Universidad de las Ciencias Informáticas
Facultad 6

Título: Cálculo en paralelo de un modelo Markoviano para la Exploración de Níquel.

Trabajo de Diploma para optar por el título de

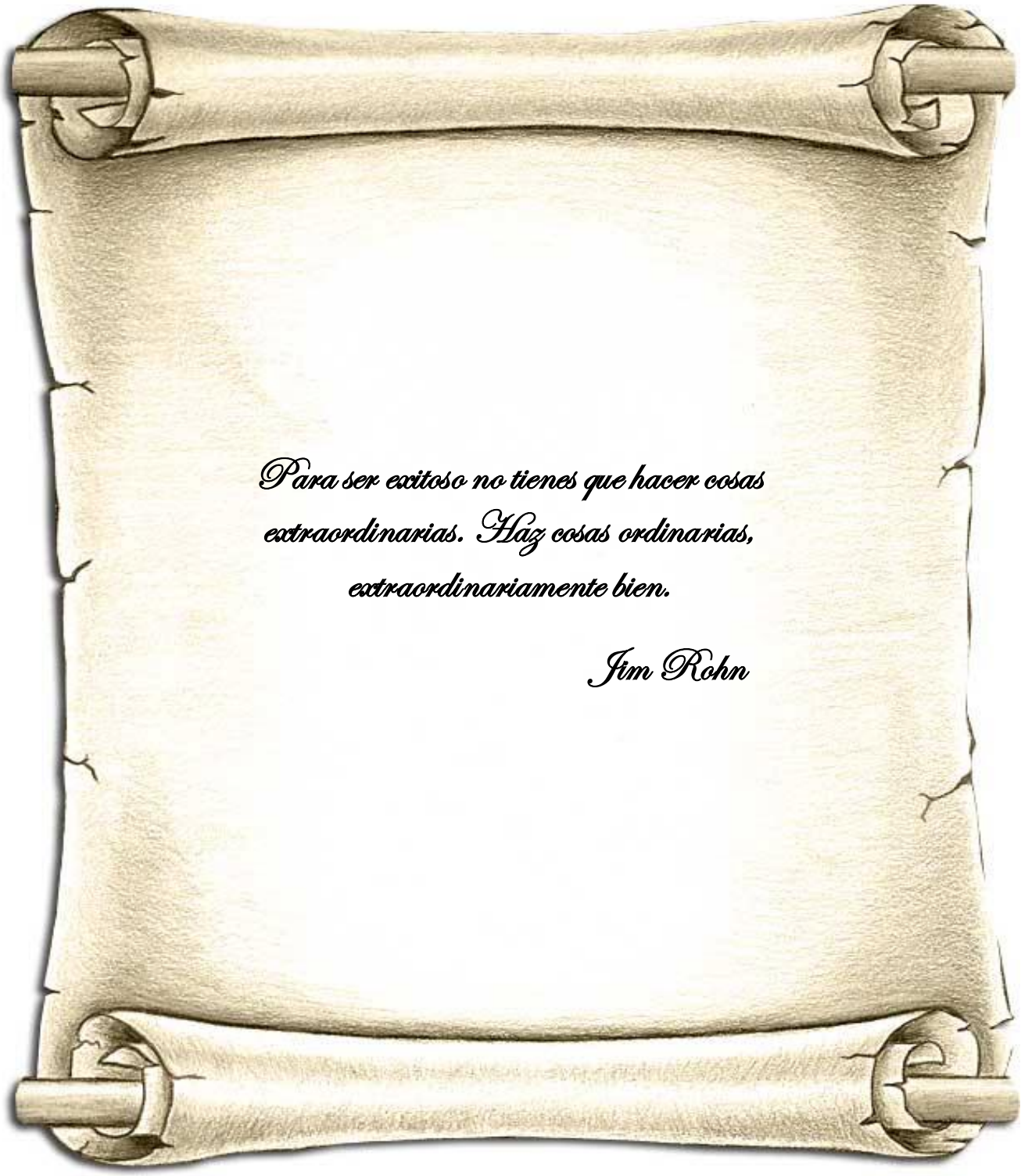
Ingeniero Informático

Autor(es): Roberto Bárbaro Beltrán Ibáñez

Yilianis Suárez González

Tutor: Dannier Trinchet Almaguer

Ciudad de la Habana
Cuba
2010



Para ser exitoso no tienes que hacer cosas extraordinarias. Haz cosas ordinarias, extraordinariamente bien.

Jim Rohn

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Yilianis Suárez González

_____
Firma del Autor

Roberto Bárbaro Beltrán Ibáñez.

_____
Firma del Autor

Dannier Trinchet Almaguer

Firma del Tutor

DATOS DE CONTACTO

Tutor: Lic. Dannier Trinchet Almaguer.
Centro Laboral: Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba.
Correo Electrónico: trinchet@uci.cu

AGRADECIMIENTOS

A Dios por darnos fuerzas para poder recorrer este largo camino.

A nuestra familia por haber hecho de nosotros las personas que somos hoy.

A todos nuestros amigos por su incondicional apoyo y cariño para poder lograr nuestras metas.

A nuestro tutor gracias por su apoyo, su exigencia y por sacrificar parte de su tiempo a despalillarnos lo que hacíamos en aras de perfeccionar nuestro trabajo.

A los chicos del grupo de primer año y aquellos que nos recibieron con los brazos abiertos en los otros grupos donde estuvimos.

A todos aquellos que durante estos cinco años nos han dado un espacio en su corazón.

Al tribunal por sus consejo para que esta tesis quedara lo mejor posible.

Al oponente por sus recomendaciones.

En general a todos los que formaron parte de nuestra educación profesional.

DEDICATORIA

A mis padres, por todo el apoyo que siempre me brindan, por hacerme sentir tan especial para ellos.

A mi hermana querida por todo su amor.

A Lazaro mi novio por ser además compañero y amigo, gracias por apoyarme en todas mis decisiones y darme fuerzas cuando la necesitaba.

A mis abuelos por confiar tanto en mí.

A todas aquellas chicas que han pasado momentos malos y buenos junto a mí (Yayi, Sury, Kaly, Ludmi, kare, Lianet, Susel y todas las niñas del 76201)

A mi abuela Lucinda que aunque ya no está conmigo, siempre la llevo en mi corazón y se sentiría feliz de verme alcanzar mi sueño.

Yilianis.

A Jehová Dios por su infinito amor y sabiduría. Y hacer de mí el hombre que soy hoy.

A mi abuela por su gran amor, siempre apoyarme y nunca perder la fe en mí hasta en los peores momentos.

A mi mamá y mi hermana porque son una parte esencial de mi vida, y ser junto con mi abuela las tres personas que más amo en esta tierra.

En general a toda mi familia por su ayuda y ser el objetivo principal de mis metas.

A todos mis grandes amigos por su cariño y saber siempre estar presentes cuando los necesité.

En fin a todas las personas que de una forma u otra tuvieron la amabilidad de utilizar una porción de su tiempo, esfuerzo y amor para dedicármelo a mí. Siempre estaré infinitamente agradecido hasta por los detalles más pequeños. Roberto.

RESUMEN

Los yacimientos lateríticos proporcionan la materia prima fundamental de la Industria Cubana del Níquel, su descubrimiento y aprovechamiento es de gran relevancia por los beneficios económicos que estos representan para el país. Por esta razón el Centro de Investigación del Níquel está interesado en realizar investigaciones en diferentes terrenos de la zona Oriental para localizar los yacimientos que sean rentables económicamente para su explotación. Teniendo esto en cuenta, geólogos cubanos han desarrollado un modelo Markoviano, que a partir de los datos que arroja una red de exploración en un área determinada predice si es factible realizar excavaciones en ella. La implementación de este modelo con un algoritmo secuencial tiene un costo temporal que no es lo suficientemente rápido para incorporarlo al proceso industrial. Además, la implementación actual trae consigo un gasto de memoria tan alto que imposibilita que el modelo se calcule con mucha más precisión. Por lo que para poder obtener algún resultado viable se ha hecho necesario diseñar un algoritmo paralelo que optimice este proceso de cálculo y además optimice el uso de la memoria. Habiendo obtenido los resultados del algoritmo en un tiempo significativo trae como ventaja a los geólogos una rápida toma de decisión en cuanto a la explotación del yacimiento. Y el haber hecho un uso eficiente de la memoria trae consigo una mayor calidad y exactitud de las predicciones en cuanto a la existencia del Níquel y la optimización de la red de exploración.

PALABRAS CLAVE

Algoritmo Paralelo: Son aquellos algoritmos que se implementan mediante un cluster para agilizar el procedimiento de cálculo.

Cluster: Conjunto de maquinas conectadas por una red con determinadas características.

Menas: Veta de substancia mineral y mineral mismo.

Meteorización: Destrucción de las rocas y minerales cercanos a la superficie por fuerzas exógenas. Depende del clima.

Lateritas: Suelo característico de las regiones tropicales, alternativamente húmedas de color rojizo o amarillento por el óxido de hierro que contiene.

Regolito: Conjunto de materiales producto directo de la meteorización de un sustrato. Se trata de un conjunto de materiales relativamente homogéneo, formado por los fragmentos de la roca original

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

TABLA DE CONTENIDOS

INTRODUCCIÓN

El níquel constituye uno de los metales más empleados en la vida moderna, debido a que contiene importantes cualidades, como son: resistencia a la corrosión, conductividad térmica y eléctrica, así como una gran estabilidad de volumen ante los cambios de temperatura. Es un elemento que se encuentra en el ambiente sólo en muy pequeños niveles y es utilizado en diferentes aplicaciones, la más común es como ingrediente del acero y otros productos metálicos, además se le puede hallar en productos metálicos de joyería.

El presente trabajo colabora con el Centro de Investigación del Níquel de Moa para facilitar la investigación y aprovechamiento de los recursos de Níquel que brinda esa zona Oriental, específicamente se han puesto en práctica redes de exploración para determinar mediante un método probabilístico si es económicamente factible realizar la explotación de Níquel. Para optimizar el proceso se hace necesario el uso de un algoritmo computacional que haga los cálculos y brinde los resultados en un tiempo viable, y con una precisión adecuada.

