Tesis defendida por

Hugo Armando Guillén Ramírez

y aprobada por el siguiente comité

Dr. Israel Marck Martínez Pérez

Director del Comité

Dr. Carlos Alberto Brizuela Rodríguez

Miembro del Comité

Dr. José Alberto Fernández Zepeda

Miembro del Comité

Dr. Miguel Ángel del Río Portilla

Miembro del Comité

Dr. José Antonio García Macías

Coordinador del Programa de
Posgrado en Ciencias de la Computación

Dr. Jesús Favela Vara

Director de Estudios de Posgrado

Enero de 2014

CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE EDUCACIÓN SUPERIOR DE ENSENADA, BAJA CALIFORNIA

Programa de Posgrado en Ciencias
en Ciencias de la Computación

Diseño de un sistema P de tejido y un algoritmo molecular para la resolución del problema MAX-CLIQUE

Tesis
para cubrir parcialmente los requisitos necesarios para obtener el grado de

Maestro en Ciencias

Presenta:

Hugo Armando Guillén Ramírez

Ensenada, Baja California, México
2014

Resumen de la tesis de Hugo Armando Guillén Ramírez, presentada como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación. Enero de 2014.

Diseño de un sistema P de tejido y un algoritmo molecular para la resolución del problema MAX-CLIQUE

Resumen aprobado por:

Dr. Israel Marck Martínez Pérez

Director de Tesis

A pesar de que el cómputo convencional se ha estudiado ampliamente, existen clases de problemas altamente complejos tales como el problema MAX-CLIQUE que no se puede atacar de manera exacta utilizando el paradigma tradicional de cómputo. En este estudio se presenta un sistema P de tejido para la resolución del problema MAX-CLIQUE y el diseño de un algoritmo de cómputo con ADN inspirado en el sistema anterior que no requiere de una biblioteca combinatoria como entrada. El sistema propuesto mejora el tiempo de ejecución contra el sistema existente en la literatura, mientras que el algoritmo molecular se presenta altamente competitivo ante aquellos del estado del arte.

Palabras Clave: **Sistemas P de Tejido, Cómputo con ADN, MAX-CLIQUE, NP-Completo**

Abstract of the thesis presented by Hugo Armando Guillén Ramírez, in partial fulfillment of the requirements of the degree of Master in Sciences in Computer Science. January 2014.

The design of a tissue P system and a molecular algorithm for the resolution of the MAX-CLIQUE problem

Abstract approved by:

Dr. Israel Marck Martínez Pérez

Thesis director

Conventional computation has been extensively studied. However, there are kinds of highly complex problems such as the MAX-CLIQUE problem that can not be solved in an exact way using the traditional paradigm of computation. In this study, we introduce a tissue P system to solve the MAX-CLIQUE problem as well as a DNA computing algorithm inspired by the previous system that does not require a combinatorial library as input. The proposed tissue P system improves the running time of the existing approach in the literature, while the latter results in a highly competitive algorithm when compared to those of the state of the art.

Keywords: **Tissue P-Systems, DNA Computing, MAX-CLIQUE, NP-Complete.**

Dedicatoria

A mis padres y hermanos.

Agradecimientos

Agradezco a mi familia, por todo su amor y apoyo incondicional.

A Shadai, por toda tu paciencia, cariño y apoyo a lo largo de todo este tiempo.

A mi director de tesis, el Dr. Israel Marck Martínez Pérez, quien tiene la culpa de despertar mi interés en la apasionante área que es el cómputo molecular. Gracias por su guía, confianza y enseñanzas.

A los miembros de mi comité de tesis, el Dr. Carlos Alberto Brizuela Rodríguez, el Dr. José Alberto Fernández Zepeda y el Dr. Miguel Ángel del Río Portilla, por su tiempo, observaciones y sugerencias durante el desarrollo de este trabajo.

A los investigadores y mis compañeros del posgrado del departamento de Ciencias de la Computación, gracias por las enseñanzas y experiencias durante esta estancia.

A mis amigos, en especial a Polo, Mayra, Saulo y David, gracias por estar al pie del cañón en todo momento, y al Maestro Carlos Hernández y la Maestra Claudia Quezada, gracias por animarme a entrar al campo de la investigación.

A CICESE por permitirme estudiar este posgrado, y al personal administrativo por la excelente atención que siempre me brindaron.

Al CONACyT por su apoyo económico para realizar mis estudios.

Contenido

Página

Página

Lista de Figuras

Figura Página

Lista de Tablas

Tabla Página

El cómputo no convencional es un área de estudio multidisciplinaria dedicada a explorar nuevos modelos de computación y superar las limitaciones del modelo tradicional. Uno de estos límites reside en los problemas combinatorios ya que no se pueden resolver de manera exacta en un tiempo polinomial. A diferencia de su contraparte, los modelos de computación no convencionales se alejan de la lógica de transistores y buscan inspiración en cualquier fenómeno que involucre un procesamiento. Ejemplos conocidos de estos modelos son el cómputo cuántico, el cómputo mecánico, el cómputo molecular y los sistemas de membrana. El cómputo molecular explota las propiedades de las moléculas de ADN y ARN, mientras que los sistemas de membrana, o sistemas P, se inspiran en la transformación de sustancias químicas dentro de la célula (Calude y Păun, 2001).

Las células nos mantienen vivos, tanto por los procesos que realizan en su interior como por su interacción con el entorno. Una célula se compone de organelos y dentro de ellos se procesan moléculas según lo requiera la célula. Los sistemas P son una abstracción de las células, donde las moléculas se representan con objetos discretos tales como símbolos, cadenas y grafos, y son procesados dentro de unidades que representan a los organelos, llamados membranas. Suponiendo que existiera la tecnología necesaria para crear células sintéticas que realicen un conjunto de reacciones preestablecidas, ¿cómo se pueden resolver problemas con dichas células?. El modelo de sistemas de tejido es una variante de sistemas P que se inspira en las relaciones célula a célula. Esta clase de sistema resulta interesante debido a que la estructura de las redes de comunicación entre las células de membranas es capaz de modelar directamente grafos (Martín-Vide *et al.*, 2003), por lo que se considera como una opción para resolver de manera teórica problemas en grafos.

1.1. Conceptos básicos

1.1.1. Sistemas P

Un sistema P o sistema de membranas es un modelo de cómputo no convencional basado en las células donde, en su sentido más simple, la computación se realiza mediante la transformación de objetos mediante reglas de transición. El sistema P básico es llamado sistema

P de transición. En este sistema existe una estructura de membranas con reglas de transición asociadas. El sistema cuenta con objetos iniciales, y a lo largo de los pasos de la computación se van transformando y comunicando entre las demás membranas del sistema.

En los sistemas P de transición, cada membrana está comunicada al menos con su membrana madre, y con sus membranas hijas en caso de que existan. En otros modelos, por ejemplo, los sistemas P de tejido, estos canales de comunicación se pueden establecer entre membranas (células) arbitrarias, e incluso se puede lograr que sean los canales de comunicación los que jueguen un papel determinante en la computación, por ejemplo en los sistemas de canales de estado.

La inspiración biológica de los sistemas P es el procesamiento de sustancias químicas dentro de las células, el cuál no ocurre en una única molécula a la vez, si no que se hace de manera paralela a todas las moléculas disponibles o que requiera procesar la célula, por lo que se dice que los sistemas P tienen un paralelismo inherente. Es posible aplicar un conjunto de reglas a un objeto dado en un instante de tiempo, sin embargo, de manera predeterminada se supone que solamente una de esas reglas se aplica. La selección de esta regla es no determinista.

Los sistemas P son Turing-completos, debido a que tienen al menos el mismo poder que una máquina de Turing. Al realizar la prueba de completitud de Turing en nuevos modelos de sistemas P (la emulación de una máquina de Turing), se parte de máquinas de Turing universales o dispositivos equivalentes, por lo que el resultado es un sistema P universal, es decir, capaz de emular cualquier otro sistema P siempre que lo reciba como entrada en la codificación que acepta el sistema universal (Păun, 2006; Jiménez *et al.*, 2003). Actualmente, los sistemas P han llamado la atención de los investigadores al modelar problemas en otras áreas tales como economía (Păun *et al.*, 2010).

1.1.2. Cómputo biomolecular

El cómputo biomolecular desarrolla modelos de computación apoyado por las características propias de las moléculas de ácido desoxirribonucleico (ADN) y ácido ribonucleico (ARN). Según el grado de intervención humana, se pueden dividir los modelos de cómputo molecular en autónomos, no autónomos e *in vivo*, también se puede considerar la clasificación constructivos y no constructivos. Un algoritmo es un procedimiento computacional bien definido que

toma un conjunto de valores como entrada y produce un conjunto de valores como salida. Los algoritmos moleculares son algoritmos que utilizan ADN o ARN como entradas, salidas, o como una herramienta para realizar el procesamiento. De manera general, un algoritmo no constructivo necesita una entrada inicial con todas las soluciones posibles de salida,a partir de las cuales descarta soluciones no válidas. Para el caso de un problema combinatorio, es necesario proporcionar a estos algoritmos una cantidad exponencial de ADN para que no se pueda descartar ninguna solución. Esto conlleva ciertos problemas técnicos, tales como la posibilidad de generar en laboratorio esa cantidad de cadenas de ADN distintas. En cambio, los modelos constructivos parten de unas cadenas de ADN diseñadas *a priori*, las cuales interactúan entre sí para generar soluciones mayores, *e.g.* el autoensamblado de ADN, donde las cadenas de ADN se diseñan de tal manera que formen mosaicos que se ensamblan entre sí.

Los sistemas de membrana y el cómputo molecular guardan una relación sinérgica debido a su fuente de inspiración. Sin embargo, mientras que el primero es meramente teórico, el segundo se ha implementado en laboratorio (Adleman, 1994) e incluso automatizado mediante dispositivos microfluídicos (McCaskill, 2001).

1.1.3. El problema MAX-CLIQUE

Un clique en un grafo es un subconjunto de sus vértices tal que para cualquier par de ellos existe una arista que los conecta. De manera formal, siendo $G = (V, E)$ un grafo no dirigido, donde $V = \{v_1, \ldots, v_n\}$ es el conjunto de vértices y $E \subseteq (V \times V)$ es el conjunto de aristas, $V' \subseteq V$ es un clique sí y sólo si para cada par de vértices $u, v \in V'$ existe una arista $(u, v) \in E$ (Cormen *et al.*, 2011).

De manera burda, un problema NP-Completo es aquel que no se sabe como resolverlo en tiempo polinomial. El problema MAX-CLIQUE pertenece a esta clase de problemas (Karp, 1972).

1.2. Antecedentes

Para que un sistema P sea capaz de resolver un problema NP-Completo es necesario que tenga el poder de generar un espacio de búsqueda exponencial, por lo que existe un compromiso entre la eficiencia en tiempo y en espacio. La eficiencia en tiempo (lineal o polinomial)

se logra mediante las operaciones de división, separación y creación de membranas, así como la replicación de cadenas en el caso de los sistemas basados en células. Para los sistemas de tejido se cuenta con la división de células y la replicación de objetos. Un enfoque inspirado en cómputo molecular es utilizar un espacio de trabajo exponencial calculado antes de la computación.

La resolución de problemas combinatorios se presenta en la literatura como una prueba del poder y eficiencia de un nuevo modelo de sistema de membrana. Se han propuesto soluciones a problemas clásicos como son SAT, k-SAT y QSAT (Păun, 2001; Obtulowicz, 2001; Alhazov *et al.*, 2003), el Camino Hamiltoniano (Krishna y Rama, 1999; Zandron *et al.*, 2000; Castellanos *et al.*, 2000; Mutyam y Krithivasan, 2001; Krishna y Rama, 2001; Păun *et al.*, 2004; Pan y Alhazov, 2006), el problema de la Suma de Subconjuntos (Graciani-Díaz y Riscos-Núñez, 2005; Gutiérrez-Naranjo *et al.*, 2005; Díaz-Pernil *et al.*, 2007; Leporati y Gutiérrez-Naranjo, 2008), el problema del Almacenamiento Mínimo (Leporati y Pagani, 2006), e incluso al problema de romper el cifrado DES (Krishna y Rama, 2003).

A pesar de que la implementación biológica debiera ser la más directa debido al origen del modelo, se han logrado pocos avances en este sentido, siendo la lógica de liposomas el único enfoque abordado a la fecha (Smaldon *et al.*, 2010). Dado el paralelismo inherente a los modelos de membranas, el cómputo paralelo ha sido el medio para la implementación *in silico*. Existe una revisión extensa de estas implementaciones sobre GPUs (Cecilia *et al.*, 2012).

A la fecha se ha abordado la implementación de los sistemas de membrana en computadoras moleculares (de manera teórica) para tres problemas: el problema del camino hamiltoniano (Ledesma *et al.*, 2004), el problema de la supercadena común más corta (Ledesma *et al.*, 2005), el problema del cliqué máximo (García-Arnau *et al.*, 2007b), y el problema de la construcción de grids de calendarización (García-Arnau *et al.*, 2007a).

El problema MAX-CLIQUE se ha abordado mediante sistemas P de transición con inhibidores por García-Arnau *et al.* (2007b). En este sistema se utiliza un sistema de transición de n membranas trabajando secuencialmente según su jerarquía, es decir, su nivel de anidamiento. Respecto al cómputo molecular, se ha propuesto el uso de diversos modelos: filtrado por marcar y destruir (Ouyang *et al.*, 1997), cómputo acuoso y *corte y empalme* (Head *et al.*, 1999), filtrado por dispositivos microfluídicos (McCaskill, 2001; García-Arnau *et al.*, 2007b)

y sistemas de etiquetas (Martínez-Pérez y Zimmermann, 2009). A excepción del algoritmo de García-Arnau *et al.* (2007b), estos algoritmos necesitan como entrada todos los cliques posibles en el grafo a resolver.

1.3. Motivación

Las investigaciones iniciales en sistemas P se enfocaron en descubrir las capacidades del modelo desde el punto de vista de poder computacional, donde como resultado se obtuvo que los sistemas P son Turing-completos (son igual de poderosos que una máquina de Turing) y universales (es posible simular sistemas P mediante un *sistema P universal*). Las aplicaciones de los sistemas P se investigaron relativamente tarde; sin embargo, actualmente es un área activa donde se tratan problemas de biología, biomedicina, optimización, economía y ciencias de la computación (Păun *et al.*, 2010). Dentro de estas aplicaciones, los sistemas de membrana representan una plataforma para el modelado gracias a sus características tales como el no determinismo en la aplicació de las reglas, paralelismo masivo, escalabilidad (posibilidad de resolver problemas cada vez mayores), extensibilidad (el poder agregar reglas a los modelos) y comunicación, entre otras (Păun, 2006).

Gracias a sus características de paralelismo y cómputo distribuido, los sistemas P se perfilan como una herramienta para la solución exacta o aproximada de problemas NP-Completos (Frisco, 2009), tales como el problema MAX-CLIQUE. Este problema es importante debido a que modela a otros presentados en la práctica (Bomze *et al.*, 1999), es decir, al resolver el problema MAX-CLIQUE, es posible encontrar la solución a una amplia gama de problemas. Por ejemplo, dentro del área de la bioquímica computacional y la genómica es necesario un modelo de detección de cliques (clique maximal, clique máximo, clique con máximo peso y todos los cliques) para la resolución de problemas como el mapeo de datos genómicos, alineamientos locales sin superposición, modelado comparativo de estructuras moleculares y el acoplamiento de proteínas (Butenko y Wilhelm, 2006).

Se demostró que el cómputo molecular no es una solución escalable para los problemas combinatorios (Hartmanis, 1995), y además no es posible que sustituya a las computadoras actuales para la resolución de problemas polinomiales debido al tiempo y alta tasa de errores de los procedimientos biológicos. Sin embargo, el diseño de nuevos algoritmos moleculares puede inspirar mecanismos aplicados al campo biomédico. Ejemplos de esto son los autómatas

basados en ADN para la detección de mutaciones a nivel molecular (Martinez-Perez *et al.*, 2007).

1.4. Planteamiento del problema

En el sistema P propuesto por García-Arnau *et al.* (2007b) se utiliza un sistema de transición de n membranas trabajando secuencialmente según su jerarquía, es decir, su nivel de anidamiento. Este diseño desaprovecha el paralelismo de los sistemas P, debido a que las membranas trabajan en secuencia. En los sistemas de tejido las células trabajan de manera independiente y se pueden comunicar arbitrariamente entre ellas, con lo que se podría aprovechar mejor el paralelismo de los sistemas P en la resolución del problema MAX-CLIQUE. Por otro lado, un algoritmo molecular inspirado en este sistema no necesitaría de todas las soluciones posibles como entrada, por lo que es necesario evaluar el impacto que tiene esta inspiración.

1.5. Objetivos de la investigación

1.5.1. Objetivo general

Diseñar un sistema de membranas que resuelva un problema combinatorio mediante un enfoque constructivo, así como un algoritmo molecular que emule el procesamiento de dicho sistema.

1.5.2. Objetivos específicos

- Definir un problema combinatorio apto para la resolución con sistemas de membranas. En este objetivo se plantea revisar la literatura de sistemas de membranas, y en base a un modelo seleccionado, buscar la clase de problemas que se adapten a este esquema de solución. Al realizar este objetivo, uno de los mayores retos consistió en la gran diversidad de sistemas de membrana existentes, dado que se generan sistemas muy distintos según sus componentes principales (tipos de reglas, modos de comunicación, manejo en las membranas, tipos de objetos, entre otros). Como resultado de este objetivo se optó por una variante más general, los sistemas de tejido, y la clase de problemas a resolver fueron los problemas combinatorios en grafos y finalmente, el problema seleccionado

fue el MAX-CLIQUE.

- Resolver el problema combinatorio seleccionado mediante un modelo de sistema de membranas. Para la resolución de este objetivo, se partió de la suposición de que el sistema de tejido mapea el grafo a resolver. Posteriormente, se definió la estructura de los objetos, para después diseñar las reglas para la resolución del problema. Mediante diferentes estrategias en las reglas, se seleccionó un conjunto tal que mantuviera la propiedad de cliques en los objetos del sistema, y al generarse uno nuevo, se reportara como solución al entorno del sistema. Un reto importante en este objetivo fue la verificación formal de su funcionamiento, el cual se basó en la demostración de que los objetos del sistema son cliques válidos mientras permanezcan en el sistema. La demostración se realizó mediante técnicas de comprobación de propiedades de conjuntos, donde dichos conjuntos son el contenido de las células en determinado instante.

- Modelar la solución al sistema de membranas utilizando cómputo molecular. Este objetivo plantea diseñar un algoritmo molecular inspirado en el sistema de membranas propuesto. Antes de describir el algoritmo como tal, se realizó un paso intermedio, donde se simuló el comportamiento del sistema mediante algoritmos paralelos. Una vez que se contó con los algoritmos, se buscó en la literatura las operaciones existentes de cómputo molecular que se adaptaran a las operaciones requeridas, las cuales son el filtrado (discriminación de objetos), la concatenación (para incrementar el tamaño de los cliques) y la replicación (para copiar y posteriormente comunicar el contenido entre células).

- Realizar una comparación de los algoritmos propuestos con los existentes en la literatura. Mediante este objetivo se evalúan las ventajas del enfoque que se utilizó para resolver el problema MAX-CLIQUE contra las propuestas del estado del arte. El resultado principal es que se superó el tiempo de ejecución del sistema P que resuelve el problema (García-Arnau *et al.*, 2007b). El tiempo de ejecución en el sistema de tejido y el algoritmo molecular propuestos depende únicamente del tamaño del clique máximo del grafo ($\omega(G)$), el cual es $O(n)$, lo cual es una ventaja sobre los otros enfoques cuyo tiempo de ejecución depende del número de aristas.

1.6. Metodología

Para cumplir con los objetivos propuestos, se siguió la metodología que se describe a continuación. Para la inmersión en el área de sistemas P se revisaron las monografías de Calude y Păun (2001), Frisco (2009) y Păun *et al.* (2010). A partir de la revisión de literatura, se seleccionó el modelo de sistema de tejido debido a su aplicación directa a los grafos. Posteriormente, de entre los problemas combinatorios con grafos se eligió el problema MAX-CLIQUE debido a que había un punto de comparación con la literatura. Al resolver el problema, se comenzó con la suposición de que se tiene un sistema de tejido cuya estructura es idéntica al grafo a resolver. Entonces, se plantearon las siguientes preguntas: ¿qué características deben tener los objetos que circulan por el sistema?, y ¿qué reglas son necesarias para validar que los objetos sean cliques?. Referente a la primer pregunta, se tomó que los objetos del sistema son conjuntos de vértices, donde los vértices son símbolos de un alfabeto formado por todos los vértices del grafo. Para la segunda pregunta, se consideraron distintas opciones: eliminar vértices de los objetos, agregar todos los vértices adyacentes, y finalmente, tomar como ventaja que en los canales de comunicación ya estaba codificada la adyacencia, por lo que las reglas se centraron en validar que los objetos del sistema fueran cliques para después incrementar dichos objetos en tamaño sólo si formaban cliques. Una vez que se describió el sistema, se verificó que resolvía el problema mediante una demostración formal. Para la demostración se utilizaron las técnicas clásicas de inducción y contradicción, pero aplicadas al contenido de las células en determinado instante. A partir del sistema verificado, se diseñaron una serie de algoritmos paralelos que simulan el comportamiento del sistema de tejido. En base a estos algoritmos, se diseñaron los algoritmos moleculares para la resolución del problema MAX-CLIQUE. Finalmente, se compararon tanto el sistema de tejido propuesto con el de García-Arnau *et al.* (2007b) y el algoritmo molecular con los existentes en la literatura (Ouyang *et al.*, 1997; Head *et al.*, 1999; McCaskill, 2001; García-Arnau *et al.*, 2007b; Martínez-Pérez y Zimmermann, 2009).

1.7. Organización de la tesis

A continuación se explica la organización del presente trabajo. En el Capítulo 2 se proveen los conceptos básicos de matemáticas, computación, biología, cómputo biomolecular y siste-

mas de membranas necesarios para el desarrollo del trabajo. Además, se hace una revisión del trabajo previo. En el Capítulo 3 se define el problema MAX-CLIQUE y después se presenta un sistema de tejido para su resolución, el cual se prueba en un caso pequeño. Posteriormente se demuestran formalmente las propiedades del sistema propuesto y se presenta un pseudocódigo basado en dicho sistema. En el Capítulo 4 se presenta un algoritmo molecular basado en el sistema de tejido del Capítulo 3. Se detallan la codificación de las hebras de ADN, las unidades de memoria y las operaciones del algoritmo, para después presentar el algoritmo y una versión paralela del mismo. En el Capítulo 5 se muestran los resultados del trabajo, comparando con el estado del arte. Finalmente, en el Capítulo 6 se presentan las conclusiones y trabajo futuro de la investigación.

2.1. Fundamentos matemáticos

En esta sección se tratan los conceptos básicos de matemáticas que se utilizan a lo largo de este trabajo, basados principalmente en el libro de Bloch (2011) y Păun (2006).

