MEJORA DE UN ALGORITMO DE CONTEO PARA MODELOS MARKOVIANOS EN YACIMIENTOS LATERÍTICOS.

Andys Marcos Ramírez Aberasturis (ponente)[1], Ramón Eddie Peña Abreu[2], Luis Yendris. Broscat Sánchez[1]

[1]Facultad de Matemática y Computación. Universidad de Oriente, yendris@csd.uo.edu.cu,

[2] *Centro de Investigaciones del Níquel. Carretera Yagrumaje km 5 ½ Moa. Holguín, Cuba. Teléf: 024-6-7976, 024-6-4184. Fax: 53 24 62202. rpena@cil.moa.minbas.cu,*

RESUMEN

El modelado Markoviano es una aplicación novedosa al estudio de yacimientos lateríticos, en su implementación computacional, se han enfrentado problemas numéricos que han conllevado a la necesidad de mejorar los algoritmos de conteo utilizado para el cálculo de las probabilidades condicionales del modelo. En este trabajo se exponen las mejoras realizadas a dichos algoritmos orientadas a su implementación en procesamiento paralelo, se muestran los resultados obtenidos en las corridas experimentales del en procesos secuenciales y paralelizadas. Los resultados que aquí se exponen son el fruto de la colaboración entre el centro de Investigaciones del Níquel (CEINNIQ) y el Departamento de Matemática de la Universidad de Oriente, en virtud de un convenio de colaboración que se estableció entre dichos centros y que ha servido de trabajo de diploma a un estudiante de la carrera de Licenciatura en Matemática.

INTRODUCCIÓN

Los yacimientos lateríticos proporcionan la materia prima fundamental de la Industria Cubana del Níquel, es muy conocida la complejidad de dichos yacimientos en cuanto a la composición química y mineralógica de los materiales que en ellos yacen (Ariosa Iznaga, 1977; Rojas Purón,1994; Vera Sardinas, 2001, Lavaut, W, 1998), la exactitud en el muestreo que a ellos se le realiza es vital para su estudio, ya que de él depende la exactitud en la toma de decisiones mineras y en el proceso metalúrgico. La influencia del muestreo en la eficiencia de la cadena productiva se convirtió en un problema que motivó la búsqueda de una solución siguiendo un nuevo enfoque (Peña Abreu y Legra Lobaina, 2005), este unificó en un modelo matemático varios aspectos que no se habían tratado en forma de sistema hasta ese momento. La modelación estuvo orientada a obtener mayor cantidad de información en menos tiempo y con menor costo, así como, minimizar los errores con respecto a los métodos actuales de muestreo. El modelo propuesto es de tipo probabilístico, en específico Markoviano y se basa en la existencia de tipos o clases de materiales patrones en dichos yacimientos, donde se asume que el conjunto de estas clases contiene todas las variedades presentes en ellos y representa el espacio muestreal del yacimiento. Las clases se obtuvieron a partir de un estudio estadístico multivariado, utilizando los métodos de clasificación no supervisada y supervisada del reconocimiento de patrones (Peña Abreu y Matos Elias, 2007). Cada clase se asume como un estado en un proceso de Markov.

El modelo propuesto por Peña Abreu y Legra Lobaina, 2007; para la optimización de las redes de exploración está compuesto de tres modelos, el primero modela el yacimiento basado en las clases patrones antes mencionadas, el segundo optimiza la red a partir del primero y el tercero optimiza los aproximadores para las variables geoquímicas de la red. En el presente trabajo se abordará solo en el abastecimiento numérico del primer modelo, o sea, de:

$$X(t,r) = \sum_i \pi(i) \cdot \rho(t,r) \qquad (1)$$

El modelo (1) ya se ha codificado para los ordenadores, lo que ha permitido evaluarlo computacionalmente (Peña Abreu, Silva Labrada et al, 2009) y se ha encontrado el problema de la complejidad numérica en la construcción del conjunto de las matrices de transferencia $\pi(i)$.

En la construcción de la matriz de transferencia es necesario comparar exhaustivamente todos los elementos de una lista entre sí, lo que implica contar las características durante esta comparación. Por la naturaleza del conteo que impone (1), se utilizan algoritmos de fuerza bruta o de búsqueda exhaustiva, en este tipo de algoritmos se puede alcanzar mayor eficiencia computacional en la medida que se divida la lista, de tal forma que se realicen mayor cantidad de conteos en una iteración y que cada iteración sea independiente de las demás. Se hace referencia a dos situaciones, la *primera* es lograr que no sea necesario realizar una cantidad de iteraciones igual a $n(n-1)$ para comparar los *n* elementos entre sí, y la *segunda* está relacionada con crear un algoritmo que pueda ser dividido en tareas independientes para facilitar la implementación del procesamiento paralelo.

