La Biología como alternativa computacional

Mario de J. Pérez Jiménez, Fernando Sancho Caparrini
marper@us.es, fsancho@us.es
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla, España

Resumen

Se presenta una visión general de un modelo de computación inspirado en la célula: la *Computación Celular con Membranas,* así como de una de sus variantes más recientes, los *Spiking Neural P Systems*, que se fundamentan en la emisión discreta de señales entre las neuronas que forman una red neuronal para realizar cálculos.

1. Introducción

Tras el establecimiento, por parte de R. Churchhouse en 1983, de las limitaciones físicas de la computación convencional (basada en las propiedades electro-magnéticas del silicio), y la constatación de la existencia de problemas duros desde el punto de vista de la eficiencia práctica, el mundo de la computación teórica ha proporcionado algunos de los modelos más sorprendentes que se han dado en la historia de la misma. Nos referimos a los modelos de Computación Natural o Bio-inspirados, en los que una interpretación audaz y novedosa de los procesos que de forma normal se dan en la Naturaleza nos proporciona nuevos paradigmas de computación. En concreto, los dos modelos más interesantes y más ampliamente estudiados en el área en los últimos años son los conocidos como *Computación Molecular* y *Computación Celular con Membranas*.

El primero de ellos, la *Computación Molecular*, surge como tal tras un experimento de laboratorio en 1994 en el cual, haciendo uso de la capacidad natural de las moléculas de ADN para codificar información y recombinarse siguiendo leyes básicas de la bioquímica, L. Adleman [1] consigue dar una solución a un problema matemático muy conocido (concretamente, proporciona por medios moleculares la solución del *Problema del Camino Hamiltoniano* sobre un grafo de 7 vértices). A pesar de que los métodos bioquímicos que usa en el laboratorio son muy rústicos, el experimento pone de manifiesto lo que unos años antes había adelantado T. Head en un marco puramente teórico [7]: el uso de moléculas de ADN como sustrato computacional es suficientemente potente como para obtener la misma potencia que proporciona la

computación convencional. Además, los modelos basados en la computación molecular presentan ventajas que están lejos de conseguirse por los medios convencionales, en cuanto a eficiencia energética, densidad de almacenamiento y velocidad global.

En la actualidad el modelo ha conseguido la suficiente madurez como para obtener primeras implementaciones reales en forma de micro-chips DNA de función específica, interruptores moleculares con aplicaciones farmacéuticas y la primera simulación bioquímica de una máquina de Turing.

El segundo modelo, el *Modelo de Computación con Membranas*, está inspirado en un nivel biológico superior: el funcionamiento de la célula como organismo vivo capaz de procesar y generar información. Por medio de una abstracción simplificadora, Gh. Păun [12] propone un modelo en el que la célula, como objeto físico compartimentalizado por medio de las membranas que contiene, opera sobre objetos (los compuestos químicos que se encuentran en su interior) por medio de reglas que permiten modificarlos, comunicarlos entre zonas adyacentes e incluso modificar la estructura de la célula por medio de la destrucción/creación de dichas membranas. El modelo obtenido es, tal y como se corresponde naturalmente con la forma de trabajar de la célula, un modelo no determinista de tipo distribuido y paralelo.

Este modelo, que ha suscitado un gran interés por parte de la comunidad científica desde el momento de su nacimiento, no dispone por el momento de una implementación biológica real, aunque se están efectuando primeras aproximaciones por medio del uso de células extremadamente simples (tanto funcional como estructuralmente, haciendo uso de algunos tipos de bacterias). Ello se debe, esencialmente, a la complejidad intrínseca que presenta el funcionamiento celular (que, a duras penas, está empezando a conocerse con algo más de profundidad en los últimos años, y que con seguridad precisa de un desarrollo previo de otras ramas de la ciencia, como es el estudio de los sistemas complejos). Sin embargo, como modelo de computación abstracto está demostrando tener una versatilidad única, aplicándose a muy diversas ramas del conocimiento [3], que van desde aplicaciones a la Biología (como es el estudio de cascadas de señales en rutas metabólicas o de interacciones proteínicas), pasando por aplicaciones en Ecología (como es el estudio de interacciones entre grupos ecológicos), Economía (modelado de procesos económicos no lineales), Computación Aplicada (al ser capaz de modelar también redes de comunicación complejas) hasta, por supuesto, la aplicación a la resolución de problemas matemáticos (proporcionando soluciones eficientes de problemas especialmente duros).