2.1.1. Teoría de conjuntos

Conjuntos

Un conjunto es una colección de objetos no repetidos llamados elementos o miembros del conjunto. Si un objeto x es un elemento del conjunto A, entonces se dice que $x \in A$, y en caso contrario, $x \notin A$. Se puede describir un conjunto listando explícitamente sus miembros en una lista dentro de llaves, o mediante una forma compacta que exprese las características de los miembros. En un conjunto no importa el orden de los elementos. El número de elementos en un conjunto A se denomina la cardinalidad de A y se escribe $|A|$. Al conjunto sin elementos se le denomina conjunto vacío y se denota como $\emptyset$.

Si todos los elementos de un conjunto A están contenidos dentro de un conjunto B, esto es, si $x \in A$ implica que $x \in B$, entonces A es un subconjunto de B y se denota como $A \subseteq B$. Para cualquier conjunto A, $\emptyset \subseteq A$ y $A \subseteq A$. Para tres conjuntos cualquiera A, B, y C, si $A \subseteq B$ y $B \subseteq C$, entonces $A \subseteq C$. Se dice que si $A \subseteq B$ y $B \subseteq A$ entonces $A = B$. Si $A \subseteq B$ pero $A \neq B$ entonces decimos que A es un conjunto propio de B y se denota como $A \subset B$.

Siendo A y B conjuntos, definimos las siguientes operaciones:

- *Intersección.*

$$A \cap B = \{x \mid x \in A \ \wedge \ x \in B\}. \tag{1}$$

  Se dice que dos conjuntos A y B son disjuntos si no tienen elementos en común, esto es, si $A \cap B = \emptyset$.

- *Unión.*

$$A \cup B = \{x \mid x \in A \ \vee \ x \in B\}. \tag{2}$$

- *Diferencia o resta.*

$$A \backslash B = \{x \mid x \in A \ \wedge \ x \notin B\}. \tag{3}$$

- *Producto cartesiano.* El producto cartesiano de dos conjuntos A y B, denotado $A \times B$, es el conjunto de todos los pares ordenados tales que el primer elemento del par es un elemento de A y el segundo es un elemento de B. De manera formal,

$$A \times B = \{(a, b) \mid a \in A \ \wedge \ b \in B\}. \tag{4}$$

  Cuando A y B son conjuntos finitos, la cardinalidad de su producto cartesiano es $|A \times B| = |A| \cdot |B|$.

- *Conjunto potencia.* El producto cartesiano del conjunto A se denota como 2^A y es el conjunto cuyos elementos son los subconjuntos de A.

  Ejemplo 2.1. Sea $A = \{a, b, c\}$. Entonces los subconjuntos de A son

$$2^A = \{\emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{a, b, c\}\}. \tag{5}$$

$\Diamond$

- *Complemento.* Puede existir un conjunto U de referencia que proporciona el contexto para los conjuntos que se trabajan en determinado momento, el cual se llama universo. Los conjuntos que son objeto de estudio son subconjuntos del universo. Dado un universo U, se define el complemento de un conjunto $A \subseteq U$ como:

$$\overline{A} = U \backslash A = \{x \mid x \in U \ \wedge \ x \notin A\}. \tag{6}$$

Familias de conjuntos

Sea $\mathcal{A}$ un conjunto. El conjunto $\mathcal{A}$ es una familia de conjuntos si todos los elementos de $\mathcal{A}$ son conjuntos. La unión de los conjuntos en $\mathcal{A}$, denotada por $\bigcup_{X \in \mathcal{A}} X$, se define de la siguiente manera:

Si $\mathcal{A} \neq \emptyset$, entonces

$$\bigcup_{X \in \mathcal{A}} X = \{x \mid x \in X \text{ para algún } X \in \mathcal{A}\}. \tag{7}$$

Si $\mathcal{A} = \emptyset$, entonces $\bigcup_{X \in \mathcal{A}} X = \emptyset$. La intersección de los conjuntos en $\mathcal{A}$, denotada por $\bigcap_{X \in \mathcal{A}} X$, se define a continuación:

Si $\mathcal{A} \neq \emptyset$, entonces

$$\bigcap_{X \in \mathcal{A}} X = \{x \mid x \in X \text{ para toda } X \in \mathcal{A}\}. \tag{8}$$

Si $\mathcal{A} = \emptyset$, entonces $\bigcap_{X \in \mathcal{A}} X$ no está definida. Sea I un conjunto no vacío. Se dice que la familia de conjuntos $\mathcal{A}$ está indizada por I si existe un elemento $A_i \in \mathcal{A}$ para cada $i \in I$, de manera que $\mathcal{A} = \{A_i\}_{i \in I}$. La unión e intersección de una familia de conjuntos indizada por I se define respectivamente como:

$$\bigcup_{i \in I} A_i = \{x \mid x \in A_i \text{ para algún } i \in I\} \text{ y} \tag{9}$$

$$\bigcap_{i \in I} A_i = \{x \mid x \in A_i \text{ para toda } i \in I\}. \tag{10}$$

Particiones

Una colección $P = \{S_{i \in I}\}$ de conjuntos no vacíos indizados por I forma una partición de un conjunto $\mathcal{S}$ si 1) los conjuntos son disjuntos entre sí, esto es, para todo $S_i, S_j \in P, i \neq j$, implica que $S_i \cap S_j = \emptyset$; y 2) su unión es $\mathcal{S}$, esto es:

$$\mathcal{S} = \bigcup_{i \in I} S_i. \tag{11}$$

En otras palabras, P forma una partición de $\mathcal{S}$ si cada elemento de $\mathcal{S}$ aparece en exactamente un $S_i \in P$.

Multiconjuntos

Un multiconjunto es un conjunto con la característica que los elementos pueden repetirse. Dado un conjunto A, un multiconjunto sobre A es un mapeo $M : A \to \mathbb{N}$. Para $a \in A$, $M(a)$ es la multiplicidad de a en M, es decir, el número de veces que a está presente en M. Al conjunto de elementos distintos de A contenidos en M se llama el soporte de M y se denota como $supp(M) = \{a \in A \mid M(a) \neq 0\}$. Si $supp(M) = \emptyset$, entonces M es el multiconjunto vacío denotado por $\emptyset$. La cardinalidad de M es la suma de las multiplicidades de los elementos de su soporte: $|M| = \sum_{a \in supp(M)} M(a)$.

Los multiconjuntos se pueden describir de distintas formas, siendo la más directa el listado de sus miembros entre corchetes. Al igual que en los conjuntos, en los multiconjuntos no importa el orden de sus elementos. Otra notación consiste en indicar los pares de elementos y multiplicidades; si un elemento no está presente se omite, y si tiene multiplicidad de 1 sólo se indica el elemento. Una derivación de esta notación es la representación por cadenas formadas por cualquiera de las posibles permutaciones de los elementos del multiconjunto repetidos por (o a la potencia de) su cardinalidad.

Ejemplo 2.2. Sea el conjunto $A = \{a, b, c, d, e\}$ y M_1 un multiconjunto sobre A tal que $M_1 = [a, b, a, b, b, c]$. La representación de pares es $M_1 = \{(a, 2), (b, 3), c\}$, mientras que una posible representación en cadena de M_1 es $M_1 = a^2b^3c$. Observe además que $supp(M_1) = \{a, b, c\}$ y $|M| = M(a) + M(b) + M(c) = 2 + 3 + 1 = 6$. $\Diamond$

Sean $M_1, M_2 : A \to \mathbb{N}$ multiconjuntos. Se dice que $M_1 \subseteq M_2$ (M_1 está contenido en M_2) si $M_1(a) \leq M_2(a)$ para toda $a \in A$. Si $M_1 \subseteq M_2$ pero $M_1 \neq M_2$ entonces se dice que M_1 está incluido estrictamente en M_2 y se denota como $M_1 \subset M_2$.

Se definen las siguientes operaciones con multiconjuntos:

- *Unión.* La unión de M_1 y M_2 es el multiconjunto $M_1 \cup M_2 : A \to \mathbb{N}$ definido por $(M_1 \cup M_2)(a) = M_1(a) + M_2(a)$, para toda $a \in A$. Sean w_1 y w_2 las cadenas que representan a los multiconjuntos M_1 y M_2, respectivamente. El multiconjunto $M_1 \cup M_2$ se puede representar por alguna de las permutaciones de la cadena formada por la concatenación de w_1 y de w_2.

- *Intersección.* La intersección de M_1 y M_2 es el multiconjunto $M_1 \cap M_2 : A \to \mathbb{N}$ definido por $(M_1 \cap M_2)(a) = min\{M_1(a), M_2(a)\}$, para toda $a \in A$.

- *Diferencia o resta.* La diferencia $M_1 \backslash M_2$ es definida solamente cuando $M_2 \subseteq M_1$ y es el multiconjunto $M_1 \backslash M_2 : A \to \mathbb{N}$ dado por $(M_1 \backslash M_2)(a) = M_1(a) - M_2(a)$, para toda $a \in A$.

Ejemplo 2.3. Sea el conjunto $A = \{a, b, c, d\}$ y los multiconjuntos sobre A, $M_1 = \{(a, 4), (b, 3), (c, 2)\}$ y $M_2 = \{a, (c, 2)\}$.

$$\begin{aligned} M_1 \cup M_2 &= \{(a, M_1(a) + M_2(a)), (b, M_1(b) + M_2(b)), (c, M_1(c) + M_2(c))\} \\ &= \{(a, 4 + 1), (b, 3 + 0), (c, 2 + 2)\} \\ &= \{(a, 5), (b, 3), (c, 4)\}. \end{aligned}$$

$$\begin{aligned} M_1 \cap M_2 &= \{(a, min\{M_1(a), M_2(a)\}), (b, min\{M_1(b), M_2(b)\}), (c, min\{M_1(c), M_2(c)\})\} \\ &= \{(a, min\{4, 1\}), (b, min\{3, 0\}), (c, min\{2, 2\})\} \\ &= \{a, (c, 2)\}. \end{aligned}$$

$$\begin{aligned} M_1 \backslash M_2 &= \{(a, M_1(a) - M_2(a))), (b, M_1(b) - M_2(b)), (c, M_1(c) - M_2(c))\} \\ &= \{(a, 4-1), (b, 3-0), (c, 2-2)\} \\ &= \{(a, 3), (b, 3)\}. \end{aligned}$$

$\Diamond$

2.2. Fundamentos de computación

En esta sección se revisan los conceptos básicos de computación necesarios para el desarrollo del trabajo. El material es condensado de Cormen *et al.* (2011), Hromkovic (2010) y Păun *et al.* (2010).

2.2.1. Teoría de grafos

Un grafo no dirigido es un par $G = (V, E)$, donde V es un conjunto finito llamado el conjunto de vértices, y $E \subseteq V \times V$ es un conjunto de pares no ordenados de vértices (v_i, v_j) denominados aristas, tal que $v_i, v_j \in V$ e $i \neq j$. El orden de un grafo G es el número de vértices y está dado por $|V|$. Una forma de representar un grafo G de orden n es mediante una matriz $n \times n$ booleana simétrica $A = (a_{ij})$ conocida como la matriz de adyacencias de G, donde $a_{ij} = 1$ si y sólo si $i \neq j$ y $(v_i, v_j) \in E$. Otra representación de grafos es mediante una lista de adyacencias, la cual consiste en un arreglo Adj de n listas, una para cada vértice en V. Para cada $v_i \in V$, la lista de adyacencias $Adj[v_i]$ contiene todos los vértices v_j tales que exista una arista $(v_i, v_j) \in E$.

Cualquier arista (v_i, v_j) de un grafo $G = (V, E)$ es incidente a los vértices v_i y v_j. Dos vértices u y v son adyacentes si la arista $(u, v) \in E$. El grado de un vértice v de G, $deg_G(v)$, es el número de aristas incidentes a v. El grado de un grafo $G = (V, E)$ es

$$deg(G) = max\{deg_G(v) \mid v \in V\}. \tag{12}$$

Para cada grafo $G = (V, E)$,

$$\sum_{v \in V} deg_G(v) = 2|E|.$$

El vecindario $N(v_i)$ de vértices adyacentes a v_i se define como:

$$N(v_i) = \{v_j \mid (v_i, v_j) \in E\}. \tag{13}$$

En un grafo completo cada par de vértices distintos está unido por una arista única, esto es, $\forall_{v \in V}, N(v) = V \backslash \{v\}$. Un grafo $G' = (V', E')$ es un subgrafo de $G = (V, E)$ si $V' \subseteq V$ y $E' \subseteq E$. Sea $H = V \times V$ el conjunto de todas las aristas posibles de un grafo $G = (V, E)$. El grafo complementario $\overline{G}$ de G es un grafo $\overline{G} = (V, \overline{E})$ tal que $\overline{E} = H \backslash E$, es decir, un grafo con el mismo conjunto de vértices V tales que dos vértices son adyacentes si y sólo si no son adyacentes en G.

Un clique de un grafo G es un subgrafo completo de G. El tamaño de un clique es el número de vértices que contiene. Un clique maximal es aquel que no se puede extender agregando algún otro vértice adyacente. En este trabajo se utiliza $\varpi(v_i)$ para denotar el tamaño del clique maximal al que pertenece el vértice v_i. El clique máximo es el clique más grande posible en un grafo. Se define $\omega(G)$ como el tamaño del clique máximo del grafo G.

2.2.2. Lenguajes

Un alfabeto es un conjunto finito no vacío cuyos elementos reciben el nombre de símbolos. Cualquier secuencia de símbolos w de un alfabeto Σ se llama una cadena sobre Σ. La longitud de w es la cantidad de símbolos que contiene y se denota como $|w|$. La cantidad de apariciones de un símbolo $a \in \Sigma$ en la palabra w se denota como $|w|_a$. Dadas dos palabras w_1 y w_2 sobre un alfabeto Σ, la operación de concatenación en w_1 y w_2 se denota como $w_1 w_2$ o $w_1 \cdot w_2$ y produce una palabra que consiste en los símbolos de w_1 en el mismo orden, seguidos por los símbolos de w_2 en el mismo orden.

Para un alfabeto Σ se denota por Σ^* la cerradura de Kleene, la cual es el conjunto (infinito) de todas las cadenas sobre Σ. La cadena vacía se denota por λ, y el conjunto de cadenas no vacías sobre Σ, $\Sigma^+ = \Sigma^* \backslash \{\lambda\}$ se llama la cerradura positiva. Cada subconjunto de Σ^* se

llama un lenguaje sobre Σ.

Para cada palabra $w \in \Sigma^*$ se define la operación potencia como:

(i) $w^0 = \lambda$, y

(ii) $w^{n+1} = ww^n$ para $n \in \mathbb{N}^+$.

Si $w = w_1w_2w_3$ para alguna $w_1, w_2, w_3 \in \Sigma^*$, entonces w_2 es una subcadena de w; si $w_1 \neq \lambda$ entonces w_1 es un prefijo de w, y si $w_3 \neq \lambda$, entonces w_3 es un sufijo de w. Los conjuntos de todos los prefijos, sufijos y subcadenas de una cadena w se denotan por *Pref(w), Suf(w), Sub(w)*, respectivamente.

Dado que los lenguajes son conjuntos, se pueden realizar las mismas operaciones de unión, intersección, resta y complemento. La concatenación de dos lenguajes L_1, L_2 es una operación que produce el lenguaje $L_1L_2 = \{w_1w_2 \mid w_1 \in L_1, w_2 \in L_2\}$. Mediante esta operación se define la exponenciación de un lenguaje L:

$$L^0 = \{\lambda\},$$
$$L^{i+1} = LL^i, i \geq 0.$$

De la misma manera se puede definir las cerraduras de un lenguaje L:

$$L^* = \bigcup_{i=0}^{\infty} L^i,$$
$$L^+ = \bigcup_{i=1}^{\infty} L^i.$$

2.3. Fundamentos de biología

En esta sección se tratan conceptos básicos de biología para las áreas de sistemas P y cómputo molecular. El material se obtuvo de Păun *et al.* (2010), Frisco (2009) e Ignatova *et al.* (2008).

2.3.1. La célula

La célula es la unidad más simple en la organización biológica de un organismo vivo. Mediante sus procesos metabólicos, la célula mantiene al organismo vivo, nutriéndolo, y

ayudando a que crezca y se reproduzca. El origen del nombre se remonta hasta 1665 cuando Robert Hook notó la similitud de la estructura de las rebanadas delgadas de corcho bajo el microscopio y las celdas de los monasterios.

Las células se dividen en dos grandes grupos: las procariotas, que tienen su material genético flotando sin una membrana que lo delimite, y las eucariotas, que tienen un núcleo bien definido que contiene dicho material. Las células procariotas abarcan un gran número de seres unicelulares como los ciliados y bacterias. Por otro lado, las eucariotas forman parte de todos los seres multicelulares, además de seres unicelulares complejos como la levadura.

Para cumplir con sus funciones, una célula dispone de múltiples organelos especializados. Por ejemplo, las mitocondrias poducen moléculas de ATP, las cuales brindan la energía química; el retículo endoplásmico se encarga de la biosíntesis de proteínas y lípidos; el núcleo de almacenar el material genético y controlar las actividades de la célula.

Los organelos y la célula misma están rodeados de membranas, las cuales están formadas en su mayoría por fosfolípidos. Los fosfolípidos poseen un extremo hidrofílico y uno hidrofóbico. Estas moléculas tienden a agruparse en forma de láminas y plegarse, formando capas cuya capa interna es hidrofílica y la externa hidrofóbica. De la misma manera, se atraen los extremos hidrofóbicos de otra sábana de fosfolípidos y se forma lo que se conoce como una bicapa. La bicapa de fosfolípidos es la parte fundamental en las membranas células. Por medio de dos tipos de proteínas (integrales o transmembrana y periféricas) es posible mantener a la célula con los nutrientes necesarios y comunicada con su entorno.

Cada célula está envuelta por una membrana plasmática con tres funciones generales: **separar** el contenido de la célula del ambiente externo, **regular** el intercambio de sustancias entre el interior de la célula y el ambiente externo, y servir como **canal de comunicación** con otras células.

2.3.2. Tejidos

Las células se pueden conectar entre ellas para formar tejidos. La piel, intestinos y músculos son ejemplos de tejidos. Las uniones celulares se pueden clasificar dentro de tres categorías:

- Oclusión: previenen a pequeñas moléculas de pasar de un lado del tejido a otro.
- Anclaje: añaden mecánicamente células a otras células.

- Comunicantes: permiten a las células pasar químicos o señales eléctricas entre células adyacentes. Este tipo de uniones son la inspiración para los Sistemas P de Tejido.

Las uniones de hendidura (*gaps*) permiten a pequeños químicos pasar directamente del citoplasma de una célula hacia otra. Este pasaje está realizado por proteínas transmembrana, llamadas proteínas formadoras de canal (conexinas), presentes en la membrana plasmática. Cuando dos de estas proteínas presentes en la membrana plasmática de dos células están alineadas, entonces se establece un canal. Las conexinas permiten el paso de moléculas con un diámetro pequeño tales como iones, azúcares, aminoácidos, etc., pero no permiten el paso de moléculas más grandes tales como proteínas. Las uniones de hendidura no están abiertas continuamente, en cambio estas uniones son estructuras dinámicas que se pueden cerrar o abrir por ellas mismas en respuesta a cambios en la célula.

El tejido neuronal

La neurona es el bloque esencial de construcción del sistema nervioso. Las neuronas son células especializadas en responder a estímulos con impulsos nerviosos, y conducirlos por distancias largas. En una neurona típica existen dos tipos de extensiones que salen desde la parte de la célula, llamada el soma, donde el núcleo y el retículo endoplásmico están localizados: las dendritas, las cuales se dividen para formar árboles de dendritas, y el axón, el cual puede tener ramificaciones llamadas colaterales de axón. Sólo una de estas extensiones puede estar presente.

La morfología de las neuronas puede variar con respecto al tamaño del soma, el número, longitud y ramificación de las dendritas así como la longitud y ramificación del axón. Los axones, por ejemplo, pueden ser desde 0.1mm hasta 1m de largo. El segmento inicial de un axón se llama el cono axómico. Los axones terminan con terminales nerviosas (o botones) normalmente encontrados cerca de la membrana plasmática de células diferentes a las neuronas o cerca de las dendritas o el soma de otras neuronas. La locación de este punto de contacto se llama una sinapsis; es aquí donde la transmisión de información de una neurona a otra toma lugar.

Los axones están especializados para la conducción de un tipo particular de impulso eléctrico llamado potencial de acción o pico (*spike*). Un potencial de acción, causado por el voltaje dependiente que abre y cierra ciertos canales iónicos, es una serie de cambios

súbitos en el voltaje a través la membrana del axón. La llegada de un potencial de acción a las terminales del nervio causa la secreción de químicos llamados neurotransmisores. Los neurotransmisores enlazan las proteínas presentes en la membrana de la neurona receptora y causan cambios en su potencial de membrana. Este potencial se llama el potencial sináptico o entrada sináptica. Estos pequeños impulsos eléctricos se conducen hacia el soma. De esta manera, las señales eléctricas llevan la información dentro de una neurona, mientras que las señales químicas transmiten información entre neuronas.

Una señal eléctrica presente en la membrana plasmática es más probable que dé lugar a un potencial de acción en el cono axómico. Allí es donde la actividad de todos los potenciales sinápticos se suma y, si cierto umbral se alcanza, la neurona dispara un potencial de acción. Un potencial de acción es todo o nada. Si los potenciales sinápticos están debajo de cierto umbral, entonces no se produce un potencial de acción. Si en cambio están sobre el umbral, entonces el mismo potencial de acción es producido. Esto significa que es la frecuencia de los potenciales de acción lo que codifica diferente información. Una neurona no puede disparar un segundo potencial de acción hasta que los canales iónicos hayan regresado a la conformación cerrada. Este intervalo de tiempo en el cual la neurona es refractaria a la estimulación se llama periodo refractario.

La manera en que una neurona reacciona a una entrada sináptica depende de los diferentes tipos de canales iónicos presentes en algunas partes de ella. Las células con diferentes combinaciones de canales iónicos reaccionan diferentes a los potenciales sinápticos constantes. Algunas células reaccionan con un solo potencial de acción, otras con una secuencia de frecuencia constante de potenciales de acción. Dichas secuencias se llaman trenes de potencial de acción o trenes de picos. Algunas neuronas incluso se pueden disparar sin ningún potencial sináptico. No hay un acuerdo general en el intervalo de tiempo entre los picos que definen un tren de picos; sin embargo, la tasa de disparo de algunas neuronas puede incrementarse por cambios en la fuerza de los potenciales sinápticos.

En adición a las señales eléctricas y químicas, la función del sistema nervioso depende de la topología de las neuronas, esto es, en cómo están interconectadas. Un axón puede tener sinapsis con cientos de neuronas e inducir respuestas a cada una de ellas simultáneamente. Estas conexiones determinan la trayectoria de las señales. Así que aunque cada neurona es un agente, sólo mediante la cooperación, las neuronas pueden completar tareas específicas.

2.3.3. Ácido desoxirribonucleico

El ácido desoxirribonucleico (**ADN**), es una molécula que codifica, regula y transmite las características de un ser vivo. El ADN se compone de subunidades llamadas nucleótidos, las cuales se conforman de tres partes: un grupo fosfato, un azúcar desoxirribosa y una de cuatro bases nitrogenadas; estas últimas se dividen en purinas (adenina (**A**) y guanina (**G**)) y piramidinas (timina (**T**) y citosina (**C**)).