Para ello se desarrollaron dos algoritmos (Peña Abreu, 2008 a y b) para construir el modelo (1), *el primero* es un algoritmo que en cada iteración realiza varias combinaciones del conteo, utilizando un sistema iterativo que se mueve de los extremos al centro de la lista, en la cual se comparan el primer elemento con el sucesor y el último con el antecesor en ambos casos hacia adelante y hacia atrás; de tal forma que en las iteraciones se comparan los elementos i con los $i+1$ y los elementos $n-i$ con los elementos $(n-i)-1$, i corre por los elementos hasta la mitad de la lista, luego siguiendo el mismo método se concibió un nuevo algoritmo que compara además los extremos entre sí (Peña Abreu y Silva Labrada, 2009). Con el primero de estos algoritmos se logra disminuir la cantidad de iteraciones en $\frac{n(n-1)}{4}$ y con el segundo $\frac{n(n-1)}{8}$. La metodología utilizada para reducir el algoritmo de búsqueda exhaustiva la denominaremos en lo adelante *Reducción Cónica* (RC) en similitud con el accionar del algoritmo entre los extremos de la lista. De esta forma se puede considerar que el segundo algoritmo resulta una reducción cónica de mayor profundidad que la utilizada en el primero y se pueden denotar como RC_4 y RC_8 respectivamente.

El segundo de estos algoritmos se codificó en un software y utilizando la característica de independencia en las operaciones entre iteraciones, se implementó un procesamiento en paralelo para la solución de (1). Aún así, no se logró la velocidad suficiente que satisface en un Software el tiempo de respuesta para el modelado de un yacimiento, de aquí que disminuir el tiempo de cálculo y de esta forma aumentar la eficiencia y la eficacia del algoritmo en la solución de (1) es el problema que se expone. Es evidente que el objeto de nuestra investigación lo constituyó el algoritmo para la solución de (1). Entonces para solucionar el problema se propuso como objetivo mejorar el algoritmo para la construcción del modelo (1) con relación a los existentes anteriormente, la metodología de Reducción Cónica orientada al multiprocesamiento es el campo de acción del trabajo. La mejora se pretende lograr bajo la hipótesis de que, si se realizan Reducciones Cónicas de mayor profundidad y se combina con la utilización del paralelismo, entonces se logrará disminuir el tiempo de respuesta en la solución de (1) y con ello la eficiencia y eficacia del algoritmo. Se ha pretendido desarrollar una reducción más, o sea, una nueva partición en la que se comparan entre sí todos los extremos en cada iteración, esto para facilitar que las iteraciones disminuyan de $\frac{n(n-1)}{8}$ hasta $\frac{n(n-1)}{32}$, se pretendió además, mantener la propiedad de independencia entre las iteraciones para facilitar el multiprocesamiento.

METODOLOGÍA EMPLEADA

Por la naturaleza del conteo que impone construir una matriz de transferencia para la obtención de la solución de (1), se utilizan algoritmos de fuerza bruta o de búsqueda exhaustiva, en este tipo de algoritmos se puede alcanzar mayor eficiencia computacional en la medida que se divida la lista de datos, de tal forma que se realicen mayor cantidad de conteos en una iteración. Lo anterior conlleva a la realización de un conteo sucesivo acompañado de comparaciones, o sea, se debe comparar todos los elementos de la tabla entre sí, sin comparar cada elemento con sí mismo.

Por esta razón el algoritmo RC_8 y el propuesto en esta investigación, tienen características comunes, los dos deben resolver un problema de conteo que genera un número de iteraciones igual a $n(n-1)$ y además resuelven la tarea utilizando un mismo sistema de reducción y comparación entre los elementos.

