^{j,1}}(t) + \varphi_{g_2^{j,2}}(t) - \varphi_{g_2^{j,1}}(t) \cdot \varphi_{g_2^{j,2}}(t) - \\
&\quad \overline{\varphi_{g_2^{j,1}}(t) + \varphi_{g_2^{j,2}}(t) - \varphi_{g_2^{j,1}}(t) \cdot \varphi_{g_2^{j,2}}(t)}] \cdot (2 \cdot l_\Sigma(g_2^{j,3}) - 1) \\
xor^j(t) &= [(1 - \varphi_{g_2^{j,1}}(t)) \cdot \varphi_{g_2^{j,2}}(t) + (1 - \varphi_{g_2^{j,2}}(t)) \cdot \varphi_{g_2^{j,1}}(t) \\
&\quad - \overline{(1 - \varphi_{g_2^{j,1}}(t)) \cdot \varphi_{g_2^{j,2}}(t) + (1 - \varphi_{g_2^{j,2}}(t)) \cdot \varphi_{g_2^{j,1}}(t)}] \cdot (2 \cdot l_\Sigma(g_2^{j,3}) - 1)
\end{aligned}
$$

  donde $\bar{b}$ *denota* $1 - b$*, para cada* $b \in \{0,1\}$*.*

  Estas operaciones proporcionan la contribución f_2^j *del gen* $g_2^{j,3}$ *junto con* ω_2^j *para saber su estado en el instante* $t+1$*.*

3.2. Un Modelo de Redes Lógicas en Computación Celular con Membranas

En esta subsección se presenta un modelo de redes lógicas basado en sistemas P de Dinámica de Redes Lógicas. Este modelo cubre cualquier posible sistema P de la familia, de modo que los multiconjuntos, reglas, etc. dependen del sistema P que representa cada instancia específica de una red lógica. La definición del modelo general requiere el uso de parámetros en sus construcciones, descritos en la tabla **??**.

Sea $LN = (\Gamma, f_1, f_2)$ una red génica. Sea ng, nu, nb el número de genes, interacciones unarias e interacciones binarias en LN, respectivamente. Sea $n = ng + nu + nb$. Este modelo consiste en un sistema PDP de grado $(1, n)$,

$$\Pi_{LN'} = (G, \Gamma, T, \mathcal{R}_{\mathcal{E}}, \mu, \mathcal{R}, \{\mathcal{M}_{ij} : \ 0 \leq i \leq q-1, \ 1 \leq j \leq m\}, \{\mathcal{M}_j : \ 1 \leq j \leq m\})$$

where:

- G es un grafo dirigido que contiene un nodo (entorno) para cada gen, interacción unaria e interacción binaria.
- En el alfabeto Γ, los estados de los genes, tipos de interacción, contribución de los pesos y objetivos se representan como se describe a continuación.

$$\begin{aligned}\Gamma = \ & \{a_i, b_i, c_i : 0 \leq i \leq 1\} \cup \{go, d_0\} \cup \{unop_j, binop_j : 1 \leq j \leq 4\} \cup \\ & \{auxDest_{i,g_{j,1},k} : 0 \leq i \leq 1, 1 \leq j \leq ng, 1 \leq k \leq nb + nu\} \cup \\ & \{dest_{i,g_{j,1},t_{k,1}+ng} : 0 \leq i \leq 1, 1 \leq j \leq ng, 1 \leq k \leq nb\} \cup \\ & \{dest_{i,g_{j,1},unt_{k-nb,1}+ng+nb} : 0 \leq i \leq 1, 1 \leq j \leq ng, nb+1 \leq k \leq nb+nu\} \cup \\ & \{e_{t_{k,4}*i+(1-i)*(1-t_{k,4}),t_{k,1}+ng} : 0 \leq i \leq 1, 1 \leq k \leq nb\} \cup \\ & \{e_{t_{k,6}*i+(1-i)*(1-t_{k,6}),t_{k,1}+ng} : 0 \leq i \leq 1, 1 \leq k \leq nb\} \cup \\ & \{e_{unt_{k-nb,4}*i+(1-i)*(1-unt_{k-nb,4}),unt_{k-nb,1}+ng+nb} : \\ & \ 0 \leq i \leq 1, nb+1 \leq k \leq nb+nu\} \cup \\ & \{eF_{t_{k,8}*i+(1-i)*(1-t_{k,8}),t_{k,1}+ng} : 0 \leq i \leq 1, 1 \leq k \leq nb\} \cup \\ & \{eF_{i,(unt_{k,1}+ng+nb)} : 0 \leq i \leq 1, 1 \leq k \leq nu\} \cup \\ & \{clock_j : 0 \leq j \leq cc+3\}\end{aligned}$$

  - El objeto *go* comienza un nuevo ciclo de evolución en los estados de los genes. Los objetos $clock_i$ sincronizan pasos críticos de cada ciclo, tales como la suma de diferentes contribuciones a cada gen.
  - Los objetos $a_i, (i \in \{0, 1\})$ representan estados de los genes: (a_0: *inactive*; a_1: *active*). Los objetos b_i representan los pesos de las interacciones de autoinfluencia.

  - Los objetos $unop_j, 1 \leq j \leq 4$ participan en las interacciones unarias, representando *promoción fuerte*, *inhibición fuerte*, *promoción débil* e *inhibición débil*, respectivamente. Los objetos $binop_j, 1 \leq j \leq 3$ participan en las binarias, representando *o*, *y* y *xor*.
  - Los objetos $dest_{i,j,k}$, $auxDest_{i,j,k}$, $e_{i,k}$, c_i y $eF_{i,k}$ son objetos auxiliares involucrados en las interacciones.

