Diseño y verificación formal de programas moleculares en el modelo débil de Amos

Sergio Rodríguez Calvo

Septiembre de 2017

Departamento de Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

Abstract. *En el presente trabajo se pretende estudiar el diseño y la verificación formal de dos programas moleculares en el modelo débil de Amos que resuelven el problema de la generación de permutaciones; y el problema del camino hamiltoniano en su versión dirigida y sin nodos distinguidos. En concreto, se centra en reescribir, así como, completar algunos detalles de las demostraciones, que se encuentran el trabajo original [1]. El desarrollo de este trabajo tiene como objetivo ser entregado como trabajo final de la asignatura de Computación Bioinspirada.*

1. Introducción

En la década de los cincuenta comienza a ser evidente que existe una analogía entre algunos procesos matemáticos y ciertos procesos biológicos. Un organismo vivo puede ser visto como el resultado de aplicar una serie de operaciones bioquímicas sobre una cadena de ácido desoxirribonucleico (ADN).

Posteriormente, en la decada de los noventa, se demostró que se pueden usar ciertos procesos biológicos para atacar la resolubilidad de problemas matemáticos difíciles. Estos problemas también son conocidos como computacionalmente intratables, y son aquellos que su solución algorítmica toma una cantidad de recursos exponenciales en el tamaño del dato de entrada.

Esta resolubilidad está relacionada con la potencia de cálculo y la densidad de almacenamiento de los ordenadores convencionales.

En la propia década de los cincuenta ya se introdujo el concepto teórico de computación a nivel molecular. En los ordenadores convencionales la paralelización y la miniturización son un objetivo importante, y la computación molecular puede suponer un paso más en este sentido.

Sobre todo, a partir de que en la década de los ochenta cuando se demostró la existencia un límite en la potencia de cálculo y en la miniturización de los componentes electrónicos empleados en los ordenadores convencionales.

Por último se enumeran las principales ventajas del uso de la computación molecular:

- Sustitución de la luz por reacciones químicas, lo que implica un ahorro del consumo energético.
- El uso de interruptores moleculares permite, según se estima, disponer de más de mil procesadores en el mismo espacio que un procesador convencional.
- Se estima que los interruptores moleculares pueden aumentar cien mil millones de veces la capacidad de procesamiento respecto a los ordenadores convencionales.
- Se estima que se podrían reproducir la capacidad de cien ordenadores en el tamaño de un grano de sal fina.

2. Modelo débil de Amos

Antes de introducir el Modelo débil de Amos, se necesita previamente conocer los detalles y principios de la computación molecular, así como, los distintos modelos previos a este, los cuales se pueden encontrar en el documento original del profesor del departamento de Ciencias de la Computación de la Universidad de Sevilla, Mario de Jesús Pérez Jiménez [1].

El Modelo débil de Amos consiste en modelo de computación basada en ADN, esto es, que utiliza como sustrato computacional el ADN y en el cual se realizan filtrados sobre el sustrato anterior. En este caso, no existe memoria de acceso aleatorio como en la computación clásica. Para almacenar el sustrato, al igual que en otros modelos de computación molecular, se utiliza un tubo de ensayo que contendrá la muestra.

Dicho tubo es un multiconjunto finito de cadenas del alfabeto $\sum_{ADN} = \{A, C, G, T\}$.

A nivel abstracto, en el Modelo débil de Amos las operaciones que se pueden realizar sobre los tubos son las siguientes:

- **Quitar (T,{$\gamma_1, ..., \gamma_n$}):** Dado un tubo, T, y un número finito de cadenas, $\gamma_1, ..., \gamma_n$, de $\sum$, devuelve el tubo obtenido de T eliminando todas aquellas cadenas que contengan, al menos, una ocurrencia de alguna de las cadenas $\gamma_1, ..., \gamma_n$.

- **Copiar (T,{$T_1, ..., T_n$}):** Dado un tubo, T, y un numero natural $k \geq 2$, devuelve k tubos, $T_1, ..., T_n$, que son copias exactas de T.

- **Unión ({$T_1, ..., T_n$}):** Dados los tubos $T_1, ..., T_n$, con $k \geq 2$, devuelve un tubo T, cuyo contenido es la unión de los tubos $T_1, ..., T_n$ como multiconjuntos.