Siguiendo la metodología ya explicada, a partir de RC_8, se realizó una nueva reducción, la cual divide la lista en 4 partes, cada una de las cuales se compara en la medida que se avanza del inicio hasta $1/4$ de la lista. Las comparaciones se logran con la misma estructura de iteración que en RC_8, utilizando dos ciclos de repetición, uno de los cuales está anidado dentro del otro. En el ciclo externo (CE), se realizan las comparaciones correspondientes a los extremos de la lista y de las particiones, y en el ciclo anidado (CA), se realiza las comparaciones de los extremos de la lista y las particiones con el resto de los elementos, además se compara el elemento corriente del ciclo externo con el resto de los elementos de la lista. Este algoritmo alcanza su máxima eficiencia cuando se le aplica a listas con longitud múltiplo de 4, en caso contrario (también es eficiente) se debe hacer un ciclo en dependencia del resto de la división entera *n mod 4*, para completar las comparaciones necesarias con el ciclo externo. En el caso de que el resto sea 1, se implementa un ciclo que compara el último elemento con toda la lista; en el caso de que el resto sea 2, se compara los dos últimos entre si y con los demás elementos de la lista, esto último implementando también un ciclo. En el caso que el resto sea 3, se compara los tres últimos elementos entre si y luego con el resto de la lista. Esto hace que el sistema combinatorio sea más complejo que en RC_8. Esta implementación del algoritmo realiza 32 comparaciones en cada CA, en el CE se realizan 12 comparaciones que completan la revisión exhaustiva de la lista, disminuyendo la cantidad de iteraciones en $n(n-1)/32$. En estos ciclos adicionales se incluyen además las comparaciones entre los índices correspondientes a *n mod 4*.

Los algoritmos RC son polinomiales, para los cuales la complejidad computacional depende del tamaño de *n*. Teniendo en cuenta que *n* en (1) suele ser del orden de 10^5, lo que genera una cantidad de iteraciones muy alta ($\approx 10^{25}$). Por esta razón se realizó un análisis detallado de la eficiencia que presentan los algoritmos RC_8 y RC_{32}. La eficiencia se mide en función de dos parámetros, el espacio de memoria que utiliza y el tiempo que tarda en ejecutar la tarea (Guerequeta y Vallecillo, 1998), aquí nos ocupamos del cálculo del tiempo de ejecución. Para el cálculo del número de operaciones elementales se utilizó las siguientes reglas (Guerequeta y Vallecillo, 1998,):

1) Se definió que el tiempo de ejecución para una operación elemental (OE) es de orden 1.
2) El tiempo de ejecución de una secuencia consecutiva de instrucciones se calcula sumando los tiempos de ejecución de cada una de las instrucciones.

   Por tanto el tiempo de ejecución para las llamadas a procedimientos y funciones $F(p_1, p_2, ..., p_n)$, se calcula de la siguiente forma:

   - Se adiciona el tiempo de una OE por la llamada.
   - Se adiciona el tiempo de evaluación de los parámetros $p_1, p_2, ..., p_n$.
   - Se adiciona el tiempo que tarda en ejecutarse F.

   Finalmente se tiene que $T = 1 + T(p_1) + T(p_2) + ... + T(p_n) + T(F)$

   Para calcular el tiempo de ejecución en todas las sentencias iterativas (FOR, REPEAT, LOOP) basta expresarlas como un bucle WHILE.

Debido a que los algoritmos RC_8 y RC_{32} están estructurados con dos ciclos que utilizan la sentencia WHILE, uno de ellos anidado, el tiempo de ejecución es $T = T(C) + (n^o\ iteraciones) * (T(S) + T(C))$.

Obsérvese que tanto *T(C)* como *T(S)* pueden variar en cada iteración, y por tanto habrá que tenerlo en cuenta para su cálculo. Es decir, la búsqueda del tiempo de ejecución se realiza después de calcular el

número de operaciones elementales para cada uno de los algoritmos, lo que permitió conocer la complejidad de ellos en función del tiempo de ejecución.

El tiempo de ejecución del algoritmo RC_{32} y del RC_8 en los distintos casos viene dado por las siguientes expresiones:

$RC_{32}, n \bmod 4 = 0$	$T_{RC_{32}}(n) = \left(\frac{3+32x}{32}\right)n^2 + \left(\frac{9-8x}{8}\right)n + 11$	$RC_{32}, n \bmod 4 = 2$	$T_{RC_{32}}(n) = \left(\frac{3+32x}{32}\right)n^2 + \left(\frac{33+24x}{8}\right)n + (24+2x)$
$RC_{32}, n \bmod 4 = 1$	$T_{RC_{32}}(n) = \left(\frac{3+32x}{32}\right)n^2 + \left(\frac{33+8x}{8}\right)n + 24$	$RC_{32}, n \bmod 4 = 3$	$T_{RC_{32}}(n) = \left(\frac{3+32x}{32}\right)n^2 + \left(\frac{33+40x}{8}\right)n + (24+6x)$
RC_8, Par	$T_{RC_8}(n) = \left(\frac{7+8x}{8}\right)n^2 + \left(\frac{25+12x}{4}\right)n + 6$	RC_8, Impar	$T_{RC_8}(n) = \left(\frac{7+8x}{8}\right)n^2 + \left(\frac{11-4x}{4}\right)n + 6$