- El alfabeto del entorno es $\Sigma = \Gamma \setminus \{d_0\}$.
- Cada paso de evolución entre ciclos en redes reales implica 15 pasos computacionales, de modo que $T = 15 \cdot Cycles$, donde $Cycles$ es el número total de ciclos a simular.
- $\mu = [\quad]_1$ es la estructura de membranas.
- Los multiconjuntos iniciales son:
  - $\mathcal{M}_{g_{k,1}} = \{\ {a_1}^{g_{k,3}}, {a_0}^{1-g_{k,3},go}, 1 \leq k \leq ng\}$. Es decir, cada entorno de tipo gen (identificado como $g_{k,1}$), contiene el estado de su gen (a_1:activo o a_0:inactivo), en función de la entrada $g_{k,3} \in \{0, 1\}$ y de *go*, que empieza un nuevo ciclo.
  - $\mathcal{M}_{ng+t_{i,1}} = \{\ binop_{t_{i,2}}, 1 \leq i \leq nb\}$. Es decir, cada entorno de interacción binaria (identificado como $ng + t_{i,1}$), contiene un objeto $binop_{t_{i,2}}$ representa la interacción (o, y, xor).
  - $\mathcal{M}_{ng+nb+unt_{i,1}} = \{\ unop_{unt_{i,2}}, 1 \leq i \leq nu\}$. Es decir, cada entorno de interacción unaria (etiquetado como $ng+nb+unt_{i,1}$, contiene un objeto $unop_{unt_{i,2}}$ que representa la interacción (promoción o inhibición fuerte o débil).
- Las reglas de $\mathcal{R}$ y $\mathcal{R}_\mathcal{E}$ se muestran abajo. Están agrupadas para seguir el orden secuencial de ejecución. Las reglas del entorno empiezan con re y las del esqueleto con rs.
  - Inicio del ciclo y contribución de cada gen sobre su estado:
    $go\, a_i[\,]_1 \rightarrow c_i {b_i}^{max*i} {b_0}^{threshold} clock_0[\,]_1,\ 0 \leq i \leq 1$
  - Por cada entorno de cada gen fuente:
    - Se crean los objetos auxiliares $auxDest$ para todas las posibles interacciones de los genes fuente:
      $$(c_i)_{g_{j,1}} \rightarrow (auxDest_{i,g_{j,1},1})_{g_{j,1}}, \ldots, (auxDest_{i,g_{j,1},nb+nu})_{g_{j,1}} \quad \begin{cases} 0 \leq i \leq 1 \\ 1 \leq j \leq ng \end{cases}$$

- Se crea los objetos de destino por cada posible interacción binaria, incluyendo información sobre el gen destino $t_{k,1} + ng$:

$$(auxDest_{i,g_{j,1},k})_{g_{j,1}} \rightarrow (dest_{i,g_{j,1},t_{k,1}+ng})_{g_{j,1}} \quad \begin{cases} 0 \leq i \leq 1 \\ 1 \leq j \leq ng \\ 1 \leq k \leq nb \end{cases}$$

- Lo mismo se hace por cada posible interacción unaria, donde $untk - nb, 1 + ng + nb$ representa el entorno de destino:

$$(auxDest_{i,g_{j,1},k})_{g_{j,1}} \rightarrow (dest_{i,g_{j,1},unt_{k-nb,1}+ng+nb})_{g_{j,1}} \quad \begin{cases} 0 \leq i \leq 1 \\ 1 \leq j \leq ng \\ nb+1 \leq k \leq nb+nu \end{cases}$$

- Por cada interacción real, en los entornos de genes, los objetos $e_{i,k}$ (valor i y destino k) se crean para la contribución de cada gen fuente implicado en una interacción diferente, de los valores fuente $t_{k,4}$ y $t_{k,6}$ (interacciones binarias) y $unt_{k-nb,4}$ (interacciones unarias):

$$(dest_{i,t_{k,3},t_{k,1}+ng})_{t_{k,3}} \rightarrow (e_{t_{k,4}*i+(1-i)*(1-t_{k,4}),t_{k,1}+ng})_{t_{k,3}} \quad \begin{cases} 0 \leq i \leq 1 \\ 1 \leq k \leq nb \end{cases}$$

$$(dest_{i,t_{k,5},t_{k,1}+ng})_{t_{k,5}} \rightarrow (e_{t_{k,6}*i+(1-i)*(1-t_{k,6}),t_{k,1}+ng})_{t_{k,5}} \quad \begin{cases} 0 \leq i \leq 1 \\ 1 \leq k \leq nb \end{cases}$$

$$(dest_{i,unt_{k-nb,3},unt_{k-nb,1}+ng+nb})_{unt_{k-nb,3}} \rightarrow$$
$$(e_{unt_{k-nb,4}*i+(1-i)*(1-unt_{k-nb,4}),unt_{k-nb,1}+ng+nb})_{unt_{k-nb,3}} \quad \begin{cases} 0 \leq i \leq 1 \\ nb+1 \leq k \leq nb+nu \end{cases}$$

- Envío de los valores a los entornos de interacción:

$$(e_{i,t_{k,1}+ng})_{t_{k,3}} \rightarrow (a_i)_{t_{k,1}+ng} \quad \begin{cases} 0 \leq i \leq 1 \\ 1 \leq k \leq nb \end{cases}$$

$$(e_{i,t_{k,1}+ng})_{t_{k,5}} \rightarrow (a_i)_{t_{k,1}+ng} \quad \begin{cases} 0 \leq i \leq 1 \\ 1 \leq k \leq nb \end{cases}$$

$$(e_{i,unt_{k-nb,1}+ng+nb})_{unt_{k-nb,3}} \rightarrow$$
$$(a_i)_{unt_{k-nb,1}+ng+nb} \quad \begin{cases} 0 \leq i \leq 1 \\ 1 \leq k \leq nb \end{cases}$$

- Cálculo del resultado de las interacciones (1/2).
  - interacciones **o**:
    $binop_1\, a_0{}^2[]_1 \rightarrow binop_1\, c_0[]_1$
    $binop_1\, a_1{}^2[]_1 \rightarrow binop_1\, c_1[]_1$
    $binop_1\, a_1\, a_0[]_1 \rightarrow binop_1\, c_1[]_1$
  - interacciones **y**:
    $binop_2\, a_1{}^2[]_1 \rightarrow binop_2\, c_1[]_1$
    $binop_2\, a_0{}^2[]_1 \rightarrow binop_2\, c_0[]_1$
    $binop_2\, a_1\, a_0[]_1 \rightarrow binop_2\, c_0[]_1$
  - interacciones **xor**:
    $binop_3\, a_1{}^2[]_1 \rightarrow binop_3\, c_0[]_1$
    $binop_3\, a_0{}^2[]_1 \rightarrow binop_3\, c_0[]_1$
    $binop_3\, a_1\, a_0[]_1 \rightarrow binop_3\, c_1[]_1$
  - interacciones de los tipos **promoción fuerte**, **inhibición fuer-**

te, **promoción débil** e **inhibición débil**, respectivamente:

$unop_1\, a_i[\,]_1 \rightarrow unop_1\, c_i[\,]_1 : 0 \leq i \leq 1$
$unop_2\, a_i[\,]_1 \rightarrow unop_2\, c_{i-1}[\,]_1 : 0 \leq i \leq 1$
$unop_3\, a_i[\,]_1 \rightarrow unop_3\, {c_i}^i[\,]_1 : 0 \leq i \leq 1$
$unop_4\, a_i[\,]_1 \rightarrow unop_4\, {c_{1-i}}^i[\,]_1 : 0 \leq i \leq 1$

- Evaluación re los resultados de las interacciones (2/2).

  Por cada interacción, los objetos de tipo eF se generan y envían al gen de destino, dependiendo de los resultados previos c_i y del tipo de contribución (+ o -).

  $(c_i)_{t_{k,1}+ng} \rightarrow$
  $(eF_{tk,8*i+(1-i)*(1-t_{k,8}),t_{k,1}+ng})_{t_{k,7}}$ $\begin{cases} 0 \leq i \leq 1 \\ 1 \leq k \leq nb \end{cases}$
  $(c_i)_{unt_{k,1}+ng+nb} \rightarrow$
  $(eF_{i,(untk,1+ng+nb)})_{unt_{k,5}}$ $\begin{cases} 0 \leq i \leq 1 \\ 1 \leq k \leq nu \end{cases}$

- La contribución de cada interacción se calcula a partir de los objetos eF. Estas reglas generan objetos b_i cuya multiplicidad depende de los pesos de las interacciones.

  $eF_{i,(t_{k,1}+ng)}[\,]_1 \rightarrow {b_i}^{t_{k,9}}[\,]_1$ $\begin{cases} 0 \leq i \leq 1 \\ 1 \leq k \leq nb \end{cases}$
  $eF_{i,(unt_{k,1}+ng+nb)}[\,]_1 \rightarrow {b_i}^{unt_{k,6}}[\,]_1$ $\begin{cases} 0 \leq i \leq 1 \\ 1 \leq k \leq nu \end{cases}$

- Una vez cada gen ha recibido contribuciones de todas sus interacciones, se calcula la influencia global sobre los genes. La siguiente regla elimina cada par de objetos (b_1,b_0), cancelando las contribuciones opuestas.

  $b_1\, b_0[\,]_1 \rightarrow [\,]_1$

- Los objetos *clock* controlan el flujo de cada ciclo, asegurando que todas las contribuciones causadas por las interacciones y autoinfluencias alcancen los genes objetivos.

  $clock_{i-1}[\,]_1 \rightarrow clock_i[\,]_1 : 1 \leq i \leq cc + 3$

- Si están presente los objetos b_0, entonces el estado del gen será *inactivo*. Se crea el objeto d_0 en la membrana 1, y en el siguiente paso se lleva a cabo un cambio de polarización. En otro caso, se consume cualquier objeto b_1, convirtiendo el estado del gen en *activo*. Los objetos restantes (objetos de destino no usados, por ejemplo) se consumen de la configuración.

$b_0[\,]_1^- \to [d_0]_1^-$
$b_1[\,]_1^- \to []_1^-$

$$dest_{i,j,t_{k,1}+ng}[\,]_1^- \to [\,]_1^- \quad \begin{cases} 0 \leq i \leq 1 \\ 1 \leq j \leq ng \\ 1 \leq k \leq nb \end{cases}$$

$$dest_{i,j,unt_{k-nb,1}+ng+nb}[\,]_1^- \to [\,]_1^- \quad \begin{cases} 0 \leq i \leq 1 \\ 1 \leq j \leq ng \\ nb+1 \leq k \leq nb+nu \end{cases}$$

$[d_0]_1^- \to [\,]_1^+$

- Una vez que se ha alcanzado el último paso del ciclo, el estado del gen se pone a *activo* (1) o *inactivo* (0) dependiendo de la polarización de la membrana etiquetada como 1. Aunque las cargas eléctricas no forman parte de la regulación génica, su uso es necesario para indicar el estado de la membrana piel de cada entorno, asegurando que los objetos restantes d_0 se consumen. Además, se generan los correspondientes objetos *go*, el reloj se elimina y la polarización de la membrana se pone a 0.

$clock_{cc+3}[\,]_1^+ \to go\, a_0[\,]_1^0$
$clock_{cc+3}[\,]_1^- \to go\, a_1[\,]_1^0$

3.3. Interpretación del estado de las LN

Después de que el sistema P haya dado un número de pasos de computación predefinido, se analiza la información de salida, que está codificada en la multiplicidad de los objetos a_1 y a_0. Los entornos con un objeto a_1 representan genes activos (aquellos con a_0 representan genes inactivos). Debido a la naturaleza del sistema, los entornos gen (es decir, los entornos que representan genes) no pueden contener objetos a_1 y a_0 simultáneamente. Si no está presente ningún objeto a_1 o a_0 en el entorno gen, entonces el estado del gen representado no puede ser evaluado todavía. Es decir, alcanzar un estado evaluable llevará algunos pasos computaciones para la red.