Se han propuesto algunos algoritmos secuenciales para su obtención, y aunque el desempeño mejora, son de complejidad cuadrática respecto al tamaño del origen de información.

En [10] se consideró bajar, cuidando en lo posible la eficacia mínima requerida para una descripción adecuada del yacimiento, la cardinalidad de las variables que definen el modelo persiguiendo reducir la complejidad tanto en tiempo como en espacio. El tiempo para $n = 28511$ fue de 17 horas 10 minutos y 16 segundos, en una PC dedicada con procesador Intel Core 2 Duo a una velocidad de 2.66 GHz y una memoria RAM de 2 GB.

Luego en [8] se realizaron optimizaciones basadas fundamentalmente en el desenrollado de ciclos y los tiempos fueron mejorados, para ese mismo tamaño de la entrada y PC, hasta 7 horas 43 minutos y 16 segundos.

Si se tiene en cuenta que $n = 28511$ puede considerarse pequeño respecto al tamaño del problema que genera comúnmente la modelación de yacimientos lateríticos reales, la cual, aunque depende de la complejidad del yacimiento a modelar, precisa en general cientos de miles de muestras incluso cantidades mayores si se combinan para un mismo yacimiento los registros de múltiples muestreos, se puede

apreciar como la ejecución de este algoritmo en una máquina convencional no es viable en términos prácticos.

Buscando una mejora en el tiempo de obtención del modelo Markoviano se desea diseñar un algoritmo paralelo que permita obtener los resultados y tomar las decisiones pertinentes en un periodo más corto de tiempo, con un óptimo manejo de memoria para una mejor precisión en los resultados. Esto traerá como beneficio para el Centro de Investigación la capacidad de realizar muchos más estudios más confiables en menos tiempo.

Teniendo en cuenta el problema planteado se define como **objeto de estudio**: Modelo matemático para el estudio de yacimientos lateríticos.

Por lo que se especifica el siguiente **campo de acción**: Algoritmo para el cálculo de matrices muestrales en modelos Markovianos**.**

Objetivo General.

Desarrollar un algoritmo paralelo para la modelación de yacimientos lateríticos a partir del modelo planteado en [10] y usado en la optimización de redes de exploración del níquel.

Objetivos Específicos.

- ✓ Diseñar el algoritmo paralelo para la modelación de yacimientos lateríticos
- ✓ Implementar el algoritmo paralelo.
- ✓ Evaluar experimentalmente el algoritmo.

Para lograr dichos objetivos se plantearon las siguientes **tareas**:

- ✓ Búsqueda Bibliográfica para lograr una base teórica de los conocimientos de algoritmos paralelos.

- ✓ Identificar las partes del modelo que pueden ser paralelizadas.
- ✓ Describir formalmente el conjunto de pasos paralelos para resolver el problema.
- ✓ Identificar e implementar las estructuras de datos fundamentales para la implementación.
- ✓ Implementar el algoritmo paralelo para calcular el modelo Markoviano.
- ✓ Realizar pruebas de ejecución para determinar el rendimiento del algoritmo implementado.

Idea a Defender

Si se realiza la implementación del algoritmo paralelo con una manipulación óptima de memoria para calcular un modelo Markoviano se podrán agilizar y desarrollar las investigaciones en menor tiempo y con buena precisión.

El documento consta de 3 capítulos:

Capítulo 1. Fundamentación Teórica:

Se exponen conceptos necesarios para el entendimiento de algoritmos paralelos y modelos Markovianos para una mejor comprensión del software y se muestran los sistemas vinculados al campo de acción. Se realiza una caracterización del modelo Markoviano aplicado. Incluye además un estudio de las estrategias de evaluación de los algoritmos paralelos y de las métricas que se usan para medir su rendimiento en cuanto a tiempo.

Capítulo 2. Descripción y análisis de la solución propuesta:

Se describe el algoritmo para una mejor comprensión. Se realiza un análisis de los materiales y métodos a utilizar.

Capítulo 3. Resultados y Discusión de la solución propuesta: Se valida la solución propuesta a través de la realización de pruebas. Se realiza una descripción de estas teniendo en cuenta objetivo, alcance, tipo de prueba y detalles de la misma.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción.

En el presente capítulo se muestra la fundamentación teórica necesaria para una mejor comprensión del tema a tratar en el trabajo de diploma. Se analizaran los conceptos básicos, se introducirá el tema de los yacimientos lateríticos viendo de esta forma la importancia que tiene en nuestro país la exploración de estos yacimientos, y la utilización del método adecuado a la hora de realizar dicha exploración. Además se expone el modelo matemático que se implementa en este trabajo y se da una breve explicación de cuáles son las formas de evaluación de los algoritmos paralelos conjuntamente con las métricas de medición de rendimiento temporal.

1.1: Yacimientos lateríticos.

1.1.1: Introducción a los yacimientos lateríticos.

“La industria del níquel y el cobalto es una de las fuentes de ingreso más importantes de Cuba; se nutre de las menas procedentes de varios yacimientos de cortezas lateríticas, minadas a cielo abierto.
Desafortunadamente los mejores depósitos han sido prácticamente agotados, aún así, las empresas involucradas en esta industria pretenden aumentar los volúmenes de producción, por lo que se enfrentan al reto de explotar con eficiencia yacimientos más complejos, menos potentes y más variables”.[9]
Estos yacimientos lateríticos producen actualmente gran parte del Níquel mundial. Sin embargo, el conocimiento sobre la distribución de Ni en las distintas fases minerales presentes en estos depósitos se conocen solo a un nivel relativo y de poco detalle composicional y estructural.
”Se describe el proceso de lateritización como la meteorización química que tiene lugar en clima húmedo, durante largos periodos de tiempo y condiciones geológicas relativamente estables, que permiten la formación de un regolito potente, con características distintivas.” [3]
“Se plantea que las lateritas ricas en níquel y cobalto son el producto de la meteorización intensa de rocas ultramáficas en la superficie terrestre, bajo condiciones climáticas húmedas. El resultado es el perfil laterítico formado por capas o estratos de material meteorizado sobreyaciendo la roca madre. En el perfil, las capas inferiores muestran los estadios más tempranos de su formación.”[7]

	Fe	MgO	Ni	Co
Ferricreta	>50%	<0.5%	<0.6%	<0.1%
"Limonita"	40-50%	0.5-5%	0.8-1.5%	0.1-0.2%
Saprolita	10-25%	15-35%	1.5-3%	0.02-0.1%
Saprock o roca alterada	5%	35-45%	0.3%	0.01%
Roca madre				

Figura 1.1: Perfil laterítico típico y contenidos promedio de elementos químicos, [1]

"Clasificación general de las lateritas, en la que se destacan tres grupos principales:

- Lateritas oxidadas (compuestas fundamentalmente por óxidos e hidróxidos de hierro en la parte superior del perfil sobreyaciendo las rocas frescas y alteradas).
- Lateritas arcillosas (compuestas fundamentalmente por arcillas esmectíticas en la parte superior del perfil).
- Lateritas silicatadas (compuesta fundamentalmente por silicatos de Mg-Ni en la parte más profunda del perfil, sobreyacidas por lateritas oxidadas)."[1]

Los yacimientos de la región de Moa, en su conjunto, constituyen un ejemplo típico de este tipo de perfil, pero no se descarta la posible existencia local de otros tipos de lateritas.

El yacimiento Moa Oriental, que es donde se aplica el modelo que se implementó, es del tipo laterítico-saprolítico o de perfil completo, según la clasificación cubana; oxidado [1], y ecuatorial húmedo, sobre rocas ultramáficas altamente serpentinizadas en áreas bien drenadas [4]. Se encuentra ubicado dentro del

bloque morfotectónico El Toldo [2] . Dicho bloque es uno de los más extensos de la región, posee valores máximos de ascenso relativo y está formado por rocas ultramáficas y máficas de la secuencia ofiolítica.

1.2: Proceso de Exploración del Níquel.

1.2.1: Descripción del Proceso de Exploración del Níquel.

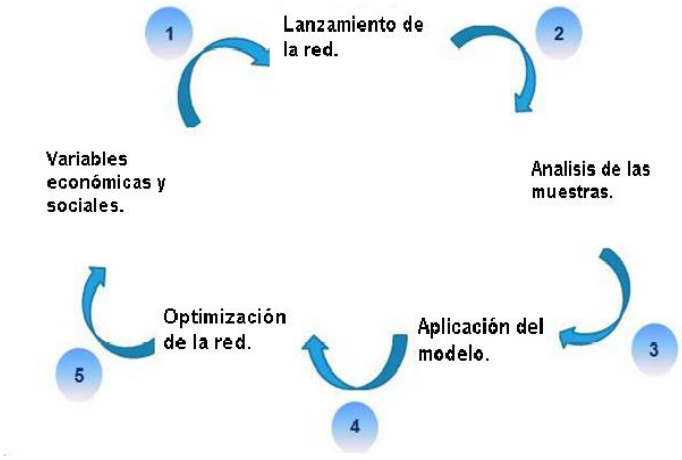


Figura 1.2: Diagrama de flujo de optimización de redes de exploración.

Como se muestra en la figura 1.2 el proceso de exploración está definido por 5 pasos, en el paso 1 es donde se procede al lanzamiento de la red de exploración y acto seguido se toman muestras del terreno. En el paso 2 estas muestras se llevan a un laboratorio y se analizan, de donde se toman los valores que interesan de ellas. En el paso 3 se le aplica un análisis matemático a estos valores para la obtención de un modelo del yacimiento. Con este modelo en el paso 4 se optimiza la red, obteniendo cual es el área dentro de la red que más probabilidad tiene de que exista Níquel. Este resultado en el paso 5 se analiza en conjunto con otras variables de tipo económicas, sociales, entre otras, para saber si es posible y factible continuar explorando, de arrojar un resultado positivo se repite el proceso hasta que ya se tenga un nivel de certeza adecuado para comenzar la explotación. En caso contrario se termina el proceso, por lo que es necesario hacer notar que todos los pasos tienen que hacerse en un tiempo lo suficientemente pequeño ya que los otros dependen de él, y es entonces de vital importancia que la implementación del modelo Markoviano que se propone tenga como principal prioridad que dé el resultado en un tiempo

óptimo. Además, del paso de modelación depende la optimización de la red, por lo que es muy importante que el modelo se calcule con la mayor precisión posible.