- **Selección (T):** Dado un tubo, T, selecciona aleatoriamente un elemento de T en el caso en que $T \neq \emptyset$; en caso contratio, devuelve **NO**.

Estas operaciones serán instrucciones moleculares primitivas del modelo débil. Además, cabe destacar que en este modelo la única operación molecular que implementa paralelismo masico es *quitar*.

El primer problema, el de la generación de permitaciones, será abordado previo al problema del camino hamiltoniano , ya que, será necesario para su resolución.

2.1. Problema de la generación de permutaciones

En esta sección se muestra como abordar desde el punto de vista de la computación molecular la resolubilidad del problema de la generación de permutaciones. Antes, se define en qué consiste una permutación para a continuación abordar el problema de la generación de las mismas.

Una permutación la definimos como *dado un numero natural, $n \geq 1$, una permutación de orden n es una aplicación biyectiva del conjunto finito {1,...,n} en sí mismo.*

Una vez tenemos definido qué es una permutación vamos a introducir el problema de generación de permutaciones, que consiste en, *dado un número natural, $n \geq 2$, generar todas las permutaciones de orden n.*

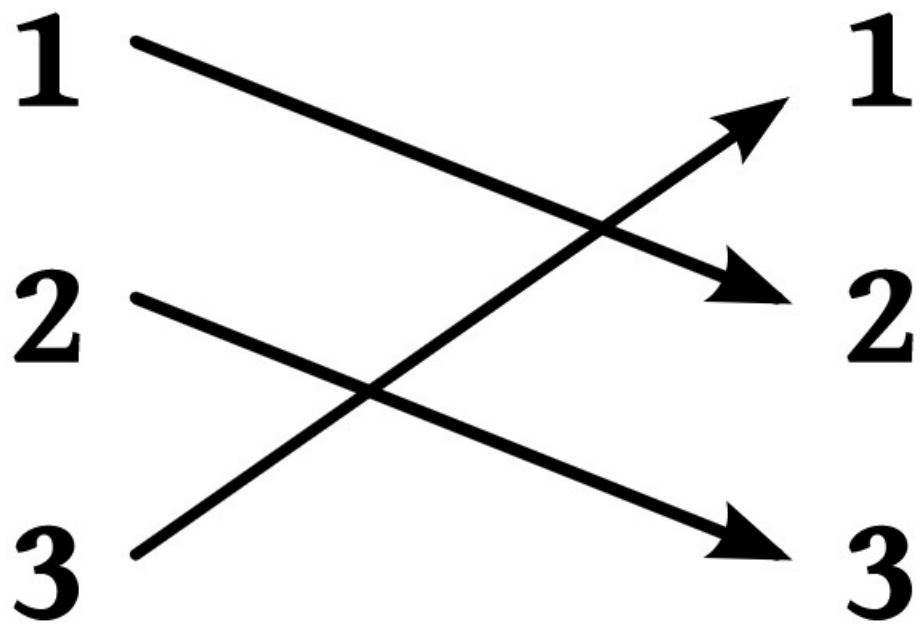

Figura 1: Ejemplo de permutación considerada como función biyectiva. En este caso, para $n = 3$.

2.1.1. Diseño del programa molecular

En primer lugar, hay que definir el modelo mediante el cual se pretende representar el problema con los elementos que tenemos en la computación molecular. Esto es, definir por un lado el alfabeto: $\sum = \{(p_i, c_j) : 1 \leq i, j \leq n\}$, donde p_i y c_j son dos oligos que codificarán respectivamente la posición i-ésima en la permutación y el número j.

A continuación, se necesita definir el tubo de entrada, T_0, para poder comenzar el experimento, es decir, aplicar las operaciones abstractas de la sección anterior según un determinado algoritmo. En este caso, use trata de un multiconjunto de entrada con un número finito de moleculas y que codifican todas las posibles sucesiones para una longitud n. Formalmente sería:

$$T_0 = \{\{\sigma \in \sum^n : \exists x_1, ..., \exists x_n(\sigma = (p_1, x_1), (p_2, x_2), ..., (p_n, x_n))\}\}$$

En la fórmula anterior, σ, es una codificación concreta, tal que, $\sigma = p_1x_1...p_nx_n$.