La desoxirribosa se compone de un anillo de cinco carbonos ($1', \ldots, 5'$). El grupo fosfato (P) une al carbono $5'$ de la desoxirribosa de un nucleótido con un grupo hidroxilo en el carbono $3'$ del siguiente. Una cadena de ADN tiene un grupo fosfato libre en un extremo y un grupo hidroxilo (OH) en el otro. Por convención, el sentido de la cadena es en esta dirección y se denota como $5'$ a $3'$.

Las bases nitrogenadas forman puentes de hidrógeno entre ellas, por lo que dos cadenas sencillas de ADN pueden enlazarse entre sí para formar una cadena doble de ADN en forma de hélice. La complementariedad se da entre los pares de bases A-T y G-C, lo cual se denomina como la complementariedad de Watson-Crick.

Para que se pueda formar la hebra doble, es necesario que las cadenas sencillas estén en sentido antiparalelo y que las bases en las posiciones correspondientes del alineamiento sean complementarias. Sin embargo, el alineamiento puede no ser perfecto y dejar extremos sin complemento. Estos extremos se llaman extremos pegajosos. Dos moléculas de ADN con extremos pegajosos complementarios pueden unirse y formar una nueva molécula de ADN mediante el enlace de los extremos pegajosos.

Los enlaces que unen una cadena doble son puentes de hidrógeno entre las bases. Estos puentes pueden romperse mediante calor, ocasionando que la cadena doble se separe en dos cadenas sencillas. Este fenómeno se llama desnaturalización. Las cadenas desnaturalizadas puede volver a unirse mediante enfriamiento lento, lo cual se llama hibridación o alineamiento. Sin embargo, es posible que una hebra sencilla se alinee consigo misma, lo que da origen a estructuras secundarias del ADN.

Captura de ADN

Debido a la complementariedad entre bases, es posible capturar cadenas de ADN de interés mediante sondas de hibridación. Una sonda de hibridación se compone por la secuencia

complementaria a la región de interés. En caso que se desee lograr la detección, entonces se aplica un marcador fluorescente. Si lo que se requiere es separar las cadenas según si tienen o no la secuencia sonda, entonces se utilizan otros mecanismos tales como las perlas magnéticas. Las sondas por hibridación son la base de los modelos de cómputo no convencional por filtrado.

Enzimas

Las enzimas son proteínas que controlan reacciones químicas en las células actuando como catalizadores, esto es, acelerando la tasa de reacciones químicas sin ser consumidas ellas mismas en el proceso. Las enzimas de restricción reconocen subcadenas específicas y continuas de una molécula doble de ADN llamadas sitios de restricción y la cortan (esto es, rompen los enlaces entre el azúcar y el grupo fosfato y los enlaces entre las bases complementarias), ya sea dentro del sitio de restricción o fuera de él. Estas enzimas ayudan a las bacterias a defenderse ellas mismas de la invasión de ADN viral cortando el ADN invasor en pedazos. El corte realizado por una enzima de restricción se hace en una manera tal que el grupo fosfato se preserva en el extremo $5'$ y el grupo hidroxilo se preserva en el extremo $3'$ de las dos moléculas de ADN creadas.

Las ligasas son enzimas que *reparan* enlaces faltantes entre un azúcar y un grupo fosfato, proveyendo los grupos hidroxilo y fosfato presentes en los extremos correspondientes $3'$ y $5'$. Las ligasas ayudan a reparar los enlaces faltantes al hibridar dos cadenas complementarias mediante extremos pegajosos.

Las ADN-polimerasas son enzimas que pueden hacer moléculas de ADN de cadena doble a partir de cadenas dobles parciales. Si la ADN-polimerasa, moléculas de ADN parcialmente dobles y nucleótidos trifosfatados se unen en una solución, entonces esta polimerasa une los nucleótidos libres a los extremos pegajosos de las moléculas de ADN extendiéndolos haciendo que se formen las moléculas de ADN de cadena doble. Las ADN-polimerasas sólo pueden extender la molécula de ADN siguiendo la orientación $5' \to 3'$.

Replicación

Cuando las células se dividen, se debe garantizar que cada célula hija reciba una copia de toda la información de la célula madre, por lo que sintetiza dos copias exactas de su ADN.

Este proceso se llama replicación del ADN. La reacción en cadena de la polimerasa (PCR) es la replicación iterativa de ADN para amplificar cadenas de ADN en el laboratorio. La PCR requiere una solución compuesta de moléculas de ADN (sencillas/dobles/parcialmente dobles) con secuencias molde conocidas, moléculas de ADN de cadena sencilla (cebadores) complementarias a las secuencias molde, polimerasa, nucleótidos y una repetición de los siguientes pasos:

- Desnaturalización: la solución se calienta para que las dos hebras de ADN se separen.
- Hibridación: la solución se enfría para que los cebadores se puedan unir a las secuencias molde.
- Extensión: la polimerasa extiende las moléculas de ADN parcialmente dobles creadas en el paso anterior.

Cada repetición de estos pasos puede doblar el número de moléculas de ADN presentes al inicio del procedimiento.

2.4. Cómputo biomolecular

Esta sección presenta una breve descripción del área de cómputo molecular, la cual está basada en los libros de Ignatova *et al.* (2008) y Calude y Păun (2001).

El cómputo biomolecular o cómputo con ADN es un paradigma de cómputo no convencional que se vale de las propiedades inherentes a las moléculas biológicas para utilizarlas como dispositivos de cómputo. El cómputo biomolecular es un campo que une las disciplinas de química, ciencias de la computación, biología molecular, física y matemáticas. El área comienza a consolidarse a partir del experimento de Adleman (1994), donde demostró la solución al problema del vendedor viajero para un grafo de 7 vértices mediante ADN y operaciones de laboratorio.

Los modelos de cómputo biomolecular se clasifican en no autónomos, autónomos e *in vivo*. Los modelos no autónomos fueron los primeros en desarrollarse, y su nombre se deriva del hecho que su ejecución la realizaban los humanos. Entre estos modelos están los modelos de filtrado, los sistemas de etiquetas, y los sistemas de empalmado. Los modelos autónomos se basan en la capacidad del ADN de auto-ensamblarse cuando se encuentra en forma de

cadena sencilla. Entre ellos se encuentran el auto-ensamblado, el modelo de computación con horquillas o lazos, desplazamiento de cadenas y el cómputo celular. En este último entra el área de sistemas de membranas. Utilizando este tipo de modelos existen implementaciones de artefactos computacionales tales como autómatas de estados finitos, redes neuronales, y máquinas de Turing.

Finalmente, los modelos *in vivo* utilizan células vivas como memoria y para ejecutar las operaciones.

Por su importancia al presente trabajo, a continuación se explican algunos modelos no autónomos.

2.4.1. Modelos de filtrado

En los modelos de filtrado, la operación fundamental es el filtro/captura/separación de ADN. El primer algoritmo de cómputo molecular (Adleman, 1994) utiliza un modelo de filtrado básico, el cual se dice que es sin memoria debido a que las hebras no cambian en el transcurso de la computación. En general, la unidad básica de memoria en cómputo molecular es el **tubo**, cuyo nombre hace alusión a los tubos de ensayo. Con esto en mente, se define la operación básica del modelo de filtrado **sin memoria**:

- SEPARAR(T, T^+, T^-, x): teniendo como entrada el tubo T y una cadena x, se producen los tubos T^+ y T^-, donde T^+ contiene todas las cadenas de T que contengan a x como subcadena y T^- contiene las hebras restantes.

Una extensión al modelo de filtrado sin memoria es el filtrado **marcar y destruir**, el cual permite eliminar y crear cadenas durante la computación. Las dos operaciones adicionales son las siguientes:

- REMOVER$(T, \{x_1, \ldots, x_m\})$: se eliminan de T las cadenas que contengan al menos a una cadena x_i, $1 \leq i \leq m$ como subcadena.

- COPIAR$(T, \{T_1, \ldots, T_m\})$: el contenido del tubo T se copia a cada uno de los tubos $T_1, \ldots, T_m$.

2.4.2. Modelos de etiquetas

En el modelo de sistema de etiquetas la memoria es de acceso aleatorio y consiste de los llamados complejos de memoria. Cada complejo de memoria es una cadena de ADN que codifica un número de n bits. Cada bit se representa como una subsecuencia o región en la cadena de longitud especificada. Si una región tiene alineado su complementario, entonces se dice que el bit está encendido o tiene un valor de 1: en caso contrario, está apagado y tiene un valor 0. La operación *separar* se extiende a este modelo si, en lugar de verificar la presencia de determinadas secuencias en las cadenas, se verifica si la subsecuencia en la posición especificada está alineada o no con su complementario. Además se cuenta con las siguientes operaciones:

- SET(T, i): a partir del tubo T se genera un tubo en el cual cada subcadena i de cada complejo de memoria está encendido.
- LIMPIAR(T, i): a partir del tubo T se genera un tubo en el cual cada subcadena i de cada complejo de memoria está apagado.
- DESECHAR(T): se vacía el tubo T.

Tanto el modelo de etiquetas como el de filtrado requiere de una biblioteca inicial de cadenas, a las cuales se les aplican las operaciones descritas anteriormente de manera paralela.

2.4.3. Modelos de corte y empalme

Los modelos de corte y empalme (*splicing systems*) son métodos generativos basados en la recombinación de ADN. Dos cadenas de ADN se cortan y se reconectan las partes para obtener nuevas moléculas de ADN. La descripción de las reglas de este modelo se presenta en la Sección 2.5.1.

Es posible automatizar la ejecución de estos modelos mediante dispositivos microfluídicos. Los dispositivos microfluídicos son plataformas biotecnológicas capaces de transportar y procesar líquido en escalas de micro, nano y picolitros. Este tipo de dispositivos tiene su aplicación en la automatización de diagnósticos y otros procesos de laboratorio.

Para el cómputo biomolecular, los microfluidos representan una forma de automatizar los modelos tanto autónomos como no autónomos debido a que muchas de las operaciones de

biología molecular ya están implementadas en estas plataformas (Li, 2008), por lo que el tipo de reglas en estos modelos depende del algoritmo de cómputo molecular a implementar.

2.5. Sistemas de membranas

Esta sección aborda los fundamentos de los sistemas de membranas, así como sus variantes básicas. El material presentado se basa en las monografías de Păun *et al.* (2010), Frisco (2009) y Calude y Păun (2001), así como en los artículos de Păun (2006) y Martín-Vide *et al.* (2003).

El área de sistemas de membranas, cómputo con membranas o sistemas P, se refiere a la rama del cómputo molecular iniciada por George Păun (2000) cuyo objetivo es diseñar modelos de computación a partir de la abstracción de la estructura y el funcionamiento de las células vivas, así como de la cooperación entre las células en los tejidos, órganos y otras poblaciones de células. Los modelos iniciales de sistemas P (los sistemas de transición) se basan en una estructura jerárquica de membranas que delimitan compartimentos donde se procesan químicos según un conjunto de reacciones o procesos biológicos. A partir de la abstracción de esta descripción general, se muestran los componentes de un sistema de transición.

2.5.1. Componentes

La estructura de membranas

La estructura de membranas es la parte fundamental de un sistema de transición. Una estructura de membranas es un conjunto de membranas jerárquicamente ordenado, contenido en una membrana externa (llamada membrana plasmática o piel). Un sistema de m membranas se llama un sistema de grado m. El espacio fuera de la piel se llama el entorno. Se dice que una membrana es de un nivel inferior a otra cuando está contenida en ella, y de nivel superior en caso contrario. Si dos membranas comparten la membrana de nivel superior, entonces son vecinas entre sí. Cuando una membrana no contiene otras anidadas, se le llama membrana elemental.

Cada membrana cuenta con una etiqueta que la identifica y un espacio que delimita, llamado región. Usualmente los términos membrana y región se usan como sinónimos. Al igual que en la célula viva, la membrana tiene la función de mediar la comunicación de

objetos entre las regiones internas y externas que delimita. Sin embargo, en los sistemas P, una región puede fungir como salida de la computación (más detalles se presentan en la Sección 2.5.3). Se puede representar una estructura de membranas de tres maneras: mediante diagramas de Euler-Venn, como árbol enraizado, y mediante paréntesis.

El sistema de la Figura 1 se presenta mediante un diagrama de Euler-Venn. En esta representación, cada membrana representa un conjunto con una etiqueta asociada, donde las membranas inferiores son conjuntos completamente incluidos en la membrana de nivel superior. A partir de esta jerarquía surge la segunda representación, como árbol enraizado (Figura 2). En la representación de árbol, la piel representa la raíz (en este caso, la membrana con la etiqueta "1"), mientras que las membranas inferiores se representan como nodos de la membrana superior. Cuando una membrana es elemental, entonces es una hoja del árbol.

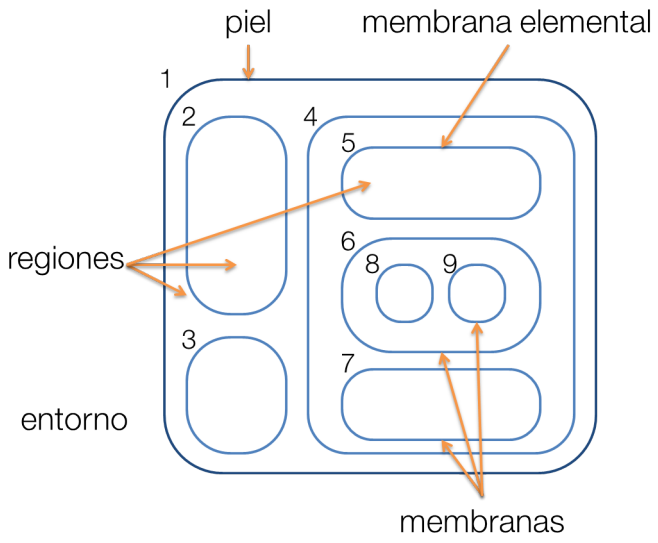

Figura 1: Ejemplo de sistema de membranas en representación esquemática. Modificado de Păun, 2006, p. 8.

Finalmente, la representación mediante paréntesis es la manera estándar de indicar la estructura de membranas en la literatura. Para generar la representación mediante paréntesis μ de una estructura de membranas S, sean $l_1, \ldots, l_n$ las etiquetas de las membranas en S, donde l_1 es la etiqueta de la piel. La membrana etiquetada por l_i, $1 \leq i \leq n$, se representa mediante un par ordenado de paréntesis $[\ \ ]_{l_i}$, y una expresión de paréntesis μ construida de la siguiente manera (Păun *et al.*, 2010):

1. $[_{l_i}\ \ ]_{l_i}$ es una forma correcta para S.

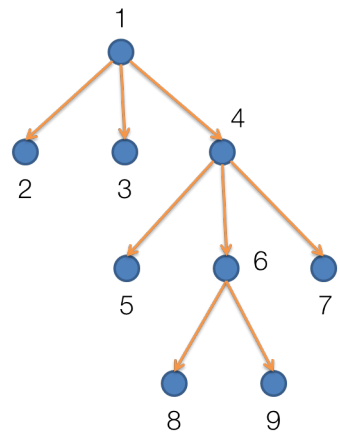

Figura 2: Árbol que describe el sistema de la Figura 1.

2. Si $z = z_1[_{l_i} \quad]_{l_i} z_2$ es una forma correcta para S, para alguna $i \in \{1, 2, \ldots, n\}$, y las membranas etiquetadas por $k_1, \ldots, k_p \in \{2, \ldots, n\}$ son todas vecinos inferiores de la membrana etiquetada por l_i, entonces por cada cadena $x = [_{k_{j_1}} \quad]_{k_{j_1}} [_{k_{j_2}} \quad]_{k_{j_2}} \ldots [_{k_{j_p}} \quad]_{k_{j_p}}$, donde $j_1, \ldots, j_p$ es una permutación de $1, 2, \ldots, p$, la estructura $z_1[_{l_i} \; x \;]_{l_i} z_2$ también es una forma correcta para S.

3. Una expresión de paréntesis μ es una forma correcta para S la cual incluye el paréntesis derecho $]_{l_i}$ para cada $i \in \{1, 2, \ldots, n\}$.

Por ejemplo, una de las representaciones de paréntesis para la estructura de membranas de la Figura 1 es

$$\mu = [_1 \quad [_2 \quad]_2 \; [_4 \quad [_7 \quad]_7 \; [_5 \quad]_5 \; [_6 \quad [_8 \quad]_8 \; [_9 \quad]_9 \;]_6 \;]_4 \; [_3 \quad]_3 \;]_1, \tag{14}$$

la cual se genera mediante la siguiente secuencia de formas correctas para S:

$$\begin{aligned}
&[_1 \quad]_1 \\
&[_1 \quad [_2 \quad]_2 \; [_4 \quad]_4 \; [_3 \quad]_3 \;]_1 \\
&[_1 \quad [_2 \quad]_2 \; [_4 \quad [_7 \quad]_7 \; [_5 \quad]_5 \; [_6 \quad]_6 \;]_4 \; [_3 \quad]_3 \;]_1 \\
&[_1 \quad [_2 \quad]_2 \; [_4 \quad [_7 \quad]_7 \; [_5 \quad]_5 \; [_6 \quad [_8 \quad]_8 \; [_9 \quad]_9 \;]_6 \;]_4 \; [_3 \quad]_3 \;]_1.
\end{aligned} \tag{15}$$

Objetos

Los objetos son otro elemento básico de los sistemas P, y están inspirados en los compuestos que están contenidos en la célula. Estos compuestos no se presentan en forma única,

sino en concentración. Por ello, se abstraen por medio de multiconjuntos. Regularmente los multiconjuntos utilizados en los sistemas P son de objetos discretos tales como símbolos, cadenas, arreglos y grafos. Sin embargo, la representación más común es la de objeto-símbolo, por lo que para indicar el contenido de una célula se utiliza la representación en cadena.

Reglas de transición

Los objetos evolucionan mediante reglas de transición asociadas a cada una de las regiones del sistema. De manera general, una regla de transición tiene la forma $u \to v$, donde u y v son cadenas que representan multiconjuntos de objetos de un conjunto O. Se define a v como el producto de la regla. Para especificar la cooperación entre membranas, las reglas adoptan la forma $u \to v$, donde u es un multiconjunto de objetos y v es una cadena sobre $O \times Tar$, donde $Tar = \{here, in, out\}$ es el conjunto de indicadores de objetivo. Los indicadores de objetivo indican la forma en que se distribuirá el producto de la regla. Sea $(a, tar) \in v$ un elemento del producto, tal que $a \in O$ y $tar \in Tar$. La aplicación de tar es la siguiente:

- $here$: el objeto a permanece en la región. Este indicador puede omitirse en la declaración de la regla.

- in: el objeto a se mueve hacia una membrana de nivel inferior. En caso de existir más de una membrana inferior, se selecciona alguna de manera no determinista. Sin embargo, puede forzarse el determinismo utilizando el indicador in_j, donde el objeto se mueve hacia la membrana inferior con la etiqueta j. Si la membrana donde reside la regla es elemental, entonces la regla no es aplicable.

- out: el objeto a se mueve hacia la membrana de nivel superior. Cuando la membrana donde reside la regla es la piel, entonces el objeto sale al entorno. Si no hay ninguna regla asignada para el entorno, entonces el objeto se disuelve y no puede recuperarse.

Se supone que el sistema cuenta con un reloj global que mide el tiempo marcado por unidades de tiempo. Una regla $u \to v$ es aplicable en la región l_i si los objetos u están disponibles en la región. El proceso de aplicación de la regla en una unidad de tiempo es el siguiente:

1. Se eliminan de l_i los objetos presentes en u en las multiplicidades especificadas.

2. Se producen en l_i los objetos presentes en v en las multiplicidades especificadas.

3. Los objetos producidos son colocados en l_i o sus vecinos según los indicadores de direccción de v.

A continuación se presentan algunas variaciones comunes en la literatura a las reglas de transición. Considere un alfabeto de objetos O.

- **Reescritura.** En un sistema P de reescritura, las cadenas las procesan reglas de la forma $a \to (u, tar)$, donde

  - $a \in O$,
  - $tar \in Tar$,
  - $u = u'$ ó $u = u'\delta$ ó $u = u'\tau$. El multiconjunto u' se representa como una cadena sobre O y $\delta, \tau \notin O$ son operadores de membrana y
  - $a \to u'$ es una regla libre de contexto sobre O.

- **Reescritura con replicación.** En los sistemas P de reescritura con replicación las reglas son de la forma $a \to (u_1, tar_1)|| \ldots ||(u_k, tar_k)$, donde

  - $a \in O$,
  - u es una cadena sobre O,
  - $tar_i \in Tar$, con $1 \leq i \leq k$ y
  - la regla $a \to u_i, 1 \leq i \leq k$, es una regla libre de contexto sobre O.

  Las reglas con $k > 1$ ejecutan lo que se conoce como reescritura replicada: una cadena $x_1 a x_2$ que se consume por la regla se transforma en un conjunto de k cadenas $x_1 u_i x_2$, $1 \leq i \leq k$. Cada cadena u_i se envía al objetivo tar_i.

- **Corte y empalme.** Estas reglas se basan en las utilizadas por el modelo de cómputo molecular con el mismo nombre, y tienen la forma $(r; tar_1, tar_2)$, donde:

  - r es una regla de corte y empalme sobre O y
  - $tar_1, tar_2 \in Tar$.

Una regla de corte y empalme sobre O es una 4-tupla $r : u_1\#u_2\$u_3\#u_4$ donde u_1, u_2, u_3, $u_4 \in V^*$. Las cadenas u_1u_2 y u_4u_4 se llaman sitios de empalme. Dos cadenas $x, y \in O^*$ son complementarias con respecto a r si x contiene un sitio de r y y contiene el otro. Si $x = x_1u_1u_2x_2$ y $y = y_1u_3u_4y_2$ con $x_1, x_2, y_1, y_2 \in O^*$, entonces la regla r se puede aplicar a x y y. El producto de la regla son las cadenas w y z, donde $w = y_1u_3u_2x_2$ y $z = x_1u_1u_4y_2$, lo cual se denota como $(x, y) \vdash_r (w, z)$.

- **Catalíticas.** En las reglas catalíticas existe un conjunto de objetos especiales $C \subseteq O$ llamados catalizadores, con la propiedad de que *ayudan* a otros objetos a evolucionar sin cambiar durante la aplicación de la regla. Una regla catalítica tiene la forma $ca \to cv$, donde

  - $a \in O - C$,
  - $c \in C$,
  - $v \in ((O - C) \times Tar)^*$.

- **Promotores e inhibidores.** Este tipo de reglas tienen la forma $a \to u|_z$ para los promotores y $a \to u|_{\neg z}$ para los inhibidores. En el caso de los promotores, la regla $a \to u$ se puede usar en una región si todos los objetos de z están presentes en la misma región y son diferentes de las copias de objetos que utilizaría a. En el caso de los inhibidores, los objetos de z no deben estar presentes en la región y además deben de ser diferentes de los objetos de a.