1.2.2: Redes de Exploración.

"Las redes de exploración" [6] son perforaciones en forma de malla que se realizan en el área que se quiere investigar, estas perforaciones se hacen a una profundidad de 52 m, y a una distancia entre ellas de 33.33 m como muestra la figura 1.3.

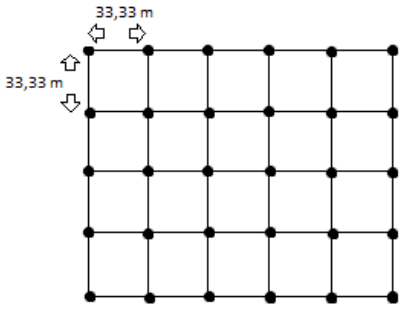


Figura 1.3: Red de Exploración del Níquel.

En cada perforación se toman muestras por cada metro extraído, y con estas se forman registros de la forma que muestra la figura 1.4, donde Bloque es el bloque al que pertenece el pozo donde se está perforando, Pozo es el número del pozo, Profundidad es a la distancia que se tomó la muestra con respecto al nivel del suelo y la Clase es el mineral que se observó en la muestra.

Bloque	Pozo	Profundidad	Clase

Figura 1.4: Formato de un Registro.

El conjunto de estos registros representan la entrada del modelo matemático que se ha implementado.

1.2.3: Modelación de los Yacimientos Lateríticos.

Existen tres enfoques principales a la hora de modelar un yacimiento laterítico[1]

1. Modelado geólogo-genético.
2. Modelado geométrico.
3. Modelado matemático o de bloques.

Estos pretenden representar fielmente la realidad del yacimiento, lográndose la estimación y simulación del fenómeno analizado de una forma lo suficientemente eficaz.

Respondiendo a las necesidades del Centro de Investigación del Níquel del municipio de Moa, [11] propone un modelo matemático de tipo probabilístico que entra dentro del enfoque 3. El mismo se basa en la existencia de clases de minerales en dichos yacimientos, donde se asume que el conjunto de estas clases contiene todas las variedades presentes en ellos, y representa el espacio muestral de todas las clases presentes en el yacimiento. Se utilizan matrices de transiciones, que no son más que matrices de 2 dimensiones donde en cada posición $p_{i,j}$ se representa la probabilidad que existe de que el elemento j ocurra después de conocer que el elemento i halla ocurrido; éstas matrices se utilizan para representar la relación entre las clases de minerales y forman las matrices muestrales; con ellas se calcula el modelo como muestra la siguiente ecuación:

$$\chi(\boldsymbol{t}, r) = \sum_{i=1}^{n} \pi(i) \cdot \rho(\boldsymbol{t}, r)$$

$\chi(\boldsymbol{t}, r)$ es una matriz de 6 dimensiones (que constituye la salida del algoritmo paralelo), esta se obtiene de la unión de todas las matrices de transiciones $\pi(i)$, donde existe una de dichas matrices por cada combinación de otras 4 variables $\rho(\boldsymbol{t}, r)$ que son: la mayor profundidad a que se tomaron las muestras, el paso (que es la menor distancia, de Norte a Sur o de Este a Oeste, que hay entre las 2 muestras donde se apreciaron esas clases, y si están en el mismo pozo ambas muestras, es la diferencia de alturas), el sector donde cae el ángulo horizontal y el sector del ángulo vertical entre las posiciones geográficas donde se tomaron las muestras.

1.2.4: Ventajas del modelo Markoviano Probabilístico.

En el modelo que se propone se ha logrado el reconocimiento de patrones, los procesos aleatorios y la optimización para formar un modelo probabilístico sobre la base de todas las variedades posibles del yacimiento que forman el espacio muestral.

El modelo optimiza la selección de los puntos de muestreo teniendo en cuenta la información de la distribución de dichas clases a lo largo del yacimiento.

Los modelos Markovianos son algunas de las herramientas más poderosas disponibles para los ingenieros y científicos para el análisis de sistemas complejos. Los rendimientos de los resultados en el análisis de ese tiempo y la evolución, dependen del sistema y su estado de equilibrio.

Se dice que un proceso es de Markov cuando verifica la propiedad de Markov: la evolución del proceso depende del estado actual y del próximo, y no de anteriores o posteriores.

1.3: Algoritmos paralelos.

Un algoritmo paralelo es una descripción que permite resolver determinado problema usando múltiples procesadores, o sea, especificar el conjunto de pasos que pueden ser ejecutados simultáneamente en un mismo instante de tiempo, para finalmente unir todas las partes y obtener un resultado óptimo, estos tienen gran importancia ya que se permiten realizar en un corto periodo de tiempo tareas muy grandes utilizando la paralelización, lo cual representa un gran beneficio a la hora de realizar el trabajo.

Posibles pasos a utilizar.

En la literatura “Introducción a la Computación Paralela” [5] se plantean algunos pasos a tener en cuenta a la hora de diseñar estos algoritmos:

- ✓ Identificar porciones de trabajo que pueden ser ejecutadas concurrentemente.
- ✓ Asignar piezas concurrentes de trabajo a los procesadores que correrán en paralelo.
- ✓ Distribuir los datos de entrada, intermedio y de salida asociados con el programa.
- ✓ Administrar el acceso a los datos compartidos por los múltiples procesadores.
- ✓ Sincronizar los procesadores durante las etapas de ejecución del programa paralelo.

Para hacer un buen uso de los algoritmos paralelos es necesario tener en cuenta los siguientes aspectos:

- ✓ Se debe contar con computadores paralelos que se puedan escalar a un número grande de procesadores y que sean capaces de soportar comunicación rápida, así como compartir datos entre procesadores.
- ✓ Diseñar algoritmos paralelos eficientes. Las metodologías de diseño de este tipo de algoritmos pueden resultar radicalmente diferentes de las utilizadas en el diseño de los correspondientes algoritmos secuenciales, ya que es necesario cuidar los aspectos de balance de carga, distribución de datos, división efectiva de tareas, minimización de tiempo de comunicación y de acceso a memoria.
- ✓ Para poder evaluar las prestaciones del sistema que resulta de implementar algoritmos paralelos sobre computadoras paralelas, es necesario poder medir qué tan rápido puede resolverse un problema utilizando el sistema paralelo, qué tan eficientemente se utilizan los procesadores de la computadora paralela, etc.
- ✓ Es necesario, para la implementación de los algoritmos paralelos, contar con lenguajes de programación adecuados para ello.

En los últimos años, ha surgido un nuevo concepto de computador paralelo constituido por redes de estaciones de trabajo u ordenadores personales, dando lugar a los denominados cluster. La idea es realmente simple: las actuales estaciones de trabajo y ordenadores personales están dotados de los mismos procesadores, o muy similares, que los grandes sistemas. Además, el gran esfuerzo de investigación y desarrollo que se realizó tiempo atrás para conseguir redes de comunicación de alta velocidad está dando sus frutos hoy en día.

1.4: Altas prestaciones y Paralelismo.

La computación de altas prestaciones es el campo de trabajo en el que se estudian el conjunto de técnicas necesarias para obtener mejores prestaciones.

Las aplicaciones de la computación de altas prestaciones y en concreto el paralelismo se extiende prácticamente a todos los ámbitos donde la programación se manifiesta como útil. En la actualidad, la computación paralela está siendo utilizada en diversos campos para el desarrollo de aplicaciones y el estudio de problemas que requieren gran capacidad de cómputo, bien por el gran tamaño de los problemas que abordan o por la necesidad de trabajar con problemas en tiempo real. De esta forma, el

paralelismo en la actualidad, además de constituir varias líneas abiertas de intensa labor investigadora, puede encontrarse en infinidad de aplicaciones en campos muy variados.

En muchas aplicaciones científicas, la formulación del problema mediante un modelo matemático y su tratamiento numérico requiere la resolución de un sistema de ecuaciones, normalmente de gran tamaño.

Teniendo en cuenta las grandes dimensiones del problema a resolver, la utilización de métodos directos se hace prácticamente inviable, por lo que se hace necesario recurrir a métodos iterativos, los cuales pueden proporcionar una solución tan satisfactoria como los métodos directos, reduciendo los errores de redondeo. Además, la resolución directa de los sistemas no lineales, al margen del tamaño de los mismos, es impracticable en la inmensa mayoría de los casos debido al carácter no lineal de las ecuaciones. Por otro lado, para sistemas de orden elevado, la implementación de métodos iterativos en máquinas secuenciales presenta serios problemas que incluso, en determinados casos, pueden hacerla inviable: entre éstos se destacan, problemas de memoria debido a que la memoria disponible en una sola máquina puede ser insuficiente para atender las necesidades de cálculo y almacenamiento de datos requeridos durante la ejecución de los programas, y el tiempo de ejecución resultan excesivamente elevados. Una forma de subsanar las limitaciones de las máquinas secuenciales es el procesamiento paralelo.

Por estos motivos, la aparición de computadoras paralelas ha originado un ajuste de los algoritmos clásicos de la computación matricial para su implementación en dichas máquinas. Además, ha dado lugar a nuevas líneas de investigación dirigidas a diseñar nuevos métodos que obtengan un buen rendimiento en las máquinas paralelas.

Los distintos tipos de arquitecturas paralelas a utilizar, convierten el diseño de las aplicaciones de computación de altas prestaciones en un arte, cuando lo que se pretende es optimizar el rendimiento del tiempo empleado en un cálculo concreto. En este arte, los fundamentos de diseño de aplicaciones paralelas, constituyen la base para abordar con éxito la implementación de un proyecto de computación de altas prestaciones.