3.4. Validación del modelo

Para validar el modelo, se ha llevado a cabo un caso de estudio en el que se analiza una LN asociada con el proceso de floración de la planta *Arabidopsis thaliana*. El modelo consigue capturar el comportamiento del método LAPP

Parámetro	Descripción
	Parámetros generales del sistema
ng	Número de genes de la red
nb	Número de interacciones binarias
nu	Número de interacciones unarias
$threshold$	Peso máximo de cada interacción
cc	Reloj de control
	Parámetros de configuración de los genes
$g_{i,1}$	Número de gen (id)
$g_{i,3}$	Estado inicial del gen
	Parámetros de interacciones binarias
$t_{i,1}$	Número de interacción binaria (id)
$t_{i,2}$	Tipo de interacción (o: 1, y: 2, xor: 3)
$t_{i,3}$	1^{st} Número de gen fuente (id)
$t_{i,4}$	1^{st} Contribución de gen fuente (positiva: 1, negativa: 0)
$t_{i,5}$	2^{nd} Número de gen fuente (id)
$t_{i,6}$	2^{nd} Contribución de gen fuente (positiva: 1, negativa: 0)
$t_{i,7}$	Número de gen de destino (id)
$t_{i,8}$	Influencia sobre el gen de destino (positiva: 1, negativa: 0)
$t_{i,9}$	Fuerza del destino
	Parámetros de interacciones unarias
$unt_{i,1}$	Número de interacción unaria (id)
$unt_{i,2}$	Tipo de interacción (promoción fuerte: 1, inhibición: 2; débiles: 3, 4)
$unt_{i,3}$	Número de gen fuente (id)
$unt_{i,4}$	Contribución de gen fuente (positiva, negativa)
$unt_{i,5}$	Número de gen de destino (id)
$unt_{i,6}$	Influencia sobre el gen de destino (positiva, negativa)

Cuadro 3.1: Parámetros para los sistemas LNDP

mejorado [56, 55], con lo que tiene una potencia predictiva equivalente al primero. Este modelo está disponible para descarga en [42], y ha dado lugar a las siguientes publicaciones:

- L. Valencia–Cabrera, M. García–Quismondo, Y. Su, M.J. Pérez–Jiménez, L. Pan, H. Yu. Modeling logic gene networks by means of probabilistic dynamic P systems. *International Journal of Unconventional Computing*, 9, 5–6 (2013), pp. 445–464. **JCR 0.43**

- L. Valencia–Cabrera, M. García-Quismondo, M.J. Pérez–Jiménez, Y. Su, H. Yu, L. Pan. Analising gene networks with PDP systems. Arabidopsis thailiana, a case study. In L. Valencia–Cabrera, M. García–Quismondo,

L.F. Macías-Ramos, M.A. Martínez del Amor, Gh. Păun, A. Riscos–Núñez (eds.) *Proceedings of the Eleventh Brainstorming Week on Membrane Computing*, Seville, Spain, February 4–8, 2013, Report RGNC 01/2013, Fénix Editora, 2013, pp. 257–272.

4

Sistemas P Probabilísticos con Guardas

En este capítulo se introduce un nuevo marco de modelización para procesos ecológicos conocido como sistemas P *Probabilísticos con Guardas* (PGP). Este marco computacional se formaliza y se complementa con dos simuladores, uno secuencial y otro basado en tecnología GPU, para reproducir su dinámica. Estos simuladores funcionan únicamente para modelos sin competición de objectos, pero se proporcionan algunas ideas para su extensión a modelos con competición de objetos.

4.1. Descripción Formal de los Sistemas PGP

A continuación de formaliza el concepto de sistema PGP y se describe su semántica.

4.1.1. Formalización de los sistemas PGP

Definición 4.1. *Un sistema P Probabilístico con Guardas (*sistema PGP*) de grado $q \geq 1$ es una tupla $\Pi = (\Gamma, \Phi, \mathcal{R}, \mathcal{G}_{\mathcal{R}}, p_{\mathcal{R}}, (f_1, \mathcal{M}_1), \ldots, (f_q, \mathcal{M}_q))$ donde:*

- *Γ y Φ son alfabetos finitos de manera que $\Gamma \cap \Phi = \emptyset$. Los elementos de Γ se conocen como **objetos** y los de Φ como **banderas**.*
- *$\mathcal{R}$ es un conjunto finito de reglas de los siguientes tipos:*

  - $\{f\}\,[u]_i \to [v]_j$ *con* $u, v \in M(\Gamma)$, $f \in \Phi$ *y* $1 \leq i, j \leq q$
  - $\{f\}\,[u, f]_i \to [v, g]_i$ *con* $u, v \in M(\Gamma)$, $f, g \in \Phi$ *y* $1 \leq i, j \leq q$. *Además, para cada* $f \in \Phi, u \in M(\Gamma), 1 \leq i \leq q$, *existe sólo un tipo de regla* $\{f\}\,[u, f]_i \to [v, g]_i$.

- $\mathcal{G}_{\mathcal{R}}$ *es el grafo directo asociado a* $\mathcal{R}$ *como sigue:* $V = \{1, \ldots, q\}$ *y* $(i, j) \in E$ *si y sólo si existe alguna regla del tipo* $\{f\}\,[u]_i \to [v]_j$, *or* $i = j$ *y existe una regla del tipo* $\{f\}\,[u, f]_i \to [v, g]_i$.

- $p_{\mathcal{R}}$ *es una aplicación de* $\mathcal{R}$ *en* $[0, 1]$ *de modo que:*

  - *Si* $r \equiv \{f\}\,[u, f]_i \to [u, g]_i$, *entonces* $p_{\mathcal{R}}(r) = 1$.
  - *Para cada* $f \in \Phi, u \in M(\Gamma), 1 \leq i \leq q$, *si* $r_1, \ldots, r_t$ *son reglas del tipo* $\{f\}\,[u]_i \to [v]_j$, *entonces* $\sum_{k=1}^{t} p_{\mathcal{R}}(r_k) = 1$.

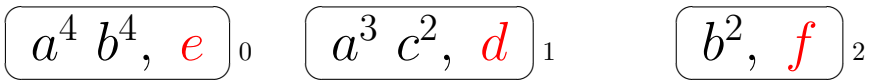
- *Para cada* $i (1 \leq i \leq q)$, *tenemos* $f_i \in \Phi$ *y* $\mathcal{M}_i \in M(\Gamma)$.