- **Reglas de frontera.** Esta variante captura la idea de que muchas reacciones toman lugar en las membranas interiores de la célula, dependiendo del contenido del interior y el exterior de la membrana. Las reglas son de la forma $xu[_iv y \to xu'[_iv'y$ donde x, u, u', v, v', y son multiconjuntos de objetos e i es la etiqueta de la membrana. La interpretación es de que en la presencia de los objetos x fuera y los objetos y dentro de la membrana i, el multiconjunto u de fuera cambia al multiconjunto u' y, simultáneamente, el multiconjunto v interno se cambia a v'.

- **Polarización eléctrica.** Para evitar el no determinismo de la instrucción *in* en las indicaciones de destino, se asigna una polarización eléctrica tanto a objetos como membranas: $+, -$, y 0. Las polarizaciones de membranas se asignan desde la inicialización

del sistema o se pueden modificar en el transcurso de la computación. Los objetos cargados tienen que ir a cualquier membrana de nivel más bajo de la polarización opuesta, mientras que los objetos con polarización neutra (0) permanecen en la misma región o salen, dependiendo de la indicación de dirección $tar \in \{here, out\}$.

- **Prioridades.** Una opción para disminuir el no determinismo de un sistema es considerar una relación de prioridad entre las reglas de una región, en la forma de una relación de orden parcial. Existen dos interpretaciones para la prioridad:

  - Sentido estricto. Si una regla r_1 tiene prioridad sobre una regla r_2 y r_1 se puede aplicar, entonces r_2 no se puede aplicar, independientemente si la regla r_1 deja objetos que no puede usar.
  - Sentido débil. La regla con la prioridad maximal toma cuantos objetos le sea posible, y si todavía hay objetos sobrantes, la siguiente regla en el orden de prioridad se usa por tantos objetos como le sea posible, y así se continúa hasta que no sea posible aplicar más reglas.

- **Simporte y antiporte.** El simporte es el proceso en el que dos moléculas pasan juntas a través de una membrana mediante un canal específico. El antiporte es cuando dos moléculas pasan simultáneamente por un canal, pero en direcciones opuestas, mientras que el uniporte es cuando un sólo elemento se desplaza por la membrana. Las reglas se formalizan como (ab, in) o (ab, out) para simporte, $(a, out; b, in)$ para antiporte, mientras que (a, in) o (a, out) para las de uniporte.

- **Membranas activas.** Mientras que las reglas anteriores se enfocan en los objetos (su transporte y modificación) las siguientes se enfocan en las membranas *per se*.

  1. Reglas de evolución de objetos:
     $[_a a \to v]_h^e$, donde $h \in H, e \in \{+, -, 0\}, a \in O$ y $v \in O^*$.
     Se asocian con las membranas y dependen de la etiqueta y la carga de las membranas, pero no las involucran directamente en el sentido de que las membranas no toman parte en la aplicación de estas reglas ni se modifican por ellas.

  2. Reglas de comunicación de entrada:
     $a[_h\ ]_h^{e_1} \to [_h b]_h^{e_2}$, donde $h \in H, e_1, e_2 \in \{+, -, 0\}$ y $a, b \in O$.

Un objeto se introduce en la membrana y posiblemente se modifica durante el proceso; también la polarización de la membrana se puede modificar, pero no su etiqueta.

3. Reglas de comunicación de salida:
   $[_h a\ ]_h^{e_1} \rightarrow [_h\ ]_h^{e_2} b$, donde $h \in H, e_1, e_2 \in \{+, -, 0\}$ y $a, b \in O$.
   Un objeto se envía fuera de la membrana, posiblemente modificado durante este proceso; también la polarización de la membrana se puede modificar, pero no su etiqueta.

4. Reglas de disolución:
   $[_h a\ ]_h^{e} \rightarrow b$, donde $h \in H, e \in \{+, -, 0\}$ y $a, b \in O$.
   En reacción con un objeto, una membrana se puede disolver cuando el objeto especificado en la regla puede modificarse.

5. Reglas de división para membranas elementales:
   $[_h a\ ]_h^{e_1} \rightarrow [_h b\ ]_h^{e_2} [_h c\ ]_h^{e_3}$, donde $h \in H, e_1, e_2, e_3 \in \{+, -, 0\}$ y $a, b, c \in O$.
   En reacción con un objeto, la membrana se divide en dos membranas con la misma etiqueta, posiblemente de diferentes polarizaciones; el objeto especificado en la regla se reemplaza en las dos nuevas membranas por posiblemente nuevos objetos; los objetos restantes se duplican y podrían evolucionar en el mismo paso por reglas del tipo (1).

6. Reglas de división generales:
   $[_{h_1} a\ ]_{h_1}^{e_1} \rightarrow [_{h_2} b\ ]_{h_2}^{e_2} [_{h_3} c\ ]_{h_3}^{e_3}$, donde $h_1, h_2, h_3 \in H, e_1, e_2, e_3 \in \{+, -, 0\}$ y $a, b, c \in O$.
   En reacción con un objeto, la membrana se divide en dos membranas con diferente etiqueta. El concepto de cambio de etiqueta también se puede considerar en las reglas de tipo (2) y (3).

Una revisión más amplia de las variantes de los sistemas P se encuentra en los libros Păun *et al.* (2010) y Frisco (2009).

2.5.2. Modelo formal de un sistema P

A continuación se presenta el modelo formal de un sistema de transición Π.

$$\Pi = (O, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, i_0), \tag{16}$$

donde:

- O es el alfabeto de objetos, finito y no vacío,
- μ es la estructura de m membranas, etiquetadas con $1, \ldots, m$; se dice que la estructura de membrana, y por lo tanto el sistema, es de *grado* m,
- $w_1, \ldots, w_m$ son cadenas sobre O representando los multiconjuntos de objetos presentes en las regiones $1, \ldots, m$ de la estructura de membrana,
- $R_1, \ldots, R_m$ son conjuntos finitos de reglas de evolución asociadas a las regiones $1, \ldots, m$ de la estructura de membrana,
- i_0 la región de salida, una de las etiquetas $1, \ldots, m$ en el caso de ser una region interna, o 0 en caso de ser el entorno.

2.5.3. Computación

Una configuración es el estado actual de las membranas y su contenido. La transformación de una configuración en el tiempo t al tiempo $t+1$ se llama una transición, la cual está dada por la aplicación no determinística y paralela maximal de las reglas de evolución (i.e., un conjunto de reglas tal que no pueda agregarse ninguna regla más). Una computación es una secuencia de transiciones, y es exitosa si llega a un estado de paro. El estado de paro es una configuración donde no se puede aplicar ninguna regla a los objetos existentes, y la región de salida todavía existe en la configuración. La salida de una computación reside en la región i_0 (la región de salida). La región de salida puede tomarse como una región única, un conjunto de regiones, o incluso el entorno ($i_0 = 0$).

Los sistemas P se utilizan como dispositivos del tipo aceptación/reconocimiento (similares a los autómatas) y como dispositivos generadores (correspondientes a las gramáticas). En una problema de decisión, el sistema P reconocedor toma como entrada un multiconjunto que representa un caso del problema, y debe arrojar una respuesta positiva o negativa por medio de un número finito de transiciones. En el caso de un sistema generador Π, existen tres maneras de interpretar los resultados de una computación exitosa:

- $N(\Pi)$. En la computación exitosa, la solución es la cardinalidad del multiconjunto de salida. Debido al no determinismo, $N(\Pi)$ denota el conjunto de enteros calculados por Π.

- $Ps(\Pi)$: La solución se denota como el vector formado por las multiplicidades de los elementos del multiconjunto de salida.

- $L(\Pi)$: Cuando se consideran las cadenas que representan a los objetos saliendo del sistema, se denota el lenguaje de estas cadenas como $L(\Pi)$.

2.5.4. Sistemas P de tejido

Los sistemas P de tejido son una generalización de los sistemas P de transición, donde la estructura de membranas se apega a un grafo arbitrario, no a un árbol. Debido a esta particularidad, esta clase de sistemas se adaptan para la resolución de problemas en grafos. La inspiración de este tipo de sistemas radica en los sistemas biológicos formados por múltiples individuos, viviendo y cooperando en un cierto entorno el cual se utiliza para intercambiar información o elementos químicos. En este contexto, las membranas se llaman células, y las relaciones entre ellas canales o sinapsis.

Los sistemas de tejido se dividen en diversas clases, según las siguientes características:

- Comunicación y evolución de objetos y cadenas (*Neural-like P Systems, Tissue P Systems with Strings*). Esta clase de sistemas utiliza tanto reglas de reescritura como de comunicación.

- Canales de comunicación (*Tissue P Systems with Communication Channels*). En esta clase de sistemas, los canales entre células tienen un papel primordial en la computación, debido a que poseen un conjunto de estados que cambia según las reglas de transición. Estas reglas de transición son de tipo simporte y antiporte, por las células no crean objetos y es necesario especificar el contenido inicial del entorno.

- Procesamiento por picos (*Spiking Neural P Systems*). Este modelo se inspira en el hecho de que las neuronas se comunican por medio de impulsos eléctricos idénticos (picos). El cómputo se realiza mediante reglas para producir y olvidar picos.

- Estructura dinámica (*Population P Systems*). En este modelo se toman las reglas de membranas activas, por lo que se considera la idea de la división y muerte de células, así como la modificación de la estructura de todo el sistema durante el transcurso de la computación.

En este trabajo se utilizan los sistemas P de tejido con cadenas, referidos como sistemas de tejido de aquí en adelante.

Modelo formal de un sistema de tejido

A continuación se presenta el modelo formal de un sistema de tejido Π (Martín-Vide *et al.*, 2003).

$$\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, i_{out}), \tag{17}$$

donde:

- O es el alfabeto de objetos del sistema.
- $syn \subseteq \{1, \ldots, m\} \times \{1, \ldots, m\}$ es el conjunto de enlaces entre las células (sinapsis),
- $i_{out} \in \{1, 2, \ldots, m\}$ es el índice de la célula de salida,
- $\sigma_1, \ldots, \sigma_m$ son células de la forma

$$\sigma_i = (Q_i, s_{i,0}, w_{i,0}, P_i), \qquad 1 \leq i \leq m, \tag{18}$$

  donde:

  - Q_i es el conjunto finito de estados de la célula σ_i.
  - $s_{i,0} \in Q_i$ es el estado inicial.
  - $w_{i,0} \in O^*$ es el multiconjunto inicial de objetos en la célula.
  - P_i es un conjunto finito de reglas de la forma $sw \to s'xy_{go}z_{out}$, donde $s, s' \in Q_i$; $w, x \in O^*$; $y_{go} \in (O \times \{go\})^*$ y $z_{out} \in (O \times \{out\})^*$, con la restricción de que $z_{out} = \lambda$ para toda $i \in \{0, 1, 2, \ldots, m\}$ diferente de i_{out}.

En el modelo presentado originalmente por Martín-Vide *et al.* (2003), la célula de salida es única, esto es, $i_{out} \in \{1, 2, \ldots, m\}$. La variante utilizada en este trabajo permite que la salida sea un conjunto de células definidas por sus índices, i. e., $i_{out} \subseteq \{1, 2, \ldots, m\}$.

Reglas de comunicación y evolución

En las reglas es necesario definir el estado actual y el multiconjunto de entrada, para poder generar un nuevo estado y un multiconjunto de salida que se distribuirá mediante los indicadores de objetivo. Las reglas tienen la forma $sw \rightarrow s'(x, here)(y, go)(z, out)$ donde los estados $s, s' \in Q_i$, y w, x, y, z son multiconjuntos de objetos sobre O. En el estado s, σ_i consume el multiconjunto w y produce los multiconjuntos x, y, z; los objetos del multiconjunto x permanencen en la célula debido al indicador de objetivo *here*, los elementos del multiconjunto y se envían a las células debido al indicador *go*, y finalmente, si la célula es de salida, el multiconjunto z reporta el resultado de la computación mediante el indicador *out*.

Las sinapsis establecen la relación de comunicación entre las células. Una sinapsis (i, j) representa comunicación de σ_i hacia σ_j. En este caso, i es el ancestro de j ($i \in anc(j)$) y j es sucesor de i ($j \in suc(i)$). La comunicación entre un par de células consiste en un intercambio de multiconjuntos a través de la sinapsis (i, j), y se lanza mediante un producto con el indicador de dirección *go*. Según el planteamiento del sistema, existen tres modos en que se puede dar este intercambio:

1. Único (*one*): todo el producto se envía a una célula única.

2. Propagativo (*spread*): el producto se distribuye de manera no determinista entre todas las células sucesoras.

3. Replicativo (*repl*): cada sucesor recibe una copia del producto.

Con respecto a la aplicación paralela de las reglas, se pueden distinguir tres modos:

1. Mínimo (*min*): si una regla es aplicable, se utiliza sólo una vez en el multiconjunto.

2. Paralelo (*par*): si una regla es aplicable, se utiliza todas las veces posibles en el multiconjunto.

3. Maximal (*max*): se escoge un conjunto maximal de reglas (ninguna regla se puede añadir al conjunto) que puedan utilizarse sobre el multiconjunto mientras todas las reglas cambien al mismo estado.

La computación e interpretación de resultados es similar a la presentada en la Sección 2.5.3.

2.6. Trabajo Previo

2.6.1. Codificación de grafos mediante ADN

A continuación se describen dos codificaciones para resolver problemas en grafos mediante cómputo con ADN.

La primera codificación de grafos utilizando ADN se remonta al experimento de Adleman (1994). En su trabajo, representa una trayectoria en el grafo mediante una hebra doble. En la codificación de Adleman cada vértice y arista se representa con una cadena sencilla de 20 nucleótidos (**nt**) de longitud. Las secuencias de ADN de los vértices se generaron de manera aleatoria, mientras que las aristas se construyeron mediante la unión sufijo-prefijo de las cadenas complementarias a los vértices (Figura 3).

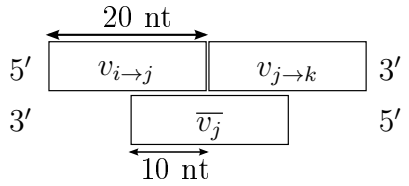

Figura 3: Representación de la trayectoria $v_i \to v_j \to v_k$ mediante la codificación de Adleman (1994).

García-Arnau *et al.* (2007b) utilizan la codificación de vértices y aristas de Adleman para resolver el problema MAX-CLIQUE. Sin embargo, la hebra doble representa un conjunto de vértices en lugar de una trayectoria, y las aristas se utilizan como hebras auxiliares para la concatenación de los vértices (Figura 4).

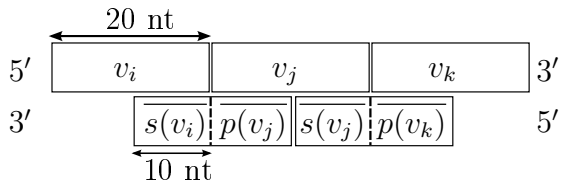

Figura 4: Representación del conjunto de vértices $\{v_i, v_j, v_k\}$ mediante la codificación de García-Arnau *et al.* (2007a).

2.6.2. Algoritmos moleculares para el problema MAX-CLIQUE

Ouyang *et al.* (1997)

Los autores presentan un algoritmo molecular que resuelve el problema mediante la aplicación de operaciones biológicas sobre una biblioteca combinatoria de ADN. Para la codificación de las soluciones se utilizó una representación binaria de los cliques, de manera que para un grafo de n vértices, un clique C se representa con una cadena binaria de n bits $b_1 \dots b_n$, donde $b_i = 1$ si $v_i \in C$ y $b_i = 0$ en otro caso.

En primer lugar, se construye una biblioteca combinatoria de ADN mediante un protocolo llamado ensamblado paralelo por superposición (POA por sus siglas en inglés, se describe a detalle en la Sección 4.3), donde cada hebra de ADN constituye un posible clique en el grafo. Después, se eliminan las cadenas que contienen vértices no adyacentes mediante enzimas de restricción, y finalmente se ordenan las cadenas restantes para la detección de resultados mediante electroforesis.

Head *et al.* (1999)

Los autores proponen un algoritmo molecular que utiliza plásmidos para representar los cliques. El plásmido tiene una secuencia conocida y regiones de interés que codifican para la representación binaria de los cliques. Por cada arista (u, v) que no pertenece al grafo, el conjunto de moléculas se divide en dos grupos, donde en el primero se elimina el segmento que codifica para u del plásmido, y en el segundo se elimina el segmento que codifica para v. Se liga el plásmido para que vuelva a su forma circular, y se unen las soluciones. De manera similar al algoritmo anterior, es necesaria la aplicación de $|\overline{V}|$ enzimas de restricción para la solución del problema.

Noort *et al.* (2002)

El enfoque presentado por los autores es el diseño de una computadora molecular para la resolución de hasta N vértices utilizando N^2 cámaras de filtrado. A partir de una biblioteca combinatoria donde cada clique posible está codificado en forma binaria mediante hebras de ADN, la computadora automatiza el proceso de filtrar los cliques inválidos mediante la comprobación de no adyacencia apoyándose en filtros de ADN, por lo que no es necesario el uso de enzimas de restricción. Además, la computadora cuenta con un módulo de ordenamiento para extraer las secuencias solución.

García-Arnau *et al.* (2007b)

Los autores proponen un sistema P constructivo con reescritura replicada e inhibidores. El sistema genera todos los cliques de un grafo tras n unidades de tiempo y los deposita en la membrana con la etiqueta $n+1$. A partir de este punto, en el instante $t = n+1$ reporta los cliques de tamaño n, en $t = n+2$ los de tamaño $n-1$ y así sucesivamente.

Sea el grafo $G = (\{a_1, \ldots, a_n\}, E)$, el sistema Π se define como la construcción:

$$
\begin{aligned}
\Pi &= (V', \mu', M_1, \ldots, M_{n+1}, R'_1, \ldots, R'_{n+1}) \text{ donde} \\
V' &= \{a_i,\ d_j \mid 1 \leq i \leq n,\ 0 \leq j \leq n\}, \\
\mu' &= [_{n+1} \ldots [_2 [_1]_1]_2 \ldots]_{n+1}, \\
M_1 &= \{d_0\}, M_i = \{\lambda\} \text{ con } 2 \leq i \leq n+1, \\
R'_i &= \{d_k w \to (d_{k+1} w a_i, out)_{\neg U_i} || (d_k w, out) \mid \\
&\quad w \in \{a_j \mid 1 \leq j \leq i-1\}^*, |w| = k, 0 \leq k \leq i-1\} \\
&\quad \text{con } U_i = \{a_j \in V \mid \{a_i, a_j\} \in E'\}, 1 \leq i \leq n \\
R'_{n+1} &= \{d_k w \to (d_{k+1} w, here) \mid w \in \{a_j \mid 1 \leq j \leq n\}^*, \\
&\quad 0 \leq |w| \leq n, 0 \leq k \leq n-1\} \\
&\quad \cup \{d_n w \to (d_n w, out) \mid w \in \{a_j \mid 1 \leq j \leq n\}^*, \\
&\quad 0 \leq |w| \leq n\}.
\end{aligned} \tag{19}
$$

Martínez-Pérez y Zimmermann (2009)

Los autores proponen un algoritmo de cómputo molecular que utiliza el modelo de etiquetas para resolver el problema k-clique en $n^2 + 5n$ pasos, donde n es el número de vértices del grafo. Se requieren $O(1)$ tubos y $O(\binom{n}{k})$ cadenas de bits iniciales, las cuales representan todos los posibles cliques de k vértices del grafo. El algoritmo examina cada posible par de posiciones en las cadenas i, j tal que $0 \leq i < n$, $i + 1 \leq j \leq n$. Si las posiciones i, j están encendidas y los vértices v_i y v_j no son adyacentes, entonces se elimina el conjunto de hebras que tengan las posiciones i, j encendidas. En caso contrario, se analiza el siguiente par i, j.

Los sistemas de tejido se basan en las redes de comunicación formadas por los canales de proteínas entre células vivas. En este capítulo se muestra un sistema de tejido para resolver el problema MAX-CLIQUE mediante el procesamiento de multiconjuntos que representan los cliques del grafo, utilizando las formas en modo maximal y una comunicación replicativa entre las células. En primer lugar se presenta la definición del problema MAX-CLIQUE, para después comenzar con una introducción de la solución propuesta y su enunciado formal. Posteriormente se realiza la demostración formal de corrección del sistema y finalmente se aborda desde un punto de vista algorítmico.

3.1. Sistema P de tejido propuesto

3.1.1. Descripción

En el Capítulo 2 se menciona que el sistema P de tejido es una generalización del sistema P de transición en el cual las membranas elementales (células) se comunican de manera arbitraria, por lo que se puede representar como un grafo donde las células se corresponden a los vértices y su interconexión por medio de aristas. Considere el problema MAX-CLIQUE para un grafo $G = (V, E)$. Partiendo de la relación entre un sistema P de tejido y el grafo arbitrario que lo representa, suponga un sistema Π tal que su representación es el grafo G. De esta manera, las células $\{\sigma_1, \ldots, \sigma_n\} \in \Pi$ se corresponden a los vértices $V = \{v_1, \ldots, v_n\}$, y las sinapsis syn se corresponden a las aristas en E, esto es, $syn = \{(i, j) \mid (v_i, v_j) \in E\}$. El resultado esperado del sistema son los cliques máximos del grafo, por lo que los objetos que serán procesados por las células representarán cliques (no necesariamente máximos). Los objetos del sistema se consideran como conjuntos, donde los elementos representan a los vértices del grafo. Finalmente, la salida se recogerá en el entorno del sistema. A continuación se describe el funcionamiento de cada una de las reglas de transición requeridas en las células para resolver el problema.

3.1.2. Reglas de transición

Las operaciones asociadas a las reglas de transición son las siguientes:

1. **Creación de objetos iniciales.** Se provee al sistema de los objetos iniciales para la computación que servirán como base para generar los demás cliques.

2. **Incorporación de vértices a los cliques.** Mediante esta función se incrementa el tamaño de los cliques, buscando objetos en el vecindario que mantengan al objeto como un clique.

3. **Eliminación de objetos que no representan cliques.** El sistema desecha aquellos objetos que no cumplan con las características de un clique, descartando la generación de respuestas erróneas.

4. **Reporte de soluciones.** Cuando el sistema detecta una solución, ésta se envía al entorno para almacenarse como resultado.

Para realizar estas cuatro funciones se vinculará la célula σ_i con el vértice v_i; en este caso se dirá que v_i es el *vértice correspondiente* a σ_i. Además, cada σ_i cuenta con información de v_i dentro de G, específicamente el complemento de la función de adyacencia $N(i)$.

3.1.3. Descripción general del funcionamiento del sistema

Una vez definido el comportamiento de las células, se detalla el funcionamiento general del sistema.

1. **Inicialización.** Cada célula crea un objeto inicial, un conjunto unitario formado por el vértice correspondiente de la célula. Dicho objeto se replica para iniciar el procesamiento y se reporta al entorno para generar la solución trivial del clique de tamaño $k = 1$. Por ejemplo, la célula σ_3 reportará como solución un objeto $z = \{v_3\}$ y enviará a cada una de sus vecinas una copia de z.

2. **Procesamiento.** De manera paralela, las células procesan los objetos recibidos por medio de la replicación de las células vecinas en el instante anterior.

   2.1. **Verificación.** La célula utiliza la no adyacencia para verificar si la unión del objeto recibido y el vértice correspondiente resultaría en un clique válido.

2.1.1. **Eliminación.** Si el clique resultara no válido, el objeto se elimina del sistema.