Después de muchos años en los que la construcción de aplicaciones de altas prestaciones implicaba el conocimiento de un lenguaje o biblioteca de funciones específicas, hoy en día se cuenta con el Message Passing Interface (MPI). Este se ha convertido en un estándar, que permite la construcción de aplicaciones ejecutables en distintos entornos de multicomputadores, aprovechando de forma transparentes las características diferenciales de las distintas arquitecturas.

1.5: Estrategias de Evaluación.

Para la evaluación y medición del rendimiento de los algoritmos paralelos se tienen en cuenta las siguientes estrategias [5]:

Evaluación experimental

- ✓ Requiere la disponibilidad del hardware y la implementación del algoritmo.
- ✓ No permite especular sobre el comportamiento del algoritmo ante modificaciones de la máquina.

Evaluación mediante simulación

- ✓ Requiere una especificación detallada de muchos detalles de la máquina y del algoritmo.
- ✓ Puede ser lento.

Evaluación mediante modelo analítico

- ✓ La construcción del modelo puede ser difícil, si se desea mucha precisión.
- ✓ Es rápido.
- ✓ Facilita la comparación de algoritmos.

1.5.1: Métricas para medir el rendimiento de los algoritmos paralelos.

En cada una de las métricas utilizadas para la medición del rendimiento de los algoritmos paralelos se toma como consideración que p es el número de procesadores que se utilizan para ejecutar el algoritmo y n es el tamaño de entrada.

Tiempo de ejecución de un algoritmo paralelo:

Denotado por la función

$T(n,p) = T_A(n,p) + T_C(n,p)$

Donde:

$T_A(n,p)$, es el tiempo empleado por el algoritmo para realizar operaciones aritméticas.

$T_C(n,p)$, es el tiempo empleado por el algoritmo para realizar comunicaciones entre procesadores.

Ganancia en velocidad de ejecución o aceleración:

Mide la ganancia de velocidad o aceleración del algoritmo paralelo respecto al mejor algoritmo secuencial correspondiente, y está dado por:

$S(n,p) = \frac{TS_{mejor}(n)}{T(n,p)}$.

Donde $TS_{mejor}(n)$ es el tiempo del mejor algoritmo secuencial para resolver el problema.

Eficiencia:

Mide el porcentaje del tiempo que los procesadores se mantienen útilmente empleados. La eficiencia se define así:

$E(n,p) = \frac{S(n,p)}{p}$.

Escalabilidad:

Es la capacidad de un determinado algoritmo de mantener sus prestaciones cuando aumenta el número de procesadores y el tamaño de la entrada, dicho de otra manera, si cuando aumentan ambos la eficiencia se mantiene constante o aumenta entonces el algoritmo es escalable. Indica la capacidad del algoritmo de utilizar de forma efectiva un incremento en los recursos computacionales.

1.6: Conclusiones Parciales.

Mediante la realización de este capítulo se ha logrado desarrollar los objetivos que se fijaron para el mismo. Se logró establecer un concepto preciso de lo que se considera es un yacimiento laterítico, se enunciaron algunas de las formas en las que pueden ser clasificadas las laterítas, se analizó el proceso que se lleva a cabo a la hora de realizar el proceso de exploración del Níquel, así como una descripción de lo que consiste una red de exploración. Se describió el modelo matemático a utilizar y se analizaron las formas de evaluar los algoritmos paralelos. Por todo lo antes planteado se espera haber cumplido con las métricas planteadas en el resumen de este capítulo.

CAPÍTULO 2: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

Introducción.

En este capítulo se hace una descripción de la solución de nuestro problema y además se realiza un análisis de nuestro algoritmo a partir de su pseudocódigo haciendo posible su total comprensión.

2.1: Propuesta de solución.

2.1.1: Descripción del Funcionamiento del algoritmo.

Como se mostró en la descripción del modelo matemático que se implementa, el resultado consiste en una matriz de 6 dimensiones que se obtiene como resultado de la unión de todas las matrices de transición que se forman por las comparaciones que se hacen entre cada par de muestra, se sabe que estas muestras vienen representadas por un conjunto de registros almacenados en un vector como se muestra en la figura 2.1.

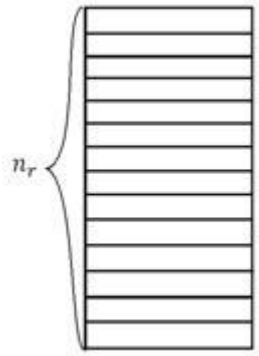


Figura 2.1: Conjunto de registros almacenados.

De este análisis se desprende que el problema principal del algoritmo es realizar las comparaciones entre cada par de registros, si se toma que la cantidad de registros almacenados en el vector es n_r y que la cantidad de procesadores con que se cuenta es igual a p, entonces se puede calcular que para cada procesador la cantidad de registros de los que tiene que encargarse para comparar con los otros es igual a $\frac{n_r}{p}$, formando éstos T_i, que no es más que la tarea que le corresponde al procesador $i\ (0 \leq i < p)$. Para que sea posible un equilibrio de la carga de trabajo entre los procesadores (lo cual permite una mayor velocidad de cálculo) se distribuyeron los registros para formar cada tarea como muestra la figura 2.2,

donde el $tramo_1$ de cada tarea tiene una cantidad de registros igual a $\frac{\left(\frac{n_r}{p}\right)}{2}$ y el $tramo_2$ tiene una cantidad de registros igual a $\frac{n_r}{p} - \frac{\left(\frac{n_r}{p}\right)}{2}$.

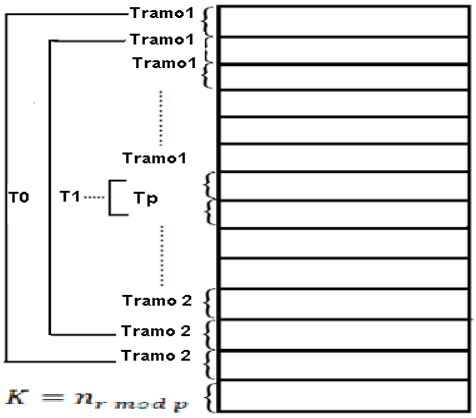


Figura 2.2: Distribución de los registros.

Como cada procesador tiene que realizar sus propias comparaciones, entonces cada uno de ellos tiene su propia matriz para realizar las modificaciones, y éstas se hacen luego de calcular el índice resultante de la comparación entre registros, se aumenta en una unidad la posición que representa ese índice en la matriz. Después que cada procesador haya realizado todas las comparaciones correspondientes, entonces procede a enviar sus resultados al procesador raíz que es el procesador p_0, éste une todos los resultados en su propia matriz y después imprime el resultado final en un fichero para su posterior utilización.

2.1.2: Análisis del Algoritmo.

```
En paralelo para k = 0 hasta p - 1

Entrada
      numeroRegistros : Entero
      registros : Real[][]

Precondiciones
      cantidadRegistrosPorProceso : Entero
      primerRango : Entero
      segundoRango : Entero

Constantes
      procesoRaiz ←  0

inicio
      cantidadRegistrosPorProceso ←  numeroRegistros / p
      primerRango ←  cantidadRegistrosPorProceso / 2
      segundoRango ← cantidadRegistrosPorProceso  - primerRango

      Algoritmo(matriz, registros, numeroRegistros, k * primerRango, primerRango)
      Algoritmo(matriz, registros, numeroRegistros,
                      p * primerRango + (p - k - 1) * segundoRango, segundoRango)

      si k != procesoRaiz y k <=  numeroRegistros mod N entonces
            algoritmo(matriz, registros, numeroRegistros, numeroRegistros - k, 1)
      fin si

      si k != procesoRaiz entonces
            EnviarMatriz(matriz, procesoRaiz)
      si no entonces
```

```
            para i ← 1 hasta p hacer
                  RecibirMatriz(matrizTemporal, i)
                  matriz ← sumarMatrices(matriz, matrizTemporal)
            fin para
            retornar matriz
      fin si
fin



algoritmo(matriz , registros : Real[][], numeroRegistros : Entero,
                        posicionEmpezar : Entero, rango : Entero)
Variables
      indicesOrigenDestino : Entero[]
      indicesDestinoOrigen : Entero[]
inicio
      para i ← posicionEmpezar hasta posicionEmpezar + rango hacer
            para j ← i + 1 hasta numeroRegistros - 1 hacer

                  indicesOrigenDestino ← CompararRegistros(registros[i], registros[j])
                  indicesDestinoOrigen ← CompararRegistros(registros[j], registros[i])

                  si validarIndices(indicesOrigenDestino) entonces
                        incrementar(matriz, indicesOrigenDestino, 1)
                        incrementar(matriz, indicesDestinoOrigen, 1)
                  fin si
            fin para
      fin para
fin


CompararRegistros (registroOrigen : Real[], registroDestino : Real[]) : Entero[]
Constantes
      registroPosBloque ← 0
      registroPosPozo ← 1
      registroPosProfundidad ← 2
```

```
      registroPosCaracteristica ← 3

      indPosCaracteristicaOrigen ← 0
      indPosCaracteristicaDestino ← 1
      indPosPaso ← 2
      indPosProfundidad ← 3
      indPosSectorAnguloHorizontal ← 4
      indPosSectorAnguloVertical ← 5
Variables
      pozoOrigen : Pozo
      pozoDestino : Pozo
      indicesResultado : Entero[]
inicio
      indicesResultado[indPosCaracteristicaOrigen] ←
registroOrigen[registroPosCaracteristica]
      indicesResultado[indPosCaracteristicaDestino] ←
registroDestino[registroPosCaracteristica]

      indicesResultado[indPosProfundidad] ←
max(registroOrigen[registroPosProfundidad],
                                 registroDestino[registroPosProfundidad])

      pozoOrigen ← crearPozo(registroOrigen[registroPosBloque],
registroOrigen[registroPosPozo])
      pozoDestino ← crearPozo(registroDestino[registroPosBloque],

registroDestino[registroPosPozo])


      indicesResultado[indPosSectorAnguloHorizontal] ←

CalcularSectorAnguloHorizontal(pozoOrigen, pozoDestino)
      indicesResultado[indPosSectorAnguloVertical] ←

CalcularSectorAnguloVertical(pozoOrigen, pozoDestino)
```

```
      si pozoOrigen == pozoDestino entonces
            indicesResultado[indPosPaso] ←
modulo(registroOrigen[registroPosProfundidad]  -
                                  registroDestino[registroPosProfundidad])
      si no entonces
           indicesResultado[indPosPaso] ← calcularPaso(pozoOrigen, pozoDestino)
      fin si
      retornar  indicesResultado
fin
```

En el pseudocódigo anterior no se tuvo en cuenta el tipo de dato que tiene la matriz pues en este trabajo se consideran 3 variantes a la hora de representar la matriz con alguna estructura de datos, y puede quedar abierto a utilizar alguna otra diferente.