Comentario 4.1. *Un sistema P Probabilístico con Guardas puede verse como un conjunto de q células etiquetadas por* $1, \ldots, q$ *de modo que: (a)* $\mathcal{M}_1, \ldots, \mathcal{M}_q$ *son multiconjuntos finitos sobre* Γ *representando los objetos inicialmente situados en las q células del sistema; (b)*$f_1, \ldots, f_q$ *son banderas que inicialmente marcan las q células; (c)* $G_{\mathcal{R}}$ *es un grafo dirigido cuyos arcos especifican conexiones entre células; (d)* $\mathcal{R}$ *es el conjunto de reglas que permiten la evolución del sistema y cada regla* $r \in \mathcal{R}$ *se asocia con un número real* $p_{\mathcal{R}}(r)$ *en* $[0, 1]$ *que significa la probabilidad de dicha regla para ser aplicada en el caso de que sea aplicable.*

Comentario 4.2. *En los sistemas PGP se usan dos tipos de símbolos:* objetos *(elementos de* Γ*) y* banderas *(elementos de* Φ*). Puede considerarse que los objetos están* ***dentro de*** *las células y los flags están* ***sobre*** *(el borde de) las celulas.*

4.1.2. Semantics of PGP systems

Definición 4.2. *Una* configuración *en cualquier instante* $t \geq 0$ *de un sistema PGP* Π *es una tupla* $\mathcal{C}_t = (x_1, u_1, \ldots, x_q, u_q)$ *donde, por cada* i $(1 \leq i \leq q)$, $x_i \in \Phi$ *y* $u_i \in M(\Gamma)$. *Es decir, una configuración de* Π *en cualquier instante* $t \geq 0$ *se describe mediante todos los multiconjuntos de objetos sobre* Γ *asociadas a todas las células presentes en el sistema y las banderas que marcan las células. Se dice que* $(f_1, \mathcal{M}_1, \ldots, f_q, \mathcal{M}_q)$ *es la* configuración inicial *de* Π. *En cualquier*



- R_0:

$$\begin{array}{l} \{e\}[a^2]_0 \overset{0.2}{\to} [c^2]_1 \\ \{e\}[a^2]_0 \overset{0.7}{\to} [b]_0 \end{array} \Big| \quad \{e\}[a,e]_0 \to [c,f]_0 \quad \Big| \quad \{f\}[b^3,f]_0 \to [e]_0$$

- R_1:

$$\begin{array}{l} \{f\}[a^2]_1 \overset{0.4}{\to} [b]_1 \\ \{f\}[a^2]_1 \overset{0.6}{\to} [c^2]_2 \end{array} \Big| \quad \begin{array}{l} \{d\}[c]_1 \overset{0.2}{\to} [c]_2 \\ \{d\}[c]_1 \overset{0.7}{\to} [a^2]_0 \end{array} \Big| \quad \{f\}[c^2]_1 \to [b]_0$$

- R_2:

$$\{f\}[b^2,f]_2 \to [a^2,d]_2 \quad \Big| \quad \begin{array}{l} \{d\}[a]_2 \overset{0.5}{\to} [b^2]_0 \\ \{d\}[a]_2 \overset{0.5}{\to} [c]_1 \end{array}$$

Figura 4.1: Un ejemplo de sistema PGP. Las flags se realzan en rojo y las probabilidades que son iguales a 1 se omiten.

instante, cada célula tiene una y solo una bandera, de manera similar a las polarizaciones de los sistemas P con membranas activas.

Definición 4.3. *Una regla r de tipo $\{f\}[u]_i \to [v]_j$ es* aplicable *a una configuración $\mathcal{C}_t = (x_1, u_1, \ldots x_q, u_q)$ si y sólo si $x_i = f$ y $u \subseteq u_i$, para todo $1 \leq i \leq q$.*
Cuando se aplica r a C_t, los objetos presentes en u se consumen en la célula i y los objetos en v se producen en la célula j. La bandera f no cambia; hace el papel de un catalizador que ayuda la evolución del objeto en u.

Definición 4.4. *Una regla r del tipo $\{f\}[u,f]_i \to [v,g]_i$ es* aplicable *a una configuración $\mathcal{C}_t = (x_1, u_1, \ldots x_q, u_q)$ si y sólo si $x_i = f$ y $u \subseteq u_i$, para cada $1 \leq i \leq q$.*
Cuando se aplica r a C_t, en la célula i los objetos de in u se reemplazan por los de v y f se reemplaza por g. En este caso, la bandera f se consume, de modo que r puede aplicarse sólo una vez en el instante t en la célula i.

Comentario 4.3. *Después de aplicar una regla r del tipo $\{f\}[u,f]_i \to [v,g]_i$, otras reglas r' del tipo $\{f\}[u]_i \to [v]_j$ todavía pueden aplicarse (la bandera se mantiene en vigor). Sin embargo, f se consume, de manera que no puedan aplicarse más reglas $\{f\}[u,f]_i \to [v,g]_i$.*

Definición 4.5. *Una configuración es una* configuración de parada *si no pueden aplicarse más reglas.*

Definición 4.6. *Se dice que la configuración* $\mathcal{C}_1$ produce *la configuración* $\mathcal{C}_2$ *es un paso de transición si puede pasarse de* $\mathcal{C}_1$ *a* $\mathcal{C}_2$ *aplicando reglas de* $\mathcal{R}$ *de un modo no determinista y maximal paralelo, en función de sus probabilidades asociadas definidas por la aplicación* $p_{\mathcal{R}}$*. Es decir, se aplica un multiconjunto maximal de reglas de* $\mathcal{R}$*, de modo que no puedan añadirse más reglas.*

Definición 4.7. *Una* computación *de un sistema PGP* Π *es una secuencia de configuraciones de modo que: (a) el primer término de la secuencia es la configuración inicial de* Π*, (b) cada término restante de la secuencia se obtiene de la anterior aplicando las reglas del sistema siguiendo la definición 4.6, (c) si la secuencia es finita (llamada computación de parada) entonces el último término del sistema es la configuración de parada.*

4.2. Algoritmos de simulación para sistemas PGP

A continuación, se presentan dos algoritmos de simulación para sistemas PGP. El primero de ellos 4.2.1 es secuencial, mientras que el segundo esta diseñado para arquitecturas paralelas. Ambos funcionan únicamente para sistemas PGP sin competición de objetos.