2.1.2. **Incorporación de vértices.** Si el clique resultara válido, entonces se crea el objeto correspondiente, se reporta como solución y se replica.

3. **Finalización.** Si en el sistema no se realiza alguna operación de replicación, entonces la computación se detiene, tomándose como solución los objetos liberados al entorno en el instante anterior. En caso contrario, se regresa al paso 2.

3.1.4. Modelo formal

A continuación se muestra el modelo formal del sistema P de tejido propuesto según la notación descrita por Martín-Vide *et al.* (2003).

Sea $G = (V, E)$ un grafo de grado n no dirigido y sin autociclos. Se construye el sistema de tejido:

$$\begin{aligned}
&\Pi(G)_{max,repl} = (O, \sigma_1, \ldots, \sigma_n, syn, \{\sigma_1, \ldots, \sigma_n\}),\\
&\quad \text{donde:}\\
&\qquad O = \{d, v_i \mid 1 \leq i \leq n\},\\
&\qquad syn = \{(i,j) \mid (v_i, v_j) \in E\},\\
&\qquad \sigma_i = (\{s\}, s, \{d\}, R_{ij}) \text{ para } i = 1, 2, \ldots, n \text{ y } 1 \leq j \leq 3,\\
&\qquad R_{i1} = \{sd \to s(v_i, go)(v_i, out)\},\\
&\qquad R_{i2} = \{sz \to s \mid \ z \in V^+ \ \wedge \ (\exists v_j \in z)(j \notin N(i))\},\\
&\qquad R_{i3} = \{sz \to s(z \cup \{v_i\}, go)(z \cup \{v_i\}, out) \mid\\
&\qquad\qquad z \in V^+ \wedge \ (\forall v_j \in z)(j \in N(i))\}.
\end{aligned} \tag{20}$$

A continuación se describen los componentes del sistema:

- $\Pi(G)_{max,repl} = (O, \sigma_1, \ldots, \sigma_n, syn, \{\sigma_1, \ldots, \sigma_n\})$: el sistema Π utiliza el modo maximal de procesamiento y replicativo de comunicación, y se compone de: el alfabeto O, las células $\sigma_1, \ldots, \sigma_n$, el conjunto de sinapsis syn, y el conjunto de células de salida $\{\sigma_1, \ldots, \sigma_n\}$.

- $O = \{d, v_i \mid 1 \leq i \leq n\}$: el alfabeto del sistema conformado por $n + 1$ símbolos (uno por cada vértice en V y el símbolo inicial d).

- $syn = \{(i,j) \mid (v_i, v_j) \in E\}$: el conjunto de sinapsis del sistema. Existirá una sinapsis entre σ_i y σ_j si y sólo si existe una arista entre los vértices v_i y v_j.

- $\sigma_i = (\{s\}, s, \{d\}, R_{ij})$ para $i = 1, 2, \ldots, n$ y $1 \leq j \leq 3$: cada célula i contiene un estado único s el cual también es inicial, su multiconjunto de entrada es $[\{d\}]$ y posee el conjunto de reglas de transición $R_{ij}, 1 \leq j \leq 3$.

- El conjunto de reglas R_{ij}:

  - $R_{i1} = \{sd \rightarrow s(v_i, go)(v_i, out)\}$: si la célula i está en el estado s y contiene un objeto d, entonces permanece en el estado s y crea un objeto $z = v_i$ que se replica (instrucción go) y envía al entorno (instrucción out).

  - $R_{i2} = \{sz \rightarrow s \mid \ z \in V^+ \ \wedge \ (\exists v_j \in z)(j \notin N(i))\}$: si la célula i está en el estado s y contiene un objeto z, cuya representación en forma de cadena pertenece a la cerradura positiva de V, y además contiene un símbolo que represente a un vértice no adyacente a v_i, entonces la célula permanece en el estado s y elimina el objeto z.

  - $R_{i3} = \{sz \rightarrow s(z \cup \{v_i\}, go)(z \cup \{v_i\}, out) \mid z \in V^+ \wedge \ (\forall v_j \in z)(j \in N(i))\}$: si la célula i está en el estado s y contiene un objeto z cuya representación en forma de cadena pertenece a la cerradura positiva de V y además todos los símbolos de z representan vértices adyacentes a v_i, entonces se crea un objeto z' mediante la unión de z y el símbolo v_i. Una vez creado, z' se replica (instrucción go) y envía al entorno (instrucción out).

Multiconjuntos del sistema

El sistema de tejido propuesto procesa multiconjuntos mediante las reglas de transición dentro de las células y la comunicación por replicación. Tomando esto en cuenta, es posible modelar los multiconjuntos con base en el instante de procesamiento y la agrupación de las células de donde proviene la replicación. Para ejemplificar los multiconjuntos, se utiliza como caso a resolver el grafo de la Figura 5A.

- I_i^t: Entrada de la célula σ_i en el instante t. Observe que por definición, el multiconjunto de entrada en cada una de las células del sistema es $[\{d\}]$ (Figura 5B).

$$I_i^1 = [\{d\}], \ 1 \leq i \leq n. \tag{21}$$

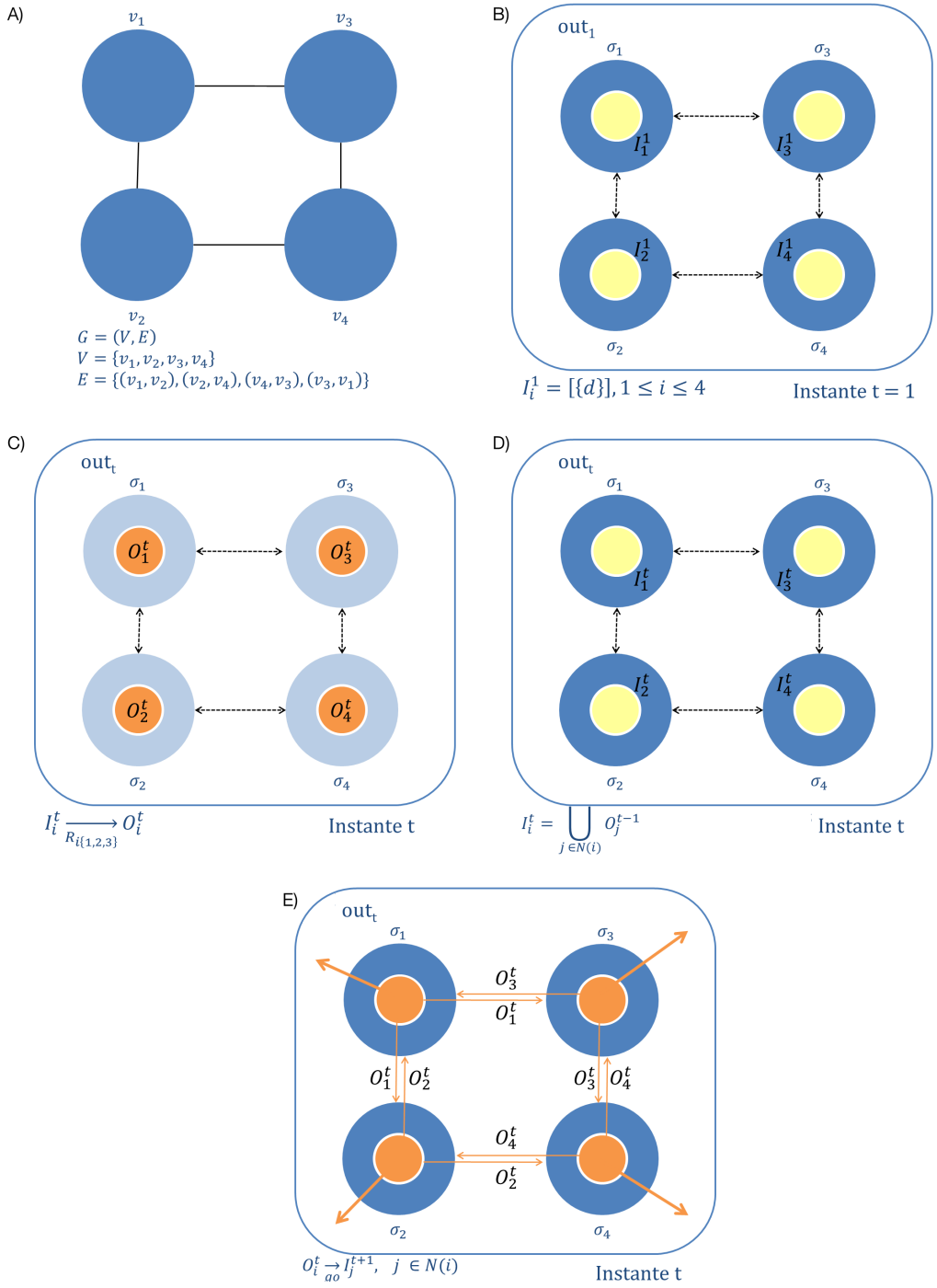

Figura 5: Relación entre las entradas y salidas del sistema de tejido propuesto: A) Caso a resolver. B) Sistema de tejido construido a partir del grafo G. C) Transformación de I_i^t en O_i^t mediante las reglas de transición. D) Entradas del sistema con base en las salidas de las células vecinas en el instante $t > 1$. E) Comunicación replicativa entre las células y el entorno por medio de las reglas $R_{i\{1,3\}}$.

- O_i^t: Salida de la célula σ_i en el instante t. O_i^t es el resultado de aplicar las reglas de transición $R_{i\{1,2,3\}}$ sobre los elementos de I_i^t (Figura 5C). Observe que las reglas R_{i1} y R_{i2} no involucran la eliminación de objetos, y el producto de dichas reglas invoca la replicación a las células vecinas. De aquí que la entrada de la célula σ_i se pueda generalizar como la unión de las salidas de las células vecinas en el instante anterior (Figura 5D):

$$I_i^t = \bigcup_{j \in N(i)} O_j^{t-1},\ t > 1. \tag{22}$$

- out_t: Contenido del entorno o salida del sistema en el instante t. El multiconjunto out_t recibe los objetos de salida de todas las células $\sigma_1 \ldots \sigma_n$ mediante la instrucción *out* de la regla R_{i1} en $t = 1$ y la regla R_{i3} en $t > 1$ (Figura 5E). De esta manera, se puede definir out_t como:

$$out_t = \bigcup_{1 \leq i \leq n} O_i^t. \tag{23}$$

Estado de paro

En el Capítulo 2 se menciona que un sistema de tejido llega al estado de paro cuando no es posible aplicar más reglas de transición a los multiconjuntos presentes en las células. Dado que ninguna de las reglas del sistema propuesto trabaja con conjuntos vacíos, una condición de paro consiste en que todas las células se encuentren vacías, ocasionando que ninguna regla se pueda aplicar.

Sea t_h el instante en que el sistema llega al estado de paro. Se cumple entonces que

$$\bigcup_{1 \leq i \leq n} I_i^{t_h} = \emptyset. \tag{24}$$

Observe que por definición, $I_i^{t_h} = \bigcup_{j \in N(i)} O_j^{t_h - 1}$, por lo que

$$\bigcup_{j \in N(i)} O_j^{t_h - 1} = \emptyset,\ 1 \leq i \leq n \tag{25}$$

también representa una condición de paro.

3.1.5. Ejemplo de ejecución

A continuación se presenta un ejemplo de ejecución del sistema de tejido propuesto. Considere el grafo de la Figura 6A. La Tabla 1 muestra el contenido de cada una de las células del sistema (Figura 6B) por unidad de tiempo, hasta encontrar los cliques de tamaño máximo dados por $\{v_1, v_2, v_5\}$ y $\{v_2, v_3, v_5\}$. Para cada célula se presenta el contenido del multiconjunto de entrada I_i^t y el multiconjunto de salida O_i^t en el instante t. De igual manera se presenta la salida del sistema al entorno en el instante t (out_t).

Figura 6: Caso de prueba para el sistema de tejido propuesto: A) Caso a resolver. B) Estructura del sistema de tejido.

Tabla 1: Resultados de la ejecución del sistema sobre el grafo de la Figura 6A.

Célula		$t = 1$	$t = 2$	$t = 3$	$t = 4$
σ_1	I_1^t	$\{d\}$	$\{v_2\}, \{v_5\}$	$\{v_1v_2\}, \{v_2v_3\}, \{v_2v_5\}^2,$ $\{v_1v_5\}, \{v_3v_5\}$	$\{v_1v_2v_5\}^4, \{v_2v_3v_5\}^4$
	O_1^t	$\{v_1\}$	$\{v_1v_2\}, \{v_1v_5\}$	$\{v_1v_2v_5\}^2$	$\emptyset$
σ_2	I_2^t	$\{d\}$	$\{v_1\}, \{v_3\}, \{v_5\}$	$\{v_1v_2\}, \{v_1v_5\}^2, \{v_2v_3\},$ $\{v_3v_4\}, \{v_3v_5\}^2, \{v_2v_5\}$	$\{v_1v_2v_5\}^4, \{v_2v_3v_5\}^4$
	O_2^t	$\{v_2\}$	$\{v_1v_2\}, \{v_2v_3\}, \{v_2v_5\}$	$\{v_1v_2v_5\}^2, \{v_2v_3v_5\}^2$	$\emptyset$
σ_3	I_3^t	$\{d\}$	$\{v_2\}, \{v_4\}, \{v_5\}$	$\{v_1v_2\}, \{v_2v_3\}, \{v_2v_5\}^2,$ $\{v_3v_4\}, \{v_1v_5\}, \{v_3v_5\}$	$\{v_1v_2v_5\}^4, \{v_2v_3v_5\}^4$
	O_3^t	$\{v_3\}$	$\{v_2v_3\}, \{v_3v_4\}, \{v_3v_5\}$	$\{v_2v_3v_5\}^2$	$\emptyset$
σ_4	I_4^t	$\{d\}$	$\{v_3\}$	$\{v_2v_3\}, \{v_3v_4\}, \{v_3v_5\}$	$\{v_2v_3v_5\}^2$
	O_4^t	$\{v_4\}$	$\{v_3v_4\}$	$\emptyset$	$\emptyset$
σ_5	I_5^t	$\{d\}$	$\{v_1\}, \{v_2\}, \{v_3\}$	$\{v_1v_2\}^2, \{v_1v_5\}, \{v_2v_3\}^2,$ $\{v_2v_5\}, \{v_3v_4\}, \{v_3v_5\}$	$\{v_1v_2v_5\}^4, \{v_2v_3v_5\}^4$
	O_5^t	$\{v_5\}$	$\{v_1v_5\}, \{v_2v_5\}, \{v_3v_5\}$	$\{v_1v_2v_5\}^2, \{v_2v_3v_5\}^2$	$\emptyset$
Salida del sistema	out_t	$\{v_1\}, \{v_2\}$ $\{v_3\}, \{v_4\}$ $\{v_5\}$	$\{v_1v_2\}^2, \{v_1v_5\}^2, \{v_2v_3\}^2,$ $\{v_2v_5\}^2, \{v_3v_4\}^2, \{v_3v_5\}^2$	$\{v_1v_2v_5\}^6, \{v_2v_3v_5\}^6$	$\emptyset$

En el instante $t = 1$ todas las celdas I_i^t contienen el conjunto $\{d\}$ debido a la inicialización del sistema. Como se puede observar, la regla R_{i1} es la única aplicable, por lo que $\{d\}$ se transforma en $\{v_i\}$ y además se envía una copia de $\{v_i\}$ tanto a las células vecinas $\{\sigma_j \mid j \in N(i)\}$ por medio de la replicación (instrucción *go*), como al entorno en el instante 1, out_1 (instrucción *out*). Al final del ciclo, los objetos enviados al entorno corresponden a los cliques triviales de tamaño 1, esto es,

$$out_1 = [\{v_1\}, \{v_2\}, \{v_3\}, \{v_4\}, \{v_5\}]. \tag{26}$$

En el instante $t = 2$, la entrada I_i^2 es el resultado de la unión de las salidas de las células vecinas a σ_i. Al verificar las condiciones en las reglas, se determina que R_{i1} y R_{i2} no pueden aplicarse ya que todos los elementos de los objetos contienen vértices adyacentes a v_i, por lo que solamente se aplica la regla R_{i2} en todas las células. Mediante esta regla, la célula σ_i agrega un elemento v_i a cada objeto en I_i^2 y, de manera similar al caso anterior, la salida O_i^2 se replica enviándose a las células vecinas y al entorno como resultado. Al final del ciclo, la salida al entorno del sistema son todos los cliques de tamaño 2:

$$out_2 = [\{v_1v_2\}^2, \{v_1v_5\}^2, \{v_2v_3\}^2, \{v_2v_5\}^2, \{v_3v_4\}^2, \{v_3v_5\}^2]. \tag{27}$$

Para el instante $t = 3$, se analizará el caso de la celda I_3^3 de la Tabla 1. Debido al procesamiento en el instante anterior, la entrada $I_3^3 = [\{v_1v_2\}, \{v_2v_3\}, \{v_2v_5\}^2, \{v_3v_4\}, \{v_1v_5\}, \{v_3v_5\}]$. De acuerdo al grafo de la Figura 6A, la función vecindario de v_3 es $N(3) = \{2, 4, 5\}$. Observe que los objetos $\{v_1v_2\}, \{v_2v_3\}, \{v_3v_4\}, \{v_1v_5\}$ y $\{v_3v_5\}$ en I_3^3 contienen elementos cuyos índices no pertenecen a $N(3)$ (v_1 y v_3), por lo que la regla R_{32} es aplicable en este grupo de objetos, eliminándose de σ_3. Por otro lado, todos los elementos de los objetos $\{v_2v_5\}^2$ tienen sus índices en $N(3)$, por lo que la regla R_{33} es aplicable y a ambos objetos se les añade el elemento v_3, generando los objetos $\{v_2v_3v_5\}^2$ y almacenándolos en O_3^3. Finalmente, se envía una copia de los objetos al entorno (out_3) y las células vecinas (I_2^4, I_4^4 y I_5^4). Una vez que se realiza el procesamiento en paralelo en todas las células, la salida out_3 es

$$out_3 = [\{v_1v_2v_5\}^6, \{v_2v_3v_5\}^6], \tag{28}$$

en la cual se encuentran los cliques máximos del grafo, $supp(out_3) = \{\{v_1v_2v_5\}, \{v_2v_3v_5\}\}$.

Observe que en el instante $t = 4$ la entrada en las células son los cliques máximos del grafo. Las células que pertenecen a los cliques máximos eliminan los objetos mediante la regla R_{i2}, debido a que su propio índice no está incluido en su función vecindario. En el caso de σ_4, que contiene los objetos $\{v_2v_3v_5\}^2$, se observa que los índices $2, 5 \notin N(4)$, por lo que la regla R_{i2} también es aplicable, eliminándose los objetos de la célula. Tras finalizar este ciclo, los multiconjuntos de salida en todas las células quedan vacíos, lo que conlleva a un estado de paro como se explica en la Sección 3.1.4.

A partir de este ejemplo se pueden observar dos características importantes:

1. El multiconjunto out_t contiene cliques de tamaño $k = t$ si $t \leq \omega(G)$, donde $\omega(G)$ es el tamaño del clique máximo en el grafo.

2. El sistema deja de producir objetos en el instante $t = \omega(G) + 1$.

Estas dos propiedades se demuestran formalmente en la Sección 3.2.

3.2. Corrección y complejidad del sistema propuesto

En esta sección se presenta la demostración formal de las siguientes propiedades del sistema:

1. **Corrección.** El sistema arroja respuestas correctas al problema que se resuelve.

2. **Determinismo.** El sistema es determinista en la aplicación de las reglas, por lo que genera la misma salida para la misma entrada durante cada computación.

3. **Ejecución en un tiempo lineal al tamaño del problema.** El sistema se ejecuta en un tiempo lineal al tamaño del problema.

La estructura para demostrar cada una de estas propiedades sigue el esquema que se muestra en la Figura 7.

Lema 3.1. (Correción para $k = 1$). *La salida del sistema en el instante $t = 1$ ($supp(out_1)$) contiene todos los cliques de tamaño $k = 1$.*

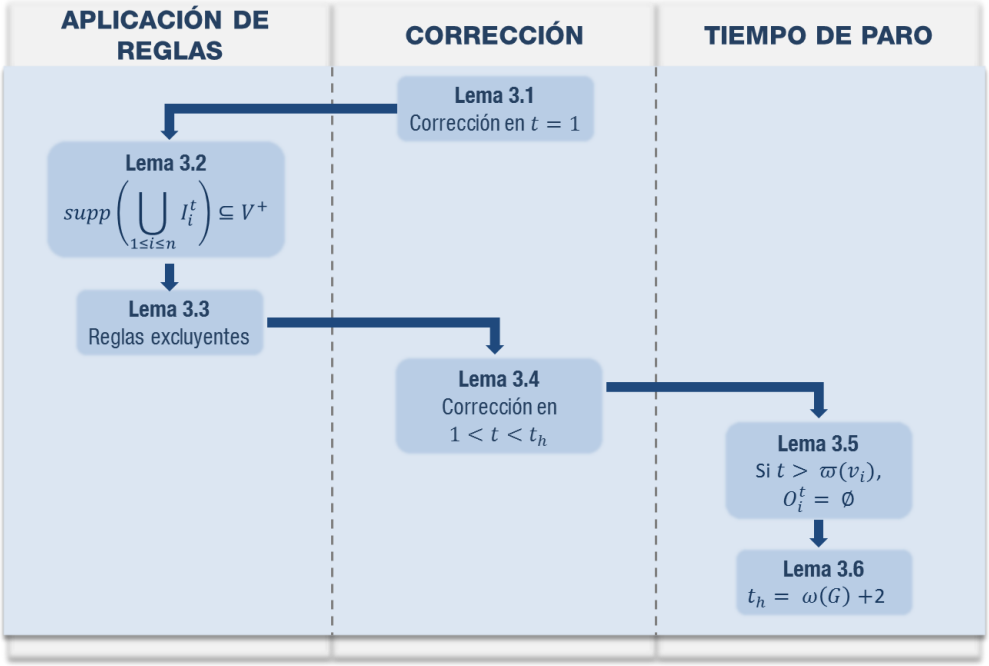

Figura 7: Estructura de la demostración de corrección y complejidad.