El objetivo de este trabajo es mostrar una visión general del segundo modelo al que hemos hecho referencia, el modelo de *Computación Celular con Membranas*, ya que puede proporcionar un marco general apropiado para el modelado de muchas de las técnicas que actualmente se están desarrollando en el estudio de sistemas colectivos o complejos, en los que la interacción entre individuos que siguen reglas relativamente simples proporciona una riqueza difícil de explicar por medio de un estudio reduccionista del sistema global, tal como es el caso de los sistemas biológicos, sistemas computacionales pervasivos, sistemas sociales, etc. Con este fin, la distribución y contenido del artículo se puede resumir como sigue: en la sección

siguiente ofrecemos una visión informal de la Computación Celular con Membranas, destacando sus puntos principales y la forma en que están concebidos los dispositivos que se generan siguiendo este modelo. En la sección 3 presentamos una variante que se engloba dentro del paradigma mostrado en la sección 2 y que está fuertemente inspirada por el modo en que funcionan las neuronas, haciendo uso de la transmisión de señales entre elementos del sistema para transformar y generar información. Por último, cerramos el trabajo presentando algunas de las conclusiones e ideas futuras en relación con los modelos presentados.

2. Computación con Membranas: una visión informal.

En Octubre de 1998 Gh. Păun introduce un nuevo modelo de computación de tipo paralelo y distribuido, denominado *P sistemas de transición* [12], dando el primer modelo de un nuevo paradigma que se conocerá más tarde con el nombre de *computación celular con membranas*.

Como primera diferencia con respecto a otros modelos de computación natural (por ejemplo, los modelos de computación molecular basados en ADN) conviene hacer notar que este modelo no viene descrito a través de un lenguaje de programación; es decir, no proporciona una serie de operaciones básicas que puedan ser secuenciadas sobre un dato de entrada para obtener un resultado final, sino que se generan *dispositivos* cuya ejecución modifica el contenido de las distintas componentes que lo integran.

Una característica que llama la atención en cuanto a la estructura interna de la célula, y que será esencial en la definición de los sistemas celulares, es el hecho de que las distintas partes del sistema biológico que la componen se encuentran delimitadas por varios tipos de *membranas* (en su sentido más amplio): desde la propia membrana que separa el exterior del interior de la célula, hasta las distintas membranas que delimitan las vesículas interiores. Es interesante notar que estas membranas no generan compartimentos estancos sino que permiten el flujo de ciertos compuestos químicos, algunas veces de forma selectiva e incluso en una sola de las direcciones. Estas ideas ya fueron consideradas con anterioridad por G. Berry y V. Manca en [2] y [11], respectivamente, con fines computacionales.

En líneas generales, el sistema está basado en una *estructura de membranas*, como modelo matemático de ordenación de la célula (básicamente, una estructura de membranas es un árbol enraizado con los nodos etiquetados, donde la relación de contención entre las membranas queda reflejada en la relación padre-hijo entre los nodos). En las distintas membranas (nodos) de esta estructura se pueden encontrar elementos (*objetos* normalmente representados como símbolos de un alfabeto) y una serie de *reglas de evolución* que nos permitirán, eventualmente, modificar y/o comunicar dichos objetos entre membranas adyacentes. Los objetos introducidos en las membranas no forman propiamente un conjunto, sino que pueden aparecer repetidos (existir varias copias idénticas de un mismo elemento), formando lo que se denomina un

multiconjunto. Adicionalmente, dependiendo de las características que queramos introducir en nuestro sistema, las reglas de evolución pueden modificar ciertas características de la membrana sobre la que se están ejecutando, tales como disolución, cambio de carga eléctrica, grosor de la misma, etc., así como considerar una relación de orden entre las reglas con el fin de priorizar la ejecución de unas sobre otras.