En la función CompararRegistros(registroOrigen, registroDestino) donde se comparan los registros cada pozo se forma por su posición geográfica en el terreno, que viene dada por el número del bloque al que pertenece (que es de 4 cifras) y la posición de él dentro de ese bloque (que es de 2 cifras), y su posición global se calcularía de la siguiente forma: Si el número del bloque es ABCD y su posición dentro del pozo se da por el número EF, entonces la posición global del pozo es el número ABECDF, con esa posición se calcula su desplazamiento de Norte a Sur y su desplazamiento de Oeste a Este. El sector del ángulo horizontal que se obtiene en la función CalcularSectorAnguloHorizontal(pozoOrigen, pozoDestino) no es más que cuando se calcula el ángulo horizontal del pozo destino con respecto al pozo origen tomando como referencia el nivel del suelo y el Norte como el ángulo cero, este se ubica en uno de los sectores angulares en que se divide el círculo alrededor del pozo Origen (figura 2.3), la cantidad de sectores es variable y depende del nivel de exactitud con que se quiera hacer el modelo, y cada sector tendría un ángulo igual a $\alpha = \frac{2\pi}{n_1}$, donde n_1 sería la cantidad de sectores que se quiere, y siempre los sectores se ubican a partir del ángulo cero y en sentido contrario a las manecillas del reloj.

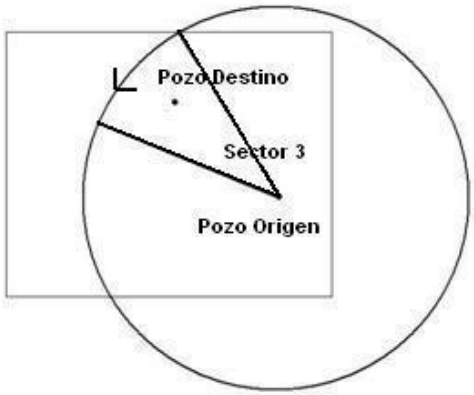


Figura 2.3: Sector del ángulo horizontal.

El sector del ángulo vertical que se obtiene en la función CalcularSectorAnguloVertical(pozoOrigen, pozoDestino), al igual que el anterior tiene en cuenta las posiciones de los pozos, pero además tiene en cuenta la profundidad a que se encuentra la muestra como se puede ver en la figura 2.4, por lo que el plano que se tiene en cuenta para tomar el ángulo vertical es el que une a los 2 puntos en los que se toman las muestras y pasa además por el centro de la tierra, en este plano se traza un círculo alrededor de la muestra origen y se divide en n_2 sectores, donde n_2 depende de la exactitud a la que se quiera calcular el modelo, y procediendo como en el caso anterior se halla el sector en el que se encuentra el ángulo que hay entre la muestra destino y la muestra origen , aquí se tiene en cuenta que el ángulo cero es la superficie del suelo y los sectores se ubican a partir de él en sentido contrario a las manecillas del reloj con una amplitud de $\frac{2\pi}{n_2}$.

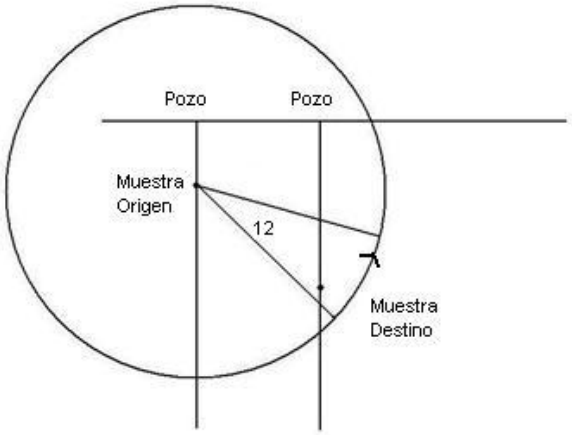


Figura 2.4: Sector del ángulo vertical.

2.1.3: Consideraciones de implementación.

A la hora de implementar el algoritmo se consideraron varias alternativas en cuanto a la estructura de datos utilizada para representar el modelo obtenido para darle más versatilidad a nuestra solución. Si se busca obtener el modelo con el mejor tiempo posible se utiliza una matriz convencional de 6 dimensiones que permite que la modificación se haga con una complejidad $O(1)$, pero esto trae consigo que la memoria tenga un gasto de $4 * \prod_{i=1}^{6} dim_i$ bytes, donde dim_i es la cardinalidad de la dimensión i, y se multiplica por 4 porque la matriz es de enteros y el tamaño de un entero es 4 bytes.

Para tener el menor gasto de memoria se puede utilizar una versión hecha del algoritmo donde se implementa la matriz en forma de un arreglo ordenado, esto se puede hacer cuando la matriz que representa el modelo quede dispersa. Se guardaran los índices de la matriz que son mayores que cero y el valor que representan, si estos valores mayores que cero representan el x porciento del total de elementos de la matriz entonces el gasto de memoria sería $\left(4 * \prod_{i=1}^{6} dim_i\right) * \frac{x}{100} * 2$ bytes, de donde se deduce que solo es aplicable para $x \leq 50$. Pero esta variante trae consigo que aumente el tiempo de ejecución ya que las modificaciones tendrían complejidad $O(\log n)$ y las inserciones $O(n)$.

Una variante mucho más equilibrada que las dos anteriores, es decir, que tenga un gasto de memoria menor que la primera variante pero no tan bueno como la segunda, y tenga una complejidad temporal

mejor que la ultima pero no tan bueno como la primera, es en la que se representa la matriz mediante un árbol AVL, que no es más que un árbol binario de búsqueda balanceado. Cada índice mayor que cero se almacena en un nodo del árbol, y además del índice guarda el valor de las veces que se repite, un puntero al hijo izquierdo y otro al hijo derecho, en total serían 4 enteros. Al igual que la variante anterior solo es factible cuando la matriz que representa el modelo queda dispersa, y si se tiene en cuenta que estos valores representan el x porciento del total, el gasto de memoria sería $\left(4 * \prod_{i=1}^{6} dim_i\right) * \frac{x}{100} * 4$, es decir que se puede tener en cuenta cuando $x \leq 25$; y como las operaciones de modificación e inserción en el árbol AVL tienen complejidad $O(\log n)$, entonces se demuestra que es más eficiente temporalmente en comparación con la implementación que utiliza el vector ordenado.

2.2: Herramientas utilizadas.

IDE de desarrollo Code: Blocks.
IDE de desarrollo para el lenguaje C y C++, ligero, multiplataforma, reconoce varios compiladores para dichos lenguajes de programación; el código de la aplicación es escrito completamente por medio de este IDE.

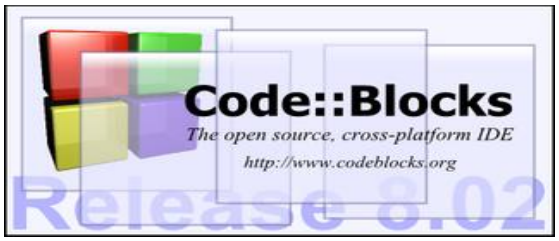


Figura 2.5: Logotipo de IDE de desarrollo Code: Blocks.

MPI (Interfaz de Paso de Mensajes).
La Interfaz de Paso de Mensajes es un estándar que define la sintaxis y la semántica de las funciones contenidas en una librería de paso de mensajes, diseñada para ser usada en programas que exploten la existencia de múltiples procesadores.

Figura2.6: Logotipo de la Interfaz de Paso de Mensajes.

2.3: Conclusiones Parciales.

En el presente capítulo se realizó una validación de la solución propuesta para mejorar el resultado del problema existente en el Centro de Investigación. Para esto se desarrolló un algoritmo paralelo para el modelo matemático probabilístico existente, que mejora el tiempo de ejecución a la hora de mostrar un resultado final en el proceso de exploración del Níquel y además permite que se calcule el modelo con un mayor grado de precisión haciendo un uso eficiente de la memoria.

CAPÍTULO 3: RESULTADOS Y DISCUSIÓN DE LA SOLUCIÓN PROPUESTA

Introducción.

En este capítulo se hace un análisis de la solución con el objetivo de determinar experimentalmente los principales parámetros que rigen el desempeño de un algoritmo paralelo: ganancia de velocidad y eficiencia. Se analiza además el comportamiento de las principales variables a tratar en esta investigación: el tiempo de ejecución en paralelo y el gasto de memoria.

El presente trabajo se basó en la estrategia de evaluación experimental ya que se contaba con un cluster que facilitó el estudio del comportamiento del algoritmo ante varios juegos de datos, esto permitió que se analizara la solución en base a los resultados reales y se pudieran hacer diferentes observaciones sobre la base de los mismos.

3.1: Consideraciones de Tiempo.

En la implementación del algoritmo Markoviano se desarrollaron 3 variantes, diferenciándose éstas en el tipo de estructura de datos que utilizan para el manejo de la matriz que representaba el modelo matemático. Se empieza analizando la complejidad temporal resultante de la corrida de cada variante tanto secuencialmente como en paralelo para diferentes números de procesadores.