Demostración. Suponga que el sistema se encuentra inicializado, esto es, cada célula σ_i, $1 \leq i \leq n$, contiene el objeto $\{d\}$. Dado que el elemento $d \notin V^+$, la regla R_{i1} es la única aplicable. De aquí que

$$O_i^1 = [\{v_i\}], 1 \leq i \leq n. \tag{29}$$

La unión de todos los multiconjuntos de salida en $t = 1$ es

$$\begin{aligned} \bigcup_{1\leq i\leq n} O_i^1 &= [\{v_1\}] \cup \cdots \cup [\{v_n\}] \\ &= \{(\{v_1\}, 1), \ldots, (\{v_n\}, 1)\}, \end{aligned} \tag{30}$$

y, por definición, se tiene que

$$out_1 = \bigcup_{1\leq i\leq n} O_i^1, \tag{31}$$

por lo que

$$\begin{aligned} supp(out_1) &= supp(\bigcup_{1\leq i\leq n} O_i^1) \\ &= \{\{v_1\}, \ldots, \{v_n\}\}, \end{aligned} \tag{32}$$

el cual es el conjunto formado por conjuntos unitarios que contienen a cada uno de los vértices

en V (cliques triviales de tamaño 1). Por lo tanto, la salida del sistema en $t = 1$ contiene todos los cliques de tamaño $k = 1$. $\square$

Lema 3.2. *Sea t_h el instante de paro del sistema. Si $1 < t < t_h$, entonces $supp(\bigcup_{1\leq i\leq n} I_i^t) \subseteq V^+$.*

Demostración. Se demuestra por inducción en t. Suponga que $t = 2$ y que el sistema no se encuentra en estado de paro. Entonces, por definición

$$I_i^2 = \bigcup_{j\in N(i)} O_j^1 \neq \emptyset, \tag{33}$$

por lo que

$$\bigcup_{1\leq i\leq n} I_i^2 = \bigcup_{1\leq i\leq n} \left(\bigcup_{j\in N(i)} O_j^1 \right). \tag{34}$$

De acuerdo al Lema 3.1, el multiconjunto $O_i^1 = [\{v_i\}]$, para $1 \leq i \leq n$.

Sustituyendo en (34)

$$\begin{aligned} \bigcup_{1\leq i\leq n} I_i^2 &= \bigcup_{1\leq i\leq n} \left(\bigcup_{j\in N(i)} [\{v_j\}] \right) \\ &= \bigcup_{1\leq i\leq n} [\{v_j\} \mid j \in N(i)]. \end{aligned} \tag{35}$$

Observe que $\{v_j\}$ aparecerá tantas veces en $\bigcup_{1\leq i\leq n} I_i^2$ como vecinos tenga:

$$\bigcup_{1\leq i\leq n} I_i^2 = \{(\{v_j\}, deg_G(v_j)) \mid v_j \in V\}, \tag{36}$$

y así,

$$supp(\bigcup_{1\leq i\leq n} I_i^2) = \{\{v_j\} \mid v_j \in V\}. \tag{37}$$

Dado que todos los elementos $v_j \in V$ y $V \subseteq V^+$, se deduce que $supp(\bigcup_{1\leq i\leq n} I_i^2) \subseteq V^+$, por lo que se demuestra el caso base.

Ahora suponga que la condición se cumple hasta un instante $t = k$, $t < t_h$. Suponga además que $supp(\bigcup_{1\leq i\leq n} I_i^k) \subseteq V^+$. Se demostrará para $t = k + 1$, $t < t_h$. Primeramente, debido a que $t \neq t_h$, se tiene que $\bigcup_{1\leq i\leq n} I_i^k \neq \emptyset$. Dado que todos los objetos de $supp(\bigcup_{1\leq i\leq n} I_i^k)$

pertenecen a V^+, entonces tanto la regla R_{i2} como R_{i3} son aplicables. Observe que si todos los objetos de $supp(\bigcup_{1\leq i\leq n} I_i^k)$ fueran procesados por la regla R_{i2}, todos los objetos se eliminarían, llegando así al estado de paro en el instante $t_h = k+1$, lo cual es una contradicción. Entonces, para que la condición se mantenga, al menos un objeto de $supp(\bigcup_{1\leq i\leq n} I_i^k)$ se tiene que procesar por la regla R_{i3} para alguna $1 \leq i \leq n$. Por definición, la regla R_{i3} no elimina objetos, sino que les agrega un elemento $v_j \in V$, por lo que $\bigcup_{1\leq i\leq n} I_i^{k+1} \neq \emptyset$. Por la hipótesis inductiva, todos los objetos de $supp(\bigcup_{1\leq i\leq n} I_i^k)$ se componen de elementos de V; dado que la regla R_{i3} agrega un elemento de V, entonces los objetos procesados por dicha regla, $supp(\bigcup_{1\leq i\leq n} I_i^{k+1})$, siguen perteneciendo a V^+, por lo tanto $supp(\bigcup_{1\leq i\leq n} I_i^{k+1}) \subseteq V^+$.

□

Lema 3.3. (Determinismo) *Cada objeto en $supp(I_k^t) \neq \emptyset$, $t > 1$, lo procesan las reglas R_{i2} o R_{i3} de manera excluyente.*

Demostración. Suponga que $z \in supp(I_k^t)$ para $t > 1$. Dado que $I_k^t \subseteq \bigcup_{1\leq i\leq n} I_i^t$ para todo $1 \leq i \leq n$, entonces $z \in supp(\bigcup_{1\leq i\leq n} I_i^t)$. Por el Lema 3.2, $supp(\bigcup_{1\leq i\leq n} I_i^t) \subseteq V^+$, por lo que $z \in V^+$ y tanto las reglas R_{i2} y R_{i3} pueden solamente aplicarse en z. Sean C_{i2} y C_{i3} las condiciones definidas para las reglas R_{i2} y R_{i3}, respectivamente. Esto es,

$$C_{i2} = (\exists v_j \in z)(j \notin N(i)), \tag{38}$$

y

$$C_{i3} = (\forall v_j \in z)(j \in N(i)). \tag{39}$$

Observe que $C_{i2} = \neg C_{i3}$, por lo que $C_{i2} \cap C_{i3} = \emptyset$ y de esta manera las reglas R_{i2} y R_{i3} dividen los objetos a procesar en una partición de dos subconjuntos: el primero contiene los objetos con al menos un elemento no adyacente, mientras que el segundo contiene los objetos que se conforman únicamente de vertices adyacentes. Por lo tanto se concluye que cada objeto $z \in supp(I_k^t) \neq \emptyset$, $t > 1$, lo procesa solamente una de las dos reglas de manera excluyente. □

Lema 3.4. *Para toda $1 \leq i \leq n$, si $O_i^t \neq \emptyset$ entonces $supp(O_i^t)$ sólo contiene objetos que representan cliques de tamaño t.*

Demostración. Se demuestra por inducción. El caso base ($t = 1$) se demuestra en el Lema 3.1.

Suponga que la hipótesis se mantiene hasta el instante $t = k$. Se demostrará que se cumple la hipótesis para la iteración $t+1$. Sean las células σ_i y σ_j tal que $j \in N(i)$ y $O_j^t \neq \emptyset$. Por la hipótesis, todos los objetos de $supp(O_j^t)$ son cliques de tamaño t. Al comenzar el ciclo $t+1$ se comunican los objetos de O_j^t a I_i^{t+1} por medio de replicación. De acuerdo al Lema 3.3, cada objeto $z \in supp(I_i^{t+1})$ se puede procesar de dos maneras:

- Caso 1: utilizando la regla R_{i2}. En este caso, existe al menos un elemento $v_l \in z$ tal que $l \notin N(i)$, por lo que $z' = z \cup \{v_i\}$ no es un clique y z se elimina del sistema. Observe que si todos los objetos los procesa la regla R_{i2}, entonces $O_i^{t+1} = \emptyset$ y la hipótesis inductiva deja de aplicar.

- Caso 2: utilizando la regla R_{i3}. En este caso, para todo $v_l \in z$ se cumple que $l \in N(i)$, por lo que el objeto $z' = z \cup \{v_i\}$ es por definición un clique. El objeto z' es entonces creado y *almacenado* en O_i^{t+1}. Observe que $|z'| = |z \cup \{v_i\}| = |z| + |\{v_i\}| = t+1$. Por lo tanto, si $O_i^{t+1} \neq \emptyset$ entonces O_i^{t+1} sólo contiene cliques de tamaño $t+1$.

□

Lema 3.5. *En el instante $t > \varpi(v_i)$ se cumple que $O_i^t = \emptyset$.*

Demostración. Se demuestra por contradicción. Suponga que $O_i^t \neq \emptyset$ y $z \in O_i^t$. Suponga además que $t = \varpi(v_i) + \epsilon$ con $\epsilon > 0$. Debido a que $z \in O_i^t$, se deduce que z lo produjo alguna de las reglas $R_{i\{1,3\}}$, por lo que que $v_i \in z$. Por el Lema 3.4, $|z| = t$. Esto implica que z es un clique con un número ϵ de vértices mayor que el clique maximal al que pertenece v_i, lo cual es una contradicción. Por lo tanto, se demuestra que $O_i^t = \emptyset$. □

Lema 3.6. (Estado de paro) *El sistema se detiene en el instante $t_h = \omega(G) + 2$.*

Demostración. Por definición se llega al estado de paro en el instante t_h cuando no hay objetos por procesar en el sistema, esto es, $\bigcup_{1 \leq i \leq n} I_i^{t_h} = \emptyset$. De aquí que $I_i^{t_h} = \emptyset$ para toda célula σ_i. Observe que $I_i^{t_h} = \bigcup_{j \in N(i)} O_j^{t_h - 1}$, por lo que $\bigcup_{j \in N(i)} O_j^{t_h - 1} = \emptyset$ también representa una condición de paro. Por el Lema 3.5 se tiene que $O_i^t = \emptyset$ cuando $t > \varpi(v_i)$, lo que implica que el sistema quedará vacío cuando las células procesen el mayor clique maximal. Dado que $\max_{1 \leq i \leq n} \varpi(v_i) = \omega(G)$, entonces

$$t_h - 1 > \omega(G). \tag{40}$$

Por lo tanto, el estado de paro se cumple en el instante $t_h = \omega(G) + 2$. □

Teorema 3.1. *El sistema de tejido* $\Pi(G)_{max,repl}$ *es correcto, determinista, y se ejecuta en un tiempo lineal al tamaño del problema.*

Demostración. La corrección del sistema se demuestra en los Lemas 3.1 y 3.4 para el instante $t = 1$ y $t > 1$, respectivamente.

El determinismo se comprueba en el Lema 3.1 al mostrar que sólo puede aplicarse la regla R_{i1} en $t = 1$. Para el instante $t > 1$, el Lema 3.3 muestra que por cada iteración, cada objeto se consume de manera excluyente por las reglas R_{i2} y R_{i3} según su composición, por lo que no hay una selección aleatoria de reglas.

El tiempo de ejecución del sistema se demuestra en el Lema 3.6, donde se prueba que el estado de paro se produce en el instante $t_h = \omega(G) + 2$, el cual es lineal en el tamaño del problema. □

3.3. Simulación del sistema propuesto

En esta sección se presenta una simulación del modelo formal del sistema de tejido mediante una serie de algoritmos paralelos. La idea general consiste en representar los multiconjuntos del sistema I_i, O_i y out_i mediante estructuras de datos y utilizar procesamiento paralelo para ejecutar a cabo las reglas de transición en los objetos que contienen.

3.3.1. Elementos del sistema

La multiplicidad se presenta en los multiconjuntos del sistema debido a que un objeto lo pueden producir distintas células en un determinado momento[1]. Sin embargo, esta redundancia se descarta debido a que la solución reside en los objetos diferentes, por lo que los multiconjuntos se tratan como conjuntos ordinarios.

Los conjuntos pueden implementarse tanto en estructuras básicas (arreglos y vectores) como algunas más complejas (conjuntos hash, diccionarios). Independientemente de la imple-

[1]En el ejemplo presentado en la Sección 3.1.5 puede observarse cómo I_3^3 recibe al objeto $\{v_2v_5\}$ tanto de O_2^2 como de O_5^2.

mentación de los conjuntos, para este problema es necesario que las estructuras que emulen tanto los multiconjuntos como los objetos del sistema cuenten con las siguientes operaciones:

- **Agregar elemento.** Esta operación añade un nuevo elemento al conjunto. Es requerida en las reglas $R_{i\{1,3\}}$ para (1) agregar el elemento v_i al objeto z y (2) agregar el objeto z' a las entradas vecinas y al entorno mediante las instrucciones *go* y *out*, respectivamente.

- **Borrar elemento.** Esta operación elimina un elemento específico de un conjunto, y se utiliza en la regla R_{i2} para borrar un objeto z que contenga vértices no adyacentes de alguna entrada I_i.

- **Verificar la existencia de un elemento en el conjunto.** Esta operación devuelve un valor *verdadero* si un elemento pertenece al conjunto, y *falso* en caso contrario. Esta función ayuda a evitar introducir elementos repetidos dentro de un conjunto.

- **Limpiar contenido.** Esta operación vacía el contenido de un conjunto y se requiere al terminar el procesamiento en el instante t.

En el caso de los objetos del sistema, se necesita la función de **intersección** en la validación de no adyacencia. Para esto, se utilizan el conjunto de aristas E de G. Dado un objeto del sistema z, en vértice v_i y el conjunto formado por los vértices no adyacentes a v_i, $NoAdj_i = \{u \mid (v_i, u) \notin E\}$, si $z \cap NoAdj_i \neq \emptyset$, entonces existirían vértices no adyacentes en el objeto $z \cup \{v_i\}$, por lo que no debe crearse.

3.3.2. Replicación

A continuación se describe de manera algorítmica la función de comunicación entre células. Recuerde que en la operación de replicación en un instante t, la célula σ_i envía una copia O_i^t a sus vecinos. La función recibe el nombre de $\textsc{go}(i)$ por el indicador de dirección usado en las reglas de transición, y se presenta en el Algoritmo 1.

El Algoritmo 1 recibe como entrada el índice i de la célula que está invocando la replicación. Primero se itera sobre los índices de las células vecinas a σ_i (línea 1). Se verifica de manera paralela que cada objeto z en O_i (línea 2) no exista en el conjunto I_j (línea 3); si es

Algoritmo 1: GO(i)

entrada : i, el índice de la célula del sistema que llevará a cabo la replicación.

1
2 **for each** $j \mid (v_i, v_j) \in E$ **do in parallel**
3 **for each** $z \in O_i$ **do in parallel**
4 **if** I_j.CONTIENE(z) = *falso* **then**
5 I_j.AGREGAR(z)
6 **end if**
7 **end for**
8 **end for**

el caso, entonces se agrega una copia del objeto z a I_j (línea 4). Esta comprobación se realiza para evitar crear múltiples entradas en I_i del mismo objeto pero procedente de distintas células.

3.3.3. Reporte de soluciones

En el Algoritmo 2 se presenta la función para el reporte de soluciones, OUT(i, t), que emula la salida de objetos hacia el entorno:

Algoritmo 2: OUT(i,t)

entrada : i, el índice de la célula del sistema que expulsará objetos al entorno;
t, el instante actual.

1 **for each** $z \in O_i$ **do in parallel**
2 **if** out_t.CONTIENE(z) = *falso* **then**
3 out_t.AGREGAR(z)
4 **end if**
5 **end for**
6 $O_i \leftarrow \emptyset$

De manera similar al Algoritmo 1, el Algoritmo 2 itera sobre cada uno de los objetos z de O_i, y en paralelo (línea 1) verifica si no existe la entrada para z en el conjunto out_t (línea 2). En caso que no esté presente, se agrega y se almacena junto a las demás soluciones para el instante t (línea 3). Una vez que se procesan todos los objetos z, se limpia el conjunto O_i (línea 6).

3.3.4. Ejecución del sistema

La ejecución del sistema se divide en dos partes: la inicialización (Algoritmo 3) y el ciclo principal del sistema (Algoritmo 4). El Algoritmo 3 mapea el comportamiento del sistema en el instante $t = 1$, donde se aplica la regla R_{i1}, y por medio de ésta, la célula σ_i crea, replica y reporta el objeto inicial $\{v_i\}$.

Algoritmo 3: INICIALIZACIÓN(G)

entrada : $G = (V, E)$ un grafo no dirigido.
1 $n \leftarrow |V|$
2 Crear los conjuntos $I_1, \ldots, I_n$, $O_1, \ldots, O_n$, $out_1, \ldots, out_n$.
3 **for** $i \leftarrow 1$ **to** n **do in parallel**
4 CREAR $z = \{v_i\}$
5 O_i.AGREGAR(z)
6 GO(i)
7 OUT($i, 1$)
8 **end for**

La entrada del Algoritmo 3 es un grafo G no dirigido compuesto por el conjunto de vértices V y de aristas E. En la línea 1 se almacena el número de vértices en la variable n. En la línea 2 se crean los $3n$ conjuntos del sistema. De manera paralela, para cada una de las células (línea 3) se crea un conjunto z (línea 4), el cual funge como objeto inicial en la célula. El conjunto z es unitario dado que sólo contiene al vértice v_i. Una vez creado z, se añade a O_i, se replica y se reporta la solución (líneas 5-7). Ahora se presenta el sistema completo en el Algoritmo 4.

La entrada del algoritmo es un grafo G no dirigido compuesto por el conjunto de vértices V y de aristas E. En la línea 1 se realiza la inicialización del sistema como se muestra en el Algoritmo 3. En la línea 2 se itera sobre el número máximo de ciclos siguientes, $n - 1$. El número de ciclos está relacionado con la propiedad del sistema de que en el instante t arroja cliques de tamaño t, por lo que en el peor de los casos son necesarias n iteraciones. En la línea 3 se inicializa la variable global booleana *replicaciones*. Esta variable tiene la función de indicar si no se realizó ninguna operación de replicación en el instante actual. En caso que tenga un valor *falso*, entonces representa la condición de paro del sistema. Posteriormente

Algoritmo 4: CLIQUESISTEMADETEJIDO(G)

```
entrada : G = (V, E) un grafo no dirigido.
salida  : entorno, el conjunto de cliques de G.
INICIALIZAR(G)
for t ← 2 to n do
    replicaciones ← falso
    for i ← 1 to n do in parallel
        O_i ← I_i
        for each z ∈ O_i do in parallel
            if z.INTERSECTA(NoAdj_i) then
                O_i.BORRAR(z)
            else
                z.AGREGAR(v_i)
            end if
        end for
        if O_i ≠ ∅ then
            replicaciones ← verdadero
            GO(i)
            OUT(i, t)
        end if
    end for
    if replicaciones = falso then
        TERMINAR
    end if
end for
```

se ejecuta el procesamiento en las células de forma paralela (líneas 4-18). Para comenzar el procesamiento de los objetos de σ_i, se mueve el contenido de I_i hacia O_i con la finalidad de dejar libre I_i para recibir los objetos procedentes de la replicación (línea 5).

De manera paralela se revisa cada objeto z en O_i (línea 6). Si z se intersecta con el conjunto de no adyacencia del vértice v_i, $NoAdj_i$, entonces z se elimina de O_i (líneas 7 y 8), tal como se observa en el comportamiento de la regla R_{i2}. Si z no presenta la intersección, entonces sólo contiene vértices adyacentes a v_i, por lo que se agrega el vértice v_i al objeto z, formando un nuevo clique (líneas 9 y 10). Una vez que se procesan todos los objetos, si al menos uno no se eliminó en la línea 8, entonces se asigna a la variable *replicaciones* el valor *verdadero*. Posteriormente, se ejecutan la replicación y el reporte de soluciones (líneas 13-16). Una vez que las células terminan de procesar los objetos, se verifica si se realizó alguna operación de replicación en todo el sistema. En caso negativo, el algoritmo finaliza su ejecución (líneas 19-20).

Realizabilidad

La implementación *in silico* de estos algoritmos tendría algunos detalles importantes por trabajar, siendo el más importante el espacio. Si bien el procesamiento se muestra distribuido, en un paso t se pueden llegar a tener hasta $\binom{n}{t}$ objetos distintos en todo el sistema (sin contar las multiplicidades). Por ejemplo, suponga un grafo completo de 50 vértices. Para $t = 25$, en el sistema habría $\binom{50}{25} = 1.264 \times 10^14$ objetos distintos. Suponiendo que se cuente con una estructura de datos eficiente que nos permita almacenar cada objeto en tan sólo 1 byte de información, para calcular el estado del sistema en $t = 25$ son necesarios 126.4 TB de almacenamiento, lo cual lo hace poco práctico en la vida real. En comparación, el algoritmo de Bourgeois *et al.* (2012) utiliza un espacio de tamaño polinomial y toma un tiempo de $O(1.2114^n)$ en el caso general. La forma de resolver el problema por el sistema propuesto se adapta mejor a otro modelo de cómputo no convencional, el cómputo molecular, lo cual se trata en el Capítulo 4.

Capítulo 4. Implementación del sistema P de tejido mediante cómputo molecular

En este capítulo se propone de manera teórica un algoritmo biomolecular para resolver el problema MAX-CLIQUE basado en el sistema P de tejido del capítulo anterior. Los algoritmos de cómputo biomolecular son inherentemente paralelos, por lo que constituyen un enfoque interesante para la implementación de los sistemas de membrana en general. Primero se muestra la codificación utilizada para representar los objetos del sistema mediante cadenas de ADN. Después se analizan las unidades de memoria y operaciones del modelo de cómputo molecular. Posteriormente se presenta el algoritmo propuesto, el cual difiere tanto en el tipo de operaciones utilizadas como en la generación de las soluciones con los presentados por Ouyang *et al.* (1997), Head *et al.* (1999), McCaskill (2001), García-Arnau *et al.* (2007b) y Martínez-Pérez y Zimmermann (2009).

4.1. Codificación

En este algoritmo se utiliza una codificación basada en las presentadas en la Sección 2.6.1. De manera general, la codificación tiene las siguientes características:

1. Cada vértice del grafo tiene una secuencia o dominio de 20 nt asociada.
2. Los cliques se representan como una cadena sencilla formada por la concatenación de los dominios correspondientes a los vértices que lo componen.
3. Existen hebras auxiliares para la unión de los vértices.

Para la ejecución del algoritmo se requieren $n+1$ dominios de ADN y dos clases de hebras construidas previas a la computación. Los dominios son secuencias de 20 nt que representan a cada uno de los n vértices, y un dominio especial denominado p, el cual funge como iniciador en la operación de agregar vértices a las cadenas mediante el ensamblado paralelo por superposición (POA por sus siglas en inglés). Utilizando los dominios se construyen los conjuntos de hebras iniciales S y las hebras auxiliares de concatenación C.

El conjunto de hebras iniciales

$$S = \{5' - pv_i - 3' \mid 1 \leq i \leq n\} \tag{41}$$

es la base para la construcción de los cliques. Se conforman por la concatenación de un dominio p y un dominio v_i (Figura 8A), por lo que su longitud es de 40 nt. En total se construyen n hebras iniciales, una por cada dominio v_i. Por otro lado, las hebras auxiliares

$$C_i = \{3' - \overline{s(v_j)v_i} - 5' \mid (v_i, v_j) \in E\},\ 1 \leq i \leq n, \tag{42}$$

se construyen por medio de la concatenación del complemento de un dominio v_i y el complemento del sufijo de un dominio v_j, $s(v_j)$, si y sólo si existe una arista (v_i, v_j) (Figura 8B). Existen $2|E|$ hebras auxiliares diferentes y cada una tiene 30 nt de longitud. Finalmente, en total es necesario sintetizar $n + 2|E|$ hebras para la ejecución del algoritmo.

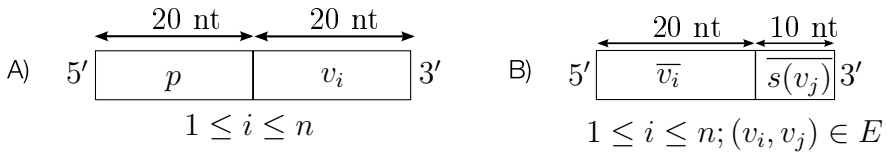

Figura 8: Codificación propuesta para el algoritmo molecular: A) Hebras iniciales pv_i. B) Hebras auxiliares de concatenación C_i.