En ambos casos la complejidad es de orden $\Theta(n^2)$, por lo que RC_8 y RC_{32} presentan el mismo orden de complejidad. Por razones de espacio se muestra la comparación de dos casos, el primero *cuando n es par* y el segundo *cuando n es impar*.

La variable *x* que aparece en cada uno de los tiempos de ejecución obtenidos, define el tiempo de ejecución de la función que realiza la comparación entre dos combinaciones, es idéntica en ambos algoritmos y puede alcanzar una gran cantidad de OE, está acotada inferiormente por 20 OE, de aquí que se toma $x \geq 20$. Para realizar las comparaciones y profundizar en el conocimiento del comportamiento de los tiempos de ejecución, se aplicó el *Principio de Invarianza*, con el cual se puede comparar la eficiencia de dos implementaciones de un algoritmo de igual orden. Según Guerequeta y Vallecillo, 1998, este principio expresa que: *Dado un algoritmo y dos implementaciones suyas I_1 e I_2, que tardan $T_1(n)$ y $T_2(n)$ segundos respectivamente, el Principio de Invarianza afirma que existe una constante real $c > 0$ y un número natural n_0 tales que para todo $n \geq n_0$ se verifica que $T_1(n) \leq cT_2(n)$.*

Siguiendo este Principio se toma $T_1(n) = T_{RC_{32}}(n)$ y $T_2(n) = T_{RC_8}(n)$.

<u>Caso 1</u>: $n \bmod 2 \neq 0$

a) Se analiza primeramente $T_{RC_{32}}(n)$ para el caso $n \bmod 4 = 1$ y $T_{RC_8}(n)$ para el caso impar.

Partiendo de la expresión $T_{RC_{32}}(n) \leq cT_{RC_8}(n)$, se crea la diferencia $T(n) = T_{RC_{32}}(n) - cT_{RC_8}(n) \leq 0$

Sustituyendo los valores correspondientes a este caso se obtiene:

$$\left(\frac{3+32x}{32} - \frac{c(7+8x)}{8}\right)n^2 + \left(\frac{33+8x}{8} - \frac{c(25+12x)}{4}\right)n + (24-6c) \leq 0$$

Evaluando $x = 20$ en la expresión anterior se encuentra la constante c. Teniendo en cuenta que *T(n)* es también cuadrático con respecto a n, el principio solo se cumplirá cuando la parábola sea decreciente, o sea, si se cumpla que: $\left(\frac{3+32x}{32} - \frac{c(7+8x)}{8}\right) < 0$ y esto solo es posible para $c > \frac{643}{668} = 0.9625$

Para un c que cumple con esta condición $\left(c = \frac{161}{167}\right)$ se obtiene: $T(n) = \left(-\frac{1}{32}\right)n^2 + \left(-\frac{53099}{1336}\right)n + \frac{3042}{167} = 0$

Para obtener el n_0 correspondiente a este c, obtenemos las soluciones de $T(n)$: $n_1 = 0.458149$ y $n_2 = -1272.29$. Se toma a n_0 como la parte entera más uno de n_1, es decir, $n_0 = 1$. La comparación resulta favorable a RC_{32}, ya que para $n_0 = 1$; se tiene que $T(n_0) = -21.56$ y por tanto se cumple el principio de invarianza. Esto garantiza que para cualquier $n \geq 1$, RC_{32} tendrá un tiempo menor que RC_8. Entonces para todos los c>0.9625 siempre se encontrará un n_0 para el cual se cumple que el tiempo de RC_{32} es

menor que RC_8. La constante real c, $x = 20$, así como el procedimiento empleado en este caso, se utilizarán para todas las comparaciones de estos dos algoritmos, ya que el coeficiente cuadrático en las respectivas funciones de los tiempos de ejecución es igual.
b) Siguiendo la misma metodología, para el caso $n \bmod 4 = 3$ de RC_{32} y RC_8 para el caso impar, se obtiene que $n_0 = 1292$, luego $T(n_0) = -16.51$ y entonces para cualquier $n \geq 1292$ se cumple el principio de invarianza y se prueba la mejora en RC_{32}, en el mkodelo (1) nunca se trabajará con valores tan bajos de *n*.