Algoritmo 4.2.1 Algoritmo de simulación para sistemas PGP

Entrada:

- T: un número entero $T \geq 1$ que representa las iteraciones de la simulación.
- R_{Iter} un número entero $R_{Iter} \geq 1$ que representa las iteraciones no maximales de las reglas (es decir, las iteraciones en las que las aplicaciones seleccionadas para cada regla no tienen por qué ser maximales).
- $\Pi = (\Gamma, \Phi, \mathcal{R}, \mathcal{G}_{\mathcal{R}}, p_{\mathcal{R}}, (f_1, \mathcal{M}_1), \ldots, (f_q, \mathcal{M}_q))$: un sistema PGP de grado $q \geq 1$.

1: **para** $t \leftarrow 1$ **hasta** T **hacer**
2: Chequear la aplicabilidad de cada bloque
3: Distribuir los objetos entre los bloques aplicables
4: Distribuir las aplicaciones de cada bloque entre sus reglas
5: Generar los objetos
6: **fin para**

Tanto el modelo como el simulador serán publicados y se harán disponibles en las páginas web [42] y [3].

Para validar el marco de modelización, se ha diseñado un modelo del ecosistema de la mariposa *Pieris napi oleracea*. Este modelo está inspirado en

Algoritmo 4.2.2 Algoritmo paralelo de simulación de sistemas PGP

Entrada:

- T: un número entero $T \geq 1$ que representa las iteraciones de la simulación.
- B_{Iter}: un número entero $B_{Iter} \geq 1$ que representa las iteraciones no maximales de distribución de objetos entre bloques.
- R_{Iter}: un número entero $R_{Iter} \geq 1$ las iteraciones no maximales de distribución de objetos entre reglas.
- Un conjunto de estructuras de datos representando un sistema PGP.

```
1: para t ← 1 hasta T hacer
2:     Chequear la aplicabilidad de los bloques
3:     Poner a 0 las aplicaciones de las reglas
4:     Calcular las aplicaciones de los bloques
5:     Poner a 0 las aplicaciones de bloques no procesadas
6:     Consumir los objetos de los bloques
7:     para n ← 1 a R_Iter hacer
8:         Distribuir aplicaciones entre las reglas
9:     fin para
10:    Distribuir aplicaciones entre las reglas de manera maximal
11:    Generar los objetos de las reglas aplicadas
12: fin para
```

otro modelo introducido en [32], que consiste en un conjunto de ecuaciones estocásticas discretas. Los parámetros para el modelo basado en Computación Celular con Membranas han sido obtenidos de [32] y [31] y proporcionados directamente por expertos en el ecosistema. La validación experimental del modelo permitió el análisis de escenarios de especial interés por parte de los expertos, concluyendo que la dinámica del sistema refleja un crecimiento continuado en el número total de mariposas y un crecimiento especialmente notable en la población de mariposas heterocigóticas con alelos dominantes y recesivos, debido principalmente a su adaptación a la planta *Alliara petiolata*, una especie invasora que está desplazando a la variante nativa *Cardamine diphylla*. Además, un análisis de rendimiento del simulador basado en tecnología GPU para modelos PGP demostró que éste simulador no fue capaz de superar en rendimiento a su homólogo secuencial, con lo que un análisis de rendimiento se propone como trabajo futuro.

5 Trabajo futuro

Por último, en la tesis se proponen algunas líneas de trabajo futuro, como la aplicación del simulador ENPSCUDA a modelos de robots colaborativos [29], con objetivos dinámicos [59], con varios objetivos [59] [51] [25] o que funcionen en entornos no deterministas [50] [4], la aplicación del marco de modelización de los sistemas LNDP a otras Redes Reguladoras de Genes, como las redes asociadas a la bacteria *Escherichia coli* [23] [22] [36] con un amplio historial de uso en biología de sistemas y sintética. También se propone adaptar en el mismo marco otros algoritmos de simulación como el método de Gillespie [21]. Por último, se propone seguir profundizando en la modelización del ecosistema de la mariposa *Pieris napi oleracea* [32] [31] y en la mejora del rendimiento del simulador paralelo PGPCUDA para modelos dentro del marco de los sistemas PGP.

Bibliografía

[1] Java web page. `http://www.java.com`. Official website.

[2] NVIDIA CUDA home page. `http://www.nvidia.es/cuda`.

[3] PMCGPU web page. `http://sourceforge.net/projects/pmcgpu/`. A website including some software tools for the simulation of P systems on GPU architectures.

[4] M. R. Abdessemed and A. Bilami. Evolutionary research of optimal strategies for exclusive positioned clustering in simulated environment of collective robotics. *Robotics and Autonomous Systems*, 58(10):1130 – 1137, 2010.

[5] F. Cabarle, H. Adorna, M. A. M. del Amor, and M. J. Pérez-Jiménez. Improving gpu simulations of spiking neural P systems. *Romanian Journal of Information Science and Technology*, 15:5–20, 06/2012 2012.

[6] F. Cabarle, H. Adorna, M. Martínez-del Amor, and M. Pérez-Jiménez. Spiking neural P system simulations on a high performance gpu platform. In Y. Xiang, A. Cuzzocrea, M. Hobbs, and W. Zhou, editors, *Algorithms and Architectures for Parallel Processing*, volume 7017 of *Lecture Notes in Computer Science*, pages 99–108. Springer Berlin Heidelberg, 2011.

[7] F. G. Cabarle, H. N. Adorna, M. A. Martínez-del-Amor, and M. J. Pérez-Jiménez. Improving GPU simulations of spiking neural P systems. *Romanian Journal of Information Science and Technology*, 15:5–20, 2012.

[8] J. Cecilia, J. García, G. Guerrero, M. Martínez-del Amor, M. Pérez-Jiménez, and M. Ujaldón. The gpu on the simulation of cellular computing models. *Soft Computing*, 16(2):231–246, 2012.

[9] J. M. Cecilia, J. M. García, G. D. Guerrero, M. A. Martínez-del Amor, I. Pérez-Hurtado, and M. J. Pérez-Jiménez. Simulation of P systems with

active membranes on cuda. *Briefings in Bioinformatics*, 11(3):313–322, 2010.