4.2. Memoria

Las unidades fundamentales de memoria en cómputo molecular son los tubos. El nombre hace alusión a los tubos de ensayo, dado que contienen los multiconjuntos de hebras de ADN. Los tubos se referencían mediante una etiqueta. En este caso, la etiqueta del tubo estará relacionada con el multiconjunto del sistema a representar: T_{I_i} para I_i, T_{O_i} para O_i y finalmente T_{out_t} para out_t. En total se requieren $3n$ tubos.

4.3. Operaciones del modelo

A continuación se muestran las operaciones de cómputo molecular requeridas para la ejecución de los algoritmos de la Sección 3.3. Estas operaciones son clásicas en los algoritmos de cómputo biomolecular (Ignatova *et al.*, 2008). Las operaciones se pueden clasificar en dos categorías: mecánicas y biológicas. Las operaciones mecánicas generalmente involucran el movimiento de hebras de ADN de un tubo a otro, por lo que se pueden automatizar con dispositivos microfluídicos o robots. El tiempo de ejecución y la tasa de error no es

significativo en comparación a las operaciones biológicas, ya que depende de la reacción que se quiera llevar a cabo.

1. **Llenar.** La operación LLENAR$(T, \{s_1, \ldots, s_m\})$ deposita el conjunto de hebras $S = \{s_1, \ldots, s_m\}$ en el tubo T. Si el tubo T no está vacío, entonces se combina el contenido de T con S. Esta operación de tipo mecánico se utiliza para llenar los tubos con las hebras iniciales.

2. **Extraer.** Mediante la función EXTRAER$(T, \{s_1, \ldots, s_m\})$ se eliminan las hebras en T que contengan al menos una ocurrencia de algún segmento $\{s_1, \ldots, s_m\}$. Esta operación se usa para emular la operación de descartar de O_i los objetos que no forman cliques con el vértice v_i. En la práctica, la operación EXTRAER puede realizarse mediante membranas de captura y perlas magnéticas utilizando como sonda de captura los segmentos $\{\overline{s_1}, \ldots, \overline{s_m}\}$. Si bien esta operación involucra una reacción de hibridación del contenido de T con el dispositivo de captura, por su facilidad de automatización se propone clasificarla como mecánica.

3. **Concatenar.** La operación CONCATENAR(T, s) añade la hebra s al extremo $3'$ de las hebras del tubo T. Para el sistema de tejido propuesto, esta operación tiene la función de crear un nuevo objeto mediante la adición de un símbolo a un objeto anterior. Actualmente existen dos procedimientos de laboratorio para realizar esta operación: el uso de la enzima ligasa (ligación) y POA (*Parallel Overlap Assembly*). En este trabajo se propone el uso de POA debido a que ha mostrado ser más eficiente experimentalmente y requerir menos tiempo para completar la reacción (Ibrahim *et al.*, 2006). El primer paso para realizar el POA es mezclar las hebras que se quieren extender junto con los reactivos necesarios para PCR. Posteriormente las hebras se desnaturalizan y se dejan alinear. Una vez que las hebras se alinean, quedan extremos pegajosos expuestos que se extienden (elongan) mediante la enzima ADN polimerasa en dirección $5' \rightarrow 3'$. Este procedimiento se puede realizar en un dispositivo para PCR estándar, debido a que utiliza los mismos ciclos termales. La diferencia entre POA y PCR es que POA incrementa la longitud de las secuencias por medio de la elongación, mientras que PCR duplica la cantidad de secuencias durante cada iteración.

   Al realizar la operación de concatenar, la entrada del POA consiste en un conjunto de

hebras que representan a los objetos de entrada de alguna célula i, y además las hebras auxiliares de concatenación C_i. Al alinearse, las hebras auxiliares dejan el extremo pegajoso $\overline{v_i}$, por lo que al extender la secuencia se añade el vértice v_i en el sentido $5' - 3'$. El cebador para la polimerasa es la secuencia $\overline{P}$, por lo que la cadena se elonga en sentido delantero y sólo se concatena la secuencia que representa al objeto (Figura 9). Una vez realizada la operación, las cadenas que representan cliques y las hebras auxiliares utilizadas pueden separarse por medio de desnaturalización y electroforesis (las hebras C_i tienen una longitud constante de 30 nt, mientras que las hebras que representan a los objetos tienen una longitud $>= 40$ nt).

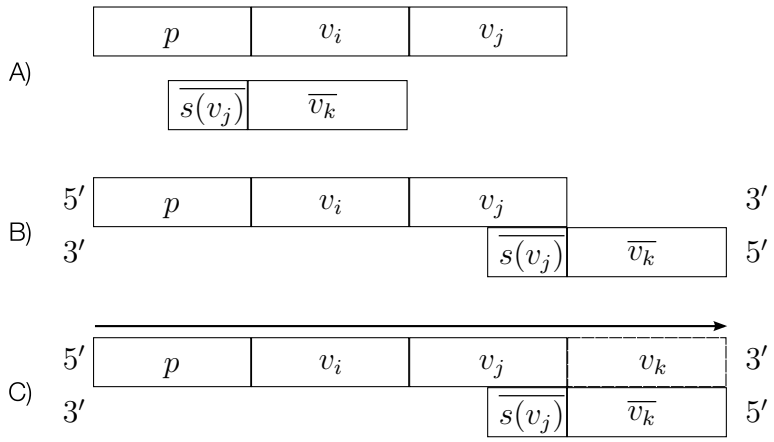

Figura 9: Concatenación del segmento v_k a pv_iv_j mediante POA: A) Las hebras sencillas se mezclan. B) Alineamiento. C) La elongación se realiza mediante la enzima ADN polimerasa únicamente en dirección $5' \to 3'$.

4. **Combinar.** La operación de tipo mecánico COMBINAR(T_1, T_2) mezcla el contenido de T_1 en T_2, dejando vacío el tubo T_1.

5. **Distribuir.** La operación DISTRIBUIR$(T, \frac{1}{\alpha}, \{T_1, \ldots, T_m\})$ es una generalización de combinar. Se mezcla $\frac{1}{\alpha}$ del contenido del tubo T a cada uno de los tubos $T_1, \ldots, T_m$. Para ejecutar esta operación, se debe cumplir que $m \leq \alpha$. Esta operación es de tipo mecánico.

6. **Amplificar.** La amplificación es una de las operaciones fundamentales en cómputo molecular. La operación AMPLIFICAR(T, m) realiza m ciclos de PCR sobre el del tubo T, generando 2^m copias del contenido del tubo. Esta operación es de tipo biológico.

7. **Copiar.** Mediante la función COPIAR$(T, \{T_1, \ldots, T_m\})$ se combina una copia del contenido del tubo T en cada tubo $\{T_1, \ldots, T_m\}$. Los tubos que reciben las copias no

están necesariamente vacíos. La operación COPIAR implícitamente realiza las operaciones AMPLIFICAR($T, \lceil log(m+1) \rceil$) y DISTRIBUIR($T, \frac{1}{m}, \{T_1, \ldots, T_m\}$). Esta operación es utilizada para emular la replicación entre células.

8. **Intercambiar etiquetas.** La operación INTERCAMBIAR_ETIQUETA(T_1, T_2) intercambia los nombres de T_1 y T_2. Esta operación se utiliza como auxiliar para emular el procesamiento paralelo del sistema, utilizando tubos como resultados temporales y transformarlos en los tubos principales.

9. **Detectar.** La operación DETECTAR($\{T_1, \ldots, T_m\}$) arroja *verdadero* si existe al menos una hebra de ADN en el conjunto de tubos $T_1, \ldots, T_m$, y *falso* en caso contrario.

4.4. Algoritmo de cómputo molecular

En esta sección se presenta el algoritmo de cómputo molecular. El primer paso en la computación es la inicialización del contenido de los tubos, la cual se muestra en el Algoritmo 5. Este algoritmo es análogo al Algoritmo 3: cada tubo almacena hebras iniciales de sus vecinos (líneas 2 - 4) y posteriormente reporta como resultado los cliques triviales de tamaño 1 (línea 5).

Algoritmo 5: INICIALIZAR(G)

entrada : $G = (V, E)$ un grafo no dirigido.
1 **for** $i \leftarrow 1$ **to** n **do**
2 **for each** $j|(v_i, v_j) \in E$ **do**
3 LLENAR($T_{O_j}, \{pv_i\}$)
4 **end for**
5 LLENAR($T_{out_1}, \{pv_i\}$)
6 **end for**

Una vez definidas la inicialización de los tubos, se presenta el Algoritmo 6, el cual resuelve el problema MAX-CLIQUE.

Se define la variable n como el número de vértices del grafo (línea 1). Tan pronto se inicializan los tubos (línea 2) se comienza el procesamiento para generar los cliques de los

Algoritmo 6: CLIQUEADN(G)

entrada : $G = (V, E)$ un grafo no dirigido.
salida : $T_{out_1} \dots T_{out_n}$, los tubos que contienen los cliques del grafo G.

1 $n \leftarrow |V|$
2 INICIALIZAR(G)
3 **for** $t \leftarrow 2$ **to** n **do**
4 **for** $i \leftarrow 1$ **to** n **do**
5 EXTRAER($T_{O_i}, \{v_j \mid (v_i, v_j) \notin E\}$)
6 CONCATENAR(T_{O_i}, C_i)
7 COPIAR($T_{O_i}, \{T_{I_j} \mid (v_i, v_j) \in E\}$)
8 MOVER(T_{O_i}, T_{out_t})
9 **end for**
10 **if** DETECTAR($\{T_i \mid 1 \leq i \leq n\}$) **then**
11 TERMINAR
12 **end if**
13 **for** $i \leftarrow 1$ **to** n **do**
14 INTERCAMBIAR_ETIQUETA(T_{O_i}, T_{I_i})
15 **end for**
16 **end for**

$n - 1$ tamaños restantes (líneas 3 - 16). La variable t almacena el paso actual del algoritmo, mientras que la variable i indica el vértice que se está procesando (línea 4).

Cuando se inicia el procesamiento para el vértice v_i, se eliminan las hebras que contengan vértices no adyacentes a él (línea 5). Las hebras que no se extrajeron son las que pueden formar cliques válidos si se unen a v_i; por esta razón se aplica la concatenación mediante POA para crear los nuevos cliques utilizando el conjunto de hebras auxiliares C_i, cuyos elementos dejan a $\overline{v_i}$ como extremo pegajoso (línea 6).

Una vez que se extienden las hebras que representan a los nuevos cliques se emula la replicación del sistema mediante la copia del contenido del tubo a los vecinos del vértice (líneas 7) y se almacenan como solución en el tubo T_{out_t} (línea 8). Las líneas 7 y 8 pueden emular la función REPLICAR, la cual se muestra en el Algoritmo 7.

Algoritmo 7: REPLICAR(T_{O_i}).

entrada : T_{O_i}, el tubo origen.
salida : El contenido de T_{O_i} en cada uno de los tubos $\{T_{I_j} | (v_i, v_j) \in E\}$.

1 AMPLIFICAR($T_{O_i}, \lceil \log_2(deg_G(v_i) + 1) \rceil$)
2 DISTRIBUIR($T_{O_i}, \frac{1}{deg_G(v_i)+1}, \{T_{I_j} | (v_i, v_j) \in E\}$)
3 MOVER(T_{O_i}, T_{out_t})

Posteriormente, se revisa la condición de paro de que al menos una célula haya arrojado un objeto, para lo que se utiliza la operación DETECTAR (líneas 10 - 12). Si la condición es *verdadero*, entonces se termina la ejecución del algoritmo. Después de procesar el contenido de todos los tubos en el instante t se intercambian las etiquetas de los tubos, esto con el objetivo de que T_{O_i} contenga todas las hebras recibidas por replicación que se procesarán y T_{I_i} quede vacío para recibir las hebras de los vecinos durante el instante $t+1$ (líneas 13-15).

Interpretación de resultados

Observe que si la operación DETECTAR de la línea 11 reporta *falso* en el instante $k+1$, la solución se encuentra en el tubo T_{out_k}. El tamaño del clique máximo es de k vértices. Por otro lado, para conocer la conformación del clique, es necesario secuenciar las hebras del tubo T_{out_k}.

4.4.1. Versión paralela

El algoritmo anterior supone que tanto las operaciones mecánicas como biológicas se realizan secuencialmente. Sin embargo, los dispositivos actuales de PCR permiten realizar pruebas en paralelo. Por ejemplo, Arktik Thermal Cycler es un dispositivo con espacio para 384 tubos (ThermoScientific). Por lo tanto, mientras el número de vértices sea menor o igual a la capacidad del dispositivo PCR, se pueden ejecutar las amplificaciones y concatenaciones en paralelo. En el caso de las amplificaciones, el número máximo de ciclos está dado por el grado máximo del grafo, $deg(G)$. En el caso de la concatenación, sólo se necesita una secuencia de ciclos termales del PCR, debido a que se concatena una sola hebra.

El paradigma trabajo-tiempo

El Paradigma Trabajo-Tiempo es un marco de trabajo para la descripción de algoritmos paralelos (Jájá, 1992). Se compone de dos niveles: el superior y el inferior. El nivel superior ayuda a describir el algoritmo como una secuencia de unidades de tiempo, donde cada unidad de tiempo puede incluir cualquier número de operaciones concurrentes. La declaración

for $l \leq i \leq u$ **do in parallel** *declaraciones*

indica que las *declaraciones* para los valores de i entre l y u se ejecutan concurrentemente. Por otro lado, el nivel inferior adapta el algoritmo para que pueda correr sobre cualquier

número de procesadores p, es decir, describe con mayor detalle la asignación de trabajo.

Tomando la idea de la paralelización de la concatenación y la amplificación, se presenta el Algoritmo 8 utilizando el nivel superior del paradigma trabajo-tiempo.

Algoritmo 8: CLIQUEADNPARALELO(G)

entrada : $G = (V, E)$ un grafo no dirigido.
salida : $T_{out_1} \ldots T_{out_n}$, los tubos que contienen los cliques del grafo G.

1 $n \leftarrow |V|$
2 INICIALIZAR_PAR(G)
3 **for** $t \leftarrow 2$ **to** n **do**
4 **for** $i \leftarrow 1$ **to** n **do in parallel**
5 EXTRAER($T_{O_i}, \{v_j | (v_i, v_j) \notin E\}$)
6 **end for**
7 **if** DETECTAR($\{T_{O_i} \mid 1 \leq i \leq n\}$) = *falso* **then**
8 TERMINAR
9 **end if**
10 **for** $i \leftarrow 1$ **to** n **do in parallel**
11 CONCATENAR(T_{O_i}, C_i)
12 **end for**
13 **for** $i \leftarrow 1$ **to** n **do in parallel**
14 AMPLIFICAR($T_{O_i}, \text{LOG}_2(deg(G) + 1)$)
15 **end for**
16 **for** $i \leftarrow 1$ **to** n **do**
17 DISTRIBUIR($T_{O_i}, \frac{1}{deg_G(v_i)+1}, \{T_{I_j} | (v_i, v_j) \in E\} \cup \{T_{out_t}\}$)
18 **end for**
19 **for** $i \leftarrow 1$ **to** n **do**
20 INTERCAMBIAR_ETIQUETA(T_{O_i}, T_{I_i})
21 **end for**
22 **end for**

En este algoritmo se introduce la rutina INICIALIZAR_PAR(G), la cual tiene la misma función de INICIALIZAR(G); sin embargo, los ciclos **for** se ejecutan en paralelo (líneas 1 - 2). Posteriormente, se realiza la extracción de hebras en todos los tubos T_{O_i} en paralelo (líneas 4 - 6), y se verifica si existen hebras en los tubos (líneas 7 - 9). Si no hay ninguna, se termina el algoritmo. En otro caso, se realiza la concatenación en paralelo utilizando las hebras auxiliares (líneas 10 - 12). Una vez creados los nuevos vértices, se realizan en paralelo las amplificaciones necesarias (líneas 10 - 12). La distribución de las hebras para simular la replicación se lleva a cabo de manera paralela (líneas 16 - 18), para finalmente intercambiar

las etiquetas de los tubos (líneas 19 - 21). A continuación se presenta el análisis de tiempo y operaciones de este algoritmo.

Análisis de operaciones

Dado que el Algoritmo 8 está descrito en el nivel superior del paradigma trabajo-tiempo, se hace la distinción entre las unidades de tiempo y las unidades de operaciones (el trabajo realizado durante la unidad de tiempo). Se definen *utb* y *utm* como las unidades de tiempo biológicas y mecánicas, respectivamente, y *ob* y *om* como las unidades de operaciones biológicas y mecánicas, respectivamente. A pesar de que la duración de una unidad de tiempo depende de la operación específica a realizar, se supondrá que tienen un tiempo uniforme.

Antes de comenzar el análisis, se establece que el valor máximo de la variable de iteración t es $t_h = \omega(G)+1$ debido a que en ese instante el sistema ya no produce objetos y la operación DETECTAR regresa un valor *falso*. Esto se demuestra en la Sección 3.2. En la inicialización se recorren todas las aristas del grafo al llenar los tubos T_O en un solo ciclo paralelo, por lo tanto se realizan $O(1)$ *utm* y $2|E|$ *om*. Después cada vértice llena con su hebra inicial el tubo T_{out_1}, por lo que se realizan $O(1)$ *utm* y $|V|$ *om*.

La instrucción EXTRAER($T_{O_i}, \{v_j|(v_i, v_j) \notin E\}$) es paralelizable si las sondas de captura se preparan previo a la computación de manera que se concentren en un sólo conjunto de filtros, por lo que por cada iteración del ciclo de la línea 4-6 se realizan $|V|$ *om* en tan solo $O(1)$ *utm*. La concatenación se realiza en los $|V|$ tubos T_O en 1 paso paralelo, por lo que toma $|V|$ *ob* y se ejecuta en $O(1)$ *utb*.

Dado que se realiza en paralelo, la amplificación toma $O(1)$ *utb*; sin embargo, en total realizan $|V|(\log_2(deg(G)+1))$ *ob*. La distribución en el ciclo de las líneas 16-17 toma $(2|E|)$ *om* y $(2|E|)$ *utm* dado que se realiza en secuencial. En total, se realizan $t_h(2|E|)$ *om* y $t_h(2|E|)$ *utm*. Finalmente, cambiar la etiqueta toma $t_h(|V|)$ *uom* y $t_h(|V||)$ *utm*.

Por lo tanto, en tiempo y operaciones biológicas, el algoritmo tiene un desempeño de $\omega(G)+1$ *utb* y $(\omega(G)+1)(|V|)(\log_2(deg(G)+1))$ *ob* respectivamente, mientras que en tiempo y operaciones mecánicas tiene un desempeño de $(\omega(G)+1)(2|E|)$ *om* y $(\omega(G)+1)(2|E|)$ *utm*. Debido a que las operaciones biológicas toman una cantidad mucho mayor de tiempo que las mecánicas, el tiempo del algoritmo es de $\omega(G)+1$ y realiza $(\omega(G)+1)(n)(\log_2(deg(G)+1)) = O(n^2\log_2(n))$ operaciones.

5.1. Sistemas de membranas para la resolución del problema MAX-CLIQUE

En la Tabla 2 se muestran las características más importantes entre el sistema de tejido propuesto y el presentado por García-Arnau *et al.* (2007b).

Tabla 2: Comparación de las características del sistema de tejido propuesto y el sistema P de García-Arnau *et al.* (2007b).

Característica	**García-Arnau *et al.* (2007b)**	**Sistema propuesto**
Membranas/Células	$n+1$	$\boldsymbol{n}$
Tiempo de finalización	$2n+1-\omega(G)$	$\boldsymbol{\omega(G)+2}$
Grado de replicación	2	$deg(G)$

El sistema propuesto utiliza una célula menos que el sistema de García-Arnau debido a que los resultados se reportan desde la célula que lo genera. Como se menciona en la Sección 2.6.2, el sistema de García-Arnau reporta los cliques de tamaño n en el tiempo $t = n+1$, los de tamaño $n-1$ en $t = n+2$, y así sucesivamente. Generalizando, este sistema arroja cliques de tamaño k en el tiempo $t = (n+1)+(n-k) = 2n+1-k$, por lo que el clique con tamaño máximo $\omega(G)$ se reporta en el tiempo $2n+1-\omega(G)$. Por otra parte, en la Sección 3.2 se demuestra que el sistema propuesto arroja los cliques de tamaño $\omega(G)$ en el tiempo $\omega(G)+1$ y se detiene en el tiempo $\omega(G)+2$.

Para comparar qué sistema necesita menos pasos para obtener la respuesta al problema, se resuelve la siguiente desigualdad:

$$\begin{aligned} \omega(G)+2 &\leq 2n+1-\omega(G) \\ 2\omega(G)+2 &\leq 2n+1 \\ 2\omega(G) &\leq 2n-1 \\ \omega(G) &\leq n-1/2 \end{aligned} \tag{43}$$

y redondeando al entero más cercano, se tiene que el tiempo de finalización del sistema propuesto es menor cuando

$$\omega(G) \leq n - 1. \tag{44}$$

Analizando el caso extremo $\omega(G) = n$, el sistema propuesto toma $n + 2$ pasos, mientras que el de García-Arnau toma $n+1$ pasos. Sin embargo, este caso se puede resolver en tiempo constante si se verifica que G es un grafo completo, es decir, $|E| = \frac{n(n-1)}{2}$. La comparación entre los tiempos de ejecución de los dos sistemas variando el tamaño del clique máximo de un grafo G de 6 vértices se presenta en la Figura 10.

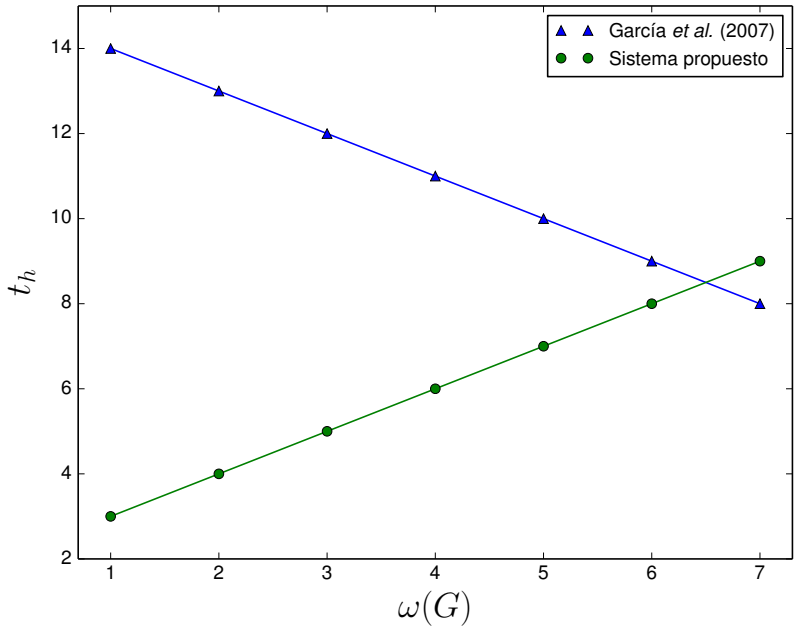

Figura 10: Comparación del tiempo de finalización t_h entre el sistema de tejido propuesto y el sistema P de García-Arnau *et al.* (2007b) para un grafo de 6 vértices.