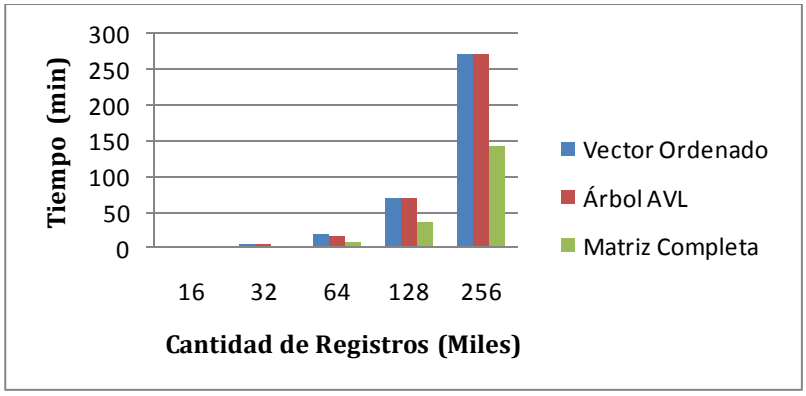


Figura 3.1: Tiempo de Ejecución para el Algoritmo Secuencial.

Se puede observar en el tiempo de ejecución del algoritmo secuencial, que en todas las variantes para un número de registros pequeño el tiempo de respuesta es tratable, pero cuando el número de éstos va aumentando hasta llegar a un valor no muy alto (si se tiene en cuenta que se desea calcular con más de 1 millón de registros) entonces los tiempos se hacen muy elevados sobrepasando las 3 horas para la variante más rápida.

La complejidad temporal del algoritmo secuencial es cuadrática, y todo algoritmo secuencial que se comporte de esta forma cuando el tamaño de entrada se duplica el tiempo se cuadruplica, con esto y queriendo determinar el comportamiento del mejor algoritmo secuencial, tomamos como tiempo base el tiempo que se calculó para el menor número de ficheros $TS_{mejor}(16000) = 0.6$ minutos, y a partir de él se puede determinar la ecuación de crecimiento del tiempo en función del tamaño de la entrada y este es

$$TS_{mejor}(n) \approx 4^{\log_{16000}^{n}} * TS_{mejor}(16000) \approx 0.6 * 4^{\log_{16000}^{n}} \qquad (1)$$

Pudiéndose comprobar que para n igual a 1 millón 24 mil el tiempo necesario para obtener el modelo es de aproximadamente $0.6 * 4^{\log 64} \approx 0.6 * 4^{6} \approx 2457.6$ minutos, lo que representa casi 2 días de cálculo.

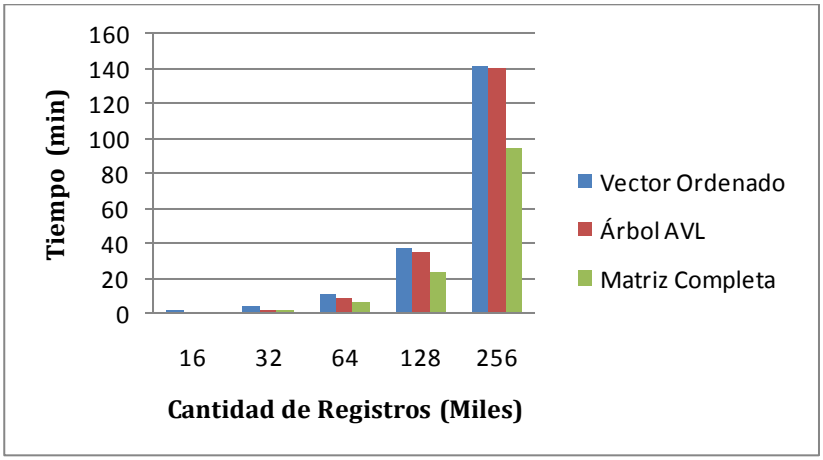


Figura 3.2: Tiempo de Ejecución para el Algoritmo Paralelo con *p=2*.

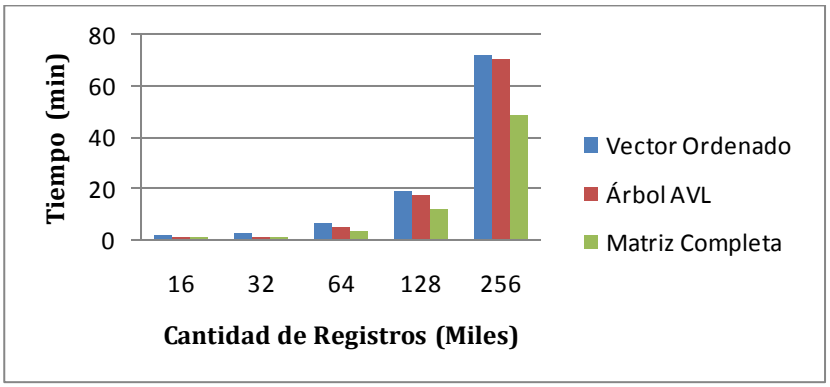


Figura 3.3: Tiempo de Ejecución para el Algoritmo Paralelo con *p=4*.

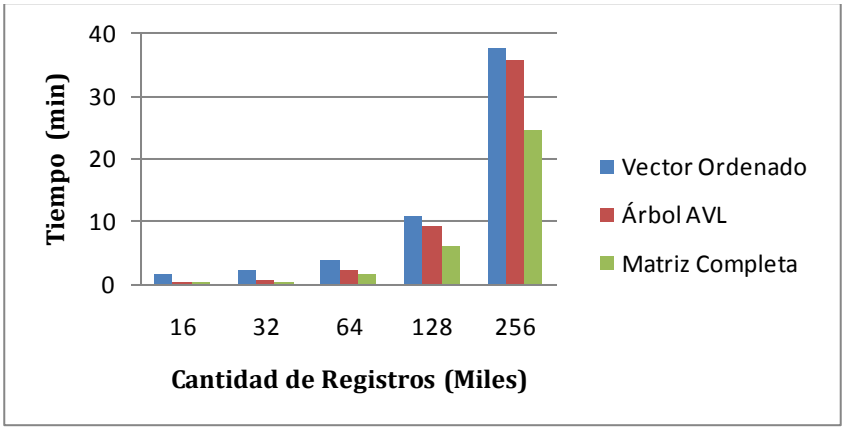


Figura 3.4: Tiempo de Ejecución para el Algoritmo Paralelo con *p=8*.

En cada una de las gráficas de las corridas del algoritmo paralelo que se realizó se pudo observar que el tiempo fue comportándose de manera inversamente proporcional a la cantidad de procesadores que se le asignaban, hecho que refleja que el algoritmo paralelo realiza el modelado en un tiempo mucho menor que la forma secuencial del mismo.

Se puede notar que efectivamente la variante de la matriz completamente almacenada en memoria es la más rápida, y que representar la matriz mediante un árbol AVL hace el algoritmo ligeramente más rápido con respecto a la variante que utiliza un arreglo ordenado para representar la misma, pero con el aumento del número de registros el tiempo de la variante del arreglo ordenado va tendiendo a ser el mismo que el de la variante del árbol AVL. Esto ocurre porque la precisión con que se realizaron los cálculos es pequeña, pero en la gráfica 3.5 se muestra el tiempo para ambos en el cálculo del modelo para una mayor

precisión al aumentar las cardinalidades de los sectores angulares hasta 64 el horizontal y 50 el vertical, y podemos apreciar que la diferencia de tiempo es muy significativa. Lo anterior se debe a la diferencia entre la complejidad temporal de ambos a la hora de insertar un elemento en sus respectivas estructuras de datos, resultando para el arreglo ordenado ser de $O(n)$ y para el árbol AVL $O(\log n)$.

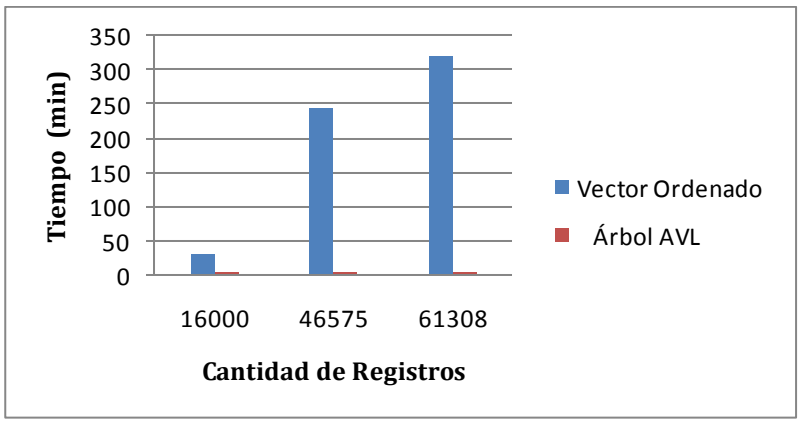


Figura 3.5: Tiempo de Ejecución para el Algoritmo Paralelo con *p=6* y 64 sectores horizontales y 50 verticales.

En las secciones siguientes se analizan los resultados de aplicar diferentes métricas para evaluar el tiempo del algoritmo paralelo. La variante de arreglo ordenado para representar la matriz no es reflejada pues las garantías de memoria que ésta ofrece (ver sección Consideraciones de Memoria) por encima de la variante árbol AVL no compensan el gasto de tiempo que posee. Para una precisión de cálculo pequeña esta variante se pudiera utilizar, pues en estas condiciones la diferencia de tiempo es pequeña y ahorra la mitad de la memoria, pero cuando la precisión del cálculo crece la variante del arreglo ordenado es poco viable.

Ganancia de Velocidad o Aceleración:

Se analiza la aceleración que alcanza el algoritmo mientras se va aumentando el tamaño de entrada, es decir, que tanto más rápido es en relación con el mejor tiempo secuencial que se obtuvo al ejecutar el algoritmo para un mismo tamaño del problema.