[10] V. M. Cervantes-Salido, O. Jaime, C. A. Brizuela, and I. M. Martínez-Pérez. Improving the design of sequences for {DNA} computing: A multiobjective evolutionary approach. *Applied Soft Computing*, (0):–, 2013.

[11] R. Ceterchi, M. Mutyam, G. Păun, and K. G. Subramanian. Array-rewriting p systems. *Natural Computing*, 2(3):229–249, Aug. 2003.

[12] G. Ciobanu and G. Wenyuan. P systems running on a cluster of computers. In C. Martín-Vide, G. Mauri, G. Păun, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing*, volume 2933 of *Lecture Notes in Computer Science*, pages 123–139. Springer Berlin Heidelberg, 2004.

[13] M. A. Colomer, A. Margalida, D. Sanuy, and M. J. Pérez-Jiménez. A bio-inspired computing model as a new tool for modeling ecosystems: The avian scavengers as a case study. *Ecological Modelling*, 222(1):33 – 47, 2011.

[14] A. Cordón-Franco, M. Gutiérrez-Naranjo, M. Pérez-Jiménez, and F. Sancho-Caparrini. A prolog simulator for deterministic P systems with active membranes. *New Generation Computing*, 22(4):349–363, 2004.

[15] L. N. de Castro. Fundamentals of natural computing: an overview. *Physics of Life Reviews*, 4:1–36, Mar. 2007.

[16] M. A. M. del Amor, J. Pérez-Carrasco, and M. J. Pérez-Jiménez. Simulating a family of tissue P systems solving sat on the gpu. In *Eleventh Brainstorming Week on Membrane Computing (11BWMC)*, pages 201–220, Sevilla, España, 08/2013 2013. Fenix Editora.

[17] M. García-Quismondo, R. Gutiérrez-Escudero, M. A. Martínez-del Amor, E. F. Orejuela-Pinedo, and I. Pérez-Hurtado. P-lingua 2.0: A software framework for cell-like P systems. *International Journal of Computers Communications and Control*, 4(3):234–243, 2010.

[18] M. García-Quismondo, M. A. Gutiérrez-Naranjo, and D. Ramírez-Martínez. How does a P system sound? In *Eighth Brainstorming Week on Membrane Computing*, pages 123–132, Sevilla, Spain, 02/2010 2010. Fenix Editora.

[19] M. García-Quismondo, L. Valencia-Cabrera, Y. Su, M. J. Pérez-Jiménez, L. Pan, and H. Yu. Modeling logic gene networks by means of probabilistic dynamic P systems. In L. Pan, G. Paun, and T. Song, editors, *Asian Conference on Membrane Computing*, pages 30–60, Wuhan, China, 10/2012 2012.

[20] M. Gheorghe, F. Ipate, R. Lefticaru, M. J. Pérez-Jiménez, A. Turcanu, L. Valencia Cabrera, M. García-Quismondo, and L. Mierla. 3-col problem modelling using simple kernel P systems. *International Journal of Computer Mathematics*, 90(4):816–830, 2013.

[21] D. T. Gillespie. Stochastic simulation of chemical kinetics. *Annual Review of Physical Chemistry*, 58(1):35–55, 2007. PMID: 17037977.

[22] J. R. Gittins, D. A. Phoenix, and J. M. Pratt. Multiple mechanisms of membrane anchoring of escherichia coli penicillin-binding proteins. {*FEMS*} *Microbiology Reviews*, 13(1):1 – 12, 1994.

[23] H. M. Grewal, W. Gaastra, A.-M. Svennerholm, J. Röli, and H. Sommerfelt. Induction of colonization factor antigen i (cfa/i) and coli surface antigen 4 (cs4) of enterotoxigenic escherichia coli: relevance for vaccine production. *Vaccine*, 11(2):221 – 226, 1993.

[24] V. Grimm. Ten years of individual-based modelling in ecology: what have we learned and what could we learn in the future? *Ecological Modelling*, 115(203):129 – 148, 1999.

[25] H. Guo, Y. Meng, and Y. Jin. A cellular mechanism for multi-robot construction via evolutionary multi-objective optimization of a gene regulatory network. *Biosystems*, 98(3):193 – 203, 2009. ¡ce:title¿Evolving Gene Regulatory Networks¡/ce:title¿.

[26] A. Gutiérrez, L. Fernández, F. Arroyo, and S. Alonso. Suitability of using microcontrollers in implementing new p-system communications architectures. *Artificial Life and Robotics*, 13(1):102–106, 2008.

[27] A. Gutiérrez, L. Fernandez, F. Arroyo, and V. Martínez. Design of a hardware architecture based on microcontrollers for the implementation of membrane systems. In *Symbolic and Numeric Algorithms for Scientific Computing, 2006. SYNASC '06. Eighth International Symposium on*, pages 350–353, 2006.

[28] M. A. Gutiérrez-Naranjo, M. J. Pérez-Jiménez, and F. J. Romero-Campero. A uniform solution to {SAT} using membrane creation. *Theoretical Computer Science*, 371(1-2):54 – 61, 2007. ¡ce:title¿Computing and the Natural Sciences¡/ce:title¿.

[29] R. Haghighi and C. Cheah. Multi-group coordination control for robot swarms. *Automatica*, 48(10):2526 – 2534, 2012.

[30] R. A. Juayong, F. Cabarle, H. Adorna, and M. A. M. del Amor. On the simulations of evolution-communication P systems with energy without antiport rules for gpus. In *Tenth Brainstorming Week on Membrane Computing*, volume I, pages 267–290, Seville, Spain, 02/2012 2012. Fenix Editora.

[31] M. S. Keeler and F. S. Chew. Escaping an evolutionary trap: preference and performance of a native insect on an exotic invasive host. *Oecologia*, 156(3):559–568, 2008.

[32] M. S. C. Keeler, F. S. Goodale, and J. M. B. C. Reed. Modelling the impacts of two exotic invasive species on a native butterfly: top-down vs. bottom-up effects. *Journal of Animal Ecology*, 75(3):777–788, 2006.

[33] R. Kuo, P. Wu, and C. Wang. An intelligent sales forecasting system through integration of artificial neural networks and fuzzy neural networks with fuzzy weight elimination. *Neural Networks*, 15(7):909 – 925, 2002.