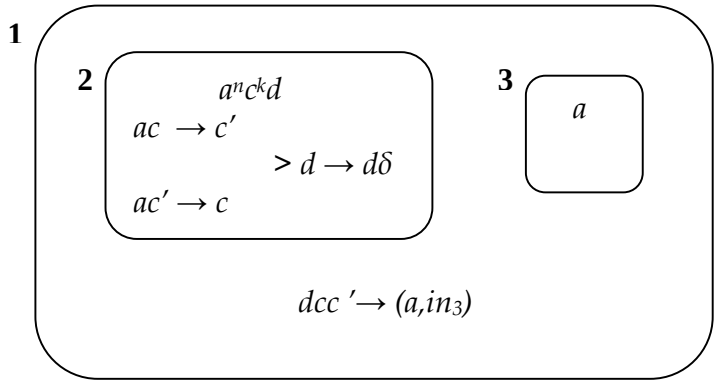

Fig 1. Ejemplo de un sistema básico: predicado de divisibilidad

En la Figura 1 podemos ver un sistema con membranas que consta de 3 membranas, donde inicialmente no tenemos objetos en la membrana 1; tenemos *n* objetos de tipo a, *k* objetos de tipo c y uno de tipo d en la membrana 2; y un objeto de tipo a en la membrana 3. Asimismo, en la membrana 1 disponemos de la regla: $dcc' \rightarrow (a,in_3)$, que interpretamos como sigue: "*si en la membrana 1 hay un objeto de tipo d, un objeto de tipo c, y otro de tipo c', entonces se genera un objeto de tipo a que se envía a la membrana 3 (y los objetos anteriores se consumen en este proceso), siempre y cuando la membrana 3 sea interior a la membrana 2*"; en la membrana 2 disponemos de 3 reglas, donde cabe resaltar la prioridad de las dos reglas de la izquierda frente a la de la derecha, y esta última se interpreta como: "*si en la membrana 2 hay un objeto de tipo d, entonces permanece igual y provoca la disolución de la membrana (acción representada por el símbolo δ)*", esta regla únicamente se puede aplicar (debido a la prioridad presente) en caso de que no se puedan aplicar aquellas de prioridad superior. En cualquier caso, la disolución de una membrana se produce tras la ejecución de todas las reglas aplicables en el sistema, y provoca la desaparición del compartimento físico junto con todas sus reglas; los objetos incluidos en dicha membrana pasan a formar parte de la membrana padre de la misma (por esta razón, la membrana más externa, conocida como *piel*, no puede aplicar nunca reglas que impliquen disolución).
En caso de que existan varias reglas excluyentes en una misma membrana que puedan ser ejecutadas en un mismo instante, el sistema actuará de manera *no determinista*; es decir, el sistema tendrá varias posibilidades de evolución, pudiéndose ejecutar cualquiera de las reglas de igual prioridad y que sean aplicables. Además, suele imponerse que las reglas deben aplicarse de forma *maximal*, en el sentido de que en cada paso de computación deben agotarse todos los objetos afectados por reglas que se puedan aplicar. Para simplificar la evolución global del sistema, supondremos que el dispositivo está sincronizado, es decir, se supone que hay un *reloj universal* que marca

las actuaciones de todos los elementos que lo integran: en un paso de reloj se considera la ejecución de todas las reglas (que pueden ser aplicadas) de cada membrana del sistema.

Partiendo de una configuración inicial (tanto de estructura de membranas como del contenido de las mismas), y aplicando las reglas de la forma descrita siguiendo los pasos del reloj universal mencionado (en cada paso todas las reglas aplicables, de forma que pasamos de una configuración a otra) obtenemos lo que se denomina una *computación*, que refleja la evolución del sistema. Diversas interpretaciones sobre configuraciones de parada (por ejemplo, que no se pueda aplicar ninguna regla en ninguna membrana) y obtención de resultados (por ejemplo, el multiconjunto presente en una cierta membrana destacada al parar la computación), así como la introducción de membranas de entrada, nos permiten definir formalmente estos dispositivos como máquinas de cálculo, generadoras de lenguajes, reconocedoras, etc… (en [17] puede encontrarse una formalización matemática completa de la variante original de estos sistemas de computación).

Por ejemplo, en el sistema que se muestra en la figura 1, si consideramos como salida el contenido de la membrana 3 cuando el sistema para (es decir, no sigue evolucionando porque no tiene más reglas aplicables), obtenemos un dispositivo que calcula el predicado de divisibilidad entre los valores de n y k.

En general, se ha demostrado que estos sistemas son computacionalmente universales y, debido al paralelismo masivo que poseen, en ellos se pueden diseñar dispositivos que proporcionan soluciones eficientes a problemas computacionalmente duros (ver [13]). Desde su creación hasta la actualidad han surgido diversas variantes entre las que destacamos la de computación *tissue*, en la que las conexiones del sistema crece en complejidad hasta considerar un tejido de células del mismo tipo (habría dos niveles de conexión, en cada célula las membranas estarían comunicadas entre sí con una estructura de árbol, y a nivel global las células podrían quedar comunicadas sobre una estructura de grafo más compleja).