Observando el comportamiento de la aceleración de cada variante se calculó un factor q que cumple la condición que:

$$S(n,p) \approx q * p \quad (2)$$

Este factor se calculó dividiendo todas las aceleraciones que resultaron de las pruebas experimentales por p y promediándolas. Con este factor es muy sencillo saber dado un tamaño del problema n, qué cantidad de procesadores se necesitan para resolverlo en un máximo de tiempo paralelo $T(n,p)$, pues si se aplica la fórmula de aceleración y se despeja la aceleración por la fórmula 2 tenemos que:

$$S(n,p) = \frac{TS_{mejor}(n)}{T(n,p)} \approx q * p$$

De donde:

$$p \approx \frac{TS_{mejor}(n)}{T(n,p) * q} \quad (3)$$

Ahora si se quisiera saber el tiempo que tomaría aproximadamente resolver cualquier problema de tamaño n teniendo p procesadores, se despeja de la fórmula 3 y se obtiene que:

$$T(n,p) \approx \frac{TS_{mejor}(n)}{p * q}$$

Es decir, con este factor q se puede saber de antemano datos relevantes para resolver un problema aun sin ponerlo a ejecutarse.

Variante con la Matriz almacenada completamente en memoria:

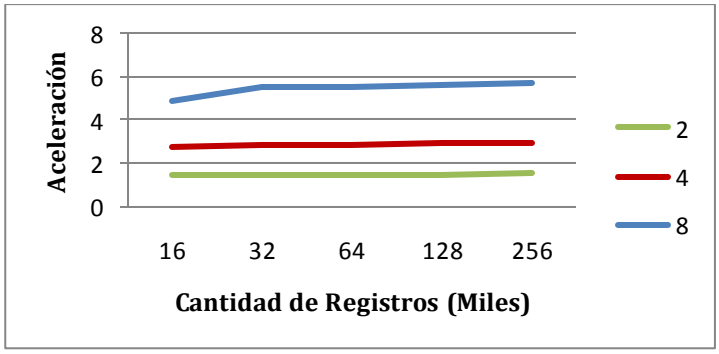


Figura 3.6: Aceleración de la variante matriz completamente en memoria, para diferentes *p*.

De la Figura 3.6 se concluye que esta variante tanto para 2, 4 o 8 procesadores mantiene un alto índice de aceleración, si tenemos en cuenta que la máxima aceleración que puede tener un algoritmo paralelo ejecutado sobre p procesadores es p [5]. Además se puede apreciar que mientras aumenta el tamaño de entrada aumenta también la aceleración, lo cual es muy bueno ya que mientras más grande se hace el problema más el algoritmo gana en velocidad.

Para esta variante el factor $q = 0.7$, con lo que por tomar un ejemplo para calcular el modelo para 1millón 24 mil registros con 8 procesadores el algoritmo paralelo de demoraría $\frac{41}{0.7*8} \approx 7.32$ horas, lo cual representa un tiempo bastante bueno en comparación con el tiempo que daba secuencialmente; por otra parte si se quisiera saber la cantidad de procesadores necesarios para poder calcularlo en 1 hora se despejaría y quedaría que se necesitan $p \approx \frac{41}{0.7*1} \approx 59$ procesadores como mínimo.

Variante Árbol AVL:

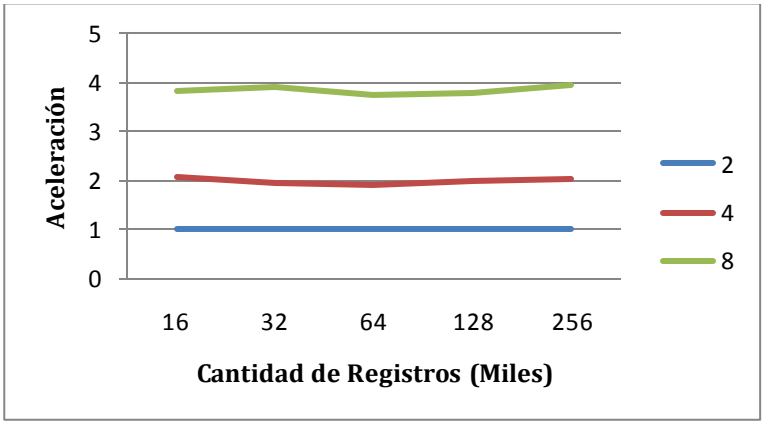


Figura 3.7: Aceleración de la variante que utiliza el árbol AVL, para diferentes *p*.

En esta variante la aceleración se mantiene prácticamente constante en un valor bastante bueno para un número fijo de procesadores, teniendo un factor $q = 0.49$. Hay que tener en cuenta que la aceleración de esta variante se calcula con respecto al tiempo secuencial de la variante que guarda la matriz completamente en la memoria, ya que es la implementación más rápida y por ende la mejor secuencialmente.

Si se quisiera saber también en esta variante qué tiempo tomaría calcular el modelo para 1 millón 24 mil registros de entrada y 8 procesadores, entonces despejando nos quedaría que se demoraría $\frac{41}{0.49*8} \approx 10.5$ horas en arrojar los resultados, que es un tiempo bueno, comparado con las 41 horas que tarda el algoritmo secuencial. Y para conocer la cantidad de procesadores que se necesitan para obtener el modelo en 1 hora utilizando esta variante se obtendría que como mínimo fueran $p = \frac{41}{0.49*1} \approx 84$ procesadores.

Eficiencia:

Se analiza qué tan eficientemente utiliza el algoritmo paralelo los recursos que se le asignan, dígase cantidad de procesadores.

Variante con la Matriz guardada completamente en memoria:

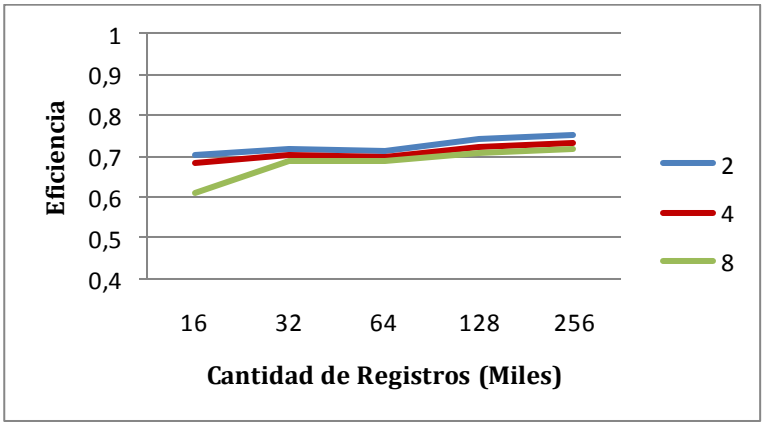


Figura 3.8: Eficiencia de la variante que utiliza la matriz completamente en memoria, para diferentes *p*.

De la observación y análisis de la Figura 3.8 se concluye que esta variante hace un uso muy bueno de los recursos que se le asignan, y mantiene este nivel de eficiencia prácticamente para todos los tamaños de entrada, aumentando ésta mientras aumenta el número de registros con que se evalúa.

Variante del Árbol AVL:

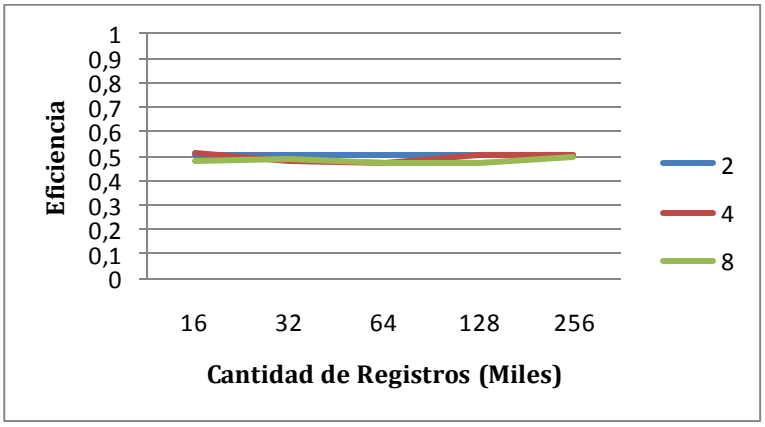


Figura 3.9: Eficiencia de la variante que utiliza el árbol AVL, para diferentes *p*.

De la gráfica representada en la Figura 3.9 se puede ver que la eficiencia de esta variante se mantiene aproximadamente constante para cualquier cantidad de procesadores debido a que la aceleración se mantenía aproximadamente constante también. El que la eficiencia se mantenga en 0.5 se considera bueno ya que esta variante perseguía el objetivo de ahorrar memoria (ver sección 3.2 Consideraciones de Memoria) tratando de afectar lo menor posible el desempeño temporal del algoritmo, y como se puede apreciar si comparamos la eficiencia de esta variante con la eficiencia de la matriz tradicional vista anteriormente, vemos que la diferencia es menor que 3.

La eficiencia en ambas variantes analizadas anteriormente no da un valor mucho mayor, ya que en la implementación del algoritmo, la suma de todas las matrices resultantes de cada proceso paralelo se hace en un solo procesador, esto trae consigo que se tenga que esperar a que termine dicho procesador para que el algoritmo termine, lo que aumenta el tiempo de ejecución en paralelo.

Escalabilidad:

Se analiza si el algoritmo mantiene sus prestaciones cuando se le aumentan los recursos a utilizar y se le aumenta la carga de trabajo, en otras palabras se quiere saber si mantiene sus niveles de eficiencia mientras aumentan el número de procesadores y el número de registros a comparar.

Si se observa la tabla 3.1, en los valores resaltados se puede comprobar que mientras aumentan la cantidad de procesadores y el número de registros, se mantiene aproximadamente constante la eficiencia, por lo que se puede concluir que la variante de la matriz tradicional es escalable.

Como se vio cuando se analizó la eficiencia de la variante del árbol AVL, esta se mantenía aproximadamente constante para cualquier tamaño del problema y cualquier cantidad de procesadores, por la tanto es trivial que es escalable.

	n=16 mil	n=32 mil	n=64 mil	n=128 mil	n=256 mil
p=2	0.70196678	0.71588413	0.71288141	0.73941545	0.75079637
p=4	0.68057894	0.70263835	0.69943367	0.72287924	0.7336839
P=8	0.61088403	0.68828282	0.68582549	0.70497695	0.71823821

Tabla 3.1: Eficiencia del algoritmo usando una matriz tradicional en función de *n* y *p*.