[34] V. Martínez, A. Gutiérrez, and L. Mingo. Circuit fpga for active rules selection in a transition P system region. In M. Kippen, N. Kasabov, and G. Coghill, editors, *Advances in Neuro-Information Processing*, volume 5507 of *Lecture Notes in Computer Science*, pages 893–900. Springer Berlin Heidelberg, 2009.

[35] M. A. Martínez-del Amor, I. Pérez-Hurtado, A. Gastalver-Rubio, A. C. Elster, and M. Pérez-Jiménez. Population dynamics P systems on cuda. In D. Gilbert and M. Heiner, editors, *Computational Methods in Systems Biology*, Lecture Notes in Computer Science, pages 247–266. Springer Berlin Heidelberg, 2012.

[36] O. Mol and B. Oudega. Molecular and structural aspects of fimbriae biosynthesis and assembly in escherichia coli. {*FEMS*} *Microbiology Reviews*, 19(1):25 – 52, 1996.

[37] V. Nguyen, D. Kearney, and G. Gioiosa. An algorithm for non-deterministic object distribution in P systems and its implementation in hardware. In D. Corne, P. Frisco, G. Păun, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing*, volume 5391 of *Lecture Notes in Computer Science*, pages 325–354. Springer Berlin Heidelberg, 2009.

[38] S. Omkar, R. Khandelwal, S. Yathindra, G. N. Naik, and S. Gopalakrishnan. Artificial immune system for multi-objective design optimization of composite structures. *Engineering Applications of Artificial Intelligence*, 21(8):1416 – 1429, 2008.

[39] L. Pan and M. J. Pérez-Jiménez. Computational complexity of tissue-like p systems. *Journal of Complexity*, 26(3):296 – 315, 2010.

[40] L. Pan, G. Păun, and M. Pérez-Jiménez. Spiking neural P systems with neuron division and budding. *Science China Information Sciences*, 54(8):1596–1607, 2011.

[41] G. Paun. *Membrane Computing: An Introduction*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[42] I. Pérez-Hurtado. P–Lingua webpage. `https://www.p-lingua.org`.

[43] M. Pérez-Jiménez and A. Riscos-Nuñez. A linear-time solution to the knapsack problem using P systems with active membranes. In C. Martín-Vide, G. Mauri, G. Păun, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing*, volume 2933 of *Lecture Notes in Computer Science*, pages 250–268. Springer Berlin Heidelberg, 2004.

[44] G. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108 – 143, 2000.

[45] G. Păun and R. Păun. Membrane computing and economics: Numerical p systems. *Fundam. Inf.*, 73(1,2):213–227, July 2006.

[46] Z. Qi, J. You, and H. Mao. P systems and petri nets. In C. Martin-vide, G. Mauri, G. Păun, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing*, volume 2933 of *Lecture Notes in Computer Science*, pages 286–303. Springer Berlin Heidelberg, 2004.

[47] F. J. Romero-Campero and M. J. Pérez-Jiménez. A model of the quorum sensing system in vibrio fischeri using P systems. *Artificial Life*, 14(1):95–109, 2008.

[48] F. J. Romero-Campero and M. J. Pérez-Jiménez. Modelling gene expression control using P systems: The lac operon, a case study. *Biosystems*, 91(3):438 – 457, 2008. ¡ce:title¿P-Systems Applications to Systems Biology¡/ce:title¿.

[49] A. Romero-Jiménez, M. Gutiérrez-Naranjo, and M. Pérez-Jiménez. Graphical modeling of higher plants using P systems. In H. Hoogeboom, G. Păun, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing*, volume 4361 of *Lecture Notes in Computer Science*, pages 496–506. Springer Berlin Heidelberg, 2006.

[50] L. Su and M. Tan. A virtual centrifugal force based navigation algorithm for explorative robotic tasks in unknown environments. *Robotics and Autonomous Systems*, 51(4):261 – 274, 2005.

[51] A. T. Tolmidis and L. Petrou. Multi-objective optimization for dynamic task allocation in a multi-robot system. *Engineering Applications of Artificial Intelligence*, 26(5–6):1458 – 1468, 2013.

[52] C. Vasile, A. Pavel, I. Dumitrache, and G. Păun. On the power of enzymatic numerical P systems. *Acta Informatica*, 49(6):395–412, 2012.

[53] S. Verlan and J. Quiros. Fast hardware implementations of P systems. In *Proceedings of the 13th international conference on Membrane Computing*, CMC'12, pages 404–423, Berlin, Heidelberg, 2013. Springer-Verlag.

[54] J. Wang, P. Shi, H. Peng, M. Pérez-Jiménez, and T. Wang. Weighted fuzzy spiking neural p systems. *Fuzzy Systems, IEEE Transactions on*, 21(2):209–220, 2013.

[55] S. Wang, Y. Chen, Q. Wang, E. Li, Y. Su, and D. Meng. Analysis for stimuli to shoot genes of arabidopsis thaliana based on logical relationships. *Optimization and Systems Biology*, 11(-):435–447, 2006.

[56] S. Wang, Y. Chen, Q. Wang, E. Li, Y. Su, and D. Meng. Analysis for gene networks based on logic relationships. *Journal of Systems Science and Complexity*, 23:999–1011, 2010. 10.1007/s11424-010-0205-0.

[57] D. Whitley, T. Starkweather, and C. Bogart. Genetic algorithms and neural networks: optimizing connections and connectivity. *Parallel Computing*, 14(3):347 – 361, 1990.

[58] X.-S. Yang and M. Karamanoglu. Swarm intelligence and bio-inspired computation: An overview. In X.-S. Yang, Z. Cui, R. Xiao, A. H. Gandomi, and M. Karamanoglu, editors, *Swarm Intelligence and Bio-inspired Computation*, pages 3 – 23. Elsevier, Oxford, 2013.

[59] Y. Zhang, D. wei Gong, and J. hua Zhang. Robot path planning in uncertain environment using multi-objective particle swarm optimization. *Neurocomputing*, 103(0):172 – 185, 2013.