La idea aquí es generar todas las posibles permutaciones, utilizando un tubo inicial, T_0, que contiene la codificación de todas las posibles soluciones de longitud n. El programa sigue los siguientes pasos hasta $n-1$ veces:

- Un primer filtro sobre T_0 para seleccionar las moleculas, σ, tal que:

  $$\forall r > 1((\sigma)_1 \neq (\sigma)_r)$$

  Esto es, para todas las posiciones, r, mayores que 1, devolver todas aquellas codificaciones, tal que, el número que ocupa la posición r-ésima sea distinto de el número de la primera posición.

- Un segundo filtro respecto del paso anterior donde se seleccionan las moleculas, σ, tal que:

  $$\forall r > 2((\sigma)_1 \neq (\sigma)_r \wedge (\sigma)_2 \neq (\sigma)_r)$$

  Esto es, del conjunto resultante del paso anterior, para todas las posiciones, r, mayores que dos, devolver todas aquellas codificaciones, tal que, el número que ocupa la posición r-ésima sea distinto de el número de la posición primera y de la posición segunda a la vez.

La notación, $(\sigma)_r$, representa el número que ocupa la posición *r-ésima* en la sucesión de longitud n codificada por σ. Es decir, $(\sigma)_r = x_r$ para todo r $(1 \leq r \leq n)$.

El algoritmo a seguir en el programa molecular, basado en la idea de filtrar sobre un determinado tubo T_0 de entrada, es el siguiente:

Require: T_0
  for $j \leftarrow 1$ in $n-1$ **do**
    copiar$(T_0, \{T_1, ..., T_n\})$
    for $i \leftarrow 1$ in n **do**
      quitar$(T_i, \{p_j r : r \neq i\} \cup \{p_k i : j+1 \leq k \leq n\})$
    unión$(\{T_1, ..., T_n\}, T_0)$
  return T_0

Este algoritmo presenta una complejidad $O(n^2)$, es decir, orden cuadrático en n. Por tanto, el número de operaciones moleculares es n^2.

2.1.2. Verificación formal del programa molecular

Para realizar la verificación formal es necesario etiquetar cada uno de los tubos que se van a obtener a los largo del algoritmo descrito anteriormente. Por tanto, el algoritmo va a ser reescrito de la siguiente forma:

Require: T_0
  for $j \leftarrow 1$ in $n-1$ **do**
    copiar($T_0^{j-1}, \{T_1^j, ..., T_n^j\}$)
    for $i \leftarrow 1$ in n **do**
      $\bar{T}_i^j \leftarrow quitar(\mathrm{T}_i^j, \{p_j r : r \neq i\} \cup \{p_k i : j+1 \leq k \leq n\})$
    unión($\{\bar{T}_1, ..., \bar{T}_n\}, T^j$)
  return T_{n-1}

Antes de continuar, hay que aclarar la notación a emplear a partir de aquí. $A_{\sigma,j}$ será el conjunto $\{\sigma_1, ..., \sigma_j\}$, es decir, número en cada posición j-ésima, para $(1 \leq j \leq n)$, en cada $\sigma \in T_0$. Esto es, un conjunto que contiene todos los números que contienen una determinada codificación, σ, desde 1 hasta la posición j en cada caso.

Esta notación que se acaba de describir será necesaria para la correción formal del programa que se acaba de reescribir, y que se utiliza en la siguiente formula:

$$\theta(j) \equiv \forall \sigma \in T^j(|A_{\sigma,j}| = j \wedge \forall r(j+1 \leq r \leq n \rightarrow (\sigma)_r \notin A_{\sigma,j}))$$

La formula, $\theta(j)$, expresa que toda molecula del tubo T^j codifica una sucesión de longitud n tal que los j primeros términos son distintos entre sí y, además, distintos de los restantes términos de la sucesión.

A continuación, se van a exponer una serie de teoremas con los que se pretende realizar la verificación formal, tomando el algoritmo que se ha reescrito en este apartado.

Teorema 1.1. $\forall j(1 \leq j \leq n-1 \rightarrow \theta(j))$. Es decir, la formula θ es un invariante del bucle principal. Esto es, que θ no cambia a lo largo de las transformaciones que sufre el tubo en el bucle principal.