A pesar de que el modelo tiene inspiración biológica, debe resaltarse el hecho de que constituye igualmente un buen modelo teórico de computación distribuida, en donde distintas unidades de cálculo trabajan independientemente pero estructuradas en una cierta jerarquía vertical. Asimismo, sistemas dinámicos complejos, como aquellos que surgen en el estudio de la dinámica de poblaciones, pueden ser también interpretados como P sistemas en los que los individuos de las distintas especies interaccionan entre sí dentro de hábitats que permiten el traspaso de los mismos.

En la sección siguiente estudiaremos un modelo que sigue esta estructura de tejido, aunque con ciertas particularidades y restricciones que la diferencian de otras variantes, ya que usan a nivel local un sistema muy simple de estructura de células con una membrana y sólo poseen un tipo de elemento.

3. Un modelo desde la neurociencia: Spiking Neural P Systems

En los últimos años, las investigaciones en neurofisiología han demostrado que muchos sistemas neuronales biológicos utilizan la duración del potencial de acción (o *spikes*) para codificar la información a transmitir entre neuronas. En base a esto, los modelos matemáticos de los *spiking neurons* intentan simular dicho comportamiento, y su utilización computacional está siendo ampliamente investigada en la actualidad ([5], [10])

En 2005 fue introducida una variante de la computación con membranas con el objetivo de definir un modelo de computación basado en ideas específicas de los spiking neurons ([9]), conocida como *Spiking Neural P systems* (SNP, para abreviar).
Brevemente, un SNP está formado por un conjunto de neuronas (que vamos a interpretar como celulas simples que constan de una sola membrana) situadas en los nodos de un grafo (un estructura más complicada que los sistemas celulares habituales) y que envían señales (*spikes*, que en lo que sigue serán denotadas por el símbolo a) a lo largo de las sinapsis (representados por las aristas dirigidas del grafo). Por tanto, su estructura sería similar a la de un sistema celular de tipo *tissue* en el que todas las células que lo componen son muy simples y contienen un solo tipo de objeto.

En este sistema, y de forma análoga a como se trabaja en otros sistemas de computación celular, los objetos evolucionan por medio de reglas, de las que hay dos tipos:

- Las *reglas spiking* (también llamadas *firing*), que son de la forma $E/a^c \rightarrow a;d$, donde E es una expresión regular sobre $\{a\}$ y c,d son números naturales, con $c \geq 1$, $d \geq 0$. La interpretación de tal regla es que una neurona que contenga k spikes con a^k verificando la expresión regular E y $k \geq c$, puede consumir c spikes (k-c permanecen en la neurona) y producir 1 spike tras un retardo de d pasos; si $d=0$, entonces el spike es emitido inmediatamente; si $d=1$ el spike se emite en el paso siguiente; en el caso $d \geq 1$, si la regla ha sido usada en el paso t, entonces en los pasos t, $t+1$, $t+2,\ldots,t+d-1$ la neurona estará cerrada y no puede recibir nuevos spikes (si una neurona tiene una sinapsis a otra que está cerrada e intenta enviarle un spike, éste se pierde), en el paso $t+d$ la neurona se vuelve a abrir y puede volver a recibir spikes (que pueden ser usados en el paso $t+d+1$). Un spike emitido por una neurona se replica y es enviado a todas las neuronas a las que se puede llegar desde ella por medio de las sinapsis dirigidas.

- Las *reglas forgetting*, que son de la forma $a^s \rightarrow \lambda$, y que se interpretan como que, supuesto que la neurona contiene exactamente s spikes, todos ellos desaparecen.

Al igual que en la variante presentada en la sección anterior, el sistema trabaja sincronizada, maximal y paralelamente, de forma que en cada unidad de tiempo, cada neurona que pueda aplicar una regla debe usarla, sin embargo, dentro de cada neurona

el comportamiento es secuencial, de forma que sólo (y a lo sumo) se usa una regla de las que tenga.

Además, una de las neuronas se considera *neurona de salida*, y sus spikes también son enviados al entorno. Los instantes de tiempo en que esta neurona envía spikes al exterior se marcan con un 1, mientras que aquellos en los que no hay envío se marcan con 0. La sucesión que se obtiene durante la evolución del sistema (que puede ser infinita) se denomina *spike train*.