3.2: Consideraciones de Memoria.

En una versión inicial del modelo los autores consideraron que el valor de las cardinalidades de la dimensión del sector angular horizontal y vertical eran 17 y 6 respectivamente, esto calcula el modelo pero con un bajo nivel de precisión. Como para el Centro de Investigación del Níquel es importante aumentar el nivel de exactitud con que se calcula el modelo, y esto se logra aumentando la cardinalidad de las dimensiones mencionadas, pudiendo llegar al valor de 64 sectores, lo que conlleva a un aumento drástico de utilización de memoria, resulta de gran importancia hacer un análisis del gasto de este recurso para las tres variantes.

Gasto de memoria:

Índice de Dispersión	2.45	13.86	14.05
Matriz Completa	46.62	46.62	46.62
Árbol AVL	4.57	25.84	26.20
Vector Ordenado	2.29	12.92	13.10

Tabla 3.2: Gasto de memoria en Mb de las tres variantes con 17 sectores angulares horizontales y 6 verticales.

En la Tabla 3.2 construida a partir de pruebas que se le hicieron a las 3 variantes donde se obtuvieron diferentes índices de dispersión, se puede observar que para 17 sectores angulares horizontales y 6 verticales el gasto de memoria de la variante que almacena la matriz completamente en memoria no es muy alto, pudiéndose calcular con los ordenadores con que cuenta el Centro de Investigación del Níquel, aunque se puede ver claramente que es mucho mayor que el gasto de las otras variantes.

Índice de Dispersión	0.70	5.21	3.34
Matriz Completa	1462.5	1462.5	1462.5
Árbol AVL	41.05	304.65	195.60
Vector Ordenado	20.52	152.32	97.80

Tabla 3.3: Gasto de memoria en Mb de las tres variantes con 64 sectores angulares horizontales y 50 verticales.

En la Tabla 3.3 se observa que cuando los valores de las cardinalidades de las dimensiones de los sectores angulares aumentan a 64 el horizontal y 50 el vertical, el gasto de memoria de la variante que almacena la matriz completa es excesivamente alto, llegando casi a los 1.5Gb. Esto hace poco viable la utilización de dicha implementación en una máquina convencional, pues la solución requeriría del empleo de memoria secundaria, lo que traería consigo pérdida en la ganancia de tiempo. Sin embargo, se puede apreciar que el gasto de memoria de las otras variantes es muchísimo menor, siendo la más ahorrativa la del vector ordenado. Si teniendo estos datos de memoria y de tiempo se quiere saber cuál de las 2 últimas variantes es más beneficiosa, se puede dividir la ganancia de tiempo sobre la de memoria para tener una idea del balance entre ambas. Se tiene en cuenta que se utilizaron 6 procesadores y que ambos datos hay que normalizarlos para tener un resultado confiable, lo que quiere decir que se deben llevar a una misma escala. La normalización de ambas medidas se hizo llevándolas a una escala de $[0,1]$, siendo para el tiempo $\alpha = 1 - \frac{T(n,p)}{T_{suma}(n,p)}$, $T_{suma}(n,p)$ es la suma de los tiempos paralelos de las dos variantes para un tamaño n y p procesadores. La memoria se normalizó de esta otra forma $\beta = \frac{M(d)}{M_{total}}$, siendo $M(d)$ la memoria consumida para un índice de dispersión d y M_{total} la memoria que se gastaría si se almacenara completamente la matriz en memoria. En la tabla 3.4 se puede ver el resultado de $\frac{\alpha}{\beta}$ y mientras mayor es el valor de este cociente más beneficiosa es la variante, de donde se deduce que la variante del árbol AVL es la mejor en cuanto a mantener ambos costos en un nivel balanceado.

Índice de Dispersión	0.70	5.21	3.34
Árbol AVL	33.93788933	5.721566374	7.204408043
Vector Ordenado	1.252729467	0.220062169	0.253422152

Tabla 3.4: Valor de equilibrio entre memoria y tiempo con 64 sectores angulares horizontales y 50 verticales, *p=6.*

3.3: Conclusiones Parciales.

De todo el análisis hecho anteriormente se puede llegar a la conclusión que la variante más rápida es la que utiliza como estructura de datos para representar el modelo una matriz tradicional, sin embargo, para realizar cálculos de alta precisión, la implementación que utiliza el árbol AVL es más viable en términos de requerimientos de memoria, siempre con un costo de tiempo adicional.

CONCLUSIONES

Con el desarrollo del presente trabajo de diploma se logró un algoritmo paralelo escalable que permite la modelación de yacimientos lateríticos. Los resultados experimentales mostraron que se logran buenas ganancias de velocidad y eficiencia durante su ejecución. Se propuso además el uso de estructuras de datos que permitieron un ahorro de memoria significativamente menor al empleado por los enfoques tradicionales.

Estos resultados permiten concluir que la solución propuesta contribuye favorablemente al proceso de optimización de redes de exploración que lleva a cabo el Centro de Investigación del Níquel en Moa, pues no solo permite bajar los costos durante la fase de modelado, sino que permite resolver problemas de mayor complejidad no solubles hasta el momento, debido a los requerimientos temporales y espaciales.

RECOMENDACIONES

- ✓ Utilizar en el algoritmo una biblioteca para manejo de matrices dispersas.
- ✓ Hacer el almacenamiento del modelo en paralelo.
- ✓ Realizar la suma de las matrices resultantes de forma paralela.

BIBLIOGRAFÍA

Adrian, Martínez Vargas. 2006. *Modelación de los contenidos de hierro en yacimientos lateríticos heterogéneos de Níquel y Cobalto. Caso de estudio, yacimiento Moa Oriental".*

Alina, Rodríguez Infante. 1998. *Estudio morfotectónico de Moa y áreas adyacentes para la evaluación de riesgos de génesis tectónica.*

Butt C.R.M. and Zeegers, H. 1992. Regolith Exploration Geochemistry in Tropical and Subtropical Terrains. [En línea] Volume 4, G.J.S Govett, Editor,

Golightly, J. P. 1979. *Nickeliferous Laterites: A general Description.*

Guía Docente de Computación de Altas Prestaciones. [En línea] www.dia.usal.es.

Kumar V, Gramar A, Grupta A, Kerypis G. 1994. Introduction to Parallel Computing: Desing and analysis of Algorithms.

Louisiana, February 19 to 21, Published by Society of Mining Laterite Symposium, New Orleans, Engineers of the American Institute of Mining, Metallurgical en Petroleum Engineering, Inc., Session 1, p. 3-23.

M, Elías. 2002. Nickel laterite deposits-geological overview, resources and Exploitation, in Giant Ore Deposits: Characteristics, Genesis, and Exploration. [En línea] http://www.csaaus.com/documents/public/publications/godpaper.pdf.

M. Ramírez, R. E. Peña, and L. Y. Broscat. La Habana, Cuba, 2009. *Mejora de un algoritmo de conteo para modelos Markovianos en yacimientos lateríticos.*

Martínez Vargas Adrian, Pérez Martínez Yusneuris. 2000. *Metodología para la modelación de yacimientos residuales de níquel .*

Peña, R. E. La Habana, Cuba, 2007. *Algoritmo de conteo para modelos Markovianos en yacimientos lateríticos.*

Peña, R. E. La Habana, Cuba, 2007. Modelo matemático para la optimización de las redes de exploración y explotación en yacimientos lateríticos. In II Convención Cubana de Ciencias de la Tierra.

REFERENCIAS BIBLIOGRÁFICAS

[1] **Adrian, Martínez Vargas. 2006.** *Modelación de los contenidos de hierro en yacimientos lateríticos heterogéneos de Níquel y Cobalto. Caso de estudio, yacimiento Moa Oriental"*.

[2] **Alina, Rodríguez Infante. 1998.** *Estudio morfotectónico de Moa y áreas adyacentes para la evaluación de riesgos de génesis tectónica*.

[3] **Butt C.R.M. and Zeegers, H. 1992.** Regolith Exploration Geochemistry in Tropical and Subtropical Terrains. [En línea] Volume 4, G.J.S Govett, Editor.

[4] **Golightly, J. P. 1979.** *Nickeliferous Laterites: A general Description*. Guía Docente de Computación de Altas Prestaciones. [En línea] www.dia.usal.es.

[5] **Kumar V, Gramar A, Grupta A, Kerypis G. 1994.** Introduction to Parallel Computing: Desing and analysis of Algorithms. [En línea].

[6] **Louisiana, February 19 to 21**, Published by Society of Mining Laterite Symposium, New Orleans, Engineers of the American Institute of Mining, Metallurgical en Petroleum Engineering, Inc., Session 1, p. 3-23.

[7] **M, Elías. 2002.** Nickel laterite deposits-geological overview, resources and Exploitation, in Giant Ore Deposits: Characteristics, Genesis, and Exploration. [En línea] http://www.csaaus.com/documents/public/publications/godpaper.pdf.

[8] **M. Ramírez, R. E. Peña, and L. Y. Broscat. La Habana, Cuba, 2009.** *Mejora de un algoritmo de conteo para modelos Markovianos en yacimientos lateríticos*.

[9] **Martínez Vargas Adrian, Pérez Martínez Yusneuris. 2000.** *Metodología para la modelación de yacimientos residuales de níquel*.

[10] **Peña, R. E. La Habana, Cuba, 2007.** *Algoritmo de conteo para modelos Markovianos en yacimientos lateríticos*.

[11] **Peña, R. E. La Habana, Cuba, 2007.** Modelo matemático para la optimización de las redes de exploración y explotación en yacimientos lateríticos. In II Convención Cubana de Ciencias de la Tierra.

GLOSARIO

MPI: Interfaz de Paso de Mensajes.

IDE: Entorno de Desarrollo Integrado.

AVL: Árbol Binario de Búsqueda.