2.2. Problema del camino hamiltoniano en versión dirigida sin nodos distinguidos

En este sección se muestra como abordar desde el punto de vista de la computación molecular la resolubilidad del problema del camino hamilto-

niano en su versión digidia y sin nodos distinguidos.

El problema del camino hamiltoniano en su versión dirigida y sin nodos distinguidos consiste en dado un grafo dirigido, determinar si existe un camino simple que pasa por todos los nodos del grafo. O lo que es lo mismo, si el grafo posee un ciclo hamiltoniano.

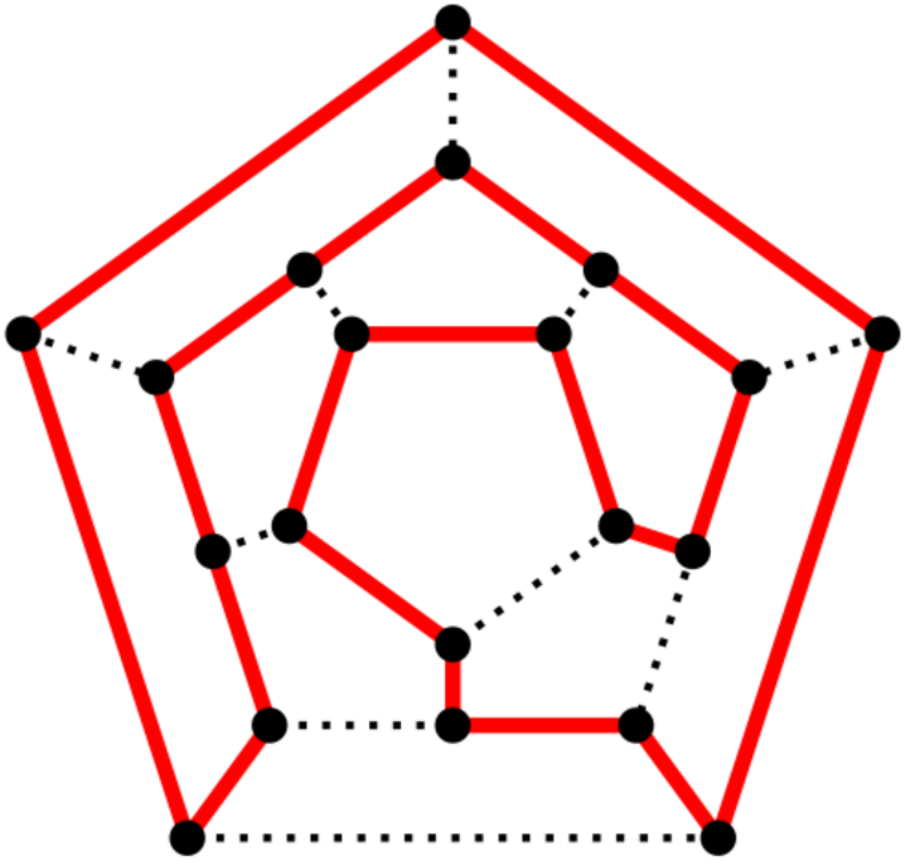

Figura 2: Ejemplo de camino hamiltoniano sobre un grafo.

2.2.1. Diseño del programa molecular

Para dar solución a este problema en el modelo débil de Amos, partimos de un $G = (V, E)$, tal que, G es un grafo dirigido con $V = \{1, ..., n\}$ nodos.

Se debe definir en primer lugar el alfabeto $\sum = \{(p_i, c_j) : 1 \leq i, j \leq n\}$. Donde, p_i y c_j, son dos oligos que codifican respectivamente la posición i-

ésima del camino, y el nodo j. Es decir, el nodo j está en la posición i-ésima del camino.

Como punto de partida, se necesita un tubo de ensayo inicial, T_0. Este problema necesita de todas las permutaciones de orden n, por tanto, el tubo de ensayo inicial es igual que el utilizado en el problema de la generación de permutaciones y una notación parecida. Esto es:

$$T_0 = \{\{\sigma \in \sum^{n} : \exists x_1, ..., \exists x_n(\sigma = (p_1, x_1), (p_2, x_2), ..., (p_n, x_n))\}\}$$

En este caso, la notación es la siguiente:

Referencias

[1] Mario de Jesús Pérez Jiménez. *Computación molecular sin memoria basada en ADN*. 2001.