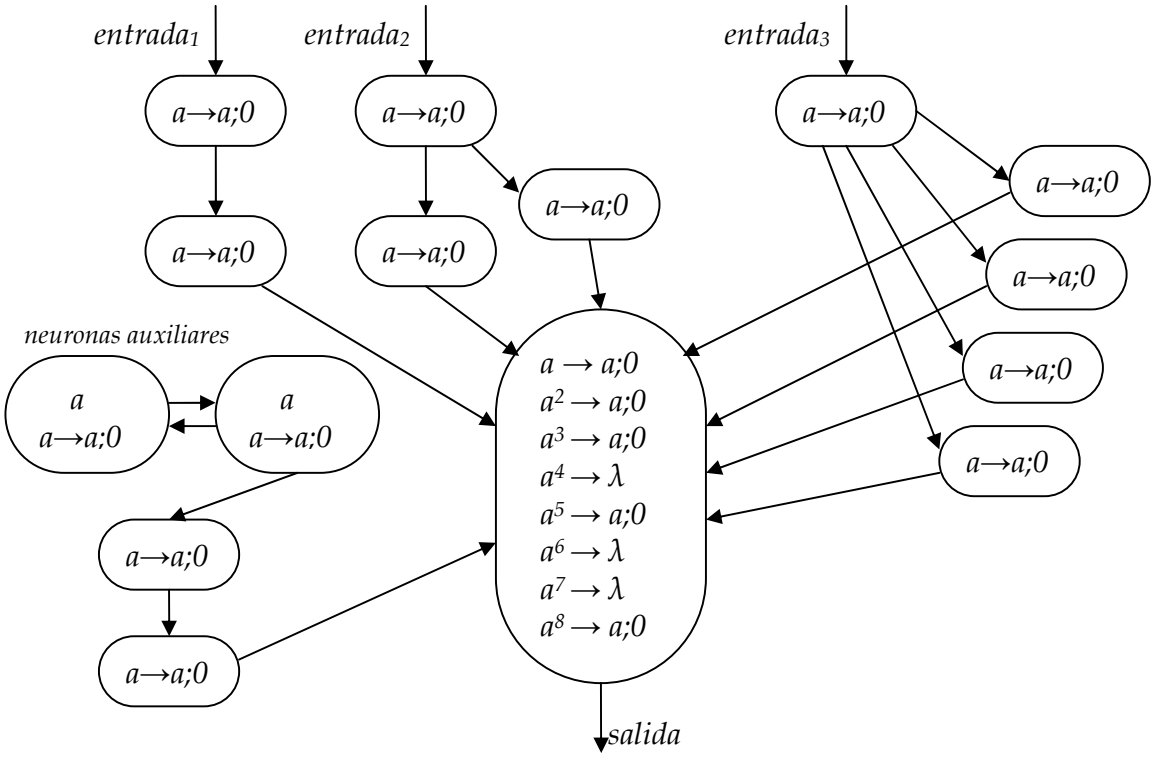

Fig 2. Ejemplo de un SNP que calcula una función booleana

Siguiendo el espíritu del modelo biológico en el que se inspira, la primera interpretación es que el resultado de una computación viene codificado en la distancia entre spikes consecutivos que se envían al entorno por la neurona de salida. Otras interpretaciones consideran solo la distancia entre los dos primeros spikes que se obtienen (o una generalización considerando las distancias entre los k primeros); añaden neuronas de entrada para crear sistemas reconocedores de números (codificando un número natural de entrada como la distancia entre dos spikes consecutivos que se introducen en dichas neuronas, y aceptando dicho número si el sistema para); etc. Una variante muy interesante consiste en tomar como resultado el propio spike train de salida, es decir, interpretando el sistema como dispositivo generador de lenguajes binarios (si añadimos, además, neuronas de entrada, el sistema puede funcionar como un traductor).

Como resultado ejemplificador de la versatilidad y potencia de esta variante presentamos el siguiente resultado de [14]:

Teorema. *Toda función, $f:\{0,1\}^k \rightarrow \{0,1\}$, puede ser calculada por un SNP (en su variante traductora) con k neuronas de entrada (y que además usa 2^k+4 neuronas, una de ellas como neurona de salida).*

El ejemplo mostrado en la figura 2 calcula la función $f:\{0,1\}^3 \rightarrow \{0,1\}$ definida por:

$$f(b_1,b_2,b_3)=1 \text{ si y sólo si } b_1+b_2+b_3 \neq 2$$

Donde, las tres neuronas de entrada son alimentadas de forma continua con los valores b_1, b_2, b_3, y la neurona de salida proporcionará, con un retraso de 3 pasos, el valor de $f(b_1,b_2,b_3)$.