Caso 2: $n \bmod 2 = 0$

a) Para el RC_{32} con resto 0 y el RC_8 caso par, se obtiene que $n_0 = 1$ y $T(n_0) = -30.32$, por lo que se prueba la mejora.
b) Para el RC_{32} con resto 2 y el RC_8 caso par, $n_0 = 2585$ y luego se prueba la mejora dado que ningún n en el estudio de los yacimientos con el modelo (1) será menor que 2585.

Se codificó en lenguaje Delphi v.7.0 el RC_{32}, con el objetivo de realizar corridas numéricas experimentales. Para las pruebas de los algoritmos, estos fueron procesados en ordenadores personales del tipo Pentium 4, con procesador Intel (R) Core (TM) 2 Duo con 2 Gb de memoria RAM. Los resultados se muestran en las tablas 1 y 2.

Tabla Nº 1. Corrida experimental # 1. n=28511.

Estadísticas de los algoritmos.

	RC_8		RC_{32}	
	Iteración	Combinaciones	Iteración	Combinaciones
Promedio	4.337268	0.0000750	3.827164	0.000035
Mediana	4.359000	0.0000733	3.578000	0.000033
Moda	0.078000	0.0000000	1.500000	0.000000
Máximo	55.516000	0.0005334	16.109000	0.000232
Mínimo	0.000000	0.0000000	0.000000	0.000000
n	28511	28511	28511	28511
$\omega = n(n-1)$	812848610	812848610	812848610	812848610

Tabla Nº 2. Corrida experimental # 2. n=28511. Estadísticas de las comparaciones prácticas de los algoritmos.

	H	M	S	Msg	Total (Seg)
RC32	7	43	16	692	27796.69
RC8	17	10	23	416	61823.42
	DIFERENCIAS				44.96%
	10	33	7	276	

CONCLUSIONES

Luego de profundizar la Reducción Cónica al algoritmo RC_8, se logró disminuir el tiempo de respuesta en la solución del modelo Markoviano, obteniéndose el algoritmo RC_{32}. Quedó demostrado teórica y prácticamente, en las comparaciones realizadas a ambos algoritmos que RC_{32} es más eficiente con respecto a RC_8.

RESUMEN BIBLIOGRÁFICO

Ariosa Iznaga, J. D., 1977. *Curso de Yacimientos Minerales Metálicos, Tipo Genético*. Editorial Pueblo y Educación.

Guerequeta R, Vallecillo A. 1998. *Técnicas de Diseño de Algoritmos*. Servicio de Publicaciones de la Universidad de Málaga. ISBN: 84-7496-666-3. http://www.lcc.uma.es/~av/Libro/indice.html. Segunda Edición: Mayo 2000.

Lavaut, W, 1998. *Tendencias geológicas del intemperismo de las rocas ultramáficas en Cuba oriental*. Minería y Geología, 15(1), 9-16.

Papadimitriou, Christos H. 1998. *Combinatorial optimization: algorithms and complexity*. Mineola, New York: Dover: 499 p. ISBN: 0-486-40258-4.

Peña Abreu R.E, 2007. *Modelo Matemático para la Optimización de las Redes de Exploración y Explotación en Yacimientos Lateríticos*. CD, Memorias de la II Convención Cubana de Ciencias de la Tierra. ISBN: 978-959-7117-16-2. La Habana. Marzo 20-23.

Peña Abreu R.E, Legrá Lobaina A. A. 2005. *Nuevo enfoque al problema de la optimización de redes de exploración en yacimientos lateríticos*. CD, Memorias de la I convención Cubana de ciencias de la Tierra. ISBN: 959-7117-03-7: MIN3-3.

Peña Abreu R.E, Matos Elías L. Perez Melo N. 2007. *Propuestas de Clases Patrones para los Yacimientos Lateríticos Cubanos*. CD, Memorias de la II Convención Cubana de Ciencias de la Tierra. ISBN: 978-959-7117-16-2. La Habana. Marzo 20-23.

Rojas Purón A.L., 1994. *Principales Fases Minerales Portadoras de Níquel en los Horizontes Lateríticos Del Yacimiento Moa*, Tesis presentada en opción al Grado Científico de Doctor en Ciencias Geológicas. ISMM, Moa, Holguín.