El grado de replicación del sistema es el número máximo de cadenas que se pueden generar después de la aplicación de una regla. En el caso del sistema de García-Arnau el grado de replicación es 2 (debido a la regla R'_i), mientras que en el sistema propuesto es $deg(G)$, es decir, el grado máximo del grafo G.

5.2. Algoritmos moleculares para la resolución del problema MAX-CLIQUE

A continuación se presenta una comparación entre los algoritmos moleculares para la resolución del problema MAX-CLIQUE presentados por Ouyang *et al.* (1997), Head *et al.* (1999), Noort *et al.* (2002), García-Arnau *et al.* (2007b), Martínez-Pérez y Zimmermann (2009) y el propuesto en este trabajo. En la Tabla 3 se muestra la comparación de las características teóricas de los algoritmos, mientras que en la Tabla 4 se muestran algunos de los valores de la Tabla 3 obtenidos durante la ejecución para la resolución del problema en el grafo G (Figura 11).

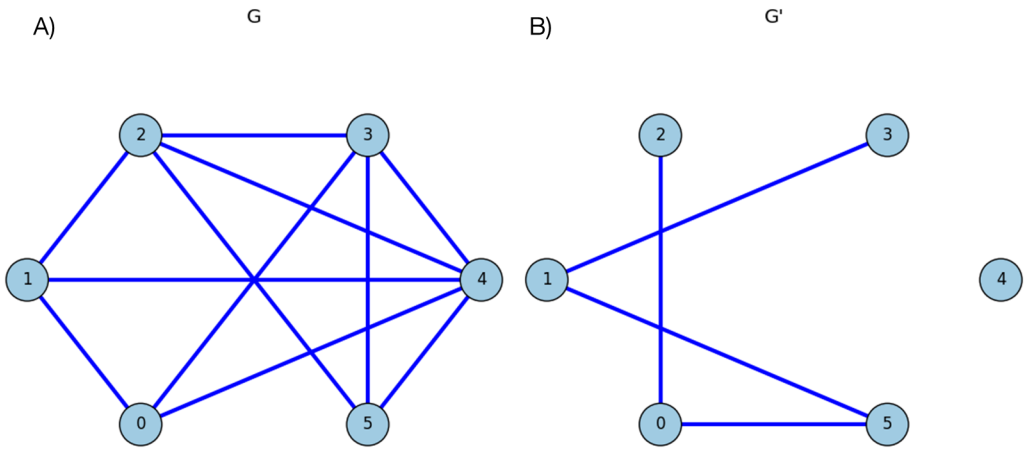

Figura 11: Caso utilizado para la comparación de los algoritmos moleculares: A) Grafo G con $n = 6$, $|E| = 11$ y clique máximo $\{v_2, v_3, v_4, v_5\}$. B) Grafo complemento G' con $|\overline{E}| = 4$.

Tabla 3: Características de los algoritmos moleculares comparados.

Característica	**Ouyang *et al.* (1997)**	**Head *et al.* (1999)**	**Noort *et al.* (2002)**	**García-Arnau *et al.* (2007b)**	**Martínez-Pérez y Zimmermann (2009)**	**Algoritmo Propuesto**
Cadenas iniciales	2^n	1	2^n	$\frac{n(n+1)}{2}$	$2^n - 1$	$n + 2\|E\|$
Iteraciones	$\|\overline{E}\|$	$\|\overline{E}\|$	$\|E\|$	n	$\frac{n(n-1)(n-\omega(G)+1)}{2}$	$\omega(G) + 1$
Tubos	$O(1)$	$O(1)$	$O(n^2)$	$O(n)$	$O(n)$	$O(n)$
Enzimas de restricción	n	n	0	0	0	0
Implementado	Sí	No	Sí	No	No	No
Categorías	Filtrado *marcar y destruir*	Cómputo acuoso, *splicing*	Filtrado sin enzimas, microfluidos	Filtrado sin enzimas, constructivo, microfluidos	Filtrado sin enzimas, etiquetas	Filtrado *marcar y destruir*, constructivo

Tabla 4: Resultados de la ejecución de los algoritmos moleculares sobre el grafo de la Figura 11A.

Característica	**Ouyang *et al.* (1997)**	**Head *et al.* (1999)**	**Noort *et al.* (2002)**	**García-Arnau *et al.* (2007b)**	**Martínez-Pérez y Zimmermann (2009)**	**Algoritmo Propuesto**
Cadenas iniciales	64	1	64	21	64	28
Iteraciones	4	4	11	6	45	5
Tubos	1	1	36	7	12	18
Enzimas de restricción	6	6	0	0	0	0

En la Tabla 3 las columnas corresponden a cada uno de los algoritmos y las filas a las características de interés. La primer fila, *cadenas iniciales*, indica la cantidad de secuencias distintas de ADN que es necesario sintetizar previo a la ejecución del algoritmo. Observe que tanto los algoritmos de Ouyang, Noort y Martínez requieren una biblioteca combinatoria que represente a todos los cliques posibles del grafo. En el caso de Martínez se necesitan $\binom{n}{k}$ hebras para resolver el problema k-clique. Sin embargo, para resolver el problema MAX-CLIQUE es necesario iterar sobre los valores de $k = n$ hasta $k = 1$, por lo que en total son necesarias $\sum_{k=1}^{n} \binom{n}{k} = 2^n - 1$ cadenas iniciales. En el caso de Head, es necesario sólo un plásmido en una concentración suficiente para derivar en el espacio de búsqueda exponencial mediante la aplicación de las enzimas de restricción. El algoritmo de García requiere n cadenas que se corresponden con cada uno de los vértices del grafo y $\sum_{k=1}^{n-1} k$ hebras auxiliares.

La variable *iteraciones* indica el total de ciclos necesarios para que el algoritmo llegue a la solución. En el algoritmo de Ouyang y en el de Head el número de iteraciones está directamente relacionado con la cantidad de aristas en G' ($|\overline{E}|$ pasos), mientras que el de Noort con el número de aristas en G ($|E|$ pasos). El algoritmo de García está inspirado en el sistema de membranas que se ejecuta en $2n+1-\omega(G)$ pasos. Sin embargo, dado que el ordenamiento se puede hacer por electroforesis, sólo son necesarios n pasos para generar todos los cliques del grafo. Finalmente, tanto en el algoritmo de Martínez como en el algoritmo propuesto las iteraciones se relacionan con el tamaño del clique máximo del grafo ($\frac{n(n-1)(n-\omega(G)+1)}{2}$ y $\omega(G)+1$, respectivamente). El peor caso para los algoritmos de Ouyang y Head se presenta en grafos dispersos, y para Noort en grafos densos, el cual es de $O(n^2)$ para los tres casos. Para el algoritmo de Martínez el peor caso es cuando el clique máximo tiene pocos vértices y es de $O(n^3)$. Por otra parte, el algoritmo de García y el propuesto tienen un tiempo de $\Theta(n)$; sin embargo, el algoritmo propuesto tiene un tiempo igual o mejor que el de García cuando $\omega(G) \leq n-1$.

El número de *tubos* representa las unidades de memoria necesarias para ejecutar el algoritmo. Los algoritmos que necesitan un menor número de tubos son los de Ouyang y Head, debido a que modifican el mismo espacio de solución conforme progresa la ejecución del algoritmo. En el caso de Noort, los tubos corresponden N^2 cámaras en un dispositivo microfluídico programable; la mitad de las cámaras emula la matriz de adyacencia de un grafo de hasta N vértices, mientras que las cámaras restantes se usan como ordenadores de la solución.

Martínez sólo necesita cuatro tubos por iteración para guardar resultados temporales de la separación, por lo que es orden lineal. El algoritmo propuesto requiere un número fijo de $3n$ tubos, mientras que el de García sólamente requiere n tubos.

El número de enzimas de restricción necesarias para los algoritmos es un indicador de la escalabilidad de las soluciones, es decir, el tamaño de problema a resolver está restringido por el número de enzimas de restricción disponibles. Los únicos algoritmos con esa limitación son los de Ouyang y Head, los demás evaden el uso de estas enzimas mediante filtrado de ADN y concatenación de hebras. Los algoritmos de Ouyang y Noort se han llevado a la práctica con éxito, ambos con casos de 6 vértices.

Finalmente, se puede observar en los diferentes enfoques que la técnica de filtrado predomina debido a que es el componente principal de los modelos no autónomos.

En la Tabla 3 se observan los valores de la ejecución de los algoritmos para el grafo de la Figura 11. La menor cantidad de cadenas iniciales (1) la reporta Head (debido a que se utiliza un único tipo de plásmido), mientras que la mayor cantidad (64, 63) se reporta en los algoritmos de Ouyang, Noort y Martínez. El menor número de iteraciones lo tienen los algoritmos de Ouyang y Head debido al valor de $|\overline{E}|$ (4), seguido por el algoritmo propuesto (5), el de García (6) y el de Martínez (45). El menor número de tubos lo reportan Ouyang y Head (1), seguido por García (7), Martínez (12), el algoritmo propuesto (18) y el de Noort (36). Finalmente, los únicos algoritmos que requieren enzimas de restricción son los de Ouyang y Head.

Conclusiones

Sumario

En este trabajo se abordaron dos modelos de cómputo no convencional para la resolución del problema MAX-CLIQUE: los sistemas P y el cómputo molecular. Se diseñó un sistema P de tejido que resuelve el problema para un grafo G de n vértices en un tiempo $\omega(G) + 2$ utilizando n células con reglas con un grado de replicación de $deg(G)$. La corrección y paro del sistema se demostraron mediante las propiedades de los conjuntos de los objetos presentes en las configuraciones del sistema. El sistema propuesto mejora el tiempo de ejecución respecto al sistema de membranas existente en la literatura de García-Arnau *et al.* (2007b), bajo la condición de que $\omega(G) \leq n - 1$. Posteriormente, se presentó un algoritmo molecular que utiliza un modelo constructivo apoyado por la operación de filtrar y destruir. El algoritmo es competitivo respecto al tiempo ($\omega(G) + 1$ pasos) y número de hebras iniciales ($n + 2|E|$ hebras) contra los existentes en la literatura, y además cuenta con la ventaja de que no utiliza enzimas de restricción, lo cual lo hace escalable en comparación a los algoritmos de Ouyang *et al.* (1997) y Head *et al.* (1999).

Conclusiones

1. En los resultados se muestra que el sistema de tejido supera en tiempo al de García-Arnau *et al.* (2007b) cuando se cumple la condición de que $\omega(G) \leq n - 1$, es decir, el tiempo no se supera cuando $\omega(G) = n$. Para que se de este caso, es necesario que G sea un grafo completo, lo que hace que resolver el problema MAX-CLIQUE sea trivial al comprobar el número de aristas.

2. El sistema propuesto tiene un alto grado de replicación, lo que añade complejidad al sistema. Sin embargo, es el grado de replicación lo que permite dos cosas: explorar el espacio de búsqueda exponencial en un número menor de pasos y arrojar las soluciones de tamaño k en el instante k. Esta última característica no se encuentra en ninguna de las propuestas del estado del arte.

3. El hecho de que el sistema propuesto resuelva el problema MAX-CLIQUE en tiempo lineal no significa que se resuelve la incógnita de si las clases de complejidad **P** y **NP**

son iguales (Cormen *et al.*, 2011). Esto se explica al observar que existe un compromiso entre los recursos de tiempo y espacio del sistema: si bien el tiempo es lineal, los objetos generados (el espacio) crecen exponencialmente según el número de pasos que ejecute el sistema. En el caso de este problema, la complejidad reside en que un grafo tiene un número exponencial de cliques (los cuales se pueden listar de manera exacta en un tiempo $O(1.2114^n)$ utilizando el algoritmo de Bourgeois *et al.* (2012) en un espacio polinomial). Por la naturaleza del sistema propuesto, todos los cliques deben generarse en algún punto, por lo que es *exponencial en espacio*.

Recomendaciones

El sistema de tejido propuesto aprovecha la propiedad de subestructura óptima del problema MAX-CLIQUE, es decir, se utilizan las soluciones óptimas de los subproblemas para la resolución del problema principal. Creemos que la metodología utilizada, suponer un sistema de tejido correspondiente al grafo a resolver, puede ayudar a encontrar la solución a otros problemas combinatorios con la propiedad de subestructura óptima.

Al desarrollar el trabajo se observó que al resolver problemas combinatorios es importante seleccionar el modelo de sistemas de membranas que más se ajuste al problema y después explorar las opciones de búsqueda exponencial. En este trabajo se optó por los sistemas de tejido debido a que su estructura puede mapear cualquier grafo arbitrario. Sin embargo, existen otros modelos interesantes tales como los de membranas activas que también permiten hacer la búsqueda exponencial mediante reglas más sencillas de visualizar.

Trabajo futuro

Dentro del trabajo futuro, el sistema de tejido propuesto se puede mejorar en términos de eficiencia del espacio de búsqueda considerando los estados de los canales de comunicación y reglas adicionales para el manejo de los objetos.

Debido al alcance de la investigación, el algoritmo molecular sólo se presenta de manera teórica, por lo que también queda por realizar una implementación práctica. En este sentido, la recomendación es apostar hacia los dispositivos microfluídicos, dado que por su organización en cámaras y reprogramabilidad permitirían modelar de manera eficaz los pasos de comunicación del algoritmo molecular propuesto.

Bibliografía

Adleman, L. (1994). Molecular computation of solutions to combinatorial problems. *Science*, **266**(5187): 1021–1024.

Alhazov, A., Martín-Vide, C., y Pan, L. (2003). Solving a PSPACE-complete problem by recognizing P systems with restricted active membranes. *Fundamenta Informaticae*, **58**(2): 67–77.

Bloch, E. D. (2011). *Proofs and fundamentals: a first course in abstract mathematics*. Springer.

Bomze, I., Budinich, M., Pardalos, P., y Pelillo, M. (1999). The maximum clique problem. En: D.-Z. Du y P. Pardalos (eds.), *Handbook of Combinatorial Optimization*, pp. 1–74. Springer.

Bourgeois, N., Escoffier, B., Paschos, V. T., y van Rooij, J. M. (2012). Fast algorithms for max independent set. *Algorithmica*, **62**(1-2): 382–415.

Butenko, S. y Wilhelm, W. E. (2006). Clique-detection models in computational biochemistry and genomics. *European Journal of Operational Research*, **173**(1): 1–17.

Calude, C. S. y Păun, G. (2001). *Computing with cells and atoms: an introduction to quantum, DNA and membrane computing*. CRC Press.

Castellanos, J., Păun, G., y Rodríguez-Patón, A. (2000). P systems with worm-objects. Reporte técnico 123, University of Auckland. Recuperado de http://www.cs.auckland.ac.nz/research/groups/CDMTCS//researchreports/123paun.pdf.

Cecilia, J., García, J., Guerrero, G., Martínez-del Amor, M., Pérez-Jiménez, M., y Ujaldón, M. (2012). The GPU on the simulation of cellular computing models. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, **16**: 231–246.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., y Stein, C. (2011). *Introduction to Algorithms, 3rd Edition*. MIT Press.

Díaz-Pernil, D., Gutiérrez-Naranjo, M. A., Jiménez, M. J. P., y Riscos-Núñez, A. (2007). Solving subset sum in linear time by using tissue P systems with cell division. En: J. Mira y J. R. Álvarez (eds.), *Proceedings of the 2nd international work-conference on The Interplay Between Natural and Artificial Computation, Part I: Bio-inspired Modeling of Cognitive Tasks*, Vol. 4527 de *Lecture Notes in Computer Science*, pp. 170–179.

Frisco, P. (2009). *Computing with cells: advances in membrane computing*. Oxford Univ. Press. New York, NY.

García-Arnau, M., Manrique, D., y Rodríguez-Patón, A. (2007a). A Parallel DNA Algorithm Using a Microfluidic Device to Build Scheduling Grids. En: J. Mira y J. Álvarez (eds.), *Bio-inspired Modeling of Cognitive Tasks*, Vol. 4527 de *Lecture Notes in Computer Science*, pp. 193–202. Springer.

García-Arnau, M., Manrique, D., Rodríguez-Patón, A., y Sosík, P. (2007b). A P system and a constructive membrane-inspired DNA algorithm for solving the Maximum Clique Problem. *Biosystems*, **90**(3): 687 – 697.

Graciani-Díaz, C. y Riscos-Núñez, A. (2005). Looking for simple common schemes to design recognizer P systems with active membranes that solve numerical decision problems. En: *Unconventional Computation 4th International Conference, UC 2005, Sevilla, Spain, October 3-7, 2005. Proceedings*, Vol. 3699 de *Lecture Notes in Computer Science*. Springer.

Gutiérrez-Naranjo, M., Perez-Jimenez, M., y Romero-Campero, F. (2005). A linear solution of subset sum problem by using membrane creation. En: J. Mira y J. Alvarez (eds.), *IWINAC 2005, Las Palmas de Gran Canaria*, LNCS 3561, pp. 258–267. Springer.

Hartmanis, J. (1995). On the weight of computations. *Bulletin of the EATCS*, **55**: 136–138.

Head, T., Yamamura, M., y Gal, S. (1999). Aqueous computing: writing on molecules. En: *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, Vol. 2, pp. –1010 Vol. 2.

Hromkovic, J. (2010). *Algorithmics for hard problems: introduction to combinatorial optimization, randomization, approximation, and heuristics*. Springer.

Ibrahim, Z., Tsuboi, Y., y Ono, O. (2006). Hybridization-ligation versus parallel overlap assembly: an experimental comparison of initial pool generation for direct-proportional length-based DNA computing. *NanoBioscience, IEEE Transactions on*, **5**(2): 103–109.

Ignatova, Z., Martínez-Pérez, I., y Zimmermann, K.-H. (2008). *DNA Computing Models*. Springer.

Jájá, J. (1992). *An Introduction to Parallel Algorithms*. Addison-Wesley.

Jiménez, M. J. P., Jiménez, A. R., y Caparrini, F. S. (2003). Complexity classes in models of cellular computing with membranes. *Natural Computing*, **2**(3): 265–285.

Karp, R. M. (1972). Reducibility among combinatorial problems. En: R. E. Miller y J. W. Thatcher (eds.), *Complexity of Computer Computations: Proceedings*, pp. 85–104. Plenum Press.

Krishna, S. N. y Rama, R. (1999). A variant of P systems with active membranes: Solving NP-Complete problems. *Romanian Journal of Information Science and Technology*, **2**(4): 357–367.

Krishna, S. N. y Rama, R. (2001). P systems with replicated rewriting. *Journal of Automata, Languages and Combinatorics*, **6**(3): 345–350.

Krishna, S. N. y Rama, R. (2003). Breaking DES using P systems. *Theoretical Computer Science*, **299**(1-3): 495–508.

Ledesma, L., Pazos, J., y Rodríguez-Patón, A. (2004). A DNA Algorithm for the Hamiltonian Path Problem Using Microfluidic Systems. En: N. Jonoska, G. Paun, y G. Rozenberg (eds.), *Aspects of Molecular Computing*, Vol. 2950 de *Lecture Notes in Computer Science*, pp. 72–84. Springer.

Ledesma, L., Manrique, D., y Rodríguez-Patón, A. (2005). A tissue P system and a DNA microfluidic device for solving the shortest common superstring problem. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, **9**: 679–685.

Leporati, A. y Gutiérrez-Naranjo, M. A. (2008). Solving subset sum by spiking neural P systems with pre-computed resources. *Fundamenta Informaticae*, **87**(1): 61–77.

Leporati, A. y Pagani, D. (2006). A Membrane Algorithm for the Min Storage Problem. En: H. Hoogeboom, G. Paun, G. Rozenberg, y A. Salomaa (eds.), *Membrane Computing*, Vol. 4361 de *Lecture Notes in Computer Science*, pp. 443–462. Springer.

Li, D. (2008). *Encyclopedia of Microfluidics and Nanofluidics*. 3 volúmenes. Springer.

Martín-Vide, C., Păun, G., Pazos, J., y Rodríguez-Patón, A. (2003). Tissue P systems. *Theoretical Computer Science*, **296**(2): 295 – 326.

Martinez-Perez, I., Zhang, G., Ignatova, Z., y Zimmermann, K.-H. (2007). Computational genes: a tool for molecular diagnosis and therapy of aberrant mutational phenotype. *BMC Bioinformatics*, **8**(1): 365.

Martínez-Pérez, I. M. y Zimmermann, K.-H. (2009). Parallel bioinspired algorithms for np complete graph problems. *Journal of parallel and distributed computing*, **69**(3): 221–229.

McCaskill, J. S. (2001). Optically programming DNA computing in microflow reactors. *Biosystems*, **59**(2): 125 – 138.

Mutyam, M. y Krithivasan, K. (2001). P systems with membrane creation: Universality and efficiency. En: Y. R. M. Margenstern (ed.), *Machines, Computations, and Universality: Third International Conference, MCU 2001 Chisinau, Moldavia, May 23-27, 2001, Proceedings*, Vol. 2055 de *Lecture Notes In Computer Science*, pp. 276–287, May 23-27. Springer.

Noort, D. V., Gast, F., y McCaskill, J. S. (2002). DNA computing in microreactors. En: *DNA7 Seventh International Meeting on DNA Based Computers*, pp. 33–45.

Obtulowicz, A. (2001). Deterministic p systems for solving sat problem. *Romanian Journal of Information Science and Technology*, **4**(1-2): 551–558.

Ouyang, Q., Kaplan, P. D., Liu, S., y Libchaber, A. (1997). DNA Solution of the Maximal Clique Problem. *Science*, **278**(5337): 446–449.

Pan, L. y Alhazov, A. (2006). Solving hpp and sat by p systems with active membranes and separation rules. *Acta Informatica*, **43**(2): 131–145.

Păun, G., Suzuki, Y., Tanaka, H., y Yokomori, T. (2004). On the power of membrane division in P systems. *Theoretical Computer Science*, **324**(1): 61–85.

Păun, G. (2000). Computing with Membranes. *Journal of Computer and System Sciences*, **61**: 108–143.

Păun, G. (2001). Computing with membranes: Attacking np-complete problems. En: I. Antoniou, C. Calude, y M. Dinneen (eds.), *Unconventional Models of Computation, UMC'2K*, Discrete Mathematics and Theoretical Computer Science, pp. 94–115. Springer.

Păun, G. (2006). Introduction to Membrane Computing. En: G. Ciobanu, G. Păun, y M. J. Pérez-Jiménez (eds.), *Applications of Membrane Computing*, Natural Computing Series, pp. 1–42. Springer.

Păun, G., Rozenberg, G., y Salomaa, A. (2010). *The Oxford Handbook of Membrane Computing*. Oxford University Press.

Smaldon, J., Romero-Campero, F., Fernández Trillo, F., Gheorghe, M., Alexander, C., y Krasnogor, N. (2010). A computational study of liposome logic: towards cellular computing from the bottom up. *Systems and Synthetic Biology*, **4**: 157–179.

Zandron, C., Mauri, G., y Ferretti, C. (2000). Universality and normal forms on membrane systems. En: R. Freund y A. Kelemenova (eds.), *Proc. Intern. Workshop Grammar Systems 2000*, pp. 61–74, July, Bad Ischl, Austria.