En general, en los resultados que se han obtenido relacionados con los SNP se usa toda la potencia que el modelo proporciona, por ejemplo, se ha probado que son computacionalmente completos en el caso en que no se imponga un límite al número de spikes que pueden estar presentes en el sistema, y una caracterización de los conjuntos semilineales en caso de que dicha cota exista. Normalmente, las pruebas realizadas hasta ahora se basan en la simulación de máquinas de registros, lo que permite (simulando pequeñas máquinas universales de registros) dar una cota superior acerca del número de neuronas necesarias para obtener una SNP universal (que, en la actualidad, rondan las 80 neuronas).

Sin embargo, puede ser interesante realizar un estudio de las características mínimas que necesitamos para obtener la misma potencia computacional, no ya con respecto al número de neuronas que hemos de usar o a la limitación de los spikes, sino a la complejidad de las reglas o del grafo subyacente. En este contexto, resaltamos el siguiente resultado, que se obtiene como combinación de dos resultados aparecidos en [14] y [8], respectivamente, en el que, además de debilitar el tipo de reglas que se pueden usar en el diseño de los sistemas, establece un grado máximo de salida y de entrada para las conexiones sinapsis de las neuronas:

Teorema. *La universalidad de los SNP puede ser obtenida para sistemas:*

i. *que no usan retardos en las reglas,*
ii. *que no usan reglas forgetting,*
iii. *que usan únicamente expresiones regulares de la forma a^+ y a^k, con $k \geq 1$, en las reglas spiking,*
iv. *cada una de las restricciones i. a iii. combinada con la restricción de tener un grafo de sinapsis con grado de salida menor o igual a 2.*
v. *cada una de las restricciones i. a iii. combinada con la restricción de tener un grafo de sinapsis con grado de entrada menor o igual a 2.*

4. Conclusiones

En este trabajo hemos presentado un ejemplo de cómo la inspiración biológica puede proporcionar interesantes modelos de computación. Aunque hemos nombrado la Computación Molecular basada en ADN, debido a las restricciones lógicas de espacio no hemos entrado en profundidad en sus características y proyecciones de futuro, sino que hemos preferido centrar el cuerpo del artículo en mostrar las características de uno de los más prometedores modelos de computación inspirados en la célula, así como en una de sus variantes inspiradas en el comportamiento de las redes de neuronas.

Por supuesto, una vez formalizado un modelo de computación se abren multitud de cuestiones acerca de sus propiedades computacionales, de complejidad, de diseño, etc. En muchas de las variantes de computación celular se ha desarrollado una teoría completa de complejidad ([15] y [16]), un análisis casi exhaustivo de sus capacidades computacionales y se ha dado un compendio bastante amplio de diversos problemas interesantes que pueden ser resueltos en ellas [3].

La variante de *Spiking Neural P Systems* es relativamente nueva, por lo que muchas de las cuestiones anteriores están por resolver en este marco. Una cuestión importante es acerca de la eficiencia de este nuevo modelo en la resolución de problemas computacionalmente duros. Habitualmente, la resolución de problemas NP-completos en tiempo polinomial ha sido posible en la computación con membranas gracias a la capacidad de producir una cantidad exponencial de objetos en tiempo lineal, sin embrago, en los SNP no disponemos de dicha capacidad, ya que esta producción está limitada a un factor polinomial en el tiempo, ¿cómo podríamos introducir nuevas posibilidades en el modelo para conseguir esta característica fundamental?

Inspirándose en el hecho de que el cerebro está formado por una cantidad inmensa de neuronas de las cuales solo se usa una pequeña parte, en [4] se propone una forma de atacar problemas computacionalmente duros suponiendo que disponemos de un SNP arbitrariamente grande, con una estructura lo más regular posible y sin spikes en su interior (podríamos decir que en estado latente). Para resolver el problema, podemos introducir una cantidad polinomial de spikes en las membranas de entrada, entonces, por el movimiento de los spikes a lo largo de las sinapsis, el sistema se auto-activa y proporciona una computación que resuelve el problema para esa instancia concreta. Esta forma de resolver problemas no es nada habitual en la computación, y hasta ahora no disponemos ni de formalización de la misma, ni de definiciones de clases de complejidad en este marco, pero tanto por la motivación biológica como por su interés inherentemente computacional, creemos que es una propuesta interesante que merece la pena ser investigada.

Adicionalmente, debido a la relación que tiene el modelo SNP con las redes neuronales reales, surgen cuestiones nuevas que puede ser interesante abordar, como por ejemplo, ¿de qué forma afecta la presencia de una compleja red de neuronas a la capacidad generativa del sistema? (en general, es una cuestión no abordada la influencia del grafo subyacente sobre la capacidad del sistema), ¿puede ocurrir que a partir de cierta

complejidad emerja una capacidad generativa que no es posible conseguir a priori?, es decir, al igual que ocurre con un sistema neuronal complejo, ¿podría emerger un comportamiento cualitativamente más complejo en el sistema que no fuera predecible por el modelo?, quizás esta variante podría servir como modelo teórico para el estudio de la emergencia de características complejas que se observan en los sistemas naturales.

Bibliografía

1. ADLEMAN, L. Molecular Computation of Solutions to Combinatorial Problems. *Science*, 268 (Noviembre 1994), 1021-1024.
2. BERRY, G.; BOUDOL, G. The chemical abstract machine. *Theoretical Computer Science*, 96 (1992), 217-248.
3. CIOBANU, G.; PĂUN, GH.; PÉREZ JIMÉNEZ M.J. (eds.) *Applications of Membrane Computing*, Natural Computing Series, Springer-Verlag, ISBN 3-540-25017-4, 2006, X.
4. CHEN, H.; IONESCU, M.; ISHDORJ, T.-O. On the efficiency of spiking neural P systems. En [11], Vol. I, 195-206.
5. GERSTNER, W.; KISTLER, W. *Spiking Neuron Models. Single Neurons, Populations, Plasticity*. Cambridge Univ. Press, 2002.
6. GUTIÉRREZ NARANJO ET AL., eds: *Proceedings of Fourth Brainstorming Week on Membrane Computing*, Febrero 2006, Fenix Editora, Sevilla, 2006.
7. HEAD, T. Formal Language Theory and DNA: an analysis of the generative capacity of specific recombinant behaviours. *Bulletin of Mathematical Biology*, 49 (1987), 737-759.
8. IBARRA, O.H.; PĂUN, A.; PĂUN, GH.; RODRÍGUEZ PATÓN, A.; SOSIK, P.; WOODWORTH, S. Normal forms for spiking neural P systems. En [11], Vol. II, 105-136.
9. IONESCU, M.; PĂUN, GH; YOKOMORI, T. Spiking neural P systems. *Fundamenta Informaticae*, 71, 2-3 (2006), 279-308.
10. MAASS, W.; BISHOP, C., eds. *Pulsed Neural Networks*, MIT Press, Cambridge, 1999.
11. MANCA, V. String rewriting and metabolism: A logical perspective. Co*mputing with Bio-Molecules*. Theory and Experiments, Springer-Verlag, Singapore, 1998, 36-60.
12. PĂUN, GH. Computing with membranes. *Journal of Computer and System Sciences*, 61, 1 (2000), 108-143, y *Turku Center for Computer Science – TUCS Report Nº 208*, 1998 (www.tucs.fi).
13. PĂUN, GH. *Membrane Computing. An introduction*. Springer-Verlag.
14. PĂUN, GH; PÉREZ JIMÉNEZ M.J.; ROZENBERG, G. Infinite spike trains in spiking neural P systems. Submitted 2005.
15. PÉREZ JIMÉNEZ M.J.; ROMERO JIMÉNEZ, A.; SANCHO CAPARRINI, F. *Teoría de la Complejidad en Modelos de Computación Celular con Membranas*, Editorial Kronos, ISBN 84-86273-57-9, 2002, II.
16. PÉREZ JIMÉNEZ M.J.; ROMERO JIMÉNEZ, A.; SANCHO CAPARRINI, F. Complexity classes in cellular computing with membranes. *Natural Computing*, Kluwer Academic Publishers, 2, 3 (2003), 265-285.
17. PÉREZ JIMÉNEZ M.J.; SANCHO CAPARRINI, F. A formalization of transition P systems. *Fundamenta Informaticae*, vol. 49 (2002), 261-272.